STUBOK

(STUdent BOdy of Knowledge)

Christophe Cérin
christophe.cerin@iutv.univ-paris13.fr

Version du 28 janvier 2012
Dernière mise à jour sur :
http://www.lipn.univ-paris13.fr/~cerin/stubok.pdf

Sommaire

1 Introduction générale

Sommaire

1.1 Objectifs et exposé des motifs

1.1.1 Objectifs principaux

CET OUVRAGE rassemble des éléments introductifs de ce que nous estimons devoir être une culture commune en informatique et qui sera reprise et développée au sein des disciplines d'une formation universitaire en informatique. Il s'agit d'une introduction à l'informatique et aux disciplines majeures pour des études raissonnées en informatique. De notre point de vue, l'avantage ici est d'avoir un chapeau commun qui regroupe les (des) différents éléments d'une culture commune. L'ouvrage s'adresse en premier lieu à des étudiants qui rentrent dans un cycle L (licence) ou en DUT informatique. Il fournit des clés au savoir scientifique et technique, explicite des méthodes, permet de se les approprier. Le chapeau commun doit contenir du vocabulaire, l'acceptation de règles communes. Cet ouvrage met le doigt dans l'engrenage, l'enseignement disciplinaire complètera les acquisitions.

Cet ouvrage s'inscrit donc dans une démarche pour que nos étudiants puissent faire des choix en toute connaissance de cause à partir d'une appréhension globale des disciplines. Nous commençons par poser dans cet ouvrage la «sous-couche d'accrochage»sur laquelle les disciplines et autres savoirs (savoir être) pourront s'appuyer. Il ne s'agit pas d'identifier tout ce qu'un étudiant doit savoir mais de proposer ce qui à notre avis forme un bagage indispensable pour voyager tout au long de sa vie professionnelle.

Partager une culture commune c'est aussi se poser la question du lien à établir entre l'apprenant et celui qui transmet. L'apriori pédagogique de l'ouvrage est de ne considérer pratiquement aucun préalable. Nous voulons nous adresser à tout étudiant désirant intégrer une formation universitaire en informatique sans nous attacher outre mesure au baccalauréat de l'étudiant qui cependant devra relever du domaine des sciences et des techniques. Nous essayons d'avoir une approche naturelle dans le sens ou les artifices de présentation sont évités et dans le sens

où nous nous référons assez souvent à des pratiques d'étudiants. Il ne s'agit pas d'un ouvrage pour des spécialistes mais une tentative de construction d'un corpus de connaissances transversales, minimales pour bien aborder des études en informatique.

L'acquisition d'une culture permet de mettre du ciment entre les disciplines et de faire du lien. L'approche par la culture, permet dès l'entrée dans une formation d'évoquer les objectifs de la formation et surtout, au delà des formations, les attentes du monde professionnel. Le fondement de notre démarche est de montrer que tout est lié... ce qui constitue un autre objectif majeur de cet ouvrage.

1.1.2 Exposé des motifs

L'honneur de l'université est d'accepter des profils d'étudiants différents, de mettre en place des dispositifs de rattrapage comme de proposer le redoublement au bout de six mois, d'organiser des groupes de discussion et de soutien individualisé.

En 20 ans l'enseignement supérieur a développé son offre de formation d'une manière considérable, a vécu de profondes modifications de ses organisations pédagogiques, modes de fonctionnement de la recherche et a recruté massivement pour faire face à une massification étudiante qui s'est accrue au fil des ans. L'université s'est maintenant imposée comme un acteur en terme de formation continue, de formation en alternance, elle prend une part active dans les projets de recherche lancés par l'Agence Nationale de la Recherche (ANR), elle développe des partenariats internationaux avec l'Europe mais aussi des pays plus lointains, elle passe des «contrats de partenariat«avec de grands organismes nationaux de recherche (CNRS, INSERM, INRIA...). L'université assure également du transfert technologique, s'intéresse à de nouveaux mode d'apprentissage et développe des programmes, des outils de formation à distance.

Cependant le défi premier pour les enseignants-chercheurs reste de travailler sur des publics hétérogènes, parfois exclus ou moins émancipés. La massification de l'université conduit de fait à une population étudiante moins homogène. Pour autant ne serait-il plus possible d'en appréhender la complexité ? Pourquoi ne serait-il pas possible de passer du temps dans les premiers cycles afin de les homogénéiser ? La culture ne serait-elle pas un élément structurant ?

L'université doit rester un lieu d'apprentissage du sens, un lieu où l'on va au cœur des choses, où l'on fait aussi l'expérience du concrèt autant que de l'abstrait, où on lutte contre le fragmentaire pour trouver ce qu'il y a de commun, un lieu où on s'ouvre aux autres et aux réalités des métiers. L'université n'est qu'un passage, mais un passage fondamental qui forme aussi à la citoyenneté. Il y a un «après université»et cet après permet également d'acquérir des compétences. Cela sera discuté dans le chapitre 4 ou au final le modèle d'organisation «salarié, entreprise, état»sera évoqué notamment au moyen des conventions collectives qui seront fouillées, analysées. Ne pas mettre en perspective cet aspect au cours des années de formation ne participe pas à la défense de notre modèle social et à une certaine citoyenneté.

Les vrais problèmes de l'université concernant ses étudiants sont d'abord de noter que ceux-ci souffrent de manques de formation fondamentale comme l'aisance avec la langue commune et technique, l'articulation des idées ou encore des rapports aux savoirs qui font qu'il n'y a pas d'investissement personnel au delà de la salle de cours.

Mais la situation n'est pas désespérée, il y a toujours de la place pour rénover les enseignements en mixant des cycles d'initiation, d'approfondissement des langages, des phases de rencontre / discussion avec des phases où nous devons instaurer les savoirs disciplinaires ; tout ceci dans un cadre national qui travaille à restaurer le prestige de l'Université.

Cet ouvrage aurait pu ou dû également traiter de l'histoire de l'informatique car notre discipline a maintenant plus de 60 ans si l'on considère que l'apparition du premier ordinateur marque la naissance de l'informatique, ou plus de 80 ans si l'on considère les travaux de Turing (1937) et la publication de "On computable numbers with applications to the Entscheidungs problem", ou plus de 100 ans si nous remontons au 8 août 1900, où Hilbert au congrès mondial de mathématique à Paris expose ses vues sur les mathématiques et les problèmes à traiter dans le siècle qui s'ouvre. Le dixième problème de Hilbert est en relation avec ce que sera l'informatique à ses débuts. Le problème demande de trouver une méthode par laquelle, au moyen d'un nombre fini d'équations (c'est cela qui est important), on pourra distinguer si un certaine forme d'équation est résoluble en nombres entiers rationnels.

Enfin, un autre point de vue culturel aurait pû être pris en étudiant le Swebok[1] (Guide to the Software Engineering Body of Knowledge) qui est un ouvrage exposant le corpus de connaissances communes en génie logiciel. L'ouvrage est divisé par grands thèmes, défis scientifiques et techniques et chaque partie du corpus est accompagnée d'une bibliographie détaillée. Une distinction est faite entre les savoirs acceptés et les savoirs avancés ou de recherche. Cela en fait à notre avis un ouvrage à consulter après quelques années d'études. Il est par ailleurs développé avec en arrière plan l'organisation nord-américaine du système de validation des acquis. Le notre est sensiblement différent, nous en reparlerons au chapître 5. Toujours est-il que le titre de notre ouvrage (STUB$_{O}$K) fait explicitement un clin d'œil au Swebok.

1.2 Organisation de l'ouvrage

NOUS AVONS CHOISI de faire en sorte autant que possible que les chapîtres soient auto-suffisants avec des pré-requis minimaux, typiquement du niveau du baccalauréat. Il faut parfois aller piocher dans des connaissances enfouies, mais à priori elles ont été vues en terminale. Cependant, certains chapîtres devront être étudiés après quelques mois dans une formation en informatique. C'est le cas par exemple des études de cas qui mettent en pratique tous les concepts et outils

[1] http://www.swebok.org

vus dans l'ouvrage. Ils nécessitent du recul et très certainement au préalable, de longues heures de pratiques d'analyse, de modélisation et de codage.

On ne peut pas dire non plus que les chapîtres soient indépendants les uns les autres. Notre objectif est précisément de montrer le continuum de connaissances et la rigeur avec laquelle nous devons les aborder pour réussir son passage à l'université et se préparer à sa vie professionnelle. Tout est lié depuis l'apprentissage d'outils mathématiques ou de programmation jusqu'à l'appréhension du monde professionnel qui nous entoure. Tout cela doit être étudié de manière précise, détaillée et systématique, en un mot, de manière rigoureuse.

Le chapître 1 fournit des éléments pour conduire un projet d'étude. Celui ci doit contenir une analyse du problème, la dérivation d'une solution, une partie expérimentale suivie d'une analyse des résultats et enfin une conclusion qui résume le travail. À partir d'écrits d'étudiants nous montrons ce qui ne va pas et comment remédier aux nombreux problèmes liés à l'écriture d'un rapport à valeur scientifique et technique.

Nous présentons ensuite les démarches de modélisation mathématique, de simulation et d'émulation. Le chapître comporte également une introduction à la typographie du français et à la présentation d'une communication orale appuyée sur des transparents.

À l'issue du chapître, le lecteur a une vue d'ensemble de la démarche de construction d'un rapport de projet (en fait, de construction d'un mini-projet) ainsi que les outils pour mettre en valeur les résultats obtenus.

Le chapitre 2 est de facture classique (on pourrait même dire «vieille facture classique») pour un ouvrage d'informatique dans le sens ou il expose les outils pour raisonner sur les programmes, les paradigmes de programmation et le langage Python qui nous sert de langage d'implémentation utilisé dans les études de cas qui suivront. Notre parti pris est de considérer ici que la rigueur s'aquiert par l'usage et la compréhension des mathématiques utilisés dans le contexte de la preuve de programmes et la modélisation de propriété d'objets informatiques.

Les mathématiques permettent de justifier et d'expliquer ce que l'on fait de manière concise en faisant ressortir les propriétés des objets. Les mathématiques sont donc utilisées dans cet objectif. C'est réconfortant mais bien entendu cette démarche n'est pas toujours suffisante pour des problèmes de plus grandes ampleurs que les études de cas que nous abordons au chapitre 3 et qui sont introduites pour faire réfléchir sur la manière de raconter la solution d'un problème relativement circonscrit. Les outils mathématiques sont là pour donner de la rigueur aux discours.

Le chapitre 3 aborde différentes études de cas. Là aussi elles sont de facture classique dans le sens où elles développent, pour la plupart, des problèmes connus dans la littérature. Sauf que les développements sont approfondis ou partent sur des pistes complémentaires afin d'aller, en particulier, en direction de l'expérimentation qui valide d'une certaine mesure les idées exposées.

Les problèmes traités sont un problème de géométrie issu du livre de Polya (voir la bibiographie du chapitre 2) ; le problème de la plus longue sous-suite croissante dans un contexte de programmation concurrente ; la construction d'un simulateur de cache ; la spécification du type abstrait multi-ensemble et un problème de dénombrement d'expressions parenthésées.

Le travail du lecteur est de bien examiner comment les arguments sont avancés, comment ils s'articulent, comment et à quels moments les hypothèses sont émises et vérifiées, de découvrir le plan de résolution, son application. . . et bien entendu de se persuader de la correction de la solution.

Après toutes ces considérations académiques, le chapitre 4 bascule, de manière tout autant rigoureuse, du coté de la profession et des métiers. Il est constitué en bonne partie d'une mise en œuvre du Projet Personnel et Professionnel (PPP) des Diplômes Universitaires de Technologie (DUT) en informatique. Au delà de la découverte d'un (des) métier(s), il s'agira pour le lecteur de vérifier si les connaissances que nous avons vues dans les chapitres précédents sont suffisantes pour bien s'intégrer et réussir une carrière profesionnelle.

La réponse est non et ce chapitre sert de pont entre les savoirs académiques et les compétences complémentaires à développer par ailleurs. L'idée est de faire prendre conscience au primo arrivant que le diplôme ne fera pas tout : il ne doit pas s'imaginer que le reclassement dans la grilles des salaires se fera directement en fonction de son diplôme. L'étudiant pourra vérifier en étudiant certaines conventions collectives que les diplômes ne sont pas explicitement mentionnés dans les grilles. Les professionnels peuvent indiquer un DUT (au sens très général) comme point d'entrée dans les fonctions de techniciens ; ce qui va les intéresser à l'embauche, c'est de vérifier que le candidat correspond au poste défini.

Le chapitre 4 est organisé en fiches de travail plus ou moins indépendantes qui permettent de progresser rapidemment vers les objectifs de réalisation d'une entrevue d'un professionel ou d'une visite en entreprise suivies par des restitutions. L'objectif est de faire en sorte que, à partir d'un métier choisi, le lecteur puisse au final faire part de son retour d'expérience, par exemple sous la forme d'un poster et de croiser ce que le professionnel à pu raconter avec ce qui était attendu pour son métier d'après la lecture de fiches métiers accessibles sur des sites spécialisés.

Pour l'enseignant, ce chapitre l'interroge sur les relations universités - entreprises, diplômes - compétences afin qu'un jour peut-être, les diplômes soient définis nons seulement en terme de connaissances mais aussi en terme de compétences. Pourquoi ne pas mentionner lors de l'habiliation d'un diplôme la correspondance, de manière explicite, entre connaissances et compétences ? Qu'avons nous tous à y gagner ? Les initiatives en la matière sont sans doute trop peu nombreuses : nous pouvons cependant noter celle de l'université de Rennes[2].

Enfin, le chapitre 5 plonge délibérément du côté des compétences complémentaires elles aussi indispensables pour maîtriser sa compréhension du champs professionnel de l'informatique, on dit dorénavent du champs professionnel des sciences

[2]http://www.univ-rennes2.fr/sfc/interface-competences

et technologies de l'information. Il s'agira d'une introduction au droit lié à l'informatique notamment le rôle du «libre», une introduction aux modèles économiques et enfin un récapitulatif des notions importantes d'anglais de l'informatique.

Sur ce dernier point, nul n'est besoin de rappeler l'importance de l'anglais dans notre contexte si ce n'est qu'il est en France langue de travail dans de nombreuses sociétés d'informatique ou bien que beaucoup de manuels utilisateurs ne sont immédiatement disponibles qu'en anglais. Nous avons donc ici une vision utilisatrice de l'anglais : il fallait bien qu'il y ait au moins une contradiction dans l'ouvrage dont le tître fait directement référence à la Culture... et la Culture n'est pas liée à une vision purement utilisatrice de la connaissance !

L'ouvrage est en fait le fruit d'un travail collectif. Je voudrais remercier les collègues d'anglais (XXX), Ahmed Elleuch pour sa contribution sur les modèles économiques, Christèle Couleau et Bernard Quino pour la rédaction de plusieurs fiches du PPP, Michel Koskas (INRA) pour les discussions sur les automates et la modélisation mathématique ainsi que Sabine Bodiroga (Service de formation continue de l'IUT de Villetaneuse) pour une relecture.

2 Éléments méthodologiques et conceptuels

Sommaire

2.1 Motivations

CES PAGES apportent un ensemble de conseils généraux pour organiser et présenter un compte rendu d'un travail académique (le compte rendu de réunion c'est autre chose). Il s'agit aussi d'une introduction à la préparation d'un exposé oral, une introduction à la problématique de la réalisation d'expériences au moyen d'ordinateurs qui sera l'occasion de préciser les notions de modélisation mathématique, de simulation et d'émulation. Enfin, le lecteur trouvera une introduction aux règles typographiques usuelles permettant de construire aussi bien un rapport qu'une présentation visuelle sur un vidéo projecteur par exemple.

Ces pages s'adressent à des étudiants du champs disciplinaire des sciences et technologies de l'information en vue de les aider à préparer leurs mémoires de fin d'étude, leurs rapports de projet élaborés dans le cadre académique, autrement dit leurs productions écrites telles qu'on peut les attendre dans des structures d'enseignement en cours ou à la fin d'une année. Nous pensons aux écrits dans tous les cursus depuis la première année universitaire jusqu'à la licence. Nous n'envisageons pas explicitement le cas des rapports de stage en entreprise bien que certaines parties de l'ouvrage puisse être d'un grand secours. Nous avons préféré une approche plus globale.

Nous avons pu observer depuis de nombreuses années que malheureusement les écrits des étudiants sont très inégaux. Cet article est donc né d'un double constat : premièrement, les étudiants de savent pas rédiger. De manière moins négative, on pourra dire : les étudiants ne savent pas quoi rédiger ou encore qu'ils ne savent pas organiser leurs idées afin de mettre en valeur leur travail. Deuxièmement :

la méthodologie et les techniques en la matière ne sont pas vraiment enseignées. Elles sont bien souvent supposées acquises.

Or, en praticien nous constatons que ce qui est acquis au niveau du baccalauréat n'est pas transposé facilement par les étudiants dans les cursus universitaires. Par exemple, on constate que très peu de sujets de travaux dirigés ou pratiques ou de rapport sont vraiment organisés autour d'un plan. Quant à la démarche scientifique et technique proprement dite, elle est très rarement mise en valeur quand elle existe dans le document écrit. Et nous ne parlons même pas des justifications.

Autrement dit, nos étudiants, en particulier ceux de la première année (là même où nous constatons les plus grands taux d'échec) ont de grandes difficultés à analyser objectivement les données, à dire l'essentiel, à le présenter clairement et simplement selon un plan rigoureux de façon qu'une «personne normale»puisse se forger une opinion du travail réalisé.

Ne soyons pas plus royaliste que le roi ! Nous voulons donner dans les pages qui suivent des recommandations, mais pas une vérité, en matière de conduite de projet (universitaire) et de rédaction d'un compte rendu d'un travail scientifique et technique dans les domaines des sciences et technologies de l'information.

Bien souvent les étudiants ont de bonnes connaissances dans un langage de programmation particulier mais quand il s'agit de dire, de commenter de manière générale ce que fait le code... c'est une autre affaire ! Idem pour justifier que le code répond bien à la spécification (même de manière informelle).

Au final, nous voulons montrer que communiquer c'est arriver à dire l'essentiel, le plus simplement possible en évitant les discours de spécialistes, le jargon technique en particulier. Y est-on arrivé ? Il nous semble au moins que nous poursuivons un but valide et que les questions ont du sens.

L'absence, à notre connaissance, de synthèses ou d'ouvrages introductifs traitant des points ici soulevés pose le problème de la difficulté de cerner les connaissances à transmettre à nos étudiants en la matière. Cependant ce travail d'explication qui traite aussi de la façon dont les évaluateurs évaluent doit être fait. Il serait très regrettable de s'apercevoir que c'est un manque d'information sur les règles du jeu de l'évaluation qui entraînerait en partie l'échec aux examens alors que les initiés (de part une position sociale avantageuse ou un décryptage rapide des règles sous entendues) obtiendraient les meilleurs résultats.

L'objectif est ici de permettre à l'étudiant d'acquérir des savoir-faire comme par exemple la capacité d'analyser divers documents, d'effectuer sa recherche d'information, d'exposer clairement sa solution, de répondre aux questions posées et d'écrire un compte rendu ayant non seulement des qualités de forme mais aussi dont le contenu dépasse la simple description par l'exposition d'un fil conducteur et d'une valorisation des résultats.

Le travail d'exposition de compétences et de techniques que nous abordons ici permet, à partir de nos pratiques et de nos vécus, de dégager des techniques générales utilisables pendant tout un cursus universitaire et même au delà. Il est utile parce qu'il se situe sur un plan transversal et permettra très certainement au futur jeune diplômé une adaptation aux exigences de la vie professionnelle.

Par exemple, la convention collective des sociétés de service et d'ingénierie (SSII), ne fait pas référence aux capacités techniques (connaissance de tel ou tel langage de programmation - Java/C#) mais aux facultés périphériques du coeur de la discipline (au sens universitaire), à savoir la capacité à communiquer, à échanger, à l'autonomie et au degré de responsabilité ainsi qu'à l'expérience. D'où la nécessité de préparer l'étudiant à développer ces formes de compétence qui ont un impact important sur son futur salaire lorsqu'on devient salarié d'une SSII[1] !

Comme justification supplémentaire, notez par ailleurs que les SSII ont la particularité d'embaucher beaucoup de jeunes diplômés (de l'ordre de 50% de leurs embauches). Elles fonctionnent comme un «sas»où les jeunes diplômés font leurs gammes avant de se diriger vers les entreprises utilisatrices ou vers des fonctions d'encadrement. À la vue du flux de jeunes diplômés informaticiens passant chez les SSII, il convient de les préparer au mieux à une certaine rigueur dans l'exposition de ce qu'ils font, ont fait et feront. D'autant plus que les autres personnes embauchées par les SSII, à savoir les jeunes diplômés non informaticiens (double compétences par exemple) ou encore les personnes expérimentés mais dans un tout autre domaine que l'informatique (étudiants d'écoles de commerces par exemple) risquent d'avoir déjà développés ce type de compétences ! Donc, un informaticien pur jus a intérêt à développer ces capacités parce qu'il sera confronté à des personnes qui les possèdent.

Nous avons décidé de nous intéresser d'abord au cas de la constitution d'un rapport correspondant à un travail d'étudiant, typiquement la rédaction d'un rapport de projet. Les difficultés rencontrées par l'étudiant viennent du fait que le sujet du projet est généralement ouvert, que le code produit ne se résume pas à 50 petites lignes. L'étudiant à donc à produire un effort de réflexion important du type : mais au fait, quel est le véritable problème, la difficulté majeure, le coeur du problème ?

Répondre à des questions fermées, du type «montrer par la méthode X que Z»offre généralement moins de difficulté à l'étudiant. Répondre à des questions guidées oriente la rédaction vers un plan : la liberté d'imaginer sa propre solution s'en trouve réduite. Notre optique serait plutôt de placer l'étudiant, pendant ses études, devant la réalité suivante : dans la vie professionnelle, une fois sortis de la salle de travaux dirigés, il faudra se poser les questions importantes à traiter pour résoudre le problème. Peut être même que vous aurez le bonheur d'avoir à imaginer les problèmes à traiter.

Ce document est constitué de 6 paragraphes. Le premier paragraphe s'occupe de l'organisation générale d'un compte rendu. Le paragraphe 2 donne, à travers des exemples tirés de copies d'étudiants, des maladresses commentées. À titre d'exercice, relisez ces exemples et essayez d'améliorer le style, de préciser le sens. Certains

[1] Syntec Informatique, la chambre patronale des SSII met en avant l'idée que s'il existait des métiers repères (basés sur des compétences techniques pures), une renégociation annuelle serait nécessaire pour tenir compte de l'évolution technologique. On peut peut être même leur accorder qu'ils font confiance aux enseignants pour transmettre un bagage métier initial suffisament large pour pouvoir s'adapter au fil des ans.

de ces exemples correspondent à la translation d'une phrase en langage naturel dans le langage mathématique de la logique avec quantificateurs, ou réciproquement. L'idée est ici de faire exprimer une idée, un concept et de s'interroger sur le sens des phrases. Le paragraphe 3 donne un ensemble de règles typographiques afin que votre compte rendu respecte des critères de forme. Le paragraphe 4 s'intéresse à des critères pour construire une expérimentation sur machine et renvoie le lecteur aux problématiques de modélisation. C'est dans ce paragraphe que nous précisons les notions de simulation, d'émulation et de modélisation mathématique. La paragraphe 5 donne un ensemble de conseils pour présenter et construire une présentation orale. Le paragraphe 6 conclue ce document qui se termine par une annexe composée d'un exemple de présentation orale plutôt orientée vers un travail de recherche à valoriser.

2.2 Organisation du travail

LES PARAGRAPHES qui suivent présentent une liste de questions auxquelles vous chercherez à répondre avant la rédaction d'un compte rendu. Elles constituent autant de critères d'évaluation pour le correcteur.

L'objet du compte rendu est avant tout de montrer que vous avez bien mesuré les difficultés du problème et que vous avez apporté des solutions «raisonnables»et «cohérentes». L'évaluateur veille à ne pas être trop permissif (vous écrivez n'importe quoi, pourvu que ça marche) ni trop restrictif (rien n'est assez bien fait). De même, il doit aussi pouvoir apprécier la méthodologie employée, la précision et la qualité de l'exposé technique ainsi que la présentation du travail.

2.2.1 Qualités générales

Un bon rapport est constitué de plusieurs parties qui s'enchaînent afin de donner un sens à votre propos. Le style est clair : employez des phrases courtes avec un sujet, un verbe et un complément. Écrivez à la voie active et n'abusez pas du passé composé. Préférez le Nous au Je et évitez le On qui est impersonnel. C'est dans un CV qu'il faut utiliser le Je ! Ne donnez pas des développement sans rapport direct avec le but recherché ou le sujet : ne semez pas le doute, l'incertitude ou pire, l'inquiétude dans l'esprit de votre lecteur.

Faites preuve d'élégance par la richesse de votre vocabulaire (utilisez habilement le vocabulaire appris dans les cours, TD, TP)

Vous commencez par exposer brièvement le problème puis vous résumez, en quatre ou cinq lignes les points importants du compte rendu et vous décrivez notamment les apports personnels.

Au cours des autres parties vous formulez des opinions solides, bien fondées et claires. L'utilisation d'un «peut être»est acceptable si vous argumentez suffisamment. Dans tous les cas, votre compte rendu contiendra assez d'informations qui justifient vos choix.

Essayer de construire chaque phrase en exposant d'abord les concepts à priori

connus puis les concepts supposés pas connus. Allez d'une idée générale à une nouvelle idée, du background vers le résultat. En règle générale, commencez par définir les objets non connus avant de les utiliser dans le texte.

Vous trouverez ci-dessous une liste d'interrogations qui vous aiderons à construire vos écrits en fonction des objectifs définis.

- Quels sont les problèmes ? Sont ils bien expliqués ? Avez vous exposé ce que vous vouliez faire ? Par exemple, si vous décrivez un algorithme, a t il été réellement implanté ou bien s'agit il d'une amélioration qu'il serait possible d'implanter dans le futur ?

- Le but recherché correspond t-il vraiment à l'énoncé du problème ? Si tel est le cas, vos explications ne contredisent t-elles pas certaines lois établies ? (vous affirmez que la complexité du problème est exponentielle alors qu'il est bien connu que le problème est linéaire).

- Est ce que votre approche est suffisante, bien construite ? Vos explications sont elles auto-suffisantes ou bien font elles appel à des connaissances «extérieures» à votre compte rendu ?

- Détaillez vous assez les méthodes de résolution : s'appuient elles sur des formules mathématiques ou relèvent t-elles du bon sens ? Sont elles réalistes ?. Est ce que les résultats que vous obtenez sont consistants avec vos hypothèses de travail ou avec les faits observés ? Sont ils plausibles ? Les conditions aux limites ont elles été vérifiées (pas d'effet de bord) ? Tirez vous les bonnes conclusions de vos résultats ?

Dans tous les cas, faites ressortir les idées qui vous semblent originales en les commentant suffisamment, ni trop, ni trop peu. N'employez pas d'affirmations trop laconiques, sibyllines ou énigmatiques : soyez clairs !

Dans la préparation à la rédaction d'un projet, je vous recommande d'écrire le plus tôt possible une première version du rapport que vous raffinerez au fil du temps. Vous avez intérêt à vous faire relire par une personne extérieure au projet. Cette personne devra «tout» comprendre à la première lecture sans votre intervention. La première version s'écrit à partir d'un plan qui listera sur au plus une page des titres, les arguments importants à faire figurer que vous détaillerez plus tard.

DON'T START WRITING A DOCUMENT ANY OTHER WAY!!!

2.2.2 Établir un plan

Un plan de résolution doit être établi lorsque les calculs, les raisonnements ou les constructions à effectuer pour trouver l'inconnue ont été identifiés.

Il est souvent intéressant, pour dresser votre plan, d'examiner l'inconnue, les données et les conditions pour un problème à résoudre et l'hypothèse et la conclusion dans un problème à démontrer. Éventuellement, il faut penser à un problème

qui vous est familier et qui porte sur une même inconnue ou une inconnue similaire. Quelle est alors l'utilisation que vous pouvez en faire ?

C'est à ce moment que vous devez vous demander qui et quand a-t'on travaillé sur le problème[2]. Vous devez alors faire une recherche bibliographique : aidez-vous des moteurs de recherche, si possible les moteurs dédiés. Lisez la documentation en ligne, les tutoriels disponibles. Dans les articles que vous lisez, qu'est ce qui a de nouveau par rapport à ce que vous connaissiez ? par rapport à ce qu'était la discipline avant cet article ?

Revenons aux questions à vous poser pour établir un plan. Pouvez vous énoncer le problème différemment ? Une modification du problème peut conduire à un problème auxiliaire qui se rattache au votre (introduction d'une nouvelle donnée ? de nouvelles hypothèses ?). Pouvez vous tirer des données un élément utile ? Pouvez vous penser à d'autres données ou bien pouvez vous changer l'inconnue, les données (les deux à la fois ?). Vous êtes vous servi de toutes les données, de la condition dans son ensemble ?

Lorsque le plan est établi, il faut le mettre à exécution : ne soyez pas infidèle à votre plan ! Vous devez alors vous assurer que les détails cadrent avec cette ligne directrice en les examinant attentivement les uns à la suite des autres pour vérifier que l'enchaînement des idées est cohérent et «prouver»l'exactitude soit en vous référant à votre intuition (description informelle) soit à l'aide d'une démonstration formelle.

Vérifiez que chaque partie du plan est organisée : vous allez pouvoir construire un discours où les idées seront ordonnées et non pas juxtaposées sans aucun lien. Soignez vos transitions. Les transitions comportent une partie synthèse / conclusion des développements qui précèdent et une annonce des développements qui vont suivre.

Pour conclure sur la rédaction du compte rendu, vous devez vous efforcer à ce que chaque paragraphe, sous paragraphe soient autant d'éléments de démonstration des idées annoncées.

2.2.3 Qualités de présentation

Le compte rendu doit être bien écrit afin de faciliter l'évaluation de son contenu. On évite l'utilisation de références arrière puis avant, à tout moment. Les paragraphes doivent se succéder dans un ordre logique, la grammaire et la syntaxe sont correctes. Le résumé, en introduction, décrit l'ensemble du compte rendu. De même la conclusion doit faire le point sur l'ensemble de votre étude ; elle se termine sur une réflexion sur les limites du travail, les perspectives de développement.

Éliminer les mots qui ne sont pas nécessaires ou les superlatifs. Par exemple, à la phase suivante : «La partie de code qui nous semble être très importante à commenter est. . . »préférez le texte suivant «Commentons la partie importante du code». En faisant cela, nous avons à la fois simplifié la présentation et enlevé le superlatif.

[2]Il y a sûrement d'autres personnes avant vous qui ont travaillées sur le problème

Si possible, bien que cela constitue probablement des éléments de votre propre style, évitez les mots creux comme Cependant, Vraiment, Il est important de noter que, Maintenant.

Les titres (du document principal, de paragraphes, de sous paragraphes...) ont une importance cruciale : ils doivent donner l'essence de l'idée qui va être défendue dans le document principal, dans le paragraphe... Par exemple, à la lecture du titre principal, on doit être capable de trouver la contribution faite dans le document. Un usage consiste à écrire un titre composé de deux parties séparées par un deux-points. Exemple : «Construction d'un simulateur et mise en évidence de la hiérarchie mémoire cache : étude de la localité spatiale et temporelle» Ce titre à plus d'effet sur le lecteur que le titre «Simulateur cache». Il est plus informatif et moins lapidaire. Il est peut être un peu long, à vous de voir.

Une autre technique possible pour construire un titre de paragraphe est de fabriquer un *titre phrase* avec un sujet, un verbe, un complément.

Les figures et les tableaux de résultats sont lisibles, référencés et commentés. Il ne doit pas y en avoir trop (pertinence du choix). Si vous présentez des courbes expérimentales (attention aux échelles), vous devez décrire la méthode qui vous a conduit aux résultats présentés et les interpréter en «termes pratiques». Vous devez aussi décrire les outils utilisés et leurs apports ainsi que commenter les coûts introduits par leurs usages.

Le compte rendu est rédigé d'une manière ni trop familière, ni trop formelle. Il est relu plusieurs fois et ne comporte pas d'erreurs typographiques majeures qui nuisent à la lisibilité. Pour ceux qui utilisent un traitement de texte, on ne souligne pas les titres de paragraphes, de sous paragraphes... Pensez aux qualités de forme. Comparez avec d'autres compte rendus afin d'éviter les maladresses de style les plus fréquentes comme celles présentées au prochain paragraphe.

Vous pouvez créer un paragraphe de motivations et/ou intitulé «Travaux connexes» qui va vous permettre de comparer vos apports avec ce qui était connu avant vous ou qui est proche de votre travail. Il ne faut ni laisser l'impression que vous êtes le seul à travailler sur le problème ni que le problème n'a pas de racine profonde et que sa résolution n'a pas d'importance pratique.

Si possible, et c'est un must, vous montrez que vous apportez un *incrément*. Avant vous il y avait ça, maintenant il y a ça. Ne le dites pas comme cela de manière explicite mais faites le comprendre subtilement à votre lecteur. Vous êtes un charmeur !

2.2.4 Le rôle des exemples et de la formalisation

Il faut bien situer le niveau du public à qui vous vous adressez afin de donner l'information adéquate qui va faciliter la compréhension du texte. Par exemple, dans un article de recherche on ne va pas rappeler que la borne du problème du tri est en $\mathcal{O}(n \log n)$ par contre, si vous utilisez un tri comme brique de base dans la décomposition d'un problème de licence ou maîtrise, vous pouvez le rappeler et vous en servir pour argumenter autour de la complexité du problème global.

Apprenez à faire la différence entre rigueur et formalisme. Une explication courte peut s'avérer suffisante. Avoir de la rigueur c'est faire des choix raisonnés.

Vous pouvez donner, au contraire de ce qui se passe en général dans un papier de recherche, une conjecture même si vous démontrez quelle est fausse. Elle peut être intéressante dans le sens ou l'on peut mieux apprécier votre démarche et votre raisonnement.

Il est toujours très pédagogique de discuter d'abord de cas particuliers puis du cas général et ceci à partir d'exemples. Ces exemples doivent révéler les réelles difficultés du problème ; on ne les choisi pas n'importe comment !

2.2.5 Imprécisions et maladresses rédactionnelles

Les phrases qui suivent sont toutes tirées de compte rendus d'étudiants. Certes elles sont un peu sorties de leurs contextes, mais cela ne doit pas vous empêcher de vous essayer à les reformuler correctement. Certaines d'entre elles visent à traduire une idée, un concept dans un autre langage, par exemple du langage naturel vers le langage des propositions et des quantificateurs ou du langage des propositions vers le langage naturel. Il est très surprenant d'observer combien les idées sont «déformées».

1. Commentez toutes vos affirmations. Exemples :

   - «Ces programmes pouvant être conçus de différentes façons, la troisième partie expose...»
     (lesquelles) ;
   - «Pour le premier programme: pour ce dernier, nous avons envisagé deux solutions pour l'algorithme des listes»(et vous ne dites pas lesquelles – vous de dites pas, non plus, ce que sont les listes dans ce cadre de travail : tout objet doit être défini avant son utilisation) ;
   - La notion suivante : la majorité des éléments du tableau T ne sont pas présents plus d'une fois, est traduite par la phrase

     $$card(card(\{x \in \mathbb{N} : 1 \leq a \leq n : T[a] \in T\}) = 1) > 1/2$$

2. Employer le bon mot au bon endroit. Exemples :

   - Le booléen `bool` sert à sortir de la boucle après le swapage et...
   - «Les jeux d'essais prouvent...»(ils ne prouvent rien, mais valident une implantation) ;
   - «Comme notre listing des jeux d'essais ainsi que notre dossier le prouvent ...»(même remarque que précédemment) ;
   - «Ce programme permet à l'utilisateur, de façon transparente, de saisir...»
     (transparent = qui laisse passer la lumière) ;

- D'après la condition d'hypothèse...
- D'après l'hypothèse d'injection...
- La fin des premières phrases doit être calquée par rapport à la dernière.
- Faisons une preuve par application numérique en prenant $x = 5$ et $y = 3$.
- Il faut ensuite prouver l'invariant : à l'ordre 0 ; puis à l'ordre n et en déduire l'ordre $n + 1$.
- Le processus est codé avec les conditions suivantes : R $\rightarrow$ nombre de boules rouge dans l'urne.
- On sait que les éléments sont distincts donc pas de doublons.
- Donc ce qui semble un tri en n n'est en fait qu'un tri en $2n - 1$ étapes.
- Si on effectue `take(l,j)` et `drop(l,i)` sur la même liste et que l'on garde les éléments identiques dans les deux listes résultantes on obtiendra alors les éléments entre les positions i et j de la liste l.

3. Des idées claires et précises. Exemples :
   - On se rend compte en fait que le travail revient à mettre en œuvre une boucle (ou un appel récursif) ce qui revient au même d'un point de vue terminal.
   - Soit $b[1..n]$ un tableau qui pour chaque phrase exprime le début de chaque mot (nombres pairs) et la fin de chaque mot (nombre impair).
   - Donc on incrémente 2 de manière à ce qu'il ne soit pas déjà utilisé avant sa position.
   - Ensuite le programme va afficher à l'écran le résultat que la fonction f va lui retourner après que le programme lui est fait appel avec les variables x et y
   - «Enfin, notre dernier jeu d'essai a été fait au cas ou on lirait un fichier d'ouvrages après l'avoir créé mais en n'y ayant pas rangé d'enregistrements » ;
   - «Nous retrouvons donc bien les livres à leurs places et aucun dans un fichier où ils ne devaient pas être.» ;
   - «Nos deux programmes fonctionnent correctement et montrent que l'on peut gérer une bibliothèque *assez* aisément.»(le assez est peut être de trop !) ;
4. Dans cette méthode, on permute à l'indice donné la valeur donnée à l'indice i de base.
5. (au sujet de la spécification d'un programme P^{-1} réalisant l'inversion du programme P) : P^{-1} ne devra plus redonner aux variables leurs valeurs initiales avant l'exécution de P mais les valeurs possibles, hypothétiques qu'il avait avant l'exécution de P.

6. (toujours au sujet de la spécification d'un programme P^{-1} réalisant l'inversion du programme P) : Les programmes inversés tels qu'on les voit ici permettent de revenir à ne rien faire. Il s'agit en quelque sorte de trouver la matrice identité en combinant $P.P^{-1}$.

7. mais ce n'est pas suffisant car on peut avoir des valeurs négatives qui une fois positives soient supérieures à n donc on est obligé de restreindre à n.

8. Lorsque l'on dit que le tableau vérifie $X[i] = I_i$ pour tout i compris entre 1 et n, en fait, on entre la position dans le tableau qui doit nous donner la valeur équivalente qui est contenue à cette position.

9. Ne dévalorisez pas votre travail. Exemples :

   - «Le but secondaire du TP est...»(il n'y a rien de secondaire) ;
   - «Cette variable qui en fait n'a aucune utilité...»(alors pourquoi en parler ?) ;
   - «Ce programme devait afficher...»(on regrette, c'est qu'il ne le fait pas !) ;

10. Pas de conclusion banale. Exemple :

    - «Ce TP nous a permis, en quelque sorte, de faire une révision sur les points vus en première année ce qui nous a paru très profitable.»;

Voici d'autres phrases tirées de vos copies qui illustrent l'effort qui vous reste à accomplir dans les directions précisées plus haut ! À titre d'exercice, je vous invite à les reformuler convenablement (si cela est possible) et à expliquer pourquoi ces phrases sont «bancales»(mal formulées, elles n'apportent rien...).

- soit $(a/b)+c$ soit $a/(b+c)$: deux expressions qui seront différentes dans la plupart des cas pour ce qui est du résultat.
- cet algorithme n'est donc d'aucune utilité (*c'est pas ce que l'on demandait d'apprécier : il s'agissait d'apprécier la difficulté du passage d'une expression bien parenthésée à une expression complètement parenthésée*).
- le choix de l'opérateur doit déterminer les définitions.
- en revanche, a_i doit être $<$ à b_i et tout ceci où $i =$ longueur de A et inférieur à celle de B.
- $\exists i \in [1..n], \forall A_i \leq X, A_i = X$
- Elle permet donc d'avoir le classement pour des chaînes qui diffèrent avant la fin.
- les difficultés résident essentiellement dans le fait de transformer les formules de l'un à l'autre.

- nous utilisons une boucle récursive.

- pour cela il faut faire un test afin de savoir si ')' est suivie d'un opérateur ou non afin de continuer le calcul (*on continue de travailler dans les deux cas de toutes les façons !*)

- Il y a impossibilité de savoir le type d'un objet car cela n'existe pas en langage de programmation (*de l'audace ! et puis c'est faux !*)

- si parenthèses trouvées alors envoyons en paramètre le contenu bien parenthésé au mécanisme.

- l'exemple donné à la question a) satisfait le problème.

- dans cette définition, l'idée principale consiste à comparer la longueur de la chaîne A (lg(A)) à la longueur de la chaîne B (lg(B)) en faisant intervenir l'indice j et en introduisant une constante i grâce à laquelle on peut la comparer à la lg(A), à la lg(B) et à l'indice j de ces deux tableaux.

- étant donné qu'une machine ne possède pas (de façon standard) la capacité de réflexion et de prise d'initiatives...

- soit $S(n) = x_1 \times x_2 \times \cdots x_m, m = 1, 2, \cdots, n$. On peut aisément calculer $S(m)$ en une seule opération chacun.

- ce passage est possible en se référant à la table de vérité du OU et à la nullité du premier membre.

2.3 Règles typographiques

NOUS CONSIDÉRONS dans les sous-paragraphes qui suivent les règles usuelles de typographie du français... et aussi de l'anglais. La typographie a une longue histoire derrière elle. Ne comptez pas en devenir un expert sans des années de pratique et d'expérience. Commençons par deux citations ; la première est relative aux buts poursuivis par les écritures, l'autre est amusante et contradictoire avec les idées de la première :

> *Toutes écritures ne sont bonnes ni profitables sinon en tant que l'on en reçoit un usage, pour conduire plus dextrement les affaires communes*
>
> **Plantin (1567)**

Sleon une édtue de l'Uvinertisé de C'mabrigde, l'odrre des ltteers dnas un mto n'a pas d'ipmrotncae, la suele coshe ipmrotnate est que la pmeirère et la drenèire soient à la bnnoe pclae. Le rsete peut êrte dnas un dsérorde ttoal et vuos puoevz tujoruos lrie snas porblème. C'est prace que le creaveu hmauin ne lit pas chuaque ltetre elle-mmêe, mias le mot cmome un tuot.

Télérama n° 2803, 1er octobre 2003, page 8

Concernant la grande histoire des écritures, on peut dire qu'elles ont d'abord étaient cursives (manuscrites). Puis, officielles elles tendent à se styliser, à devenir baroques. La réforme carolingienne et la renaissance (14e et 15e siècle) ont provoqué des réactions. Depuis, il n'y a eu que des progrès dans la mise en forme des textes. Par exemple, les grandes caractéristiques d'une police ont été définies depuis bien longtemps.

En fait, la typographie c'est l'étude

1. des signes de l'alphabet, les chiffres et les ponctuations ;
2. de la ligne d'écriture : comment la composer ;
3. de la colonne de texte ;
4. de la surface de la page ;
5. de la répartition des blancs ;

Il y a une phonétique de l'œil»selon le mot de Maximilien Vox qui fut un grand typographe. Le *rythme* est bon ou pas selon que l'on rapproche plus ou moins les caractères, selon leur proportion respective :

Exemples : WE et W E et WE, OIJ et O I J, Oij, ff, ffl, ffi. Noter les ligatures des trois derniers caractères. Une police de caractères est aussi composée de caractères ligaturés : ff par exemple n'est pas la juxtaposition de deux f.

Une lettre peut être ronde, étroite, épaisse ou maigre. Pour gagner de la place on peut sacrifier les contrastes, le rythme et la couleur. Pour gagner de la couleur, pour augmenter l'impact des titres, on force sur la graisse et **l'encombrement des caractères**.

En typographie, les auteurs qui désirent mettre en valeur un terme peuvent le faire en utilisant l'*italique* ou le **gras** (qui est réservé aux titres), mais il ne pas utiliser le soulignement qui est une faute typographique rappelons-le. Le gras ne s'utilise pas au sein d'un paragraphe : il faut utiliser l'*italique*.

Les *MAJUSCULES* sont un héritage de l'antiquité romaine et ne sont pas ligaturés. Elles sont réservées aux titres, aux initiales. Elles s'alignent entre deux lignes parallèles horizontales.

Les *minuscules* sont un héritage de la renaissance carolingienne. Il faut 4 lignes pour les aligner (2 pour : aeimnorsuvwxz), (1 pour gpqy) (1 pour bdfhkl).

En résumé :

- *l'italique évoque la Renaissance italienne*. Utilisation : mettre en exergue, dans les citations, termes techniques.
- le Romain évoque l'antiquité.

L'italique n'est pas un romain penché !!! L'italique s'utilise aussi dans les bibliographies : on peut mettre en italique les noms d'œuvres (les livres, journaux, rapports).

Les symboles des chiffres et de la ponctuations sont de deux sortes : alignés ou non, majuscules ou minuscules (elzéviriens) :

- les chiffres alignés sont utilisés dans les entêtes de lettre pour aligner avec les intitulés en lettres capitales
- les elzéviriens pour aligner sur les énoncés en lettres minuscules ;

Les ponctuations sont Re-dessinées pour chaque police

- sans sérif :;,!?''/$# % « »
- romain :;,!?"/$# % «»

Une classification des caractères se fait en cinq familles : Maximilien Vox (1950), Thibaudeau (1920-24), Novarese (1958), Alessandrini (1979), Bertin (1980). La classification VOX se présente quant à elle avec les familles suivantes :

Vox ATYPI	Humanes	Linéales
	Garaldes	Incises
	Réales	Scriptes
	Didones	Manuaires
	Mécanes	

Voici des exemples de familles et des instanciations.

Humanes : Palatino (H. Zapf, 1951)
Garaldes : Bembo
Réales : Baskerville, Times New Roman
Linéales : Helvetica (Max Meidinger, 1957), Univers (Adrian Frutiger, 1956), FUTURA (Paul Renner, 1928), ITC Avant Garde (Herb Lubalin, 1970)

Il n'est pas toujours facile même pour un spécialiste d'affecter un caractère à sa famille. Les mécanes, Linéales, Manuaires et Incises servent de caractères de titrage. Les scriptes pour la publicité.

Voici maintenant des termes techniques qui sont définis et qui correspondent à des propriétés des caractères.

Corps

Historiquement, c'est la hauteur de la ligne de plomb qui portait le caractère – assimilé à sa *taille* ou encore à son *œil*. Il est donné en point : 1 point $= 0,376\text{mm}$

en Europe et $= 0,351mm$ aux US. Son multiple en Europe : *cicéro* $= 12pts$, aux US 12 points $= 1pica$.

Œil

d'un caractère bas de casse : hauteur du a, e, x (pas ascendantes ni de descendantes). Au sein d'un même corps, l'œil est plus ou moins important.

Graisse

caractérise l'épaisseur du trait (maigre, normal, gras, extra-gras)

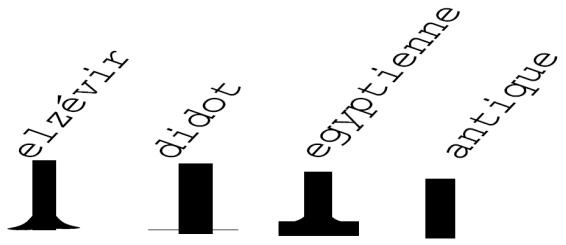
Chasse

largeur de l'œil d'un caractère. Elle est fonction :

- de la lettre (m versus n)
- du corps
- du dessin des caractères (serré ou étroit)

Empattement

C'est un ornement ajouté à la base des jambages ou des obliques (un des critères de classification). Voici les différents types d'empattement.



Sur des transparents, nous vous conseillons d'éviter d'utiliser des polices avec des empattements. Par exemple, les copies de transparents que vous trouverez en annexe sont composés en Times ce qui est un peu maladroit puisque le times est une police de journal (The Times en l'occurrence) et donc ce n'est pas approprié.

Résumé

La *chasse* = largeur visible du caractère.

L'*encombrement* = chasse + blancs de part et d'autre du caractère.

L'*approche* = blanc entre lettres.

L'*espace* = blanc entre les mots

Le *cadratin* = blanc dont la largeur est égale au corps. C'est généralement l'unité pour les rentrées de paragraphes.

Lisibilité

Première loi : l'œil occidental lit haut en bas et de gauche à droite. Ainsi, le titre de chapitre doit se situer en haut de page et il faut aligner les titres sur la gauche (éventuellement au centre).

Deuxième loi : cet œil est attiré par ce qui est visuellement gros ! (rôle de la force, de la couleur et du contraste que l'on contrebalance par le blanc qui les entoure).

Troisième loi : cet œil ne supporte pas la profusion de composants typographiques (unité dans la page et dans l'ouvrage). Il ne faudrait pas changer de police dans un même document. Dans les courbes de résultats en couleur, il ne faudrait pas utiliser plus de 5 couleurs.

Quatrième loi : cet œil recherche certaines informations comme la numérotation à droite des pages de droite (pages impaires) et à la rigueur à gauche des pages de gauche (ou paires).

Cinquième loi : une illustration en couleur attire plus l'attention qu'une illustration en noir.

On peut pratiquer l'habillage du texte (encadrer sur un bloc de texte). Pour marquer les séparations entre paragraphes d'un même bloc typographique on utilise des blancs verticaux : il s'agit alors d'un *interlignage* de la valeur d'une ligne ou d'une demi ligne.

Le début du paragraphe suivant peut être repéré par une entrée de type : un numéro ou un renfoncement.

En typographie française on retrouve bien souvent un blanc en début de ligne d'un «premier paragraphe»(style anglo américain) : il n'y a pas de raison de le faire !

Le fin du fin

Le *débord* consiste à déplacer du texte dans la marge : c'est le cas des titres de paragraphes, ce qui n'est pas pratiqué dans le document présent. C'est le meilleur procédé pour une lecture de recherche. Recommandée pour la composition des bibliographies et des glossaires.

2.3.1 Les espaces entre les ponctuations du français

Les règles pour répartir le blanc entre les symboles de ponctuation sont les suivantes :

AVANT		APRèS
pas de blanc	,	espace justifiante
pas de blanc	.	espace justifiante
espace fine insécable	;	espace justifiante
espace fine insécable	!	espace justifiante
espace fine insécable	?	espace justifiante
espace mots insécable	:	espace justifiante
espace justifiante	–	espace justifiante
espace justifiante	«	espace mots insécable
espace mots insécable	»	espace justifiante
espace justifiante	(	pas de blanc
pas de blanc	)	espace justifiante
espace justifiante	[	pas de blanc
pas de blanc	]	espace justifiante

Une espace insécable signifie qu'on ne peut pas couper à cet endroit, une espace justifiante est un blanc qui peut «s'étendre ou s'allonger», un espace mot est une

espace de largeur déterminé selon la fonte. Le *cadratin* est une espace de valeur déterminée.

2.3.2 Les espaces entre les ponctuations de l'anglais

Rappel : il n'y a pas de lettres accentuées en anglais... le I n'est jamais en minuscule. En français, on met un cadratin après un . ? ! terminant une phrase. Il n'y a aucune espace qui sépare le tiret long du mot ou du signe qui le précède ou le suit, à l'intérieur d'une phrase.

Les guillemets anglais sont " et " alors que les guillemets du français sont «et ».

Les noms des jours, des mois, les adjectifs de nationalité, les titres de civilité pas de capitale initiale (12 janvier, français, maître de conférences).

2.3.3 Abréviations conventionnelles

En français nous avons par exemple av. av.J.-C. c.-à.-d cf. etc. ex. id. i. e. vol. p. art.

M. (et non Mr) MM. – M^{lle}, M^{lles}, M^{me}, M^{mes}, n

En anglais : st (Saint) St. (street) Co. No. et no Dr Mr Mrs i. e., (for example) e. g., (that is) cf. (meaning : compare) donc, cf. ne doit pas être utilisé à la place de "see".

Voici d'autres symboles du français : 1er, 2^{e}, 3^{e}

Autres symboles de l'anglais : 1st, 2nd, 3rd, 4th, 5th.... dans les nombres, une virgule sépare les tranches de trois chiffres et un point sépare les unités des décimales.

2.3.4 Les signes de ponctuation

Rappels : ils contribuent à la logique du discours.

- Le point termine une phrase. Confondu avec les ... Supprimer le dans les titres ;
- Le ? termine une phrase interrogative. On le garde dans les titres ;
- Le ! peut être gardé dans les titres centrés ;
- La virgule sépare sujets, compléments, épithètes, attributs et propositions de même nature non unis par une conjonction de coordination ;
  - Deux «ni»peu éloignés ne sont pas séparés par ,
  - pas de , avant une (, –, [à moins que le crochet annonce une restitution ;
  - etc précédé par une virgule ;

- Le ; s'emploie pour séparer dans une phrase les parties dont une au moins est déjà subdivisée par la virgule ou pour séparer des propositions «longues»– Dans une liste ;
- Les : introduisent une explication, une citation ou un discours.

2.3.5 La coupure des mots du français

On dit encore la *césure* et elle doit être évitée autant que possible. Les règles de césure sont les suivantes :

- pour les mots simples (deux syllabes) : pas de difficulté. Pour les mots composés, la division devra tenir compte de l'étymologie ;
- la division étymologique n'exclut pas la coupure syllabique ;
- la division d'après la prononciation est la seule admise si la coupure étymologique entraîne un changement de prononciation.

Enfin,

- Les coupures isolant une seule lettre sont à proscrire ;
- de même pour les coupures de début et de fin qui isolent deux lettres ;
- Les mots composés sont coupés au – ; on évitera de couper le dernier mot d'une page impaire ; on ne coupera pas un mot après l'apostrophe.
- la coupure des mots étrangers se fera selon la langue étrangère.

2.3.6 Les notes (en bas de page)

Ce sont des commentaires explicatifs[3]. Leur caractère accessoire justifie leur composition dans un corps inférieur. Emplacement : en général en bas de page ou encore en fin de chapitre (sur deux colonnes). Dans un tableau, la note se trouve à l'intérieur du cadre. La note[4] est séparée soit par une ligne de blancs, un (amorce) filet maigre. La note est numérotée.

2.3.7 Les titres

Il est, dans la mesure du possible, informatif et concis. Pour faire un effet de style vous pouvez faire un titre phase avec un sujet suivi d'un verbe suivi d'un complément.

Voici la hiérarchie des titres utilisables : tome ou volume, livre, partie, titre, sous-titre, chapitre, sous-chapitre, section, sous-section, article, paragraphe, alinéa.

On utilise le système numérique international suivant :

[3] Ceci est un renvoi en bas de page

[4] Ceci est un deuxième renvoi en bas de page

```
1.
1.1.
1.1.1.
1.2.
1.2.1.
```

2.3.8 Les majuscules

Seule la première lettre d'un titre prend une majuscule. Il n'y a pas d'article défini en début ni de point en fin de titre. Concernant l'utilisation des majuscules, il y a beaucoup de cas d'espèce !!! Pas de majuscules pour :

- les organismes qui ne sont pas uniques : l'université de Picardie ;
- les noms de jours et de mois ;
- les titres et qualités s'écrivent avec une minuscule : le président de la République, le pape, l'ayatollah Rominé, le général Lebol, le ministre de l'éducation nationale.

2.3.9 Les sigles

Les tendances actuelles sont les suivantes :

- plus de points dans les sigles ;
- majuscule sur la première lettre d'un sigle lorsqu'il est prononçable ;
- en petite capitale un sigle que l'on épelle : Lipn, SNCF

2.3.10 Les listes

En gros, il y a deux classes de listes. Il y a celles qui font partie d'une phrase unique et celles qui sont composées de plusieurs phrases :

1. les éléments d'une liste commencent par une minuscule et se terminent par un ; sauf le dernier élément, s'il termine la phrase, prend un point ;
2. les éléments de liste formés de plusieurs phrases se comportent comme des phrases.

2.3.11 Format

Examinons maintenant quelques format usuels. Les pages possèdent les caractéristiques suivantes :

- format A4 (21cm $\times$ 29,7cm) ;
- largeur des textes (ou justification) : 16cm (2cm de marge, et 1cm de reliure) ;

- hauteur des textes, y compris les notes : 23cm (2,5cm de marge haute et 2cm de marge basse) ; 1ère page de : 36pts d'espacement avant le titre ;
- Tous les textes sont justifiés.

2.3.12 Les alinéas

Ils sont d'un renfoncement (ou retrait de 1ère ligne) de 5 mm et utilisent le tiret (et non le point comme en anglais).

2.3.13 Figures et Tables

Les figures et illustrations peuvent être fournies à part ; elles doivent être bonnes pour la reproduction. Les copies d'écran sur fond blanc sont préférables. Les figures sont numérotées de 1 à n à l'intérieur du document. Elles sont nécessairement accompagnées de légendes explicites qui sont de véritables commentaires et toutes figures sont référencées dans le texte. Elles se présentent comme suit :

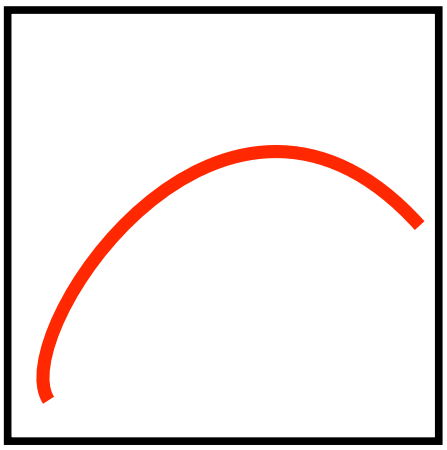


FIGURE 2.1: Une Figure animée en PostScript

TABLE 2.1: Exemple de légende

2.3.14 Composition des mathématiques

Les lettres majuscules sont toujours composées en romain. Par contre les minuscules représentent les variables ou inconnues, les fonctions, les constantes littérales, les paramètres entiers (i, j, k) sont en *italique*.

Sauf dans :

$$\sin(a+b) = \sin a \cos b + \sin b \cos a$$

et e (base des log) et i (complexes) sont en romain !!! ainsi que les constantes fondamentales de la physique et de la chimie.

Des lettres capitales sont ajourées ou grasses sont employées pour désigner certain ensembles :

$$\mathbb{NQRZ} \quad \text{et} \quad \mathbf{NQRZ}$$

Les caractères d'anglaise et de ronde permettent d'éviter des notations semblables dans un même texte :

P représente la pression $\mathcal{P}$: probabilité

C représente un point $\mathcal{C}$: une courbe

Il y a des blancs plus ou moins importants entre les symboles (avant et après) unaires et binaires.

On peut définir ses propres symboles :

$$\boxplus \rtimes \leqslant \circledcirc \ltimes \hbar \complement \eth \angle$$

Quelques symboles particuliers

Composés en romain :

e	base des logarithmes népérien
i	base des nombres complexes
$\mathrm{N}, \mathrm{N_A}, \mathcal{N} = 6,02.10^{23}\,\mathrm{mol}^{-1}$	Nombre d'Avogadro
$\mathrm{h} = 6,63.10^{-34}\mathrm{J.s}$	Constante de Planck
$\mathrm{R} = 8,31\mathrm{J.K}^{-1}.\,\mathrm{mol}^{-1}$	Constante des gaz parfaits
$\epsilon_0 = 8,854.10^{-12}\mathrm{F.m}^{-1}$	permitivité du vide
$\mathcal{F} = 9,65.10^{3}\mathrm{C}$	Constante de Faraday

Récréation :

```
\def\mol{\mathrm{mol}}
\def\mol1{\rmfamily{mol}}

${\cal N}=6,02.10^{23}\, \mol^{-1}$
${\cal N}=6,02.10^{23}\mol1^{-1}$
```

$\mathcal{N} = 6,02.10^{23}\,\mathrm{mol}^{-1}$ et $\mathcal{N} = 6,02.10^{23}\mathrm{mol}^{-1}$

Alphabet grec

En principe, seules les lettres capitales qui ne présentent pas de ressemblance avec des latines sont utilisées :

$$\Gamma, \Delta, \Theta, \Lambda, \Sigma, \Pi, \Phi, \Omega, \Psi, \Upsilon, \Xi$$

Idem pour les minuscules grec mais il reste des risques de confusion : ϵ et $\in$, ζ et ξ, υ, ν et ν, ρ et p, θ et Θ, δ et ∂, n et $\eta \ldots$

Exposants et indices

Ils sont soit littéraux (i^n), soit numérique (i^2). Les exposants littéraux sont en italique. Cas des indices littéraux : en italique sauf «s'ils représentent des abréviations ou des repères destinés à différencier des grandeurs du même ordre »(Code Typographique). Lorsque la lettre en indice est l'abréviation d'un nom propre elle est en capitale.

Composition des formules

Il faut utiliser 0 (zéro) et pas le symbole O (la lettre o majuscule)!

Il convient d'aligner les exposants et les indices : A_2^2

Le produit de facteurs sont composés collés : $ax^2 + bx + c = 0$

La ponctuation qui suit une expression est composée en romain.

Les (), [] et {} (dans cet ordre de choix) refermant que des termes «simples»doivent être du même corps : $[n - (2p + 1)(2p + 2)]$,

MAIS

$$y = a\left[\left(x + \frac{b}{2a}\right)^2 - \frac{b^2 - 4ac}{4a^2}\right]$$

(remarquer la hauteur des divers éléments qu'ils réunissent !!!)

Pour faciliter la lecture, les formules mathématiques seront le plus souvent sorties du texte et centrées. Lorsqu'une formule est longue $\leftrightarrow$ éviter de couper dans l'intérieur des (), [] et {} ...les lignes doublantes doivent commencer par un signe opératoire ou relationnel (première ligne justifiée à gauche, seconde ligne justifiée à droite).

Les minuscules peuvent aussi représenter des unités de mesure et dans ce cas elles sont en romain : 15g et g (accélération).

De même : $\cos(x), \sin(x), \log_2(16)$ sont en romain !
$cos(x), sin(x), log_2(16) =$ NON !!!

Bad:	If $x > 1 f(x) < 0$
Fair:	If $x > 1, f(x) < 0$
Good:	If $x > 1$ then $f(x) < 0$
Bad:	Since $p^{-1} + q^{-1} = 1, \| \cdot \|_p \ldots$
Good:	Since $p^{-1} + q^{-1} = 1$, the norm $\| \cdot \|_p \ldots$
Bad:	It suffices to show that $H_p = n^{1/p}, 1 \leq p \leq 2$
Good:	It suffices to show that $H_p = n^{1/p}$ for $1 \leq p \leq 2$
Good:	It suffices to show that $H_p = n^{1/p}(1 \leq p \leq 2)$
Bad:	For $n = r$ (2.2) holds with ...
Good:	For $n = r$, (2.2) holds with ...
Good:	For $n = r$ inequality (2.2) holds with ...
Bad:	Let the Schur decomposition of A be QTQ^*
Good:	Let a Schur decomposition of A be QTQ^*

Bibliographie

Les références sont rassemblées en fin de document ; leur numéro, du type [1], [2] etc est placé entre crochets dans le texte. Noter qu'en français les noms d'auteurs peuvent être typographiés en petites capitales. Les noms de livres ou les titres peuvent être typographiés en italique : voir les instructions dans la référence [3] pour les détails. L'élément principal est de conserver les mêmes règles de présentation tout au long de la bibliographie.

Voici des exemples.

[1] Brassard (Gilles) et Bratley (Paul). – Algorithmics - theory & practice. – Prentice Hall, 1988.

[2] Ward (A.C.) et al. – Extending the constraints propagation of intervals. – Proceeding of the 11th International Joint Conference on AI, p. 1453–1458, 1989.

[3] Lexiques des règles typographiques en usage à l'imprimerie nationale, 1990, ISBN 2-11-081075-0

[4] University of Chicago Press, The Chicago Manual of Style, thirteen edition, Chicago and London, 1982 –ISBN 0-226-10390-0

[5] Oxford University Press, The Oxford English Dictionary, second edition, 1989

[6] François Richardeau – Manuel de typographie et de mise en page – Retz, 1989.

[7] Fernand Baudin – La typographie au tableau noir – Retz, 1984

[8] Les sites sur TEXet LATEX: http://www.gutenberg.eu.org, http://tex.loria.fr/ (voir aussi http://tex.loria.fr/general.html) et http://yann.lecacheux.free.fr/LaTeX/sommaire_latex.htm. Voir aussi le site http://fr.wikibooks.org/wiki/Programmation_LaTeX/Vade_mecum

2.4 L'expérimentation

L'INFORMATIQUE en tant que discipline scientifique à part entière repose elle aussi sur quatre piliers : la modélisation mathématique, la simulation, l'émulation et la confrontation avec le monde réel. Nous nous concentrons ici sur les trois thématiques de la modélisation, de la simulation et de l'émulation. Pour les premiers cycles universitaires, la modélisation peut s'aborder aussi par exemple par l'étude de la logique, de la spéficication et de la résolution de problèmes concrêts faisant ressortir une rigueur d'analyse. Nous y reviendrons ultérieurement dans un chapitre spécifique. La partie de confrontation avec le monde réel renvoie à la connaissance du monde qui nous entoure. Dans tous les cas, l'ensemble des activités nécessaires à de bonnes pratiques d'expérimentation requiert des compétences multiples, souvent disponibles dans des équipes de personnes plutôt que chez une seule et même personne.

Nous pensons par ailleurs que l'informatique, caractérisée par un rapport aux machines, glisse progressivement vers une discipline ou la simulation et l'approche

expérimentale doit prendre de plus en plus de place, y compris dans les enseignements. Cela pose également pour la recherche et l'industrie des problèmes de construction de grands instruments comme la Physique le fait depuis bien longtemps. En effet, les problèmes, par exemple numériques, auquels nous sommes confrontés requiert de plus en plus de ressources (processeurs, mémoires, disques, réseaux) alors qu'Internet permet potentiellement de connecter des millions d'ordinateurs qui peuvent jouer tantôt le rôle de clients (demande service) tantôt le rôle de serveurs c.à.d offrant un service.

2.4.1 Introduction

Dans cette première partie, nous adoptons une démarche simple, spontanée une nouvelle fois qui part presque d'évidences. Le deuxième paragraphe commencera à «théoriser»un peu les notions (les quatre piliers) essentielles.

Ici, il s'agit donc de conseiller et de discuter de la méthodologie de l'expérimentation, de la bonne manière de commenter des courbes, des tableaux de résultats. . . et de les présenter convenablement. Par exemple, le point à remarquer dans le tableau de résultat doit *sauter* aux yeux.

Les expérimentations sont là pour vérifier les hypothèses. Une hypothèse est une idée, une suggestion qui concerne la nature des choses. Tant qu'elle n'a pas été *testée*, une hypothèse ne doit pas être confondue avec une *loi*.

Les hypothèses doivent être suffisamment détaillées. Par exemple, supposez que vous avez construit deux structures de données A et B. Si vous dites que votre test d'hypothèses est simplement que A est supérieur à B alors ceci est incorrect. Par exemple, si A est la structure de données des listes chaînées et B celle des tableaux, il est bien évident que l'une des deux n'est pas supérieure à l'autre, sinon nous n'enseignerons que la meilleure. Il faut alors parler du contexte d'utilisation.

Donc, notre hypothèse à tester pour notre exemple doit se compléter, par exemple de la façon suivante :

- Comme structure de données pour rechercher en mémoire, A est plus rapide et plus compacte que B ;

Cette première formulation peut se compléter par d'autres détails, par exemple :

- Par ailleurs, nous nous assurons qu'il y a une majorité d'accès à la mémoire qui provoque un défaut de cache (ou qui ne provoque pas de défaut de cache, c'est selon) ;
- Nous supposons que la majorité des accès s'effectue sur une petite proportion des données.

En informatique, une implémentation testée sur un jeu de données sert non seulement à vérifier des hypothèses mais aussi que l'implémentation remplit bien la tâche demandée *lorsque* le système dispose de ressources (RAM, disque, réseaux) appropriées.

L'expérimentation et son commentaire est une discipline difficile à mettre en œuvre. Combien de fois nous avons pu observer des mesures expérimentales sur des jeux de données de taille inadaptée ? Par exemple, dans la construction d'un simulateur de cache, tous les phénomènes physiques importants que l'expérimentation est censée révélée ne sont pas captés si la taille du problème tient dans le cache et qu'il n'y a donc pas d'interaction avec la mémoire centrale (RAM) !

De même un commentaire de courbe n'est pas simplement la mise en français de ce que l'on voit sur la courbe mais il s'agit d'apporter des explications et de faire des liens avec les hypothèses théoriques du modèle et/ou les conditions expérimentales et/ou avec la machine sur laquelle on expérimente.

Quelque fois on a bien une idée intuitive de ce qui doit ressortir mais une présentation 2-D n'est peut être pas suffisante. Essayez alors une présentation en 3-D en incluant dans votre discours un paramètre supplémentaire. Construisez alors des graphes en 3-D.

Cette idée n'est pas forcément payante à tous les coups : vous n'allez peut être pas réussir à montrer des relations particulières entre les paramètres autres que celles attendues mais peut être que vous allez faire apparaître de nouveaux phénomènes «physiques»qui vous permettrons d'explorer d'autres formes de commentaires.

Vous serez alors en mesure de dire en conclusion (par exemple) : «Les expérimentations des algorithmes naïfs et optimaux étudiés mesurent les variations de X en fonction de Y. Les comportements obtenus étaient attendus pour notre plateforme. Cependant, en considérant aussi Z, nous avons observé que la convergence de X vers le bord de la grille se faisait en suivant tel phénomène. Nous pensons que cela est dû à tel phénomène. Par ailleurs, il y a là matière à explorer. En effet, si l'on arrive à corriger, réduire telle chose, alors il se passera telle autre chose et nous obtiendrons de la sorte un gain au niveau de. . . »En rédigeant de cette façon vous avez soigné la chute et on peut estimer les perspectives du travail.

Discutez des options de compilation qui ont un impact significatif sur les performances. Présentez complètement la machine, le compilateur, le système d'exploitation utilisés.

Si vous mesurez des temps d'exécution, il faut dire un mot avec quels outils cela est fait et il faut faire des moyennes et des calculs d'écart type ou de variance. Le choix d'un générateur de nombre aléatoire est important : ne vous contentez pas d'utiliser la fonction *random* de votre outils car il y a de nombreuses bibliothèques de qualité disponibles sur le WEB. Renseignez-vous. L'idée étant de construire un test équitable et pas fait pour vérifier les hypothèses.

Êtes vous sur que la machine sur laquelle vous avez expérimenté vous était complètement dédiée et qu'il n'y avait pas d'autres travaux occupant le CPU qui tournaient concurremment avec vos tests ?

Les jeux d'essais, comme les étudiants les appellent fréquemment, ne sont pas donnés pour le plaisir. Ils doivent révélés que vous traitez convenablement dans votre code telle ou telle difficulté. Ils ne prouvent pas que votre code est correct mais sont des illustrations de points particuliers de votre travail. Il est donc im-

portant de bien les choisir. Ne cherchez pas à donner une liste exhaustive de jeux d'essais. Sélectionnez ceux qui montrent que vous résolvez une difficultés importante, majeure.

Vérifiez également que l'interprétation des résultats que vous faites est la bonne. Par exemple, si vous comparez deux implémentations A et B et que A s'exécute plus rapidement que B, est-ce que cela est dû au fait que A utilise moins d'instructions que B ou alors que B exécute plus d'instructions que A mais que les instructions «coûtent moins chères»? Attention aux évidences !

2.4.2 Prenons un peu de recul vis à vis de l'expérimentation

Quand un concepteur de solutions informatiques se pose des questions du genre :

- quel algorithme est le meilleur pour cette application sur cette architecture matérielle ?
- quelle conception est la meilleure pour implémenter tel ou tel service ?
- quelle stratégie de placement des données est la meilleure pour faire en sorte que leur analyse soit facilité ?
- est-ce que le gestionnaire de service X se comporte bien vis à vis de la propriété (comportementale) Y ?
- ...

il peut se référer à une analyse fondée sur un modèle mathématique. Ceci lui permet généralement de prouver des théorèmes interessants mais l'approche peut être trop simple pour convaincre le praticien. Généralement cette approche est utile pour comprendre des grands principes sans que cela puisse s'exploiter directement ou simplement dans les programmes.

Une approche expérimentale est alors nécessaire. Vous pouvez alors commencer à recruter des collaborateurs pour qu'ils vous fasse une synthèse des outils standardisés, des outils acceptés par la profession, des outils émergents. Vous pouvez aussi leur demander quelle méthodologie expérimentale utiliser dans votre contexte d'exploitation sachant que vous voulez juste vérifier une et une seule propriété.

Votre expérience et votre éducation vous ont appris depuis longtemps que l'expérimentation permet de convaincre le praticien ou l'utilisateur et que, par dessus tout, elle démontre que l'approche proposée peut être implémentée en pratique, dans le monde réel.

Cependant, l'expérimentation est une activité consommatrice de temps et souvent elle constitue une activité laborieuse : l'application dans son entier doit être construite et être fonctionnelle. Ceci inclue l'élaboration des phases de conception, de choix d'algorithmes, de codage ainsi que le déploiement de l'application et des données, la configuration de la plateforme et des tests.

Sur ces derniers points, si vous construisez un outil qui utilise Internet comme mode d'échange des données, comment pouvez-vous dire que votre système est

performant (vis à vis des temps de réponse à une requête) sachant que vous ne contrôlez pas les débits sur Internet, ni le routage des paquets, ni les pannes de transmission sur les liens ?

D'un point de vue scientifique, si vous ne maîtrisez pas votre environnement vous n'allez pas pouvoir reproduire les résultats expérimentaux et donc personne ne pourra avoir confiance dans vos résultats. N'oubliez pas que la reproductibilité des faits caractérise la Science.

Plaçons nous tout de même dans un contexte d'exploitation où vous contrôlez peu de chose. Plutôt que de tout vouloir tester, vous construisez un banc d'essais. Les expérimentations sont limitées à votre banc d'essais. Vous les lancez et vous observez les résultats. Plusieurs questions se posent alors et les approches possibles sont discutées dans les paragraphes qui suivent :

- quelle est la part des résultats dûe au banc d'essai seul ?
- est-ce que je peux extrapoler les résultats ? Si oui, sont-ils convainquants ? Puis-je imaginer d'autres scénari expérimentaux du type «qu'est ce qui se passerait si on disposait de. . . » ?
- est-ce qu'une autre personne que moi peut reproduire les résultats ?

2.4.3 La modélisation mathématique

Exemple introductif

Ce paragraphe illustre l'idée de modélisation mathématique appliquée à un premier problème informatique. Supposez que vous vouliez échanger un fichier entre une machine A et une machine B reliées par un réseau informatique longue distance comme Internet par exemple. Vous savez que pour arriver à B il vous faut traverser différents équipements (routeurs) et que le débit entre deux routeurs n'est pas forcément le même. Par exemple le débit entre chez vous et votre fournisseur Internet (C) n'est pas le même que celui entre le fournisseur Internet de votre correspondant (D) et sa machine. La situation est décrite par $A \rightarrow C \rightarrow D \rightarrow B$

Oublions dans la suite cette dernière contrainte quant aux débits distincts. Considérons que les débits de A vers C, de D vers C et de C vers B soient égaux. Il faut par exemple 10s pour transférer un bloc de taille T de A vers C. Votre fichier à échanger est constitué de 4 blocs de taille T.

Pour le faire passer de A à B, le premier bloc a besoin de $3 * 10 = 30$s pour arriver ; on peut ensuite envoyer le deuxième bloc ce qui prend aussi 30s, puis le bloc 3 et enfn le bloc 4. Au total, l'envoi a consommé $4 * 30 = 120$s.

On peut envisager une autre solution. On n'attend plus qu'un bloc soit arrivé à B pour transmettre le suivant, mais ici, dès qu'un bloc a été traité par un routeur, alors il reçoit un autre paquet. Cette stratégie s'appelle le pipeline.

On peut alors vérifier qu'au moyen de cette statégie, quand le site C envoie son premier paquet, le site A lui envoie un deuxième paquet.

Dans un système parfait et sans pipeline, pour transférer un fichier de taille M à N nœuds via un réseau de vitesse S, le temps total est alors :

$$T_{np} = \frac{M}{S} * N \tag{2.1}$$

alors que celui pour un réseau à P niveaux, le temps total d'exécution) est :

$$T_p = \frac{M}{S * P} * (N + P - 1) \tag{2.2}$$

On remarque alors que plus P est élevé (on intercale beaucoup de sites dans le pipeline), plus T_p sera petit et meilleures seront les performances. Cependant, le traitement d'un bloc se fait avec un surcoût, par exemple celui de copier le message en entrée vers la carte réseau en sortie. Supposons que le système soit homogène (l'architecture est partout la même) et définissons δ comme étant le surcoût. Les équations 2.1 et 2.2 deviennent :

$$T^*_{np} = \left(\frac{M}{S} + \delta\right) * N \tag{2.3}$$

$$T^*_p = \left(\frac{M}{S * P} + \delta\right) * (N + P - 1) \tag{2.4}$$

Exercice 1 (*Application numérique*) : À partir de l'équation 2.4, tracer les courbes du temps d'exécution en fonction de P (nombre d'étages dans le pipeline) et pour $M = 100\text{Mo}, S = 10\text{Mbps}, N = 4, \delta = 850\text{ms}$. Tracer sur le même graphique la courbe correspondante à l'équation 2.3. Déterminer graphiquement la valeur de P qui correspond au temps minimal d'exécution et celui qui fait que le temps d'exécution avec un pipeline devient plus important que celui sans l'utilisation d'un pipeline.

Note : vous pouvez également vous aider de la dérivée pour calculer de manière exacte les points recherchés à l'exercice précédent.

Transfert d'information

Supposons que nous ayons à transférer des données d'un processeur de stockage (nœud maître) à deux processeurs de calcul (nœuds esclaves) qui trient les données reçues.

Les processors esclaves i sont caractérisés par :

1. une vitesse de téléchargement u_i;
2. une puissance de calcul k_i (établie par rapport à un processeur de référence)

Le processeur i reçoit une proportion p_i de données. La somme des p_i est égale à D. Le problème ici est de déterminer les p_i c'est à dire ce que chaque processeur doit avoir comme données. On veut par ailleurs que les deux processeurs terminent leur travail de tri en même temps. On sait par ailleurs qu'il existe des algorithmes de tri séquentiels sur n données qui coûtent $n \log n$ opérations élémentaires.

Dans notre cas, le temps d'exécution est donné par :

$$\max_i\{u_i p_i D + k_i p_i D \log(p_i D)\}$$

ce qui correspond au temps de transfert plus le temps de tri. On montre que le temps d'exécution est optimal lorque ce nombre ne dépend pas de i c'est à dire lorsque les processeurs terminent en même temps. On obtient alors les équations suivantes :

$$\sum_i p_i = 1 \tag{2.5}$$

et

$$\forall i > 1,\ u_i p_i D + k_i p_i D \log(p_i D) = u_1 p_1 D + k_1 p_1 D \log(p_1 D) \tag{2.6}$$

Considérons le cas suivant à deux processeurs : $p = 2$, $k_1 = 1$, $k_2 = 4$, $u_1 = 10$ et $u_2 = 1$.

Les équations à résoudre sont alors $p_1 + p_2 = 1$ et $10p_1 D + p_1 D \log(p_1 D) = p_2 D + 4p_2 D \log(p_2 D)$.

En remplaçant p_1 par x et p_2 par $1 - x$ on trouve (quand on divise par D):

$$10x + x\log(xD) = (1-x) + 4(1-x)\log((1-x)D)$$

Pour résoudre cette équation, on utilise un développement limité à l'ordre 1 de la fonction logarithme comme suit : $\log x = x-1+o(x-1)$ et $\log(1-x) = -x+o(x)$. Cette manipulation conduit à une appoximation du résultat mais nous permet de résoudre le système d'équations. On obtient alors :

$$10x + x\,[(x-1) + \log(D)] = (1-x) + 4(1-x)\,[-x + \log(D)]$$

On résoud cette équation du second degré sans problème. L'équation a une unique solution entre 0 et 1 et nous obtenons, pour différentes valeurs de D et les valeurs numériques précédentes :

D	p_1
16M	0.7103%
32M	0.7136%
64M	0.7166%
128M	0.7194%
256M	0.7220%
512M	0.7245%
1024M	0.7268%

On observe que la quantité de données qui doit résider sur le processeur le plus lent est importante (72% de la taille totale) et elle reste à peu près constante entre 16M et 1024M. Cela signifie aussi qu'une bande passante de réseau dix fois plus importante pour le second processeur n'est pas suffisante pour contre-balancer le coût du tri.

Maintenant, si on choisit $p = 2$, $k_1 = 1$, $k_2 = 4$, $u_1 = 100$ et $u_2 = 1$ (le réseau est dix fois plus rapide pour P_1 comparé au cas précédent et 100 fois plus rapide que pour l'autre processeur), on obtient les résultats suivants :

D	p_1
16M	0.3623%
32M	0.3704%
64M	0.3782%
128M	0.3857%
256M	0.3929%
512M	0.3999%
1024M	0.4066%

Ici nous concluons que la vitesse de transmission du processeur P_1 l'autorise à contre-balancer sa vitesse de calcul inférieure à celle de P_2.

Modélisation via des automates

Nous présentons dans ce paragraphe un exemple de modélisation via un automate. La formalisation de la notion d'automate est sans aucun doute la théorie la plus aboutie des informaticiens car elle permet de capter de nombreux cas pratiques indépendamment du champs disciplinaire ; elle a aussi des liens profond avec la discipline de la compilation qui vise la translation d'un langage vers un autre langage plus proche de la machine. L'exemple qui suit, qui de prime abord n'évoque pas un problème informatique est cependant lié à une situation concrète. La démarche de modélisation doit ici être comprise comme étant la démarche permettant de s'abstraire des détails d'implémentation pour se concentrer sur l'essence des règles régissant le fonctionnement du jeu.

Un barman aveugle aime faire un tour de magie à sa clientèle. Il dispose quatre verres sur un plateau rond, aux sommets d'un carré inscrit. Les verres sont chacun à priori dans une situation quelconque (c'est-à-dire qu'ils peuvent être à l'endroit ou a l'envers). Le barman porte des gants de boxe, si bien que lorsqu'il saisit un verre (ou deux verres), il ne sait pas si celui-ci (ou ceux-ci) est (ou sont) à l'endroit ou à l'envers. Le but est d'arriver à une situation dans laquelle les quatre verres sont à l'endroit ou les quatre verres sont à l'envers.

Au début les verres sont donc dans une situation quelconque et il peut retourner un verre ou deux verres. Mais à chaque fois qu'il a effectué un mouvement (c'est-à-dire à chaque fois qu'il a retourné un verre ou deux verres) quelqu'un tourne le plateau d'un angle $\frac{k\pi}{2}$ ($k \in \mathbb{N}$). Quand les quatre sont dans la même position, les clients du bar applaudissent à tout rompre et le barman sait ainsi qu'il a terminé.

Comment s'y prend-il ?

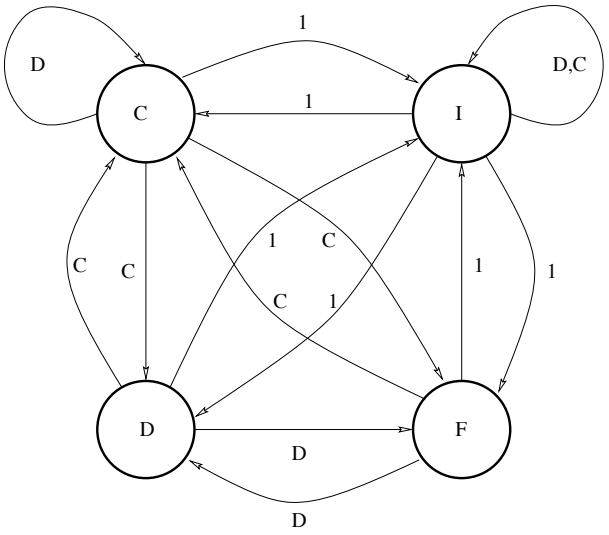
Le plateau peut être dans quatre états possibles : il peut être dans un état final (les quatre verres sont dans la même position, nous nommerons cet état du plateau $\mathcal{F}$), il peut y avoir deux verres à l'endroit et deux à l'envers, cette situation se divisant en deux états : les deux verres dans la même position sont côte-à-côte (nous nommerons cet état $\mathcal{C}$), les deux verres dans la même position peuvent être sur la même diagonale (nous nommerons cet état du plateau $\mathcal{D}$) et enfin il peut y avoir un verre dans une position et les trois autres dans l'autre (nous nommerons cet état $\mathcal{I}$, pour "impair").

Remarquons déjà que cette représentation est indépendante de l'orientation du plateau.

Le barman peut retourner à chaque mouvement un verre ou deux verres. S'il retourne un verre le plateau va passer d'un état à un autre. De même s'il retourne deux verres. Mais il convient là aussi de distinguer deux façons possibles de retourner deux verres : il peut retourner deux verres qui se trouvent côte-à-côte ou retourner deux verres qui se trouvent sur une diagonale. Nous nommerons ces mouvement 1, D et C. (1 : le barman retourne un verre, D il retourne deux verres se trouvant sur une diagonale et C il retourne deux verres qui se trouvent côte-à-côte.)

Nous allons figurer les quatre états possibles du plateau par quatre cercles contenant chacun le nom de son état puis nous allons les relier par trois séries de flèches, un pour chaque façon possible de retourner un ou deux verres, nous verrons ainsi comment chaque mouvement fait passer d'un état à l'autre du plateau.

Nous obtenons ainsi le diagramme suivant :



Ce que "voit" le barman ...

L'objet représenté ci-dessus *est* un automate. Nous pouvons remarquer dans ce diagramme un certain nombre de particularités : on ne peut pas par exemple parler de *fonction* permettant de passer d'un état à un autre car certains mouvements font passer d'un état à plusieurs autres dans ce sens qu'on ne peut pas savoir à priori vers lequel d'entre eux on ira. Par exemple, le mouvement C fait passer l'état $\mathcal{C}$ soit vers l'état $\mathcal{F}$, soit vers l'état $\mathcal{D}$, selon qu'on sera tombé sur les «bons»verres ou les autres...

Néanmoins, on peut remarquer que la suite de mouvements DCD1DCD, partant de n'importe lequel des états de l'automate, fait toujours passer par l'état final (avant même de commencer à retourner le moindre verre si on est parti de l'état final).

Ceci résoud le problème.

2.4.4 La simulation

Pour étudier un «système»nous pouvons réaliser des expérimentations sur le système... à condition qu'il soit construit ! Si cela n'est pas le cas, nous pouvons réaliser des expérimentations sur un modèle du système.Le modèle peut être un modèle physique ou un modèle mathématique. Pour résoudre les équations du modèle mathématique nous pouvons mettre en œuvre une solution analytique ou procéder par simulation. La simulation est une approche pour prédire des aspects de comportement d'un système en créant un modèle (mathématique) approché du système.

La simulation a potentiellement la capacité de résoudre les difficultés mentionnées ci dessus : il n'y a pas besoin de construire un système réel, vous pouvez contrôler et répéter les expérimentations, vous n'avez pas de limite pour les scénari d'expérimentation et il est possible à n'importe qui de reproduire les résultats. Une fois la simulation réalisée, il reste à analyser les résultats et de tirer des conclusions pour prendre des décisions.

Dans la communauté informatique, l'activité de simulation est particulièrement développée en architecture des machines[5] mais aussi dans la communauté des réseaux avec des simulateurs au niveau des paquets (NS-2[6], DaSSF[7], TCNG[8]) ou des émulateurs (ModelNet[9]) par exemple).

Prenons comme exemple le simulateur TCSIM (inclus dans le projet TCNG). TCNG permet par exemple de contrôler l'émission et la réception de paquets réseaux. Par exemple vous pouvez configurer votre ordinateur pour que les paquets reçus de l'adresse 192.168.1.23 le soient à la vitesse de 160Kb/s et ceux reçus de la machine 192.168.1.2 à la vitesse de 512Kb/s. Pour arriver à cela, il faut écrire des scripts qui vont configurer les différents composants de gestion de la carte réseau. Avant de déployer ces codes, on aimerait tester s'ils répondent bien à ce que l'on voulait. TCSIM le permet.

C'est donc un outil qui permet de valider les scripts générés par TCNG/TC. TCSIM prend en entrée des scripts de configuration (ceux qui seront utilisés ultérieurement par TCNG/TC), des données (les contenus des messages à échanger dans le réseau) et génère une trace d'exécution rendant compte de ce qui s'est passé. Cette trace peut ensuite être injectée dans des outils de post-traitement afin de tracer des graphes.

Tous ces outils, qui peuvent servir pour des études de cas en complément à notre présentation, se posent la question de savoir quelles sont les métriques représentatives et extractibles (mesurables dans la pratique). Tous ces outils posent aussi la question du choix de l'infrastructure, des tests représentatifs (benckmarks), de la méthode qui ont un impact sur la qualité des mesures effectuées.

[5] Voir http://www.cs.wisc.edu/~arch/www/tools.html, en particulier SimpleScaler sur http://www.simplescalar.com
[6] Voir http://www.isi.edu/nsnam/ns/
[7] Voir http://www.ssfnet.org
[8] Voir http://tcng.sourceforge.net/
[9] Voir http://modelnet.ucsd.edu/

Une des techniques est la simulation statistique qui combine la modélisation mathématique (développer un nombre limité de formules mathématiques qui résument les performances en mettant en relation les caractéristiques du programme étudié et les paramètres de la machine) et la simulation pour générer une trace synthétique c'est à dire un résumé des propriétés, des caractéristiques comportementales du programme (par exemple les accès mémoire, cache). Elle peut être obtenue en faisant par exemple des statistiques sur les motifs d'accès à la mémoire, d'où le nom de statistique. Cette trace synthétique est alors passée dans un simulateur.

L'intérêt de la démarche est d'arriver à résumer les comportements par un comportement type et d'étudier ce comportement type, puis de généraliser. Un autre intérêt est de prédire le comportement dans des temps très courts sans avoir à construire un vrai système mais en construisant un simulateur ce qui est beaucoup moins coûteux en développement comme en argent sonnant et trébuchant.

Afin de réduire encore plus l'espace des exécutions possibles, les chercheurs et les concepteurs ont également mis au point des techniques permettant de tronquer les exécutions, de réduire la taille des données dès le départ ou de faire de l'échantillonnage (pas seulement des données mais aussi des exécutions). La aussi se pose alors le problème de la représentativité de ce que l'on a extrait. Plutôt que de faire que des statistiques, l'idée est de choisir autant de représentants qu'il y a de comportements uniques dans le programme.

Outre l'étude des microprocesseurs, la technique de simulation statistique peut également avoir comme application la caractérisation de programmes (gros/réels) sur des jeux de données (importants/réels) que l'on appelle les workloads.

La question clef qui se pose alors est la validation c'est à dire la correspondance entre la simulation et le monde réel. Valider consiste à justifier que le modèle est plausible, à convaincre que le simulateur implémente le modèle (étape de vérification), à montrer au travers de quelques graphiques que sur des cas (spéciaux) il est raisonable de penser que la validation est correcte, à argumenter que les tendances sont respectées.

2.4.5 L'émulation et la virtualisation

Pour aller vers plus de réalisme, autrement dit pour être plus proche du monde réel, nous pouvons aussi pratiquer l'expérimentation par simulation par évènements discrêts ou par émulation.

La simulation par évènements discrêts s'attache à construire une abstraction du système étudié en un ensemble d'actions et d'évènements dépendants alors que l'émulation consiste à virtualiser les actions réalisées à bas niveaux de votre application ou par votre système.

Généralement, en émulation on travaille sur une platforme de plus grande ampleur que pour un simulateur (qui généralement tourne sur un seul processeur). Comme nous ne sommes pas encore à grandeur réelle, il est nécessaire de virualiser certaines ressources, par exemple en faisant en sorte qu'un seul processeur de l'émulateur joue le rôle de 10 processeurs de la plateforme réelle.

Reprenons la discussion vue au paragraphe sur la modélisation mathématique et qui concerne l'envoi de données à partir d'un nœud maître vers deux processeurs esclaves. Le contexte architectural de la modélisation est l'hétérogénéité des nœuds/processeurs et des communications.

Pour vérifier si cette modélisation est acceptable pour «le monde réel», nous devons nous munir, dans le cadre de l'émulation, d'outils permettant de faire en sorte que les processeurs soient hétérogènes et d'outils permettant de jouer également sur la vitesse des nœuds de communication.

Sans aucun outil, on pourrait également vérifier que pour une architecture hétérogène donnée, le comportement observé du temps d'exécution à partir du modèle est bien celui prédit. Cependant, une telle démarche ne permet que de valider l'architecture en question mais ne valide pas la démarche dans son ensemble. Par contre, si on se munit d'outils (autement configurables) alors on sera en mesure de vérifier et d'observer les predictions à partir d'une architecture (homogène) configurée selon nos souhaits (à partir d'une large gamme de paramètres), d'observer éventuellement des tendances, des minima... La richesse des résultats est donc plus importante en émulation.

Sur un système Linux dont le noyau est une version 2.6, nous trouvons les réponses suivantes quant aux outils nécessaires à notre émulation :

Module CPUfreq : c'est un outil permettant d'attribuer une fréquence d'horloge à un exécutable et donc d'émuler un processeur hétérogène. Une autre technique consisterait à consommer du temps CPU en changeant la priorité de l'ordonnanceur Linux et en passant la main à un processus consommateur de temps et non interruptible pendant le temps désiré et calculé selon une méthode non détaillée ici.

Module IProute : c'est un module qui s'interpose au niveau de la pile TCP (l'endroit ou le système stocke les messages pour la carte réseau) et qui ralentit les packets envoyés ou reçus. Le module TC (Traffic Control) permet nottament de configurer le débit et les latences des communications.

Notons que l'émulation consiste ici en une dégradation des performances. Faisons remarquer également que les techniques peuvent interférer les unes les autres. En effet, si on réduit la vitesse d'horloge du processeur, on va envoyer les messages sur la carte réseau à cette nouvelle vitesse et on ne serait peut être pas en mesure d'assurer le débit configuré par TC. Il faudra donc vérifier expérimentalement qu'il n'y a pas de problème de ce genre.

Pour que notre émulation soit encore plus réaliste, il s'agira de faire en sorte de disposer de deux cartes réseaux au niveau du maître. En effet, notre modélisation (revenez aux équations pour vous en convaincre) n'est valable que dans ce cas. Si l'on ne dispose pas de deux cartes réseaux, les envois ne pourront se faire que de manière multiplexée : le canal de communication est partagé.

Enfin, parmi les projets connus qui réalisent de l'émulation en virtualisant certaines ressources, nous pouvons citer MOC (Mac on Linux[10] qui à partir d'une

[10] Voir http://www.maconlinux.org

version de Linux sur un processeur PowerPC permet de faire tourner MacOS. Nous pouvons aussi mentionner Bochs[11] qui est un émulateur d'architecture IA-32 (x86) c'est à dire de l'architecture PC. Dans le domaine du multimédia, on se reportera au site http://www.emulator-zone.com. Pour les domaines de l'émulation de systèmes d'exploitation les projets Xen, Parallel Desktop (pour Mac OSX) ou des produits comme ceux de SWsoft sont particulièrement intéressants. Par exemple, en 2007, Xen est fournit gratuitement dans de nombreuses distribution Linux et s'installe comme une autre application. Les autres lectures recommandées sont :

[1] Yi, Eeckhout, Lilja, Calder, John, Smith – The future of Simulation: A Field of Dreams – IEEE Computer magazine, Nov 2006, pages 22-29.

[2] Arnaud Legrand:
http://www-id.imag.fr/Laboratoire/Membres/Legrand_Arnaud/research.html

[3] http://www.dmem.strath.ac.uk/ pball/simulation/simulate.html : notes sur la technique de simulation par évênements discrêts.

[4] Techniques de virtualisation :
http://fr.wikipedia.org/wiki/Virtualisation_(informatique).

[5] Définitions larges (c.à.d. qui recouvrent beaucoup de domaines applicatifs) de simulation et émulation sur Wikipédia :
http://fr.wikipedia.org/wiki/Simulation_informatique et
http://fr.wikipedia.org/wiki/%C3%89mulateur

[6] Le projet Linux Vserver inclut des pages introductives sur la virtualisation : http://linux-vserver.org/Overview

2.4.6 Gros plan sur la caractérisation d'un modèle de simulation

Après ces tous premiers éléments, revenons sur ce qui caractérise un modèle de simulation. Le modèle de simulation peut être déterministe ou stochastique. Il est stochastique quand des décisions sont prises selon un processus aléatoire. Si c'est la cas, deux exécutions de simulation du même jeu de données pourront donner des résultats qui varient par un facteur que l'on cherche à maintenir le moins impactant possible. Pour une simulation déterministe, deux exécutions de simulation du même jeu de données donnent le même résultat.

Une simulation peut aussi être soit statique soit dynamique. Si le temps est une variable significative alors elle est dite dynamique, sinon statique.

Une simulation peut être soit continue soit discrète. Elle est discrète quand l'état du système change à des intervalles de temps dicrèts. Dans ce cas on peut mettre en œuvre les outils de la théorie des files d'attente.

On appelle alors «simulation à évèvements discrêts»une simulation stochastique, dynamique (l'évolution du temps a une grande importance) avec des évènements qui arrivent à des instances de temps discrêts.

[11] Voir (http://bochs.sourceforge.net

On appelle alors «simulation de Monte Carlo»une simulation stochastique qui a aussi la propriété d'être statique (l'évolution du temps n'a pas d'importance).

Prenons un exemple où la simulation de Monte Carlo est mise en valeur. Notez par ailleurs qu'ici le terme de simulation se réfère à l'activité de calcul ou encore à l'activité de développement d'un programme qui résoud le problème.

La technique dite de Monte Carlo vient en fait de la géométrie et plus précisément du calcul d'aire. Cette technique est particulièrement viable à notre époque grâce au développement d'ordinateurs de plus en plus rapide.

L'exemple que nous développons ici est le problème de calculer la valeur du nombre π, ou de manière équivalente l'aire d'un cercle qui est égale à $\mathcal{A} = \pi * r^2$. Pour calculer π nous allons en fait considérer que $r = 1$ et l'équation d'un cercle : $x^2 + y^2 = r^2$. Un point est dans le cercle si $x^2 + y^2 \leq 1$. On tire de manière aléatoire N couples (x, y) et nous comptons le nombre de fois ou $x^2 + y^2$ vérifient l'inégalité. Cela donne une estimation de π lorsqu'on divise cette somme par N. La valeur de N a bien entendu un impact sur la précision du résultat.

L'idée est donc d'approximer l'aire par un pourcentage de fois où nous tombons à l'intérieur d'un cercle de rayon 1. En fait dans le code Python qui suit nous considérons le quart de plan supérieur et nous générons des valeurs de x, y comprises entre 0 et 1. L'estimation de π sera égale à quatre fois la somme moyennée calculée.

```
#
# fichier pi.py
#
import random

def my_gene():
    N=100000; # Nombre de tirages aléatoires
    cpt=0;
    # initialisation du germe du générateur
    random.seed();
    for i in range(1,N+1):
        x=random.uniform(0.0,1.0);
        y=random.uniform(0.0,1.0);
        if (x*x + y*y) < 1:
            cpt = cpt+1;
    return 4.0*(cpt/float(N))

#
# Programme principal
#
MAX=10;
somme = 0.0;
for i in range(1,MAX+1):
    new_val=my_gene();
```

```
    print "#",i,":",new_val
    somme = somme + new_val;
print "Moyenne des valeurs obtenues:", somme/float(MAX);
```

Lorsqu'on lance le programme nous obtenons :

```
$ python pi.py
# 1 : 3.15196
# 2 : 3.1484
# 3 : 3.14
# 4 : 3.1444
# 5 : 3.14208
# 6 : 3.13812
# 7 : 3.14124
# 8 : 3.1404
# 9 : 3.14236
# 10 : 3.15652
Moyenne des valeurs obtenues: 3.144548
```

Remarquons dans l'exécution précédente que 10 estimations sont calculées ainsi qu'une moyenne. Comme le modèle de simulation est stochastique, le résultat d'une seule exécution n'est pas suffisant pour conclure. Peut-on avoir confiance dans les résultats produits ?

Pour répondre à cette question nous allons contruire un intervalle de confiance de la manière suivante à partir des définitions suivantes. Soient n le nombre d'expériences et x_i la valeur du résultat de l'expérience i :

Définition (*Moyenne*) :

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

En statistique on montre que la variable $\bar{x}$ suit la loi normale d'espérance μ et de variance σ^2/n.

Définition (*Écart type*) :

$$s = \sqrt{\frac{\sum_{i=1}^{n} [x_i - \bar{x}]^2}{n-1}}$$

En statistique on montre que la variable s suit la loi du χ^2 à $n-1$ degrés de liberté.

Il est ensuite possible de caractériser un intervalle de confiance rigoureux pour la variable suivante :

$$T_0 = \frac{\bar{x} - \mu}{\sqrt{S/n}}$$

avec

$$S = \frac{s\sigma^2}{n-1} = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

Le lecteur est invité à reprendre les justifications sur la page suivante de Wikipédia : http://fr.wikipedia.org/wiki/Loi_de_Student.

On arrive alors à la conclusion que la distribution de la variable T_0 est connue indépendemment de σ^2 et par conséquent les intervalles de confiance qui lui sont associés sont également connus. Il est alors possible d'obtenir un intervalle de confiance pour μ à partir d'une réalisation des variables $x_1, \ldots, x_n$, de laquelle on déduit des valeurs de $\bar{x}$ et s.

Nous avons encore besoin des définitions suivantes:

Définition (*Quantile d'ordre* k) : Pour une variable T suivant une loi de Student à k degrés de liberté, on définit t_γ^k comme la quantité telle que la probabilité d'obtenir $T > y_\gamma^k$ soit égale à γ.

La quantité t_γ^2 est également appelé le quantile d'ordre $1-\gamma$ de la loi de Student à k degrés de liberté et leurs valeurs peuvent se consulter à partir du lien Internet de Wikipédia cité au dessus. Dans ce cadre, si $t_\gamma^k > 0$, alors la probabilité d'obtenir $-t_\gamma^k < T_0 < t_\gamma^k$ est égale à $1 - 2\gamma$.

Or on a :

$$\mu = \bar{x} - T_0\sqrt{\frac{S}{n}}$$

La probabilité d'obtenir

$$\bar{x} - t_\gamma^{n-1}\sqrt{\frac{S}{n}} < \mu < \bar{x} + t_\gamma^{n-1}\sqrt{\frac{S}{n}}$$

est elle aussi égale à $1 - 2\gamma$. Le niveau de confiance α associé à cet intervalle est donc $\alpha = 1 - 2\gamma$.

Le niveau de confiance α correspond à la probabilité que l'espérance μ de la loi normale se trouve à l'intérieur de l'intervalle de confiance.

Nous avons repris le code Python précédent pour donner l'intervalle de confiance pour $\alpha = 0.95$ correspondant à $\gamma = (1 - \alpha)/2 = 0,025$. D'après le tableau des valeurs du quantile (cf. cite Wikipédia déjà donné), on a $t_{0,025}^9 = 3,690$.

Le nouveau code Python ainsi qu'une exécution donne :

```
#
# fichier pi1.py
#
import random
import math

def my_gene():
```

```
    N=100000;
    cpt=0;
    random.seed();
    for i in range(1,N+1):
        x=random.uniform(0.0,1.0);
        y=random.uniform(0.0,1.0);
        if (x*x + y*y) < 1:
            cpt = cpt+1;
    return 4.0*(cpt/float(N))

MAX=10;
new_val=[];
somme = 0.0;
for i in range(0,MAX):
    new_val.append(my_gene());
    print "#",i,":",new_val[i]
    somme = somme + new_val[i];
moy= somme/float(MAX)
print "Moyenne des valeurs obtenues:", moy;

somme=0.0;
for i in range(0,MAX):
    somme = somme + (new_val[i] - moy)*(new_val[i] - moy);
S=math.sqrt(somme/float(MAX-1))
print "Calcul de l'ecart type",S;

print "Intervale de confiance (alpha=0.95) :
  [",moy-3.69*math.sqrt(S/MAX) ,",", moy+3.69*math.sqrt(S/MAX),"]";

$ python pi1.py
# 0 : 3.1382
# 1 : 3.13992
# 2 : 3.14616
# 3 : 3.14016
# 4 : 3.1348
# 5 : 3.13796
# 6 : 3.13528
# 7 : 3.14156
# 8 : 3.13116
# 9 : 3.14596
Moyenne des valeurs obtenues: 3.139116
Calcul de l'ecart type 0.00474907991089
Intervale de confiance (alpha=0.95) : [3.058702 , 3.219529]
```

Il nous reste à valider l'ensemble du travail de modélisation en déterminant

si le modèle de simulation se comporte comme supposé. Différentes techniques sont utilisées pour calculer si la sortie du modèle est statistiquement similaire au système physique. Cela se réalise par un test d'hypothèse qui est une démarche consistant à rejeter ou à accepter une hypothèse statistique, appelée hypothèse nulle, en fonction d'un jeu de données (échantillon).

Pour notre cas, on étudie $t_n = (\bar{x} - \mu_0)/\sqrt{s^2/n}$ et on prend les règles suivantes : si $| t_n | > t_\gamma^{n-1}$ alors rejeter l'hypothèse, si $| t_n | \leq t_\gamma^{n-1}$ on accepte l'hypothèse.

Note : μ_0 correspond à l'espérance de la loi supposée, ici la loi normale. On peut montrer que l'espérance de la loi normale centrée réduite est nulle.

Exercice : faire le test d'hypothèse pour notre exemple numérique.

2.5 Construire une présentation orale

GÉNÉRALEMENT, après avoir retourné votre compte rendu de projet à vos évaluateurs, ceux-ci vous demandent de venir exposer votre travail devant un public. Examinons maintenant quelques conseils pour bien vivre son exposé. Un bon travail qui n'est pas bien présenté n'est pas nécessairement un bon travail au final !

Il faut être à l'aise et il faut impérativement respecter le timing ! Si on vous donne 10 minutes d'exposé il ne faut pas chercher à construire un exposé de 15 minutes et le jour de la présentation, si vous débordez c'est grave : vos évaluateurs ne vont pas apprécier. Pour 10 minutes d'exposé, il faudrait faire cinq transparents, pas plus !

Habillez vous convenablement (faites un effort), ne mettez pas les mains dans les poches, parlez fort et distinctement. Soyez vivant et montrez de la conviction et une joie certaine dans ce que vous dites. Ne gesticulez pas, ne montrez pas que vous êtes de mauvaise humeur. Dans une présentation orale, la forme a de l'importance : ne soyez pas ennuyeux ni dans la forme ni dans le fond. En un mot :

MAKE THE DIFFERENCE!!!

Avant de commencer l'exposé oral, vérifiez que vous n'êtes pas devant le vidéo-projecteur et donc que votre présentation est bien visible par tous. Parlez au public et non au tableau : tournez-vous du coté du public. Si le vidéo-projecteur vous semble mal positionné, demandez à changer sa place afin de faire en sorte d'être bien en face de votre public : les yeux dans les yeux !

Ne vous laissez pas perturber par une question au cours de l'exposé : si vous ne savez pas répondre sur le moment, plutôt que d'avouer explicitement que vous ne savez pas, tournez le problème en suggérant à la personne qui vous à posé la question de prendre quelques minutes après l'exposé pour y répondre. C'est le métier qui parle ici ! En plus, vous n'avez pas trop perdu de temps par rapport à votre timing. Mais ne cherchez pas à bluffer, préférez alors d'être franc et admettez votre ignorance sur le point en question.

De manière générale, les transparents ou les pages vidéo-projetées par les étudiants sont trop chargées. Chaque page a un titre suivi l'une liste de termes indé-

pendants en principe et qui vont vous servir à discuter. Chaque point listé sur un transparent doit correspondre à une idée différente - de trois à quatre points au grand maximum doivent être listés par transparent. Par ailleurs, il faut veiller à la coupure comme suit (la partie gauche donne ce qu'il ne faut pas faire, la partie droite ce qu'il convient de faire) :

```
o bla bla bla ;                                | o bla bla bla
o Les écritures sont satisfaites, cela dépend  | o Les écritures sont satisfaites,
  de l'implémentation sous jacente ;           |   et cela dépend de l'implémentation ;
o bla bla bla.                                 | o bla bla bla.
```

L'idée étant de regrouper sur une ligne tout ce qui constitue une même idée. Ne laisser pas faire la coupure par votre outil informatique.

Concernant les polices de caractères à utiliser, préférez les polices sans empattement. Évitez[12] donc le Times qui est une police inventée en 1932 pour le journal The Times. C'est donc une police pour journaux et pas pour faire des transparents !

Il ne faut pas écrire des longues phrases sur un transparent. Dans un exposé oral, il ne faut pas lire un document annexe à vos transparents. Vous devez être capable de parler sans support.

La difficulté majeure rencontrée par les étudiants est de choisir ce qu'il faut couvrir. En effet, un travail de projet est généralement conséquent ; il faut faire des choix. Posez-vous la question de savoir quel était le cœur du problème et du point fort de votre compte rendu. Puis, bâtissez votre présentation en fonction des réponses.

Voici une autre technique. Vous pouvez aussi construire votre exposé en vous demandant ce que votre auditoire doit connaître pour comprendre le résultat principal de votre travail. Fournissez-lui alors l'information minimale. Ne rentrez pas dans les détails : ils doivent être l'objet d'une discussion «d'après l'exposé».

Ne cherchez pas à compliquer les choses : ne cherchez pas à détailler ce qui est dur à suivre, par exemple n'exposez pas une longue série de lemmes. Une fois que votre auditoire est perdu, il reste dans cet état jusqu'à la fin de l'exposé. Voila bien ce qu'il faut éviter. Cherchez plutôt dans votre exposé à dégager l'idée, le résultat principal, et de manière générale. Les détails techniques se trouvent dans le compte rendu.

L'exposé oral peut reprendre l'organisation de votre document écrit :

1. Un transparent pour le titre, le nom et les coordonnées professionnelles de l'orateur ;
2. Un transparent pour l'organisation de l'exposé ;
3. Un transparent pour exposer le problème ; c'est ici qu'il faut glisser des aspects problématiques peut être un peu plus généraux que le problème spécifique ;
4. Un transparent pour exposer l'existant ;

[12] Notez que l'auteur des transparents en annexe ne respecte pas ce conseil.

5. Des transparents sur vos propositions et vos apports : méthodologie, résultats obtenus ; Justification théorique : conditions et hypothèses sur vos études ;

6. Preuve analytique (si possible) pour donner une idée de la preuve de correction, de la validité, du calcul des valeurs initiales pour la simulation, pour l'estimation des performances, de la complexité ;

7. Des transparents sur l'expérimentation et de l'implication qu'elle a dans la «vraie vie»(la réalité concrète) ;

8. Un résumé des apports ;

9. Bilan et perspectives (en faisant des liens avec vos apports). Qui cela touche ? Que faut-il pensez du travail ?

10. Éventuellement vous donnez des références bibliographiques et/ou WEB.

11. Remerciez votre auditoire et demandez s'il y a des questions ;

Vous avez maintenant une structure logique mais ce n'est pas suffisant, elle doit être apparente pour votre auditoire. Pour cela, au cours de l'exposé, utilisez des phrases du type «comme je l'ai montré précédemment...». Il s'agit de faire des références en arrière ou en avant... mais pas trop afin de montrer combien la discussion présente est liée à ce qui précède ou à ce qui va suivre.

Dans le même ordre d'idée, vous pouvez faire un transparent qui marque une chute sur les concepts puis sur un autre transparent bien séparé, vous lancez les discussions sur les implémentations. Autrement dit, séparez bien les parties et faites en sorte que votre discours le fasse apparaître : soigner les transitions.

Revenons un instant sur les tous premiers transparents. Vos premiers mots vont faire en sorte que l'auditoire va se faire une opinion de vous et du sujet. Pour accrocher votre auditoire, commencez par une phrase surprenante : argumentez par exemple que les solutions intuitives ou familières ne sont pas adaptées ou dites que le problème que vous avez traité a d'importantes conséquences pratiques. Affirmez que vous avez fait «vachement mieux».

C'est seulement après avoir dit où vous alliez que vous pourrez passer au transparent sur l'organisation de l'exposé (transparent 2) qui explique comment vous y allez. Éventuellement vous pouvez glisser une anecdote mais uniquement si cela se rapporte à votre sujet. Il s'agit là aussi de capter l'audience mais soyez sûr de votre coup !

Ne traînez pas en longueur : il faut arriver à parler de son travail rapidement. Ne construisez pas un exposé où les trois quart du temps sont consacrés à exposer ce que d'autres personnes ont fait ! Ceci est vrai aussi pour la construction du rapport écrit.

Poursuivons par un mot sur le transparent de conclusion. Ne soyez pas fade. La chute doit être heureuse, positive. Il doit y avoir des perspectives. Rappelez les points importants que vous souhaitez que votre auditoire retienne. Considérez aussi le travail en cours.

Ne terminez pas par une phrase négative, par exemple : «Voilà, c'est tout ce que je voulais dire, sauf qu'il y a une implémentation mais elle ne fonctionne pas». Dites plutôt : «Voilà, je terminerai cet exposé par vous signaler que nous développons actuellement une implémentation en Java. Elle représente déjà 300 lignes. J'attends maintenant vos questions».

En un mot, il doit y avoir une *chute glorieuse*. Ceci est aussi vrai pour les fins de paragraphe de votre rapport écrit.

L'annexe de ce document vous donne des recommandations pour réaliser un exposé avec un contenu de orienté recherche. Inspirez vous de cet exemple.

Si vous êtes habile, vous pouvez anticiper les questions et donc préparer des réponses. Par exemple, si vous avez répété votre exposé (ce qui est bien sûr indispensable) avec un ami et qu'il a posé des questions, notez les et prévoyez (sur de nouveaux transparents) déjà les réponses : la question pourrait revenir. Dans tous les cas, ne parlez pas de ces transparents supplémentaires si cela n'est pas demandé : ce sont des extras !

Rappelez-vous que la discussion qui suit l'exposé est un révélateur de qui vous êtes réellement. Laisser les personnes poser entièrement les questions (sans les interrompre) puis prenez la parole. Soyez diplomate. Généralement il n'y a pas de sens caché aux questions, les (bons) évaluateurs ne sont pas là pour vous piéger mais pour faire en sorte que vous vous révéliez.

En complément des discussions, vous pouvez également visiter les sites WEB suivants :

http://www.cs.rmit.edu.au/~jz/writing.html

http://www.presentations.com

http://www.presentersuniversity.com

`http://www.rightbrain.com/pages/book-download.shtml`

http://www.perrousseaux.com, (éditeur), `http://www.typographie.org`, `http://www.gutenberg.eu.org/`, `http://tex.loria.fr/` sont des sites relatifs à la typographie.

Les instructions typographiques données dans ce document ont été prises dans l'ouvrage «Les règles de typographie en usage à l'imprimerie nationale», Imprimerie nationale, 1990, ISBN 2-11-081075-0.

2.6 Conclusion

Nous avons souligné dans ces pages que le compte rendu et l'exposé oral sont des parties entières d'un travail. Ils répondent à une organisation précise, exposée

dans un plan (de résolution). La première des règles du jeu du savoir scientifique et technique consiste alors à exprimer clairement ses idées dans un discours articulé autour d'hypothèses, de faits et de déductions.

Par l'apprentissage de techniques et l'utilisation raisonnée d'outils, ce chapître doit aussi encourager le travail personnel et une certaine démarche de groupe. Faites relire votre production écrite, rédigez à plusieurs le document de synthèse de fin de projet. Lorsque vous êtes dans une démarche de recherche de solutions à un «gros»projet, vous n'êtes pas en cours, bien au chaud dans l'amphithéâtre ! Il faut interagir avec vos enseignants et entre vous. En un mot il faut pratiquer, sortir de votre environnement bien confortable de l'amphithéâtre ou de la salle de travaux dirigé et ne plus croire à ce vieil adage qui dit «qu'un cours est le processus de transfert d'information depuis les notes du professeur jusqu'aux notes de l'étudiant sans jamais passer par le cerveau de l'un ou de l'autre».

Des critères de jugement de l'évaluateur ont été aussi exposés. Vos enseignants ne sont pas uniquement une source d'information ; ils vous demandent de ne pas vous arrêter à la simple rédaction d'un programme : vous devez être capable d'analyser, d'apprécier votre propre production. Voir–juger–agir : une démarche scientifique et technique accomplie passe inévitablement par ces trois termes.

> *The reasonable man adapts himself to the world; the unreasonable man persists to adapt the world to himself. Therefore, all progress depends on the unreasonable.*
>
> **George Bernard Shaw**

Annexes

L'exemple qui suit est auto-suffisant et il concerne plutôt une présentation d'un chercheur. Cependant il comporte aussi des idées qui dépassent le cadre «recherche». A titre d'exercice, examinez la possibilité d'adapter le schéma à votre situation.

3 Programmer et raisonner sur les programmes

Sommaire

3.1 Motivations

L'OBJECTIF de ce chapître est double. Premièrement il s'agit de donner au lecteur des éléments fondamentaux pour comprendre comment les programmes sont construits et comment on peut raisonner formellement sur les programmes. Deuxièmement, nous proposons Python comme langage d'implémentation des solutions aux problèmes qui seront exposés au prochain chapître.

En fait l'activité intellectuelle mise en œuvre dans ce chapître est beaucoup moins grande et notre ambition relativement limitée par rapport à ce que nous avons vu précedemment et concernant l'ensemble de l'activité de l'informaticien. En effet, nous nous concentrons sur des exemples d'école qui par définition sont de taille raisonnable pour être traités par une seule personne en un temps réduit. Nous nous plaçons du coté de la spécification[1] dans le premier paragraphe puis du coté de l'implémentation de la solution dans le deuxième paragraphe.

Nous avons aussi fait le choix de présenter les outils de preuves de programmes (ce qui peut apparaître comme vouloir dépoussiérer un vieux sujet jadis enseigné). En fait cette notion vient avec les notions de logique des propositions et se veut être là pour faire le lien entre la logique et les programmes, les raisonnements comme l'induction et les mathématiques. Il faut peut être aussi voir tout cela comme faisant partie maintenant de notre histoire et replacer l'utilité des preuves de programmes lorsque la programmation se limitait à un cadre de programmation impérative c'est-à-dire à un cadre ou seules les notions de mise en séquence, choix et itération importaient. On sait que d'autres types de programmation ont pris de l'importance depuis comme l'objet, l'évènementiel, par contrats... et que les

[1] Texte qui décrit ce que l'on doit faire - modélisation de la solution du problème

outils présentés ici ne sont plus utilisables dans ces cadres. Ainsi va l'histoire de la programmation !

Le lecteur est par ailleurs invité à lire les pages de Henri Habrias[2] sur le génie logiciel car dans ce chapître nous nous intérogeons précisemment sur comment les programmes sont construits. Du même auteur, vous trouverez également un lien[3] référençant en particulier les livres de Jacques Arsac[4] publiés dans les années 70 en France sur le sujet de la programmation et qui sont maintenant rentrés dans la grande Histoire de la programmation (du point de vue de ceux qui ont été initiés par ces méthodes). La grande histoire des fondements des systèmes d'exploitation quant à elle peut être en partie consultée à partir des articles en ligne de Edsger W. Dijkstra[5]. On trouvera également des articles sur la vérification de programmes, l'algorithmique, la conception de programmes.

3.2 Raisonner sur les programmes

AVANT de programmer il est d'usage de faire un énorme travail de réflexion sur ce que doit être et doit faire la solution au problème rencontré. Le «comment on fait»est abordé dans le dernier paragraphe de ce chapître. Lorsqu'on programme, nous avons besoin de trois concepts fondamentaux qui sont la mise en séquence d'actions, le choix et l'itération. Ce sont ces trois concepts que nous alons modéliser et qui vont nous permettre de raisonner ensuite sur les programmes.

Remarquons que nous n'envisageons pas ici les programmes avec des supports pour la programmation objet mais des programmes réduits aux trois concepts précédents.

L'organisation de ce paragraphe est la suivante. Premièrement nous présentons les problèmes qui sont en face de nous puis, dans le deuxième sous paragraphe nous introduisons des éléments de logiques enfin nous montrons comment nous servir des formalismes de la logique pour effectuer des dérivations rigoureuses de codes. Cette partie qui concerne le comment raisonner sur les programmes termine par une autre technique de spécification qui s'appelle les types abstraits algébriques et par une section relative à comment poser et résoudre un problème en général. Cette dernière sous-partie annonce les études de cas du chapître qui suit.

3.2.1 Introduction

Nous allons aborder maintenant les thématiques suivantes :

1. **Qu'est ce qu'une spécification** : motivations et introduction générale ;
2. **Éléments de logique :** écrire des propositions, utilisation de règles de déduction, schémas de preuve ;

[2] http://www.iut-nantes.univ-nantes.fr/%7Ehabrias/spec1/Spec1sommaire.html
[3] http://www.iut-nantes.univ-nantes.fr/~habrias/dessGledn/dessbiblio.html
[4] http://fr.wikipedia.org/wiki/Jacques_Arsac
[5] http://www.cs.utexas.edu/users/EWD/

3. **Introductionà la dérivation rigoureuse de code** : pré et post conditions, notion d'invariant – preuve de programmes : correction partielle et totale ;

4. **Introductions aux méthodes d'analyse - Comment poser et résoudre un problème** : les étapes du développe ment d'un logiciel – tactiques, paradigmes.

5. **Spécifications Algébriques** : écrire des équations pour décrire ce que doit faire un système ;

A l'issue de ces parties, le lecteur sera en mesure de mettre en œuvre des méthodes (ensembles structurés de principes qui orientent la façon de concevoir) et leurs raffinements suivants (les éléments qui suivent sont repris du livre ANAGRAM [1] - voir les références bibiographiques en fin de chapître) :

- **Compréhension :** passage d'un langage à un autre (verbal, graphique, symbolique...). Il s'agit d'un effort de transposition. Le matériel est présenté sous un jour, un ordre nouveau : il s'agit d'interpréter. Extrapoler.

- **Appliquer :** utiliser des représentations abstraites dans des cas particuliers ;

- **Analyser :** rechercher, séparer, trier des faits, des hypothèses, des principes d'organisation afin de faire apparaître leur hiérarchie et leurs relations.

- **Synthétiser :** réunion des éléments précédents dans le but de former un tout ; élaborer un plan d'action.

- **Évaluer :** formation de jugements, remettre en cause, critiques.

Les concepts fondamentaux qui servent à développer des programmes ont été clairement identifiés depuis de nombreuses années [1] par les profesionnels. Il s'agit de (liste non exhaustive mais hautement étudiée dans notre ouvrage) :

- l'abstraction : concentrer l'effort de modélisation et de résolution a un niveau général plutôt que sur les détails ;

- Analyser, concevoir, médhodologie. Quels outils pour une spécification non ambigüe ? La conception doit tenir compte des contraintes. Il est fait usage de notations spécifiques qui sont des outils pour communiquer dans une discipline d'engéniérie.

- Prototypage : construire un premier code exécutable pour obtenir un premier retour afin d'améliorer l'usage de l'application.

- Modularité et architecture des programmes : facteur influant la qualité. Un *Module* est une interface bien définie pour autres modules et qui cache une décision d'implantation et qui peut être testé, prouvé independamment des autres modules.

On oublie pour le moment que par essence la tâche d'un programmeur est : étant donnée une spécification, développer un programme qui satisfait la spécification. Notre nouveau rôle est de contrôler le **cycle de vie du logiciel** et :

1. d'analyser les besoins du client ;
2. de **définir ce que l'on doit faire ;**
3. de **concevoir une solution ;**
4. de passer la main au programmeur ;
5. de valider le système (respecte t'il les besoins ?) ;
6. de vérifier que le système «fait» bien ce que l'on souhaitait ;
7. de demander l'installation et la maintenance ;

ε

On va s'intéresser uniquement ici aux points (ii), (iii) et (iv). Pour cela nous allons principalement travailler sur le plan de comment, de manière générale «poser, concevoir et résoudre un problème» (voir les lectures complémentaires de Polya [2], Solow [3], Gries [5], Dromey [4], Dijkstra [7], Gaudel [14], Bentley [12] qui doivent avoir été lues par tout étudiant). À ce niveau, le rôle des mathématiques et de la logique est résumé par la citation suivante :

> *The problem is to recognize computing science for what it really is, a mathematically-based discipline concerned with the application of rigourous methods for the specification, design, and implementation of computer systems.*
>
> **Geoff Dromey**

En effet, les programmes peuvent être prouvés corrects de la même façon que les théorèmes des mathématiques ; l'assertion selon laquelle un programme satisfait une spécification est une «phrase mathématique». Les principales étapes dans la dérivation d'un programme sont : la spécification, la conception d'un algorithme, l'implantation.

Une *spécification* est une collection de critères qui, s'ils sont vérifiés par le programme la codant, le qualifiera de *correct*.

> *Testing can reveal the presence of errors, not their absence.*
>
> **E.W Dijkstra**

Pour formuler de tels critères on passe par un langage (avec une syntaxe et une sémantique rigoureuse) capable d'être manipulé formellement (pour raisonner). Ce langage exprime ce qui est requis.

Une implantation correcte (Python, Java, C...) décrit comment les critères sont concrétisés en pratique.

L'abstraction permet quant à elle de se concentrer sur les propriétés essentielles du dispositif à construire ; les détails qui peuvent être ignorés seront à priori cachés. Il y a en informatiques différentes formes d'abstraction accéssibles au programmeur: l'écriture de procédures, les modules (collection de variables (data) et de procédures (opérations), constantes et types), les types de données définis par l'utilisateur, les objets (un système est vu comme une interaction d'objets qui envoient et reçoivent des messages) dans une classe.

L'abstraction s'obtient également en cachant de l'information (*information hiding*) afin de rendre les programmes plus faciles à lire et à maintenir.

3.2.2 Éléments de logique

Nous commençons par rappeler quelques notions autour du calcul des propositions. C'est un système de déduction qui est muni des opérations $\wedge$ (et), $\vee$ (ou), $\Rightarrow$ (implique), $\neg$ (négation), $\Longleftrightarrow$ (si et seulement si) qui sont utilisées pour combiner et modifier des propositions (une phrase écrite au moyen des opérations/connecteurs). Nous rappelons ici la définition de implique et de si et seulement si (dans les tables, le symbole T signifie True, le symbole F signifie False) :

p	q	$p \Rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

p	q	$p \Longleftrightarrow q$
T	T	T
T	F	F
F	T	F
F	F	T

Les *Tautologies* (propositions toujours vraies) et les *Contradictions* (propositions toujours fausses) sont aussi des moyens pour transformer de «grosses propositions»en propositions «plus petites». On peut donner un certain nombre de tautologies utiles :

Tautologie	Commentaire
$(p \Rightarrow q) \wedge (p \Rightarrow r) \Leftrightarrow (p \Rightarrow q \wedge r)$	simplication de $\Rightarrow$
$(p \Rightarrow r) \wedge (q \Rightarrow r) \Leftrightarrow (p \vee q \Rightarrow r)$	"
$(p \Rightarrow q) \vee (p \Rightarrow r) \Leftrightarrow (p \Rightarrow q \vee r)$	"
$((p \Rightarrow r) \vee (q \Rightarrow r)) \Leftrightarrow ((p \wedge q) \Rightarrow r)$	"
$p \Rightarrow q \Leftrightarrow \neg q \Rightarrow \neg p$	contraposée
$p \Rightarrow p \vee q$	'addition'
$(p \Rightarrow q) \Rightarrow (p \Rightarrow (p \wedge q))$	absorption
$p \wedge q \Rightarrow p$	simplification

On peut aussi rappeler les règles de De Morgan (par exemple : $\neg(p \wedge q) \equiv$

$\neg p \vee \neg q$) et les tautologies pour éliminer les symboles $\Rightarrow$ (par exemple : $p \vee q \equiv \neg(\neg p \wedge \neg q)$).

Pour dériver de nouveaux faits, on dispose des *règles d'inférence* du Modus Ponens (de $p \Rightarrow q$ et de p alors je déduis q) et du Modus Tollens (de $p \Rightarrow q$ et $\neg q$ je déduis $\neg p$). Ces deux règles sont basées sur des tautologies que nous vous invitons à préciser.

D'une manière générale, un *système de déduction formel* consiste en :

1. un ensemble fini de symboles et de règles spécifiant les phrases bien écrites (en ce qui nous concerne : variables propositionnelles, connecteurs, parenthèses, T, F et les règles de $\wedge, \vee, \Rightarrow$)
2. Un ensemble de phrases que l'on vous donne : les axiomes ;
3. un ensemble de manières pour construire de nouvelles phrases à partir des autres : *les règles d'inférence* ;

Un système avec des règles d'inférences mais pas d'axiomes s'appelle un *système de déduction naturelle*. La règle d'inférence du **modus ponens** est capitale dans le calcul des propositions.

Techniques de preuves

Pour prouver un théorème de la forme générale : $p_1 \wedge p_2 \wedge \cdots \wedge p_n \Rightarrow q$, on dégage à partir de $p_1, p_2, \cdots, p_n$ un résultat intermédiaire r_1.

Ensuite, à partir de $p_1, \cdots, p_n$ et r_1, on déduit un autre résultat r_2 et ainsi de suite jusqu'au moment où la proposition q est *logiquement* déductible de $p_1, \cdots, p_n$ et des résultats intermédiaires $r_1, \cdots r_k$.

Chaque étape doit pouvoir se justifier par une règle d'inférence et à tout moment, on peut remplacer une proposition par une proposition qui lui est logiquement équivalente.

Une autre technique courante de démonstration s'appelle la preuve par contraposition. Elle est basée sur la tautologie : $(p \Rightarrow q) \Leftrightarrow (\neg q \Rightarrow \neg p)$. Ainsi, démontrer $p \Rightarrow q$ par contraposition revient à supposer que q est fausse et à en déduire que p est fausse également.

La preuve par contraposition est un cas particulier de la technique de démonstration par *l'absurde*. Cette dernière est basée sur la tautologie :

$$((p \Rightarrow q) \wedge \neg q) \Rightarrow \neg p$$

On dispose donc de la règle d'inférence du **modus tollens**, dont le sens intuitif est évident : s'il est vrai que la proposition p implique une proposition q que l'on sait être fausse alors p ne peut être que fausse. En pratique, la proposition q est une contradiction du genre $t \wedge \neg t$.

Nous reprenons maintenant la présentation de Slow [3] pour introduire les prochains éléments de logique car elle est progressive et ilustrée de nombreux exemples. Rappelons que tous ces éléments vont nous servir ensuite à prouver

les programmes en associant à ces programmes des formules de la logique et en raisonnant sur ces formules.

3.2.3 Calcul des prédicats

Il a été défini pour apporter plus de richesse au calcul des prédicats par les notions de *termes* (constantes, variables, fonctions de termes) et de *prédicats* (relation entité-propriété).

Les *variables* jouent le jeu des noms et des pronoms du langage naturel et permettent des écritures concises... en particulier avec l'introduction des **quantificateurs**.

Manipulation des quantificateurs

Le quantificateur «il existe»: il est présent dans des phrases du genre :

Définition 1 *Un entier* n *est un carré si il existe un entier* k *tel que* $n = k^2$.

On note que dans cet exemple on a la possibilité de choisir plus d'un entier (par exemple 3 ou -3) qui vérifient la proposition.

La plupart du temps, le constructeur $\exists$ est présent dans des phrases de la forme suivante : «Il y a un objet vérifiant une certaine propriété tel que quelque chose arrive». Pour montrer qu'une telle phrase est vraie, vous pouvez employer la *méthode constructive* qui consiste à développer un algorithme de telle façon que l'objet produit vérifie la propriété demandée et que le «quelque chose arrive».

Le quantificateur «pour tout»: il est présent dans des phrases du genre :

Définition 2 *Un ensemble* S *est un sous-ensemble d'un ensemble* T *si et seulement si pour tout élément* x *de* S, x *est aussi dans* T.

Comment montrer qu'un ensemble est un sous ensemble d'un autre ensemble ? Tout simplement en vérifiant que pour tous les éléments x du premier ensemble, x est aussi présent dans le deuxième. Ainsi le quantificateur «pour tout»($\forall$) apparaît dans des phrases types : pour tout objet ayant une certaine propriété, quelque chose se passe.

Pour montrer (un théorème) qu'un énoncé exprimé sous la forme ci dessus est vrai, on choisit un objet qui a la propriété. Vous cherchez alors, par des dérivations successives, a obtenir que le quelque chose se passe. Si vous réussissez, votre «démonstrateur de théorème»est capable de répéter ce raisonnement pour tous les objets ayant la propriété.

Exercice : utilisez cette méthode pour montrer le théorème suivant :

Proposition 1 *Si* S *et* T *sont deux ensembles définis de la manière suivante :* $S = \{x \in R : (x^2 - 3x + 2) \leq 0\}$ *et* $T = \{x : 1 \leq x \leq 2\}$ *alors* $S = T$.

Il y a également des phrases contenant le mot clef «pour tout»pour lesquelles une méthode de preuve distincte de celle que l'on vient de voir existe. Cette technique s'appelle la *preuve par induction*. Ces phrases sont de la forme : pour tout entier $n \geq 1$, quelque chose se passe. Il s'agit de l'induction sur les naturels. C'est une méthode très utilisée pour montrer que toutes les instances du problème (pour n depuis 1 jusqu'à l'infini) sont vraies.

L'induction

Supposons que la propriété soit P. On procède de la manière suivante : on vérifie que $P(1)$ est vraie, puis vous faites l'hypothèse[6] que $P(n)$ est vraie et vous cherchez à montrer, à partir de cette connaissance, que $P(n+1)$ est encore vraie. Assez souvent, on peut procéder de la façon suivante :

1. écrire $P(n+1)$;
2. puisque l'on a supposé $P(n)$, on essaye d'exprimer $P(n+1)$ en fonction de $P(n)$.
3. conclure en faisant référence à l'hypothèse d'induction ;

Examinons quelques sujets d'exercices sur l'Induction sur les entiers et les invariants. Sopit le programme suivant :

```
function triple(a:integer):integer;
var x,y:integer;
begin
  x:=a;y:=a;
  while y<>0 do begin
    x:=x+2;y:=y-1
  end;
  triple:=x
end;
```

Le nom triple nous laisse à penser que la fonction calcule le triple de l'entier positif a.

problème : soient x_n, y_n les valeurs des variables x et y après n passages dans la boucle. Soit $p(n)$ la proposition : $x_n + 2y_n = 3a$. Montrer que $p(n)$ vraie pour tout $n \geq 0$.

Abordons maintenant un autre exercice. Considérons la suite infinie suivante : $1, 3, 2, 2, 6, 4, 4, 12, 8, 8, 24, 16, 16, 48, 32 \cdots$. Écrire une fonction qui génère la valeur du n-ieme terme, passé en paramètre, de cette suite.

On observe que :

1. la suite commence par 1,3,2

[6] On appelle cela l'hypothèse d'induction

2. pour $n > 3$ le n-ieme terme est le double du terme $n-3$

On écrit alors (en langage C) :

```
int terme(int n);
{
  if (n==1) then return 1;
  else if (n==2) then return 3;
    else if (n==3) then return 2;
      else return 2*terme(n-3);
}
```

On remarque que l'on vient de définir `terme` en fonction de terme. C'est ce que l'on appelle une fonction récursive... et c'est un moyen très élégant de spécifier.

considérons maintenant un troisième exemple. On veut calculer $T(n) = 8+13+18+23+\cdots+(5n+3), (n>0)$ avec $T(0) = 0$. Supposons (conjecture = "guess") que $T(n)$ soit une expression quadratique, disons de la forme $c_1n^2+c_2n+c_3 = G(n)$

- si $n = 0 \Rightarrow c_3 = 0$
- (1) si $n = 1 \Rightarrow 1c_1 + 1c_2 = 8$ (c_3 est nul à priori)
- (2) si $n = 2 \Rightarrow 4c_1 + 2c_2 = 13 + 8$

On multiplie (1) par 2 et on soustrait à (2). On obtient $2c_1 = 5 \Rightarrow c_1 = 2.5$ et $c_2 = 5.5$. Supposons donc que $G(n) = 2.5n^2 + 5.5n$ soit la bonne expression.

Il nous reste à prouver (par induction) que $G(n)$ c'est bien $T(n)$.

Le quatrième problème consite à convertir un nombre décimal en binaire. Nous allons utiliser le principe d'induction sur le nombre de fois qu'une boucle est exécutée. Nous proposons la procédure suivante pour résoudre le problème :

```
procedure Convert-2-bin(n:integer);
{ n est un entier positif, le résultat
  est rangé dans un tableau "b" de
  bits qui représente n en binaire}
var k,t:integer;
begin
  t:=n;k:=0;
  while t>0 do begin
    k:=k+1;b[k]:=t mod 2;t:=t div 2
  end
end;
```

Preuve : l'invariant doit être une expression entre variables indépendantes du nombre de fois où l'on exécute la boucle.

Hypothèse d'induction : si m est un entier représenté par le tableau binaire `b[1..k]`, alors $n = t.2^k + m$

Retour sur le quantificateur «il existe»

On le rencontre dans les phrases du type : il y a un objet avec une certaine propriété tel que quelque chose se passe.

Quand on montre un théorème de la forme $A \Rightarrow B$, on fait l'hypothèse que A est vraie (justifiez le à partir de la définition de implique). Dans ce cas, vous pouvez assurer qu'il existe un objet avec une certaine propriété tel que le quelque chose se passe.

En faisant la preuve vous commencez par dire : «Soit x un objet avec la certaine propriété machin bidule et pour laquelle le quelque chose se passe»... L'existence de cet objet permet (assez souvent) de conclure que B est vraie. Choisir un objet comme on vient de le faire s'appelle une *spécialisation*.

La spécialisation peut aussi s'appliquer de la manière suivante dans des phrases de la forme : A : pour tous les objets avec un certaine propriété, quelque chose se passe.

- faire l'hypothèse que A est vraie et identifier les objets, la propriété, le quelque chose qui arrive.
- prendre un objet particulier (cet objet est souvent le résultat d'une dérivation passée)
- vérifier que l'objet a les propriétés demandées (et il peut avoir plus de propriété que nécessaire)
- conclure, en écrivant une nouvelle phrase, que le quelque chose se passe pour cet objet

Exemple :

Définition 3 *Un nombre réel* u *est une borne supérieure pour un ensemble de nombre réel* T *si, pour tous les éléments* $t \in T, t \leq u$.

Proposition 2 *Si* R *est un sous ensemble d'un ensemble* S *de réels et* u *est une borne supérieure de* S*, alors* u *est une borne supérieure de* R.

Analyse de la preuve : Soit r un élément de R. Par hypothèse R est un sous ensemble de S et donc r est aussi un élément de S. De plus, de nouveau par hypothèse, u est une borne supérieure pour S, donc chaque élément de S est plus petit ou égal à u. En particulier, r est un élément de S donc $r \leq u$.

3.2.4 Imbrication des quantificateurs

On les rencontre dans des phrases types du genre «pour tous les objet x vérifiant P, il existe un y vérifiant Q tel que quelque chose se passe».

Attention, il faut décomposer cette phrase de la manière suivante.

- Objet : x

- La certaine propriété : P
- Le quelque chose qui arrive : il existe un y vérifiant Q tel que quelque chose se passe. Dans cette partie de phrase, vous devez également isoler l'objet, la propriété, le quelque chose qui arrive.

L'ordre d'apparition des quantificateurs a donc de l'importance. Exemple : comparez les deux phrases qui suivent.

1- Pour tous les réels $M > 0$, il existe un élément x de l'ensemble T tel que $x < M$.

2- Il existe un réel $M > 0$ tel que pour tous les éléments x de l'ensemble T, $x < M$.

Méthodologie : quand une phrase contient des quantificateurs imbriqués, utilisez à tour de rôle les méthodes discutées au préalable (induction, spécialisation...) en partant de la gauche vers la droite de la phrase.

Exemple : vous voulez montrer que la proposition suivante est vraie. B : il existe un réel $M > 0$ tel que pour tous les éléments x dans l'ensemble T, $x < M$. Vous pouvez décomposer le problème de la façon suivante puisque le premier quantificateur est un «il existe»i.e. vous devez montrer que :

- B_1: $M > 0$
- B_2 : pour tous les éléments $x \in T$, $x < M$

Quand vous cherchez à montrer B_2, vous avez à choisir un élément $x \in T$ pour lequel vous devez montrer que $x < M$.

Exemple :

Définition 4 *Une fonction a une variable* f *est une application si et seulement si pour tout réel* y*, il existe un réel* x *tel que* $f(x) = y$.

Proposition 3 *Si* m *et* b *sont des réels avec* $m \neq 0$ *alors la fonction* $f(x) = mx + b$ *est une application.*

Analyse de la preuve : (on note, que dans la proposition, il n'y a à première vue ni le quantificateur pour tout ni le quantificateur il existe) la question clef est ici la suivante : comment montrer que la fonction f est une application i.e vous avez à montrer que pour tout réel y, il existe un réel x tel que $mx + b = y$.

On vient de faire apparaître une phrase avec des quantificateurs imbriqués. Le quantificateur qui apparaît le premier (on regarde les choses de la gauche vers la droite) est le quantificateur «pour tout».

On choisit donc un réel y pour lequel on doit montrer qu'il existe un x tel que $mx + b = y$. On reconnaît alors dans la deuxième partie de la phrase précédente le mot clé «il existe» : on va construire l'objet x désiré.

On veut que $mx + b = y$ et on note que $m \neq 0$ par hypothèse. On peut donc construire $x = (y - b)/m$. Il faut alors vérifier que la certaine propriété est vraie

pour ce x et que le quelque chose se passe. On doit donc vérifier que $mx+b = y$ de la dérivation précédente on a :

$$\begin{aligned} mx + b &= m[\tfrac{y-b}{m}] + b \\ &= (y - b) + b \\ &= y \end{aligned}$$

et donc la preuve est complète.
Exercice : écrire la preuve condensée.

3.2.5 La preuve par contradiction

Les méthodes précédentes ne sont pas suffisantes pour montrer le théorème suivant. Soit n un entier et n^2 pair, alors n est pair. On a une phase du type $A \Rightarrow B$.

Avec la méthode par contradiction, vous commencez par faire l'hypothèse que A est vraie. Comment arriver à la conclusion que B est vraie ? Ici, vous vous posez la question pourquoi B ne peut pas être fausse ?. L'objectif de la méthode par contradiction est de découvrir ce(s) raisons. i.e. on assure que A est vraie et que B est fausse ($\neg$B est vraie) et on montre que cela ne peut arriver.

3.2.6 La preuve par contraposée

Comme la preuve par contradiction on fait l'hypothèse que A et $\neg$B sont vraies mais cette fois on part uniquement de la connaissance de $\neg$B vraie. L'objectif est alors de montrer la contradiction que A est fausse.

3.2.7 Écrire la négation d'une phrase avec des quantificateurs

A : Pour tous les objets avec une certaine propriété, quelque chose arrive.
$\neg$A : il existe un objet avec une certaine propriété pour lequel le quelque chose n'arrive pas.
B : il existe un objet avec une certaine propriété tel que quelque chose arrive.
$\neg$B : Pour tous les objets avec la certaine propriété, le quelque chose n'arrive pas.

Vous pouvez utiliser la méthode suivante pour trouver la négation d'une phrase avec des quantificateurs.

1. Mettre le $\neg$ devant la phrase ;

2. Si le $\neg$ apparaît à la gauche d'un quantificateur, déplacez le $\neg$ à la droite du quantificateur et placez le juste devant le «quelque chose arrive». Puis "inverser" les quantificateurs : il existe devient pour tout et pour tout devient il existe.

3. quand tous les quantificateurs apparaissent à la gauche du $\neg$, éliminer le $\neg$ en l'incorporant dans le morceau de phrase à sa droite.

3.2.8 Dérivation rigoureuse de code

Introduction

Nous abordons maintenant une partie qui nous fait revenir au niveau des programmes. La première intention est d'annoter les programmes. Il est d'usage que les notations pour les programmes, les spécifications et les assertions ressemblent à (ici x' est la valeur de x au début du programme) :

```
{ x=x' }
procedure increment(x:in out integer)
-- global: NO
-- modifies x
-- requires ?
-- ensures  x = x'+1
begin x:=x+1 end;
{ x = x'+1 }
```

La première ligne renseigne le lecteur sur le fait que x vaut x' avant de lancer la procédure, la troisième dit qu'il n'y a pas de variable globale utilisée dans la procédure, la quatrième dit que la procédure modifie la valeur du paramètre x, la cinquième dit qu'il n'y a pas de pré-requis sur les valeurs des variables et enfin, la sixième ligne dit que la procédure produit $x' + 1$ et le range dans x.

Voici un autre exemple de notations pour les assertions :

```
{int i;
  /* assert: total=total' */
  for(i=0;i<n;i++) {
  /* assert TOTAL: *total = *total'+sum
     of positive element in A[0..i-1] */
  /* assert I: 0 =< i< n */
     if(A[i]>0)
     /* assert TOTAL,I,A[i]>0 */
        *total = *total + A[i]
  /* assert *total = *total + sum of positive
     elements in A[0..i], and 0=<i<=n */ ;
  }
```

Remarques :

- les notations utilisées doivent être consistantes, uniformes tout au long du programme ;
- les annotations doivent être conçues de sorte qu'un scientifique normalement intelligent puisse les vérifier facilement ;

Pour le langage C, nous pouvons utiliser des clauses assert pour vérifier qu'une propriété est bien vérifiée à un moment donné dans le programme :

```
     #include <assert.h>

     int myfunc(int a, double b)
     {
          assert(a <= 5 && b >= 17.1);
          ...
     }
\endd{verbatim}

\subsubsection{Ce dont on a besoin}

Nous allons utiliser maintenant des notations pour le langage de
sp\'ecification afin de \ggf mod\'eliser\gdf l'it\'eration, la mise en
s\'equence et le choix qui sont les trois concepts fondamentaux (de la
programmation élémentaire). Pour cela nous allons soit utiliser les
notations \ggf \`a la Dijkstra\gdf, soit les notations \ggf \`a la
C\gdf et dans ce cas la diff\'erence entre les boucles do-od et les
boucles while est la quantification sur des instructions gard\'ees, on
y reviendra.

Nous avons aussi besoin de la notion de pr\'edicat. C'est une fonction
bool\'eenne portant sur un espace d'\'etats~; L'ex\'ecution d'une
instruction transforme un \'etat du programme en un autre \'etat du
programme~:

\begin{enumerate}
\item avec une instruction d\'eterministe on obtient un unique prochain \'etat~;
\item avec une instruction non d\'eterministe on obtient un ensemble de prochains
\end{enumerate}

\begin{unedef} Une postcondition est une propriété qui qualifie la sortie du prog
\end{unedef}

\`A partir de là, on peut uniquement s'int\'eres\-ser aux programmes qui
terminent (en v\'erifiant la bonne postcondition) et ce quelque soit
la pr\'econdition. Ceci se modélise par : $$ \{TRUE\} s \{ q\}$$

Nous avons aussi besoin des notions suivantes :

\begin{unedef} {\bf Correction partielle } (Partial correctness) : le
programme produit le bon r\'esultat si son ex\'ecution termine ; {\bf
Correction totale} (Total correctness) : correction partielle $+$
terminaison (le programme produit le bon r\'esultat) \end{unedef}
```

```
\subsubsection{Composition s\'equentielle et le choix}

Nous pouvons maintenant modéliser ces deux premiers concepts. notre
problème est le suivant :

\begin{itemize}

\item Comment prouver $\{p\} s;t \{q\}$ ?
(s\'emantique : si $p$ est verifi\'e lorque que $s;t$ est initi\'e alors
$s;t$ termine et $q$ est alors v\'erifi\'ee.)

\item Le sch\'ema de preuve est le suivant : proposer un pr\'edicat
$r$ tel que~: $$\{p\} s \{r\}\wedge \{r\} t \{q\}$$

\end{itemize}

Pour le choix, notre problème est le suivant :

\begin{itemize}

\item comment prouver $\{p\}$ if $b$ then $c$ else $d$ $\{q\}$ ?

\item Montrer : $$(\{p\wedge b\} c \{ q\})\wedge(\{p \wedge
\neg b\} d\{ q\})$$

\end{itemize}

\subsubsection{Boucle Do-od}

\def\choice{\fbox{\vbox to2mm{\hsize=2mm }}\ }

Cette structure itérative se présenta sous la forme suivante :

{\parindent=0pt do\\
\hbox to 6mm{}\choice $B_0 \Rightarrow S_0$\\
\hbox to 6mm{}\choice $B_1 \Rightarrow S_1$\\
\hbox to 6mm{}\vdots\\
\hbox to 6mm{}\choice $B_{n-1} \Rightarrow S_{n-1}$\\
od\par}

S\'emantique : si une des gardes est v\'erifi\'ee alors choisir (de
mani\'ere non d\'eterministe) une des gardes v\'erifi\'ees et
ex\'ecuter l'action correspondante ; sinon, passer \'a la suite.
```

```
Notre problème est : comment prouver \{p\} do\ldots od \{q\} ? On
procède de la manière suivante :


\begin{itemize}

\item proposer un invariant I:\\ $[p\Rightarrow I]\wedge
(\forall i: 0=<i<n: \{I\wedge B_i\}S_i\{I\})$

\item I est v\'erifi\'e initialement~;

\item I est v\'erifi\'e \`a la fin des it\'erations\ldots de cette
v\'erit\'e (et de petits compl\'ements) on doit pouvoir d\'eduire
$q$. Autrement dit, on doit v\'erifier~:
$$
(I\wedge (\forall i::\neg B_i))\Rightarrow q
$$
($q$ arrive si l'invariant est v\'erifi\'e et aucune des gardes)
\end{itemize}

Pour \^etre totalement rigoureux nous avons besoin de la notion de
borne. On appelle {\it variant} (Variant function) une fonction sur
les naturels utilis\'ees pour caract\'eriser les propri\'et\'es de
{\it terminaison} d'une boucle (elle donne quelque fois une borne
supérieure au nombre d'it\'erations avant la terminaison). On
s'arrange pour utiliser une fonction monotone. Géné\-ra\-le\-ment
c'est un \ggf bout\gdf de l'invariant.  $$(\forall k:(\forall j:\{ I\wedge
B_j\wedge M_j\leq k\} S_j\{M_j< k\}))$$

\subsubsection{Exemple de variant}

Ici la propri\'et\'e essentielle utilis\'ee est : toute suite strictement
d\'ecroissante d'entiers positifs ou nuls est n\'ecessairement finie.

\begin{verbatim}
tant que x<>0 faire x := x-2 fintantque
```

ne termine pas si x est initialement impair. Ceci est détecté par le variant $v = x$. D'après la condition $x \in \{1,2,3,4\cdots\}$ au début ; après la première itération $v \in \{-1,0,1,2,\cdots\}$ et donc v peut ne plus être un naturel !

L'idée est donc la suivante : *tant que l'on n'aboutit pas à un état satisfaisant, faire évoluer l'état en maintenant l'invariant et en diminuant le variant.*

Cas des boucles WHILE

Comment prouver $\{p\}$ while b do s $\{q\}$? On procède de la manière suivante :

- Trouver un invariant I pour lequel $[p \Rightarrow I] \wedge \{I \wedge b\}s\{I\})$
- Variant function M:
  1. $\{I \wedge b \wedge M \leq k\}s\{M < k\}$
  2. $I \Rightarrow M \geq 0$ (ou une autre borne inf)
- Condition de terminaison : $[I \wedge \neg b \Rightarrow q]$

Renforcer/Affaiblir un prédicat

Nous venons de voir que la notion essentielle est la notion de prédicat. Pour trouver le bon invariant par exemple, il est utile de connaître quelques techniques.

Si $P(x_1, \cdots, x_n)" \subset "Q(x_1, \cdots, x_n)$ on dit que le prédicat P est plus restrictif que le prédicat Q ou encore que Q est plus faible que P.

Si le prédicat $P \Rightarrow Q$ alors Q est au moins aussi faible que P c.a.d que Q caractérise tous les états caractérisés par P et éventuellement plus !

Le point de vue du client est : si le programme S' est meilleur que le programme S on écrit $S \sqsubseteq S'$ et on dit que S' est un *raffinement* de S (S est plus général que S').

Exemple informel : besoin d'au moins 4Mo $\sqsubseteq$ besoin d'au moins 2Mo.

Nous allons maintenant travailler sur les notions de pré et post-conditions en reprenant ici les exemples et une partie de la présentation de[4].

Pré-conditions et Post-condition

Le programme S satisfait sa *spécification* (Q, R) si, quand il est exécuté dans un état satisfaisant Q il termine dans un état satisfaisant R :

$$\{Q\}S\{R\}$$

La pré-condition définie des contraintes sur les entrées, les données que le programme doit capter ; la post-condition doit caractériser **ce que** le programme doit assurer.

Programmer de façon systématique c'est effectuer des transformations sur le Q (et/ou R) initial afin de dériver un **invariant** qui identifie les conditions qui doivent être maintenues lorsque certaines variables changent.

Exemples :

$$Q : X \geq 0; R : Y^2 = X$$

Le programme à implanter doit calculer la racine carré de X et la ranger dans Y.

$$Q : s \neq \emptyset; R : e \in s$$

Le programme doit choisir un élément de l'ensemble s qui n'est pas vide.

Une pré-condition et un programme pour une recherche dichotomique dans un tableau trié est :

$Q : 1 \leq N \wedge (\forall j : 1 \leq j < N : A_j \leq A_{j+1}) \wedge A_1 \leq X \leq A_n$

```
i=0;j=N+1;
while(i != j-1) {
   m=div(i+j,2);
   if(A[m]<X) i=m; else j=m;
}
return A[j]==X;
```


Si $N = 0$ ou que X est plus grand strictement... la fonction n'est pas bien définie. Mais dans ce cas Q n'est pas vérifiée : le code doit il tenir compte de ces cas (NON) ?

Une spécification est un code pour une recherche séquentielle est :

$Q : N \geq 1 \wedge (\exists j : 1 \leq j \leq N : X = A_j)$

```
j=1;
while(A[j] != X) j++;
```

Si X n'est pas présent, il y a accès en dehors du tableau – le code est mal conçu (pas assez contraint). Mais Q ne dit pas que X n'est pas présent éventuellement : il faudrait reformuler l'expression de Q pour dire ce qui se passe dans ce cas.

Généralisation d'une pré-condition

Revenons sur l'exemple de la recherche dichotomique de X dans un tableau ordonné avec $N \leq 0$ comme unique contrainte :

$Q : 0 \leq N \wedge (\forall j : 1 \leq j < N : A_j \leq A_{j+1})$

```
i=0;j=N+1;
while(i < j-1) {
   m=div(i+j,2);
   if(A[m]<X) i=m;
   else if(A[m]==X) i=j=m
        else j=m;
}
return i==j;
```

Cette implantation retourne `false` lorsque $N = 0$; c'est ce que nous attendons d'un tableau «vide»!

Post-condition

C'est la partie la plus importante d'une spécification :

- précise exactement ce qu'une fonction fait ;
- utilisée dans une preuve constructive ;
- plus elle est générale plus le programme qui l'établit sera général ;
- plus elle exprime de contraintes moins il y a de manières d'assister le développement du programme ;

> *Programming is a goal-oriented activity.*
>
> **D. Gries**

Quelques comparaisons de post-conditions

Nous allons commencer par étudier le problème d'un tri.

But : trier un ensemble de N entiers codés dans le tableau A. Si le tableau $a[1..N]$ est utilisé pour ranger le résultat, alors une post-condition possible est :

$$R : perm(a, A)$$

$$perm(a, A) \equiv (\forall k : 1 \leq k \leq N :| j : 1 \leq j \leq N : a_k = a_j |= | j : 1 \leq j \leq N : a_k = A_j |)$$

Cette post-condition est trop faible parce qu'il y a beaucoup «trop»de façons d'effectuer les permutations : pour les éliminer, on introduit comme suit une relation qui spécifie que les éléments du tableau sont triés :

$$R_1 : (\forall k : 1 \leq k < N : a_k \leq a_{k+1}) \wedge perm(a, A)$$

Remarque : il n'est pas suffisant d'écrire

$$R_2 : (\forall k : 1 \leq k < N : a_k \leq a_{k+1})$$

car un programme qui établirait $a_1 = a_2 = \cdot = a_n = 0$ à partir d'un jeu de données comportant un 0 serait une solution acceptable !

Intéressons nous maintenant à un problème d'une recherche.

But : recherche de la première occurrence d'un élément X dans une suite (non ordonnée) $< A_1, A_2, \cdots, A_n >$.

$$R : X = A_{i+1}$$

ne nous dit rien sur l'index i ($i = -312$!!!)

$$R_1 : 0 \leq i < N \wedge X = A_{i+1}$$

X est nécessairement un élément de la suite.

$$R_2 : 0 \leq i \leq N \wedge (i = N \vee X = A_{i+1})$$

X n'est pas nécessairement un élément de la suite, dans ce cas $i = N$. S'il est en première position alors $i = 0$, s'il est en dernière $i = N - 1$. Cependant le programme `i=N` satisfait la post-condition et il ne s'agit pas d'une recherche !

Quelques post-conditions types

Bien que les problèmes rencontrés soient très divers, on peut catégoriser deux grandes familles de post-conditions :

1. celle pour laquelle on effectue une recherche ou bien un examen systématique jusqu'à rencontrer un certain état ou encore une (des) condition(s) ;
2. celle pour laquelle on applique un traitement jusqu'à établir une configuration ou un résultat soit pour une donnée initiale soit pour une donnée construite au cours du calcul (approche constructive) ;

Examinons alors quelques exemples :

On veut coder la phrase suivante : Aucun des éléments de $A[1..N]$ n'apparaît plus d'une fois.

nous pouvons aborder le problème en disant : Pas éléments satisfaisant P $\sim$ Tous les éléments satisfont $\neg P$ ce qui se traduit par :

$$(\forall j : 1 \leq j \leq N : \neg P)$$

P : "pas d'occurrence multiple".

$$1 \leq j \leq N \wedge (\forall k : 1 \leq k \leq N : A_j \neq A_k)$$

Attention, on autorise la situation $A_j \neq A_k \wedge j = k$ ce qui est clairement faux ! Donc on écrit :

$$(\forall j : 1 \leq j \leq N \wedge (\forall k : 1 \leq k \leq N : j \neq k \Rightarrow A_j \neq A_k))$$

On veut maintenant coder la phrase : Un seul des éléments de $A[1..N]$ apparaît plus d'une fois.

On approche le problème en disant : Un élément apparaît en position j ou avant :

$$1 \leq j \leq N \wedge (\exists k : 1 < k \leq N : A_j = A_k)$$

Ceci n'est pas assez fort pour garantir «un seul». La notion «avant»n'est pas codée. On regarde alors la première occurrence (gauche) : la relation

$$(\forall k : 0 < k < j : A_j \neq A_k) \wedge 1 \leq j \leq N$$

doit alors être vérifiée et il peut y avoir un ou plusieurs A_j à droite de j :

$$(\exists k : j < k \leq N : A_k = A_j)$$

En combinant ces deux relations, on obtient :

$$| \ (0 < j \leq N \wedge (\forall k : 0 < k < j : A_j \neq A_k) \wedge (\exists k : j < k \leq N : A_k = A_j)) \ | = 1$$

On veut maintenant spécifier la phrase suivante : Quelques uns des éléments de $A[1..N]$ apparaissent plus d'une fois.

À partir de la solution précédente :

$$1 \leq j \leq N \wedge (\exists k : j < k \leq N : A_j = A_k)$$

nb éléments apparaissent plus d'une fois

On remplace "1" par "nb" :

$$nb = | \ (0 \leq j \leq N \wedge (\forall k : 0 < k < j : A_k \neq A_j) \wedge (\exists k : j < k \leq N : A_k = A_j)) \ |$$

Synthèse

Dans les sous-paragraphes pécédents, nous venons de pratiquer les éléments suivants :

- un texte en langage naturel a été Re-formulé avec des quantificateurs ;
- nous avons identifier les propriétés et/ou prédicats du problème ;
- nous avons formaliser ces propriétés ;
- nous avons recomposer le résultat.

Invariant de boucle et spécifications

On l'utilise pour modéliser et/ou caractériser le comportement dynamique des boucles. Il doit énoncer les propriétés essentielles de la boucle qui doivent être **vérifiées** à chaque passage.

Un invariant c'est à la fois une pré-condition et une post-condition : il caractérise les états inititiaux finaux et intermédiaires.

Il doit être utilisé pour raisonner à propos de la terminaison : on peut le voir comme une «espèce»d'hypothèse d'induction.

Examinons quelques problèmes faisant intervenir un invariant de boucle.

Problème : Recherche du maximum m et de sa position p dans un tableau d'entiers $A[1..N]$.

Pré-condition Q – post-condition R – invariant de boucle P_D.

$Q : N \geq 1$

$R : 1 \leq p \leq N \wedge m = A_p \wedge (\forall j : 1 \leq j \leq N : m \geq A_j)$

R ne nous aide pas vraiment pour trouver l'invariant. On va introduire une nouvelle variable libre i à la place de N et l'identité $i = N$

$$R_D : 1 \leq p \leq i \wedge m = A_p \wedge (\forall j : 1 \leq j \leq i : m \geq A_j) \wedge i = N$$

L'invariant est alors dérivé en identifiant les «composants»de R_D rendus vrais par les initialisations sur p, i, m :

$$P_D : 1 \leq p \leq i \wedge m = A_p \wedge (\forall j : 1 \leq j \leq i : m \geq A_j) \wedge 1 \leq i \leq N$$

L'initialisation des variables pour établir P_D est alors

pour $i = 1$ **à** N **faire**

$\uparrow i$ jusqu'à N **et** maintenir P_D

finpour

terminer la boucle avec $i = N$ et P_D encore $=$ TRUE

Vérifions que l'invariant est convenable. Il est clair que les initialisations des variables sont : $p = 1; m = A[p]; i = 1$. En substituant ces valeurs dans P_D, on obtient :

$$1 \leq 1 \leq N \wedge A_1 = A_1 \wedge (\forall j : 1 \leq j \leq 1 : A_1 \geq A_j) \wedge 1 \leq 1 \leq N$$

qui se simplifie en :

$$TRUE \wedge TRUE \wedge A_1 \geq A_1$$

qui est TRUE. Donc, les initialisations établissent l'invariant.

Remarque : on a en quelque sorte prouver le cas de base dans une preuve par induction – *idée de preuve-et-développement.*

Notre tâche maintenant consiste à établir R_D tout en maintenant P_D. Pour l'instant, on s'intéresse à ce qui fait que P_D est le bon invariant.

```
p=1;m=A[p];i=1
```
tantque `i!=N` **faire**
```
    si m>=A[i+1] alors i++
    sinon p=i+1;i++;m=A[i]
    finsi
```
fintantque

Examinons maintenant le problème de la somme des éléments d'un tableau. Les invariants que l'on peut utiliser ici sont :

$R : s = \Sigma(A_j : 0 < j \leq N)$

$R_D : s = \Sigma(A_j : 0 < j \leq i) \wedge i = N$

$P_D : s = \Sigma(A_j : 0 < j \leq i) \wedge 0 \leq i \leq N$

Le programme dérivé à partir de P_D est alors le suivant :

```
i=0;s=0
```
tantque `i!=N` **faire**
```
    i++;s=s+A[i];
```
fintantque

$\{P_D \wedge i = N\}$

On veut maintenant une implantation pour la spécification $Q : N \geq 0$ et $R : a = N! \times N^N$. On écrit alors :

(le programme qui suit ne termine pas – il a été construit sans invariant)

```
i=0;a=1;N2=2*N
```
tantque `i!=N2` **faire**
  si `i<N` **alors** `i++;a=a*i;`
  sinon `a=a*i;`
  finsi
fintantque

$\{R : a = N! \times N^N\}$

On peut en effet atteindre un état où i n'est pas changé.

On propose alors l'amélioration suivante :

```
i=0;a=1;c=0;N2=2*N
```
tantque `i!=N2` **faire**
  si `i<N` **alors** `i++;a=a*i;c=i;`
  sinon `a=a*c;i++`
  finsi
fintantque

$\{R : a = N! \times N^N\}$

L'invariant associé est très difficilement exprimable :

$$P : 0 \leq i \leq N2 \wedge (0 \leq i \leq N \wedge (a = i! \wedge c = i)) \vee (N < i \leq N2 \wedge (a = N! \times N^{i-N} \wedge c = N))$$

Attention, les variables a, c sont caractérisées de deux manières différentes (on a essayé de calculer à part $N!$ et N^N). Ne convient pas !

Pour concevoir le bon invariant, on part de la post-condition $R : a = N! \times N^N$ que l'on transforme en une nouvelle post-condition R_D (où $R_D \Rightarrow R$)

$$R_D : a = i! \times N^i \wedge i = N$$

Cette post-condition permet de dériver l'invariant de boucle P_D.

$$P_D : a = i! \times N^i \wedge 0 \leq i \leq N$$

L'implantation est alors la suivante :

```
i=0;a=1;
```
tantque `i!=N` **faire**
  `i++;a=a*N*i;`
fintantque

$\{R : a = N! \times N^N\}$

Problème de synthèse : partage d'un ensemble

Nous étudions maintenant une stratégie pour imposer un «ordre partiel»entre des éléments. Les contraintes sont moins fortes que pour un tri.

Problème : soit X un élément, on vous demande de trouver un indice i pour partager le tableau $A[1..N]$ (tous les éléments sont distincts) en utilisant $a[1..N]$

de tel sorte que à gauche de i on trouve tous les éléments $\leq X$ et à droite tous les éléments $> X$.

$Q : N \geq 0$

$R : 0 \leq i \leq N \wedge (\forall p : 0 < p \leq i : a_p \leq X) \wedge$
$(\forall q : i+1 \leq q < N+1 : a_q > X) \wedge perm(a, A)$

Cette post-condition n'est pas facile à établir : on va remplacer $i+1$ par j (nouvelle variable : espace d'états plus grand) et on obtient :

$R_D : 0 \leq i \leq N \wedge (\forall p : 0 < p \leq i : a_p \leq X) \wedge$
$(\forall q : j \leq q < N+1 : a_q \geq X) \wedge perm(a, A) \wedge j = i+1$

$S_0(X_0)$ **Initialisations :**

$\{Q\}$

```
i=0;j=N+1;a:=A;
```

si `j=i+1` **alors** R_D établit

sinon $\{Q_1\}S_1(X_1)\{R_D\}$ (backward process)

finsi

La question qui arrive maintenant est de savoir quel sous ensemble d'objets parmi $\{i, j, a\}$ doit on changer pour établir $j = i+1$ ce qui permet de rendre R_D vraie (et donc de terminer).

On peut soit faire croître i soit faire décroître j. On prend la première solution ce qui conduit à :

$P'_1 : (0 \leq i \leq N) \wedge (j = N+1) \wedge (a = A) \wedge$
$(\forall p : 0 < p \leq i : a_p \leq X) \wedge (\forall q : j \leq q < N+1 : a_q > X)$

On va «itérer sur i». Mais rien ne garantit que $a_{i+1} \leq X$ à l'itération i ; ainsi dans le cas ou $a_{i+1} > X$ il faut permuter les deux valeurs. En autorisant aussi j à décroître on obtient :

$P''_1 : 0 \leq i < j \leq N+1 \wedge (\forall p : 0 < p \leq i : a_p \leq X)$
$\wedge (\forall q : j \leq q < N+1 : a_q > X)$

$S_1(X_1)$ **premier raffinement** $X_1 = \{i, j, A\}$ **:**

```
i=0;j=N+1;a:=A;
```

si `j=i+1` **alors** R_D établit et donc **fin**

sinon

si `a[i+1]<=X` **alors**

```
    i++
```

sinon `swap(a[i+1],a[j-1]);j=j-1;`

finsi $\{P''_1\}$

si `j=i+1` **alors** R_D établit et **fin**

sinon $\{Q_2\}S_2(X_2)\{R_D\}$ (backward process) **finsi**

finsi

Pour établir R_D il est clair que l'on peut mettre en œuvre soit une boucle, soit un appel récursif (on a dégagé la structure de contrôle principale du programme) pour réaliser :

$P''_1 \wedge j = i+1 \Rightarrow R_D, \quad P''_1 \wedge j \neq i+1 \sim Q_2$

$S_2(X_2)$ **deuxième raffinement** $X_2 = \{i, j, A\}$ **:**

```
i=0;j=N+1;a:=A;
```

tantque `j!=i+1` **faire**
  si `a[i+1]<=X` **alors**
    `i++`
  sinon `swap(a[i+1],a[j-1]);j=j-1;`
  finsi
fintantque
$\{R_D\}$

Critique : cette implantation conduit à bouger éventuellement des éléments qui sont déjà à leurs places dès le départ. Exemple avec $X = 5$ et $A[1, 2, 6, 4, 7, 8]$.

3.2.9 Introduction aux types abstraits algébriques

Dans cette sous-partie nous nous intéressons à écrire des équations qui rendent compte des propriétés des systèmes. Il s'agit de la manière de faire des spécifications algébriques qui vise à construire des types de données et nous reprenons le discours et certains exemples depuis [14].

Continuons cette introduction par un autre exemple. Étant donnés deux tableaux `A[1..M]` et `B[1..N]` d'élements tous distincts (cependant, on peut retrouver un élément dans les deux tableaux à la fois), écrivez une proposition, au moyen des quantificateurs et des opérateurs logiques, qui rend compte, dans un tableau `D` de l'union (au sens ensembliste) de `A` et `B`.

Solution 1 : On cherche à construire une application de `Tableau A` $\rightarrow$ `Ensemble ens(A)` avec `#A = #ens(a)`. Si A et B sont deux tableaux, un tableau D représentant $Ens(A) \cup ens(B)$ peut être spécifié comme suit :
i) #D=#$ens(A)$+ #$ens(B) - \#(ens(A) \cap ens(B))$
ii) $\forall i \in [1 \cdots \#A], D[i] = A[i]$
iii) Soit $B - A = [B[j] : j \in [1 \cdots \#B]$ tel que $B[j] \notin Ens(A)]$ alors $\forall j \in [1 \cdots \#B - A], D[\#A + j] = (B - A)[j]$

Vous pouvez écrire le programme.

Solution 2 : on pré-suppose un ordre sur les éléments qui sont des entiers. Le tableau résultat a une propriété supplémentaire : il est trié en ordre croissant.

On commence par qualifier un tableau `C[1..M+N]` :

$$(\forall i)(1 \leq i \leq M, C[i] = A[i]) \wedge (\forall j)(1 \leq j \leq N, C[M + j] = B[j])$$

(Écrire la proposition précédente avec un seul symbole $\forall$)

On qualifie maintenant un tableau `D[1..M+N]` qui est une permutation ordonnée de `C[1..M+N]` :

$$(\forall i)(1 \leq i < M + N, D[i] \leq D[i + 1]) \wedge perm(D, C)$$

Maintenant le nombre déléments dans l'union est donné par :

$$p = | (i, 1 \leq i < M + N, D[i] < D[i + 1]) | +1$$

L'union des deux tableaux peut maintenant être qualifiée par les deux prédicats suivants :

$(\forall i)(1 \leq i \leq p, union[i] \in A[1..M] \vee union[i] \in B[1..N])$
$(\forall i)(1 \leq i < M + N, D[i] < D[i+1] \Rightarrow D[i] \in union[1..p]) \wedge (D[M+N] \in union[1..p])$

Vous pouvez écrire le programme en codant les équations ci-dessus qui préfigurent le travail à réaliser pour écrire des spécifications algébriques. Revenons donc à la notion de spécification algébrique. Les constituants d'une spécification algébrique sont :

- une **signature** qui décrit la syntaxe du type (nom + type des arguments – sortes) ;
- une description des propriétés des opérateurs du type par des **axiomes** – donner une sémantique aux noms de la signature – exprimer des contraintes sur le domaine de définition de la fonction définie ;

Considérons la spécification suivante du type Vecteur :
SORTE : Vecteur
UTILISE : Element, Entier, Boolean
OPÉRATIONS :
ième : Vecteur $\times$ Entier $\rightarrow$ Element
changer-ieme : Vecteur $\times$Entier $\times$ Element $\rightarrow$ Vecteur
borne-sup : Vecteur $\rightarrow$ Entier
borne-inf : Vecteur $\rightarrow$ Entier
True : $\rightarrow$ Boolean
$\wedge$: boolean $\times$ boolean $\rightarrow$ boolean
borne-inf(v) $\leq$ i $\leq$ borne-sup(v) $\Rightarrow$
AXIOME :
ieme(changer-ieme(v,i,e),i) =e

On interprète le début de l'axiome de la façon suivante : pour tous les i compris entre... alors la partie à doite du symbole implique s'applique.

Noter que l'écriture précédente n'est pas **purement** équationnelle puisque l'on exprime des contraintes définies par une **pré-condition** sur le domaine de définition de i-eme

Cependant, celà est plus constructif. Par contre, on n'a pas de construction if then else dans les axiomes et on fera un simulation avec plusieurs axiomes comme ci dessous :
borne-inf(v) $\leq$ i $\leq$ borne-sup(v) &
borne-in(v) $\leq$ j $\leq$ borne-sup(v) & i $\neq$ j
$\Rightarrow$ ieme(changer-ieme(v,i,e),j)=ieme(v,j)
(Seul le i^{eme} élément à changé)

Nous avons aussi besoin de quelques définitions très importantes :

Définition 5 *Opérations internes : ce sont celles qui rendent un résultat d'une sorte définie.*

Définition 6 *Observateur : une opération est un observateur si elle a au moins un argument d'une sorte définie et si elle rend un résultat d'une sorte prédéfinie : ieme.*

Définition 7 *Les générateurs : les opérations qui sont ! Par exemple, on construit les naturels à partir de la constante 0 et du générateur* succ.

À partir de ces définitions, nous pouvons maintenant énoncer quelques propriétés que l'on demande aux spécifications algébriques :

1. N'y a t'il pas d'axiomes contradictoires ? (consistance) ;
2. A t'on donné suffisamment d'axiomes pour décrire toutes les propriétés du type que l'on veut spécifier au départ ? (complétude des axiomes)

La complétude est pour l'informaticien un critère trop fort, on se contente alors de la notion de complétude suffisante qui s'énonce comme suit : les axiomes doivent permettre de décrire une valeur d'une sorte prédéfinie pour toute application d'un observateur à un objet d'une sorte définie.

Autrement dit : le membre droit d'une équation quelconque ne peut pas se réduire au membre gauche d'une autre équation (chaque équation caractérise des objets différents).

La **propriété à vérifier** dans l'écriture des axiomes pour assurer la complétude suffisante et puisque les objets sont obtenus par les opérations internes, est qu'il faut écrire des axiomes qui définissent le résultat de la composition des observateurs avec toutes les opérations internes.

Comme les opérations ne sont pas définies partout, on doit pouvoir déduire une valeur pour tous les observateurs sur tout objet de sorte définie appartenant au domaine de définition de cet observateur.

Nous développons maintenant un exemple complet. On rappelle qu'un multi-ensemble est un ensemble avec répétition(s). Pour la suite, x et y sont de sorte Element et m de sorte Multi-Ensemble. La spécification de départ est :

SORTE Multi-Ensemble
UTILISE Element, Booleen
OPÉRATIONS
M-ens-vide : $\rightarrow$ Multi-Ensemble
ajout : Multi-Ensemble $\times$ Element $\rightarrow$ Multi-ensemble
sup : Multi-Ensemble $\times$ Element $\rightarrow$ Multi-ensemble
app : Element $\times$ Multi-Ensemble $\rightarrow$ Booleen
AXIOMES
1- app(x,M-ens-vide) = faux
2- $x = y \Rightarrow app(x, ajout(m, y)) = vrai$
3- $x \neq y \Rightarrow app(x, ajout(m, y)) = app(x, m)$
4- sup(M-ens-vide,x) = M-ens-vide
5- $x = y \Rightarrow sup(ajout(m, y), x) = m$

6- $x \neq y \Rightarrow sup(ajout(m, y), x) = sup(m, x)$

Attention : on peut aussi choisir (c'est même un peu plus parlant) l'axiome 6 de la manière suivante :

6- $x \neq y \Rightarrow sup(ajout(m, y), x) = ajout(sup(m, x), y)$

Dans cette spécification, est-ce que l'on supprime toutes les occurrences d'un élément ? la spécification est-elle consistante ? Que se passe t-il si nous ajoutons les axiomes :

7- $x = y \Rightarrow app(x, sup(m, y)) = faux$

8- $x \neq y \Rightarrow app(x, sup(m, y)) = app(x, m)$

Qu'en est-il de la complétude ?

Donner les spécifications de l'opération card : Multi-Ensemble $\rightarrow$ Entier, qui donne le nombre d'éléments d'un multi-ensemble et celle de l'opération nb-occurrences : Element $\times$ Multi-Ensemble $\rightarrow$ Entier qui donne le nombre d'occurrences d'un élément donné dans un multi-ensemble.

Solution : la spécification donnée recherche la *première occurrence* de l'élément cherché. si l'on veut enlever toutes les occurrences, on peut alors écrire :

4 sup(m-ens-vide,x)=m-ens-vide

5' $x = y \Rightarrow$ sup((ajout(m,y),x)=sup(m,x)

6' $x \neq y \Rightarrow$ sup((ajout(m,y),x)=ajout(sup(x,m),y)

de plus elle est consistante, celà provient intuitivement du fait que le membre gauche d'un équation quelconque ne peut pas se réduire, par application d'un ou plusieurs axiomes, au membre gauche d'une autre équation : chaque équation caractérise des objets différents. mais si on ajoute les axiomes :

$x = y \Rightarrow app(x, sup(m, y)) = faux$

$x \neq y \Rightarrow app(x, sup(m, y)) = app(x, m)$

alors l'évaluation de app(4,sup(ajout(ajout({{ }},4),4),4)) conduit à dériver soit vrai soit faux. en effet, on a :

app(4,sup(ajout(ajout({{ }},4),4),4)) = app(4,ajout({{}},4)) (par l'axiome 5)
= vrai (par axiome 2)

mais on peut aussi faire les identifications suivantes à partir de l'axiome 7 rajouté : $x = 4$, $y = 4$, $m = ajout(ajout(\{\{\ \}\}, 4), 4)$. ainsi l'expression app(4, sup(ajout(ajout({{ }},4),4),4)) = faux par application de l'axiome 7 : la spécification n'est pas consistante (cqfd).

On a déduit à la fois le «vrai»et le «faux»donc la spécification n'est pas *consistante*.

Complètude : cette partie présente un exemple complet de démonstration de complètude d'un type abstrait (en fait, on montre une propriété de suffisante complétude). on rappelle qu'une spécification d'un type abstrait de données t est *suffisament complet* si, pour tout terme de la forme $f(x_1, \cdots, x_n)$ où f est un observateur, il existe un théorème démontrable à partir des axiomes du type t de

la forme $f(x_1, \cdots, x_n) = u$ où u est un terme qui ne contient pas d'opérations de t (u est un terme externe – de sorte prédéfini). ceci est encore équivalent à dire que tout terme $f(x_1, \cdots, x_n)$ se *réécrit* en un terme u externe en considérant l'ensemble des axiomes comme des règles de réécriture.

lorsqu'on assure une propriété de suffisante complétude, on a décrit complètement le comportement du type vis à vis des observateurs du type i.e ceux qui rendent compte des propriétés du type, sans se préocuper de la description des opérations internes qui ne permettent pas d'observer de propriétés.

la spécification de multi-ensemble donnée en exercice est suffisamment complète.

Preuve : il s'agit de prouver que, pour tout élement $e \in element$ et pour tout terme u de type multi-ensemble on peut démontrer l'un des deux théorèmes :

app(e,u)=faux ou app(e,u)=vrai

on raisonne sur la longueur du terme u (le nombre d'opérations qui composent le terme).

1. si u est le multi-ensemble vide alors l'application de l'axiome 1- nous donne le résultat ;

2. sinon, supposons que u est au moins un terme de longueur 1, deux cas sont a envisager :

   - u=ajout(v,e') : le résultat se déduit directement à partir de l'axiome 2- ou par hypothèse de récurrence (app(e,v') est suffisamment complet) appliqué sur l'axiome 3- ;
   - u=sup(v,e') : alors si v est le multi-ensemble vide ou v=ajout(w,e") alors le résultat est une conséquence des axiomes 4- 5- 6- et de l'hypothèse d'induction ; sinon il existe un entier $n \geq 1$ tel que $v = sup^n(w, e'') = sup(sup(\cdot sup(t, e_n), e_{n-1}), \cdot e_1)$ avec t étant soit le multi ensemble vide soit un terme ajout(m,e) :

     (a) lorsque t est le multi ensemble vide, le résultat est une conséquence de l'application de l'axiome 4- ;

     (b) lorsque t=ajout(m,e) alors si $e = e_n$ alors $v = sup^{n-1}(m, \cdots)$ qui a la propriété de suffissante complétude par hypothèse d'induction, sinon $v = sup^n(m, \cdots)$ et là aussi on déduit le résultat par hypothèse d'induction ;

exercice : discuter de la complétude de la spécification précédente lorsque l'on remplace l'axiome 6) par l'axiome suivant :

$$6') \quad x \neq y \Rightarrow sup(ajout(m, x), y) = ajout(sup(m, y), x)$$

que veut t'on signifier par là ?

Extensions du type : pour que l'enrichissement maintienne les propriétés de consistance et de complétude, il nous faut appliquer card sur les opérations de suppressions et d'ajouts. des axiomes possibles pour card sont :
card(m-ens-vide) = 0
card(ajout(x,m)) = card(m) + 1
$(x \in m) = vrai \Rightarrow$ card(sup(x,m)) = card(m)-1
$(x \in m) = faux \Rightarrow$ card(sup(x,m)) = card(m)

dans ce cas, on obtient, avec les axiomes précédents, une inconsistance. en effet, $sup(3, ajout(\{\}, 4)) = sup(3, \{4\})$ (axiome 6) ; ce qui ce réduit encore à $\{\}$ par l'axiome 4. donc, on peut conclure, en évaluant le terme interne d'abord que $card(supp(3, ajout(\{\}, 4))) = (card(sup(3, \{4\})) = card(\{\}) = 0$.

mais avec les nouveaux axiomes précédemment introduits nous avons aussi : $card(supp(3, ajout(\{\}, 4))) = card(ajout(\{\}, 4)) = card(\{\}) + 1 = 1$ ce qui amène la contradiction et l'inconsistance. on peut alors proposer de réécrire les axiomes pour le cardinal comme suit :

- card(m-ens-vide) = 0
  - card(ajout(x,m)) = card(m) + 1
  - $x = y \Rightarrow card(sup(ajout(m, y), x)) = card(m)$
  - $x \neq y \Rightarrow card(sup(ajout(m, y), x)) = card(sup(x, m)) + 1$

mais on peut ici remarquer que l'on ne peut rien dire du cardinal d'un terme comme sup(sup(...)). (à faire)

Les axiomes pour nb-occurences :
nb-occurrences(x,m-ens-vide) = 0 1)
$(x = y) \Rightarrow$ nb-occurrences(x,ajout(y,m)) = nb-occurrences(x,m) + 1 2)
$(x \neq y) \Rightarrow$ nb-occurrences(x,ajout(y,m)) = nb-occurrences(x,m) 3)
nb-occurrences(x,m) =0 $\Rightarrow$ nb-occurrences(x,supp(x,m)) = 0 4)
nb-occurrences(x,m) $\geq$ 1 $\Rightarrow$ nb-occurrences(x,supp(x,m)) = nb-occurrences(x,m) - 1 5)
$(x \neq y) \Rightarrow$ nb-occurrences(x,supp(y,m)) = nb-occurrences(x,m) 6)

(les trois axiomes 4 et 5 ne sont pas convaincants car les préconditions portent sur nb-occurrences !)

3.2.10 Comment poser et résoudre un problème

Cette sous-partie est très générale dans le sens où elle ne fait pas référence explicitement à des techniques spécifiques à l'informatique et elle s'appuie sur les éléments de Polya [2] pour comprendre comment analyser un problème.

Cependant, il s'agit bien toujours en arrière plan d'appréhender de façon systématique l'analyse et la programmation par l'étude des principaux raisonnements [1] qui guident la construction des logiciels.

Pour résoudre un problème vous devez successivement :

1. Comprendre le problème ;
2. Concevoir un plan ;
   - trouver le rapport entre les données et l'inconnue ;
   - vous pouvez être obligés de considérer des problèmes auxiliaires si vous ne pouvez trouver un rapport immédiat ;
   - vous devez obtenir finalement un plan de la solution ;
3. Mettre le plan à exécution ;
4. Examiner la solution obtenue et «la critiquer»;

Vous pouvez commencer à vous poser les questions suivantes :

- D'où faut'il partir ? (de l'énoncé)
- Que faire ?
  - ne pas s'occuper des détails pour le moment ;
  - dégager les points principaux et les combiner pour établir des rapports – examen sous différents angles – hypothèse et conclusion (pb à démontrer) – l'inconnue, les données, les conditions (pb à résoudre) ;
    1 soit garder l'inconnue et changer les données et la condition ;
    2 soit garder les données et changer l'inconnue et la condition ;
    3 soit changer à la fois l'inconnue et les données ;

On peut alors raffiner quelques points essentiels :

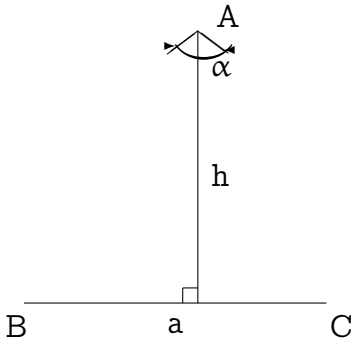
(i) et (ii) «bien regarder l'inconnue»concerne les problèmes qui ont une même inconnue – ne garder qu'une partie de la condition, négliger l'autre – mais ne pas introduire de clauses ou de données nouvelles ;

(iii) introduire une nouvelle inconnue utile et plus accessible : elle s'obtient à partir des données primitives (pouvez-vous tirer des données un élément utile ?) **Exemple :** trouver le centre de gravité d'un cube : avec nouvelle inconnue "centre de gravité d'une face"

Une façon intéressante de changer à la fois l'inconnue et les données consiste à remplacer l'inconnue par l'une des données et vice et versa.

Exemple : Construire un triangle connaissant un coté a, la hauteur h perpendiculaire à a et l'angle α opposé au côté a.

Ce problème de la construction d'un triangle (qui vient de Polya dans [2]) sera repris dans les études de cas en spécialisant une solution. Le problème est décrit dans le dessin qui suit :



Inconnue : un triangle
Données : deux longueurs a et h, un angle α

Idées : ramener le problème à la construction du point A après avoir tracé le segment BC.

On obtient alors un nouveau problème :

- **inconnue** : le point A
- **données** : h, α et 2 points B et C
- **condition** : la distance du point A à la droite BC mesurée perpendiculairement à cette droite doit être égal à h et $\widehat{BAC} = \alpha$.

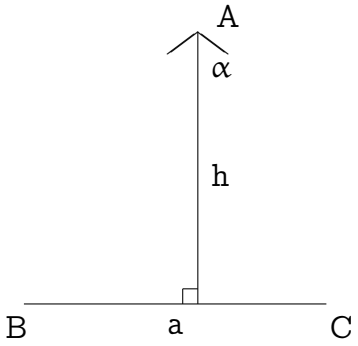
On a changé à la fois l'inconnue et les données.

La *condition* comprend deux parties. Si l'on néglige la 2ème partie, les points situés sur la droite parallèle à BC à la distance h de cette droite satisfont la condition.

Le lieu géométrique des points qui satisfont la 2ème partie est l'arc de cercle dont les extrémités sont B et C.

L'intersection de ces deux lieux géométriques est le point recherché.

Le dessin qui suit illustre la construction du point A



1. On trace une droite quelconque Δ qui passe par B.
2. Δ coupe le cercle en un point P obtenu en faisant glisser un triangle d'angle α le long de Δ jusqu'à rencontrer C.

3. Il reste à déterminer le cercle circonscrit au triangle BPC : nouveau problème plus simple que l'on sait résoudre.

    - on considère les *médiatrices* des segments BP et PC

Nous reprendrons ce problème dans les études de cas avec pour objectif de construire un programme qui dessine la solution dans une fenêtre graphique de l'écran de l'ordinateur. nous avons donc besoin maintenant de nous attarder sur un langage de programmation particulier pour avoir les moyen de programmer cette solution graphique.

3.3 Le langage Python

LE LANGAGE de programmation Python est introduit ici pour mettre en œuvre les solutions logicielles des études de cas qui seront exposées dans le chapître qui suit. Ce langage a de bonnes vertus de base pour l'enseignement des langages notamment celle d'être disponible pour de nombreuses plate-formes (Unix, Windows, MacOS) et d'être libre. Python a aussi l'avantage d'être élégant parce que dépourvu de sucre syntaxique autant que possible. Par exemple la notion de pointeur qui est juste une notion technique permettant d'accéder à une structure dynamique est absente du langage. La notion de marqueur de bloc permettant de repérer un bloc de code sur lequel porte une instruction est contournée par le biais d'une indentation appropriée.

Python est un langage interprété et interactif (toute phrase dans le langage peut être immédiatement évaluée sans avoir à attendre le texte complet du programme), orienté objet (le programmeur peut compter sur des structures de données évoluées pour organiser ses programmes et ses données). Il est souvent comparé à des langages comme Perl, Tcl, Scheme ou Java. Il peut être vu comme un langage de scripts c.à.d des programmes qui réalisent des tâches simples sur des données textuelles par exemple ou encore comme un langage pour des applications qui ont besoin d'une interface de programmation.

Concernant son orientation objet, Python permet donc de structurer les données et les codes à partir des notions de modules, classes, exceptions, types de données dynamiques que nous détaillerons utérieurement. Le site de référence du langage est `http://www.python.org`.

3.3.1 Les éléments de base

Sous Unix, on lance l'interpréteur de commande Python de la façon suivante :

```
[christophe@localhost licenc]$ python
Python 2.3a1 (#4, Jan 19 2003, 12:22:48)
[GCC 3.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> help()
```

```
Welcome to Python 2.3!  This is the online help utility.

If this is your first time using Python, you should definitely check out
the tutorial on the Internet at http://www.python.org/doc/tut/.

Enter the name of any module, keyword, or topic to get help on writing
Python programs and using Python modules.  To quit this help utility and
return to the interpreter, just type "quit".

To get a list of available modules, keywords, or topics, type "modules",
"keywords", or "topics".  Each module also comes with a one-line summary
of what it does; to list the modules whose summaries contain a given word
such as "spam", type "modules spam".

help>
```

ε

On remarque à la lecture des quelques lignes ci dessus que de l'aide en ligne est disponible (à partir du moment ou elle a été installée) à tout moment. Les commandes qui suivent vous montrent comment interagir avec Python (sortir de l'interpréteur) et réaliser quelques commandes. Quand vous avez les symboles >>> cela veut dire que c'est à vous de rentrer du texte. On dit que l'on est sous le prompt d'invite de Python.

```
[root@m11 Python-2.2.1]# /usr/local/bin/python
Python 2.2.1 (#1, Aug 20 2002, 09:56:09)
[GCC 2.96 20000731 (Mandrake Linux 8.1 2.96-0.62mdk)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> quit
'Use Ctrl-D (i.e. EOF) to exit.'
>>> quit
'Use Ctrl-D (i.e. EOF) to exit.'
>>> print 'bonjour'
bonjour
>>> 12 + 3
15
>>> x = 12 * 2
>>> x + 1
25
>>> a = 3
>>> b = 4
>>> b / 2 + a
5
>>> b / (2+a)
0
>>> b / (2.0 + a)
0.80000000000000004
>>> 99999999999999999999999999 + 1 # ATTENTION LONG INT en 2.2
100000000000000000000000000L
>>> a = 4 ; b = 5
```

```
>>> a * b
20
>>> b % a
1>>> a % b
4
>>> print 'Le %s est %d' % ('résultat',1)
Le résultat est 1
```

L'exemple qui suit vous montre comment ranger la valeur 1 dans les variables a, b, c, de faire un affichage au moyen de l'instruction print puis de construire une conditionnelle (un choix). Remarquez que les symboles ... sont donnés par Python pour vous signaler que la phrase (ici la construction if-then-else) n'est pas terminée. Vous n'avez pas à donner ces symboles mais à terminer la construction de choix.

Vous noterez le mot clé `elif` qui est un équivalent pour `else if`. Enfin, vous noterez comment réaliser vos premières opérations booléennes et que l'opérateur d'égalité (pour les entiers) est ==.

Par ailleurs, il est très important de noter que l'indentation a un rôle particulier en Python : elle permet de se passer de certains mots clés comme ceux servant à délimiter un bloc de texte. Vous indentez vos codes dès lors que vous souhaitez construire un nouveau bloc d'instructions sous influence d'un bloc englobant. Dans l'exemple, les blocs d'instructions sont réduits à la seule instruction `print`.

```
>>> a = b = c = 1
>>> print a, 'bonjour', c
1 bonjour 1
>>> a = 1; b=1; # ceci est un commentaire
>>> if a == 1:
...     print 'a vaut 1'
... elif a==2:
...      print 'autre' # il faut mettre une indentation
...
a vaut 1
>>> if 1<2<3: print 'Hello'
...
Hello
>>> if 1 < 2 and 'fifi' == 'fifi': print 'Okay'
...
Okay
>>> if not 5 == 5 or (1 < 2 and 3 != 4): print 'Hello'
...
Hello
>>>
```

L'exemple qui suit permet un dialogue avec l'utilisateur via un appel à la primitive `raw_input` qui permet de saisir un entier. On effectue ensuite une vérification sur la valeur de l'entier saisi. Vous noterez le mot clé `else` qui a la signification «dans tous les autres cas, faire»

```
>>> x = int(raw_input("Please enter #:"))
Please enter #:4
>>>
if x < 0:
        x = 0
        print('Negative changed to zero')
elif x == 0:
        print('Zero')
elif x == 1:
        print('Single')
else:
        print('More')... ... ... ... ... ... ... ...
...
More
>>>
```

Note : il n'y a pas d'instruction case (cas parmi, comme pour le langage C)

3.3.2 Les itérations

Examinons maintenant les structures syntaxiques permettant d'itérer un traitement.

```
>>> i = 0
>>> while i < 3:
...     print 'Hello\n'; i = i + 1
...
Hello

Hello

Hello

>>>
>>> y = 17 ; x = y / 2
>>>
while x > 1:
        if y % x == 0:
                print y, ' has factor ', x
                break
        x = x - 1
else:
        print y, ' is prime'... ... ... ... ... ...
...
17  is prime
>>> # noter l'indentation du else: vis à vis du while
>>> for i in range(3): print 'Une fraise !'
...
Une fraise !
```

```
Une fraise !
Une fraise !
>>>
```

La boucle `for` permet d'itérer, ici pour les valeurs de `i` comprise entre 2 et 10 (non compris) dans le cas de la boucle externe. sur l'exemple qui suit on remarquera également l'instruction `break` qui fait sortir de la boucle qui l'englobe ainsi que l'instruction `else` qui permet d'exécuter du code dès que `x` n'est plus dans la plage de valeur spécifiée, c'est à dire à la sortie de boucle.

```
>>> for n in range(2,10):
        for x in range(2,n):
                if n % x == 0:
                        print n, 'equals', x, '*', n/x
                        break
        else:
                # loop fell through without finding a factor
                print n, 'is prime'... ... ... ... ... ... ...
...
2 is prime
3 is prime
4 equals 2 * 2
5 is prime
6 equals 2 * 3
7 is prime
8 equals 2 * 4
9 equals 3 * 3
>>>
```


3.3.3 Types de données prédéfinis

Nous pouvons maintenant commencer à utiliser les types de données prédéfinis (nous n'examinons pas les types entiers, réels etc). Rappelons qu'un type de donnée est un élément organisationel des données qui est muni d'opérations de gestion (typiquement l'ajout et la suppression d'un élément) voire d'itérateurs qui permettent de parcourir la structure. Les types de données livrés en standard dans le langage sont :

- Les types «liste de, dictionnaire de, tuple de»qui sont donc directement accessibles à l'utilisateur.
- L'utilisateur peut définir ses propres types de données abstraits via les notions de `modules, classes et extensions C`.

Commençons par passer en revue les listes. Les listes se notent comme dans d'autres langages (par exemple comme en SML) et on remarquera dans les exemples qui suivent que l'instruction `for` est un itérateur générique, ici permettant le parcours des éléments d'une liste. Dans le premier exemple qui suit on remarquera

également opérateur `in` qui permet de tester si un élément est présent dans une liste.

```
>>> L = ['ABC', 'DEF', 'GHE']
>>> for x in L: print x
...
ABC
DEF
GHE
>>> for x in 'Hello world': print x
...
H
e
l
l
o

w
o
r
l
d
>>> x = 'hello' ; y = 'world' ; L = [x, y]
>>> 'hello' in L
1  # 1<=> true
>>> 'helo' in L
0  # 0 <=> false
```

Le deuxième exemple illustre également l'itérateur `for` sur les listes ainsi que l'opérateur `range` qui retourne une liste d'entiers puis les opérations de gestion de listes `append` (concaténation), `remove` (suppression). On remarquera également la notation point-fixée permettant d'appliquer à une structure de donnée considérée une opération de gestion.

```
>>> L = ['C', 'i', 'w', 'u', 'n', 'e']
>>> for x in 'Christophe cerin':
        if x in L:
                print x,... ...
...
C i e e i n
>>>
>>>
>>>
>>> range(5)
[0, 1, 2, 3, 4]
>>> range(1,3)
[1, 2]
>>> r = []
>>> low = 0; high = 5
```

```
>>> while low<high:
...     r.append(low); low = low + 1
...
>>> r
[0, 1, 2, 3, 4]
>>> r.append(6)
>>> r
[0, 1, 2, 3, 4, 6]
>>> r.remove(3)
>>> r
[0, 1, 2, 4, 6]
```

Le troisième exemple continue l'exemple précédent en montrant d'autres opérations prédéfinies de gestion des listes dont les sens se déduisent du nom de l'opération.


```
>>> r
[6, 4, 2, 1, 0]
>>> r.extend([7,8])
>>> r
[6, 4, 2, 1, 0, 7, 8]
>>> r.index(1)
3 # return index of first occurrence of value 1
>>> r.insert(0,9)
>>> r
[9, 6, 4, 2, 1, 0, 7, 8]
>>> r.__contains__(5)
0
>>> r.__contains__(0)
1
>>> r.__len__() # longeur d'une liste
8
>>> r.__eq__([9, 6, 4, 2, 1, 0, 7, 6]) # test d'égalité
0
>>> r.__ne__([9, 6, 4, 2, 1, 0, 7, 6]) # non égal ?
1
>>>
```

Rappelons que l'ensemble des opérations prédéfinies peut être connu sous le prompt de commande Python au moyen de la commande `help()`.

Examinons maintenant la structure des dictionnaires. Un dictionnaire est une structure de données qui associe une valeur à une clé. Dans l'exemple qui suit, le dictionnaire `d` contient deux clés (`christophe, gil`) qui ont pour valeurs respectives `cerin, utard`. Le lecteur remarquera au passage les éléments syntaxiques pour définir un dictionnaire et pour accéder à une valeur pour une clé.

```
>>> d = {'christophe':'cerin', 'gil':'utard'}
>>> d['christophe']
'cerin'
```

```
>>> for x in range(2):
...      print d[x]
...
Traceback (most recent call last):
  File "<stdin>", line 2, in ?
KeyError: 0
>>> l = ['cerin','utard']
>>> for x in range(2):
...     print l[x]
...
cerin
utard
>>>
```


La dernière partie de l'exemple précédent montre que l'on ne peut pas indexer par des entiers un dictionnaire dont les clés sont des chaînes.

Examinons maintenant un autre type prédéfini : les tuples. Ils permettent de construire également des collections d'objets, comme les dictionnaires. Dans l'exemple qui suit on remarquera différentes notations pour les définir (avec ou sans parenthèses) ainsi que la partie de code qui effectue une permutation de valeurs (sans passer explicitement par une variable intermédiaire).

```
>>> t = ('chris', 'tophe', 'cerin')
>>> for x in t: print x,
...
chris tophe cerin
>>> x = 1, 2, 3
>>> x
(1, 2, 3)
>>> a, b ,c = 1, 2 ,3
>>> a
1
>>> b
2
>>> c
3
>>> x = 1; y = 2
>>> x, y = y, x
>>> x
2
>>> y
1
>>>
```

Examinons maintenant les liens entre les notions de tuples et de listes. Dans l'exemple qui suit on remarquera que les crochets permettant d'accéder à un élément de liste ou de tuple mais que l'on ne peut pas modifier une composante d'un tuple via l'opérateur d'affection = alors que cela est possible pour un élément de type liste.

```
>>> L = [1, 2, 3]
>>> T = (1, 2, 3)
>>> L[2]
3
>>> T[2]
3
>>> L[2] = 'x'
>>> L
[1, 2, 'x']
>>> T
(1, 2, 3)
>>> T[2] = 'x'
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
TypeError: object doesn't support item assignment
# il est interdit de faire cette affectation sur les tuples !!!
>>> L = ['c','e','r','i','n']
>>> T = None
>>> for x in L:
...     T = x, T
...
>>> T
('n', ('i', ('r', ('e', ('c', None)))))
>>>
>>> while T:
...     x, T = T
...     print x,
...
n i r e c
>>>
```

Passons maintenant en revue quelques facilités pour parcourir, modifier des dictionnaires ou des listes. Dans l'exemple qui suit on remarque que les contenus d'un dictionaire sont des tuples représentant les pairs clés, valeurs. Ensuite, le code qui suit permet de prendre toutes les pairs clés, valeurs du dictionaire D pour les placer dans la liste L.

```
>>> D = {'a':1, 'b':2, 'c':3, 'd':4}
>>> D.items()
[('a', 1), ('c', 3), ('b', 2), ('d', 4)]
>>> L = []
>>> for (key, value) in D.items(): L.append(value)
...
>>> L
[1, 3, 2, 4]
>>> D
{'a': 1, 'c': 3, 'b': 2, 'd': 4}
>>> D.__contains__
<method-wrapper object at 0x81cd004>
```

```
>>> D.__len__()
4
>>> D.__contains__('c')
1
>>> D.clear()
>>> D
{}
>>> help(dict) # pour une vue complète, par exemple :
 |  iteritems(...)
 |      D.iteritems() -> an iterator over the (key, value) items of D
 |  iterkeys(...)
 |      D.iterkeys() -> an iterator over the keys of D
 |  itervalues(...)
 |      D.itervalues() -> an iterator over the values of D
```


En Python, les structures de données prédéfinies sont encore plus élaborées dans le sens où :

1. elles sont hétérogènes/génériques (elles peuvent contenir des objets de n'importe quel type).
2. elles supportent aussi un niveau d'imbrication quelconque. Voici un exemple d'une liste avec des séquences imbriquées : `L = ['abc', (1,2), ([3], 4), 5]`.

La structure L est vraiment un arbre d'objets Python. Voyons les opérations permises sur cet arbre, notamment le mécanisme d'indexation c'est à dire le mécanisme permettant d'accéder aux objets :

```
>>> L = ['abc', (1,2), ([3], 4), 5]
>>> L[0]
'abc'
>>> L[1]
(1, 2)
>>> L[0][1]
'b'
>>> L[1][1]
2
>>> L[2][0]
[3]
>>> L[2][0][0]
3
>>> L[2][0][0][0]
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
TypeError: unsubscriptable object
>>>
>>> L
['abc', (1, 2), ([3], 4), 5]
```

```
>>> L.__len__()
4
>>> range(L.__len__())
[0, 1, 2, 3]
>>> for x in range(L.__len__()): L.__getitem__(x)
...
'abc'
(1, 2)
([3], 4)
5
>>>
```

Le mécanisme d'indexation est très général comme le montre les exemples plus élaborés suivants :


```
>>> table = {
 'licence': { 'prof':('cerin','christophe'), 'cours':['GL', 'TECHNO']},
 'maitrise': { 'prof':('utard','gil'), 'cours':['GL', 'ALGO']}
}... ... ...
>>> table['licence']['prof']
('cerin', 'christophe')
>>> table['licence']['prof'][1]
'christophe'
>>> table['licence']['prof'][1][5]
't'
>>> table['licence']['cours'][0]
'GL'
>>> # on peut tester l'égalité
...
>>> table['licence']['cours'][0] == table['maitrise']['cours'][0]
1
>>> table['licence']['cours'] == table['maitrise']['cours']
0
>>>
```

En python, nous avons deux opérateurs d'égalité qui sont == pour signifier «est ce que deux objets sont égaux ?»et l'opérateur `is` qui dit si deux entités partagent le même objet.

```
>>> L = ['hello', 'world'] ; M = L ; L.append('!')
>>> L
['hello', 'world', '!']
>>> M
['hello', 'world', '!']
>>>
>>> x = [1,2]
>>> y = ['a', x, 'c']
>>> z = {'a':x, 'b':2}
>>> x[0] = 3
```

```
>>> y
['a', [3, 2], 'c']
>>> z
{'a': [3, 2], 'b': 2}
>>># opérateur is
>>> x is y
0
>>> L is M
1
>>> x is y[1]
1
>>> x == y[1]
1
>>>
```

Nous retournons maintenant sur les opérateurs sur les listes ou sur les tuples. Les exemples qui suivent montrent comment réaliser de la concaténation (autrement que par l'opération append) de listes. Ainsi l'opérateur + a pour profil liste $\times$ liste.

```
>>> [1,2] + [3,4]
[1, 2, 3, 4]
>>> 'abc' + 'def'
'abcdef'
>>> (1,2) + (3,4) + (5,6)
(1, 2, 3, 4, 5, 6)
>>> ['a','b'] + 'e'
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
TypeError: can only concatenate list (not "str") to list
>>> 'cerin' * 3
'cerincerincerin'
>>> ['cerin','c'] * 3
['cerin', 'c', 'cerin', 'c', 'cerin', 'c']
>>> {2:'hell', 1:'o'} * 3
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
TypeError: unsupported operand type(s) for *: 'dict' and 'int'
```

Les listes peuvent contenir des objets de types différents et on peut composer par tranche des listes ou récupérer des tranches :

```
>>> L = [3.14, 'cerin', [7,8], 'christophe', {2:'world', 1:'hello'}]
>>> L[1:]
['cerin', [7, 8], 'christophe', {1: 'hello', 2: 'world'}]
>>> L[2:4] # on prend une tranche !
[[7, 8], 'christophe']
>>> L[1:1] + L[5:5]
[]
```

```
>>> L[1:] + L[5:]
['cerin', [7, 8], 'christophe', {1: 'hello', 2: 'world'}]
>>> L[1:2] + L[2:3]
['cerin', [7, 8]]
>>> L[:1]
[3.14]
>>> L[0:len(L)]
[3.14, 'cerin', [7, 8], 'christophe', {1: 'hello', 2: 'world'}]
>>> L[-1:]
[{1: 'hello', 2: 'world'}]
>>> L[1:-1]
['cerin', [7, 8], 'christophe']
>>> S = 'agathe'
>>> S[-1] # attention
'e'
>>> S[-3:-1]
'th'
>>> S[:-1]
'agath'
>>> S[:]
'agathe'
>>>
```


On peut aussi faire de l'affectation sur une structure liste de la façon suivante à partir de l'exemple précédent :

```
>>> L[1:1] = 'agathe'
Traceback (innermost last):
  File "<stdin>", line 1, in ?
TypeError: illegal argument type for built-in operation
>>> L[1:1] = ['agathe']
>>> L
[3.14, 'agathe', 'cerin', [7, 8], 'christophe', {1: 'hello', 2: 'world'}]
>>>
>>> L[2:] = ['and', 'jeanne']
>>> L
[3.14, 'agathe', 'and', 'jeanne']
>>>
>>> L[1:2] = []
>>> L
[3.14, 'and', 'jeanne']
>>>
```

Attention, contrairement aux listes et aux dictionnaires, la modification des chaînes et des tuples requiert quelques précautions supplémentaires pour le programmeur. Pour modifier une chaîne ou un tuple, le principe est de créer un nouvel objet en utilisant le 'slicing' (tranche) et la concatenation.

```
>>> S = S[3:]
```

```
>>> S
'xyz'
>>> S[1] = ''
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
TypeError: object doesn't support item assignment
>>> L
['hello', 'world', '!']
>>> del L[1]
>>> L
['hello', '!']
>>> del S[1]
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
TypeError: object doesn't support item deletion
>>> S = S[0] + 'n' + S[2:]
>>> S
'xnz'
```

Le cas des tuples se traite de la façon suivante (le lecteur est invité à suivre pas à pas les transformations sur les structures de données) :

```
>>> T = ([1,2,3],3)
>>> T[0] = 4
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
TypeError: object doesn't support item assignment
>>> T[0][0] = 4
>>> T
([4, 2, 3], 3)
>>> T = T[0] + 5
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
TypeError: can only concatenate list (not "int") to list
>>>  # qui a causé l'erreur : T[0] ou 5 ?
...
>>> T = (T[0])
>>> T
[4, 2, 3]
>>> T = (T[0] + 5)
>>> T
9
>>> # c'est plus un tuple !!!
...
```

Examinons un autre exemple :

```
>>> T = ([1,2,3], 3)
>>> T[1:]
```

```
(3,)
>>> T[1]
3
>>> T[1:1]
()
>>> T[1:2]
(3,)
>>> T[1:] + 'abc'
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
TypeError: can only concatenate tuple (not "str") to tuple
>>> T[1:], 'abc'
((3,), 'abc')
>>> 'abc', T[1:]
('abc', (3,))
>>> T[1], 'abc'
(3, 'abc')
>>> T
([1, 2, 3], 3)
>>> T = T[1], 'abc'
>>> T
(3, 'abc')
>>>
```


3.3.4 Programmation objet en Python

Passons maintenant à une notion qui permet de structurer les programmes plutôt que les données : les fonctions. En travaillant sous l'évaluateur, les définitions sont vues comme appartenant à un module particulier appelé `__main__`. L'exemple qui suit introduit une fonction `foo` dont on remarque que l'on ne décrit pas le type des paramètres (ni le type du résultat d'ailleurs). La fonction `count` admet quant à elle deux paramètres et elle retourne un entier :

```
>>> def foo(item): print item
...
>>> foo('hello world')
hello world
>>> foo(30)
30
>>> def count(string, chars):
        total = 0
        for x in string:
                if x in chars:
                        total = total + 1
        return total... ... ... ... ...
...
>>> count('hello world','hello world')
11
>>> count(['H','a','l'], 'hello world')
```

```
1
>>> count([1,2,3,4],[2,4])
2
>>>
```

Ce genre de fonctionnement générique (sans typage des arguments des fonctions ni du résultat) est propre à Python. De plus, en Python (comme dans d'autres langages, SML par exemple), nous avons à notre disposition un moyen simple d'écrire des fonctionelles (des fonctions qui admettent en paramètre des fonctions). L'exemple qui suit vous présente la façon de les décrire. Les points remarquables dans l'exemple concernent l'application d'une fonction à une liste d'arguments via l'opérateur `apply`, alors que pour la fonction `argumentMAGIC`, le troisième paramètre est un tuble :

```
>>> x = foo
>>> x('hello')
hello
>>> def indirect(fun, arg): fun(arg)
...
>>> indirect(foo,'hello')
hello
>>> sched = [(foo, ('hello',)), (count, ([1,2,3],[2,4]))]
>>> for (fun, arg) in sched:
...     apply(fun, arg)
...
hello
1
>>> def argumentMAGIC(a, b, (c,d,e)):
...     a = 'cde'
...     b[1] = 9
...     e = c + d
...
>>> l = 'ABC'
>>> m = [1,2,3]
>>> n = 1.1, 2.2, 0.0
>>> argumentMAGIC(l, m, n)
>>> l, m, n
('ABC', [1, 9, 3], (1.1000000000000001, 2.2000000000000002, 0.0))
```

Une autre manière de travailler avec les fonctionnelles est la suivante :

```
>>> def argumentMAGIC(a,b,(c,d,e)):
...     a = 'xyz'
...     b[1]=9
...     (c,d,e) = c,d,c+d
...
>>> l = 'ABC'
>>> m = [1,2,3]
>>> n = 1.1, 2.2, 0.0
```

```
>>> argumentMAGIC(l,m,n)
>>> l,m,n
('ABC', [1, 9, 3], (1.1000000000000001, 2.2000000000000002, 0.0))
>>>
```

On remarque qu'ici m a été modifié mais pas le tuple n. Cela veut dire que c,d,e ont été passé par valeur et pas par référence (au sens C ou Pascal).

Le programmeur peut également initialiser par défaut les paramètres d'une fonction :

```
>>> def ask_ok(prompt, retries=4, complaint='Yes or no, please!'):
        while 1:
                ok = raw_input(prompt)
                if ok in ('y', 'ye', 'yes'): return 1
                if ok in ('n', 'no'): return 0
                retries = retries - 1
                if retries < 0: raise IOError, 'refusenik error'
                print complaint ... ... ... ... ... ... ...
...
>>> ask_ok('Really?')
Really?tit
Yes or no, please!
Really?tt
Yes or no, please!
Really?t
Yes or no, please!
Really?t
Yes or no, please!
Really?t
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
  File "<stdin>", line 7, in ask_ok
IOError: refusenik error
Really?y
1
>>> ask_ok('Really?')
Really?n
0
```

Pour le programmeur, certaines fonctionelles d'un usage fréquent ont été implémentées dans le langage. Il s'agit de :

filter(function, sequence):

```
>>> def f(x): return x%2 != 0 and x%3 != 0
>>> filter(f, range(2,25))
[5, 7, 11, 13, 17, 19, 23]
```

map(function, sequence): call function for each item, return list of return values

```
>>> def f(x): return x%2 != 0 and x%3 != 0
>>> map(f, range(2,25))
[0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0]
```

reduce(function, sequence): return a single value, call BINARY function on the first two items then on the result and next item iterate

```
>>> reduce(lambda x, y: x+y, [1, 2, 3, 4, 5])
15
```

Le problème que nous abordons est le suivant : comment créer des listes sans usage de map(), filter(), lambda ? On utilise une expression suivie par la clause `for` suivie éventuellement d'autres clauses `for` comme suit :

```
>>> vec = [2,4,6]
>>> [3*x for x in vec]
[6, 12, 18]
>>> [{x: x**2} for x in vec]
[{2: 4}, {4: 16}, {6: 36}]
>>> vec1 = [2,4,6]
>>> vec2 = [4,3,-9]
>>> [x*y for x in vec1 for y in vec2]
[8, 6, -18, 16, 12, -36, 24, 18, -54]
>>> [vec1[i]*vec2[i] for i in range(len(vec1))]
[8, 12, -54]
>>> [3*x for x in vec if x > 3]
[12, 18]
>>> [3*x for x in vec if x < 2]
[]
>>>
```

Les modules, les classes

Nous envisageons maintenant un moyen plus général que les fonctions pour organiser les programmes : les modules. Les modules adressent le problème de la réutilisation de code. Ils sont ouverts à l'intérieur d'autres modules et il semble normal que les objets définis dans un module et ouverts à l'intérieur d'un autre module ne soient visibles qu'à l'intérieur de ce module sauf si des régles de partages explicites ont été mis en oeuvre. On importe un module à l'aide de la commande `import` comme suit :

```
>>> import sys
>>> sys.modules
{'tokenize': <module 'tokenize' from '/usr/local/lib/python2.2/tokenize.pyc'>,
 'stat': <module 'stat' from '/usr/local/lib/python2.2/stat.pyc'>,
'__future__': <module '__future__' from
```

```
'/usr/local/lib/python2.2/__future__.pyc'>, 'copy_reg':
...........
...........
>>> import time
>>> time.modules
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
AttributeError: 'module' object has no attribute 'modules'
>>> time.asctime()
'Tue Aug 20 14:00:50 2002'
>>> time.time()
1029844951.535313
>>> time.time()
1029844959.250191
```


Une introduction à la programmation objet en Python peut maintenant être consultée, par exemple en lisant le chapitre "Data model" dans le manuel "Python Reference Manual", Guido van Rossum, Fred L. Drake, Jr., editor, Release 2.2.1, April 10, 2002. On remarquera que les objets sont vus comme des moyens d'abstraire des données ce qui veut dire se concentrer sur les propriétés des objets plutôt que sur le comment implémenter tel ou tel type de données de manière très concrète. Les définitions fournies par Guido van Rossum sont donc les suivants :

- Objects are Python's abstraction for data. All data in a Python program is represented by objects or by relations between objects.
- Every object has an identity, a type and a value. See id(), type()
- Some objects contain references to other objects; these are called containers. Examples of containers are tuples, lists and dictionaries.

Voici quelques exemples pour commencer avec les modules :

```
>>> s = 'abcdef'
>>> id(s)
135403136
>>> type(s)
<type 'str'>
>>> import os
>>> id(os)
135437564
>>> type(os)  # l'operateur type permet de trouver
              # à l'exécution, le type d'un objet
<type 'module'>
```

La notion de classe est également associée à celle d'objet. Nous laissons ici en anglais le vocabulaire Python des classes vues par Guido van Rossum. Pour faire simple, une classe regroupe des fonctions et des données :

```
Classes

    Class objects are created by class definitions (see section 7.6,
    ‘‘Class definitions’’). A class has a namespace implemented by a
    dictionary object. Class attribute references are translated to
    lookups in this dictionary, e.g., "C.x" is translated to
    "C.__dict__["x"]". When the attribute name is not found there, the
    attribute search continues in the base classes. The search is
    depth-first, left-to-right in the order of occurrence in the base
    class list. When a class attribute reference would yield a
    user-defined function object, it is transformed into an unbound
    user-defined method object (see above). The im_class attribute of
    this method object is the class for which the attribute reference
    was initiated.

    Class attribute assignments update the class's dictionary, never
    the dictionary of a base class.

    A class object can be called (see above) to yield a class instance
    (see below).

    Special attributes: __name__ is the class name; __module__ is the
    module name in which the class was defined; __dict__ is the
    dictionary containing the class's namespace; __bases__ is a tuple
    (possibly empty or a singleton) containing the base classes, in
    the order of their occurrence in the base class list; __doc__ is
    the class's documentation string, or None if undefined.
```

Le vocabulaire complémentaire d'instance de classe est alors introduit par Guido van Rossum :

```
Class instances
    A class instance is created by calling a class object (see
    above). A class instance has a namespace implemented as a
    dictionary which is the first place in which attribute references
    are searched. When an attribute is not found there, and the
    instance's class has an attribute by that name, the search
    continues with the class attributes. If a class attribute is found
    that is a user-defined function object (and in no other case), it
    is transformed into an unbound user-defined method object (see
    above). The im_class attribute of this method object is the class
    of the instance for which the attribute reference was
    initiated. If no class attribute is found, and the object's class
    has a __getattr__() method, that is called to satisfy the lookup.

    Attribute assignments and deletions update the instance's
    dictionary, never a class's dictionary. If the class has a
    __setattr__() or __delattr__() method, this is called instead of
    updating the instance dictionary directly.
```

```
Class instances can pretend to be numbers, sequences, or mappings
if they have methods with certain special names. See section 3.3,
‘‘Special method names.’’

Special attributes: __dict__ is the attribute dictionary;
__class__ is the instance’s class.
```

D'un point de vue syntaxique, une classe s'écrit et s'utilise de la manière suivante d'après Guido van Rossum :

```
A class definition defines a class object (see section 3.2):

    classdef     ::=  "class" classname [inheritance] ":" suite
    inheritance  ::=  "(" [expression_list] ")"
    classname    ::=  identifier


A class definition is an executable statement. It first evaluates the
inheritance list, if present. Each item in the inheritance list should
evaluate to a class object. The class’s suite is then executed in a
new execution frame (see section 4.1), using a newly created local
namespace and the original global namespace. (Usually, the suite
contains only function definitions.) When the class’s suite finishes
execution, its execution frame is discarded but its local namespace is
saved. A class object is then created using the inheritance list for
the base classes and the saved local namespace for the attribute
dictionary. The class name is bound to this class object in the
original local namespace.

Programmer’s note: variables defined in the class definition are class
variables; they are shared by all instances. To define instance
variables, they must be given a value in the the __init__() method or
in another method. Both class and instance variables are accessible
through the notation ‘‘self.name’’, and an instance variable hides a
class variable with the same name when accessed in this way. Class
variables with immutable values can be used as defaults for instance
variables.
```

Le dernier paragraphe est très important car il vous dit que les variables définies dans une classe doivent être initialisées dans la méthode `__init__()`.

La notion de classe Python peut alors être comparée à la notion de classe C++[7] comme suit :

- En termes C++ : toutes les classes sont publiques (il n'y a pas de notion "private" au sens C++) ;

[7] Ce point technique n'est à lire que si vous avez déjà des notions de C++.

- toutes les fonctions membre de la classes sont virtuelles (vitual au sens C++)
- il n'y a pas en Python de constructeur ni de destructeurs de classes.

Voici un exemple d'une classe (non paramétrée) contenant un commentaire, la définition d'une variable et d'une fonction. Le lecteur remarquera également comment faire un appel à une fonction membre de la classe.

```
>>> class MyClass:
        "A simple example class"
        i = 123
        def f(self):
                return 'hello world'... ... ... ...
>>> MyClass.i
123
>>> MyClass.f()
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
TypeError: unbound method f() must be called with
MyClass instance as first argument (got nothing instead)
>>> x = MyClass()
>>> x.f()
'hello world'
>>>#ATTENTION AU NOM DES OBJETS
>>> y = MyClass
>>> y
<class __main__.MyClass at 0x81167b4>
>>> x
<__main__.MyClass instance at 0x811611c>
>>>
```

Notez également que dans l'exemple précédent, la variable i n'est pas initialisée dans la méthode `__init__()`. Cela ne nous empêche pas de travailler :

```
>>> class MyClass:
...     i=123
...     def f(self):
...             return 'hello'
...
>>> x=MyClass()
>>> x.i
123
>>> x.i=124
>>> y=MyClass()
>>> y.i
123
>>> x.i
124
>>>
```

Pour utiliser la méthode `__init__()`, nous pouvons faire comme suit :

```
# fichier titi.py
class MyClass:
     def __init__(self,j):
          self.data=j
     def f(self):
          return 'hello'

x=MyClass(123)
print x.data
y=MyClass(124)
print y.data

Résultat de l'exécution :
> python titi.py
123
124
```


Notez que MyClass ne prend pas de paramètre mais que `__init__()` en prend un : celui qui nous sert à initialiser la variable d'instance.

Reprenons l'exemple précédent de la classe `MyClass` pour appliquer quelques méthodes spécifiques :

```
>>> type(MyClass)
<type 'classobj'>
>>> id(MyClass)
1075790684
>>> MyClass.__name__
'MyClass'
>>> MyClass.__module__
'__main__'
>>> MyClass.__dict__
{'i': 123, '__module__': '__main__',
'__doc__': 'A simple example class',
'f': <function f at 0x401c87d4>}
```

Nous pouvons maintenant nous concentrer sur un exemple à peine plus compliqué. Soit le fichier `bag.py` suivant :

```
 class Bag:
    def __init__(self):
        self.data = []
    def add(self, x):
        self.data.append(x)
    def addtwice(self,x):
        self.add(x)
        self.add(x)
```

```
l = Bag()
l.add('first')
l.add('second')
l.addtwice('third')
print(l.data)
```

On charge et on exécute le fichier sous l'évaluateur de la manière suivante :

```
[christophe@m11 licenc]$ python
Python 2.2.1 (#1, Aug 20 2002, 09:56:09)
[GCC 2.96 20000731 (Mandrake Linux 8.1 2.96-0.62mdk)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from bag import *
['first', 'second', 'third', 'third']
>>>
```


Évolutions de la notion de classes. Il y a eu des nouveautés avec la version 2.2 de Python par rapport aux versions ultérieures du langage :

- Il était impossible de créer des sous classes à partir de built-in types (types prédéfinis standards) : on ne pouvait pas ajouter une fonction à la classe prédéfinie des listes par exemple ;
- Il est maintenant possible de définir des classes et des méthodes statiques en supplément aux méthodes d'instance initialement implémentées.

Pour continuer à mesurer les artefacts de programmation objet de Python, travaillons à partir d'un exemple tiré d'une version antérieure à la version 2.2 de Python : la notion de classe dans Python supporte aussi l'héritage d'attributs et l'instantiation multiple :

```
# Rational numbers
from types import *
# La fonction qui suit retourne soit à un réel soit
# à un objet Rat. (Attention les minuscules et les
# majuscules ne sont pas identiques dans rat et Rat)
def rat(num, den):
        if type(num) == FloatType or type(den) == FloatType:
                return num/den
        return Rat(num, den)
def gcd(a, b):
        while b:
                a, b = b, a%b
        return a
class Rat:
        def __init__(self, num, den):
                if den == 0:
                        raise ZeroDivisionError, 'rat(x, 0)'
                if type(den) == FloatType or type(num) == FloatType:
```

```
                g = float(den)
        else:
                g = gcd(num, den)
        self.num = num/g
        self.den = den/g
# La méthode qui suit hérite des attributs num et den
def __repr__(self):
        return 'Rat(%s, %s)' % (self.num, self.den)
def __str__(self):
        if self.den == 1:
                return str(self.num)
        else:
                return '%s/%s' % (self.num, self.den)
def __cmp__(a, b):
        c = a-b
        if c.num < 0:
                return -1
        if c.num > 0:
                return 1
        return 0
def __float__(self):
        return float(self.num) / float(self.den)
def __long__(self):
        return long(self.num) / long(self.den)
def __int__(self):
        return int(self.num / self.den)
def __coerce__(a, b):
        t = type(b)
        if t == IntType:
                return a, Rat(b, 1)
        if t == LongType:
                return a, Rat(b, 1L)
        if t == FloatType:
                return a, Rat(b, 1.0)
        if t == InstanceType and a.__class__ == b.__class__:
                return a, b
        raise TypeError, 'Rat.__coerce__: bad other arg'
def __add__(a, b):
        return rat(a.num*b.den + b.num*a.den, a.den*b.den)
def __sub__(a, b):
        return rat(a.num*b.den - b.num*a.den, a.den*b.den)
def __mul__(a, b):
        return rat(a.num*b.num, a.den*b.den)
def __div__(a, b):
        return rat(a.num*b.den, a.den*b.num)
def __neg__(self):
        return rat(-self.num, self.den)
```

Le programme de test de la classe précédente est le suivant :

```
def test():
        print Rat(-1L, 1)
        print Rat(1, -1)
        a = Rat(1, 10)
        print int(a), long(a), float(a)
        b = Rat(2, 5)
        l = [a+b, a-b, a*b, a/b]
        print l
        l.sort()
        print l
        print Rat(0, 1)
        print a+1
        print a+1L
        print a+1.0
        try:
                print Rat(1, 0)
                raise SystemError, 'should have been ZeroDivisionError'
        except ZeroDivisionError:
                print 'OK'

test()
```

```
>>> test()
-1
-1
0 0 0.1
[Rat(1, 2), Rat(-3, 10), Rat(1, 25), Rat(1, 4)]
[Rat(-3, 10), Rat(1, 25), Rat(1, 4), Rat(1, 2)]
0
11/10
11/10
1.1
OK
>>>
```

Les classes en Python 2.2

Examinons maintenant l'héritage de propriétés à travers un exemple :

```
# Class method
class C:
        def foo(cls, y):
                print "classmethod", cls, y
        foo = classmethod(foo)

C.foo(1)
c = C()
c.foo(1)
```

```
#But notice this:

class E(C):
        def foo(cls, y): # override C.foo
                print "E.foo() called (cls=", cls,")"
                C.foo(y)
        foo = classmethod(foo)

>>> E.foo(1)
E.foo() called (cls= __main__.E )
classmethod __main__.C 1
>>> e = E()
>>> e.foo(1)
E.foo() called (cls= __main__.E )
classmethod __main__.C 1
```


Ici, il faut noter le passage en paramètre de la classe C dans la définition de la classe E. La notation avec un point permet de clarifier les situations c'est à dire de lever les ambiguités sur la méthode qui est appelée !

Résumé sur l'héritage (Inheritance)

```
class DerivedClassName(BaseClassName)
  <statement-1>
  ...
  <statement-N>
May override methods in the base class

Call directly via BaseClassName.method
```

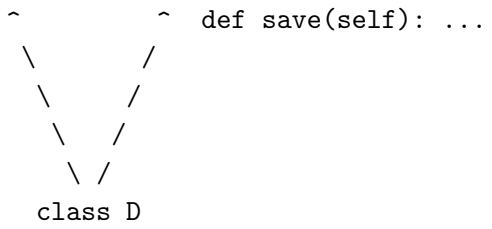
L'héritage multiple (une classe hérite des méthodes de plusieurs autres classes dites classes de base) se présente quant à lui de la manière suivante :

```
class DerivedClass(Base1,Base2,Base3):
   <statement>

Recherche : depth-first, left-to-right
```

PROBLEM: class derived from two classes with a common base class?

```
              class A:
               ^ ^  def save(self): ...
              /   \
             /     \
            /       \
           /         \
        class B     class C:
```



Dans la classe D, on fait un appel à save. Quel est l'algorithme de recherche du code à exécuter et quelle save est exécutée (celui de A ou celui de C) ?

Complément

Comment représenter en Python un type abstrait ? Dans ce paragraphe, nous allons écrire une classe qui représente un objet (abstrait). Examinons successivement trois exemples. Le premier concerne le codage d'un type décrit par extension, le deuxième concerne un type paramétré alors que le troisième s'intéresse au type abstrait «arbre de»(ici arbre binaire) qui est une structure bien connue et permettant des recherches efficaces.

```
# Une ellipse est soit un objet ER soit un objet ED
# soit un objet CR, soit un objet CD.
# Dans le langage SML, on écrit :
datatype ellipse = ER | ED | CR | CD;

# En Python, la simulation est :
class ellipse:
"simulation datatype sml ellipse"
    def __init__(self,a):
        if a in ('ER','ED','CR','CD'):
           self.val = a
        else:
           print "mauvaise saisie objet ellipse"
```

Autre exemple :

```
# Une couleur est définie par un nom (par
# exemple le rouge) et un entier (représentant
# l'intensité de la couleur).
# Dans le langage SML on écrit le type abstrait
# paramétré couleur comme suit :
datatype color = Color of int;

# En Python on simule par :
class Color:
    "datatype paramétré"
    def __init__(self,nom,x):
```

```
        if(nom==’COLOR’) and (type(x) == int):
           self.val=nom
           self.data=x
        else:
           print "valeur incorrecte"
c=Color("COLOR",30)
print c.val, c.data
```

On s’intéresse maintenant à coder en Python et avec une classe la définition SML suivante d’un arbre binaire :

```
# Définition : un arbre est soit vide (empty), soit un objet
# feuille de type ’a (Leaf of ’a), soit un noeud de type ’a auquel
# on adjoint deux arbres (’a tree). En SML celà donne :
datatype ’a tree = empty | Leaf of ’a
        | Node of ’a * ’a tree * ’a tree;
```

En Python, une description d’arbre est alors :

```
class Tree:
    def __init__(self, data=None, left=None, right=None):
        self.data = data
        self.left  = left
        self.right = right

    def __str__(self):
        return str(self.data)

    def print_tree(self, tree):
        if tree is None: return 0
        self.print_tree(tree.left)
        print tree.data,
        self.print_tree(tree.right)

    # Parcours de l’arbre et affichage des
    # valeurs des noeuds
    def print_postorder(self, tree):
        if tree is None: return 0
        self.print_postorder(tree.left)
        self.print_postorder(tree.right)
        print tree.data,

    # Parcours de l’arbre et affichage des
    # valeurs des noeuds (variante)
    def print_preorder(self, tree):
        if tree is None: return 0
        print tree.data,
        self.print_preorder(tree.left)
```

```
        self.print_preorder(tree.right)

    # Construction d'un arbre
    def build123(self):
        root  = Tree(2)
        left   = Tree(1)
        right  = Tree(3)
        root.left = left
        root.right = right
        return root
```

Les autres méthodes de gestion des arbres sont les suivantes dans la classe ainsi que le programme de test :

```
    # Recherche si la clé 'x' est présente dans l'arbre
    def search(self,x):
        if x==self.data:
           return 1
        else:
           if (self.left != None) and (self.right != None):
              return self.left.search(x) or self.right.search(x)
           elif (self.left == None) and (self.right == None):
              return 0
           elif (self.left == None):
              return self.right.search(x)
           else:
              return self.left.search(x)

t = Tree()
A = t.build123()
print "Printing  A:", t.print_tree(A)
print "PreOrder  A:", t.print_preorder(A)
print "PostOrder A:", t.print_postorder(A)
Tree1 = Tree(4,A,None)
Tree2 = Tree(5,Tree(3,None,Tree(6,None,None)),Tree("TITI",None,None))
Tree3 = Tree(10,Tree1,Tree2)
print "Printing  Tree3:", Tree3.print_tree(Tree3)
print "Recherche de 10 dans Tree3:", Tree3.search(10)
print "Recherche de 3 dans Tree3:", Tree3.search(3)
print "Recherche de 6 dans Tree3:", Tree3.search(6)
print "Recherche de 7 dans Tree3:", Tree3.search(7)
""">>> execfile("tree.py")
Printing  A: 1 2 3 None
PreOrder  A: 2 1 3 None
PostOrder A: 1 3 2 None
Printing  Tree3: 1 2 3 4 10 3 6 5 TITI None
Recherche de 10 dans Tree3: 1
Recherche de 3 dans Tree3: 1
```

```
Recherche de 6 dans Tree3: 1
Recherche de 7 dans Tree3: 0"""
```

3.4 Repères bibliographiques

[1] AnaGram – *Raisonner pour programmer* – Dunod – 1988.

[2] George Polya *Comment poser et résoudre un problème* – Édition Jacques Gabay, 151bis rue St Jacques 75005 Paris – édition 1989.

[3] D. Solow – *How to Read and Do Proofs* – John Wiley & Sons – 1990.

[4] G. Dromey – *Program Derivation, The Development of Programs from Specifications* – Addison Wesley – 1989.

[5] D. Gries – *The Science of Programming* – Springer Verlag – 1989.

[5] N. Francez – *Program Verification*, Addison Wesley – 1992.

[7] E.W Dijkstra – *A Discipline of Programming* – Prentice Hall – 1976.

[7] K. Apt et E. Olderog, *Verification of Sequential and Concurrent Programs*, 2nd Edition, Springer Verlag, 1997.

[8] J.F. Morin, *Comprendre les méthodes formelles, Panorama et outils logiques*, Masson, 1996.

[9] Habrias, *Introduction à la spécification*, Masson.

[10] Habrias, *Dictionnaire encyclopédique du génie logiciel*, Masson.

[11] The Mythical Man-Month, Frederick P Brooks, Jr. AW, ISBN 0-201-83595-9

[12] Programming Pearls, Jon L Bentley, AW, 2000, ISBN 0-201-65788-0

[13] The Practice of Programming, Brian W. Kernighan and Rob Pike, Addison Wesley, 1999 ISBN 0-201-61586-X

[14] Christine Froidevaux, Marie Claude Gaudel, Michèle Soria – *Types de données et algorithmes* – McGraw-Hill – 1990.

[15] Udi Manber – *Introduction to Algorithms – A creative approach* – Addison Wesley – 1989.

[16] C.Pair, R.Mohr, R.Schott – *Construire les algorithmes* – Dunod – 1988.

[17] Carroll Morgan – *Programming from Specifications* – Prentice Hall – 1990.

[18] Cliff B.Jones – *Systematic Software Development Using VDM* – Prentice Hall International – 1986.

[19] C.B Jones and R.C Shaw – *Case Studies in Systematic Software Development* – Prentice Hall – 1990.

4 Études de cas

Sommaire

4.1 Introduction

Ce chapître est dédié à des études de cas et des exercices qui mettent en œuvre les concepts vus prédédemment depuis le début de l'ouvrage. Il s'agit d'études de cas ou d'exercices modestes (dans le sens ou la solution se dérive en deux minutes) à des études plus ambitieuse où il faut à la fois reformuler le problème, spécifier une solution qui vérifie un ensemble de propriétés, puis dériver un code Python qui implémente partiellement ou l'ensemble des propriétés de la solution.

Les études de cas font intervenir des mathématiques pour construire et prouver des propriétés d'un programme ou alors, il s'agit d'une étude visant à mettre en œuvre une implémentation à partir d'une spécification déjà écrite (c'est le cas par exemple pour le prototype qui génére des fichiers au format Xfig). Dans ce cas, la rédaction fait ressortir les choix d'implémentation en dégageant les idées maitresses. Il s'agit alors de prendre de la hauteur par rapport au code, de faire ressortir l'essentiel plutôt que le détail, de montrer son savoir faire, sa compréhension du problème et les limites de la solution retenue.

N'oublions pas qu'un de nos objectifs est de montrer aux étudiants comment rédiger dans une discipline scientifique et technique et avec de la rigueur. Pour

cela nous avons quelque fois inséré des commentaires dans la marge gauche pour attirer l'attention du lecteur sur un fait qui donne de l'ampleur à la discussion.

Dans tous les cas, le lecteur est invité à lire l'ensemble du problème depuis la redaction du sujet du problème jusqu'à la dérivation d'une solution. Il est invité à faire un travail de compréhension et aussi, à titre d'exercice, celui de critique de la solution retenue. La critique peut aller jusqu'à remettre en cause la solution, à imaginer des compléments, à explorer des pistes alternatives. Les études de cas peuvent se présenter aussi comme des exercices à faire ou plus exactement à compléter.

Enfin, le lecteur est invité à lire l'ensemble de la bibilographie du chapître précédent et plus particulièrement [2], [3], [4], [5], [7], [12] parce qu'elle introduit également des exemples particulièrement pédagogiques dans la lignée de ceux présentés ici. Il est vraiment indispensable, à notre avis d'étudier cette littérature pour débuter des études en informatique. Certaines de nos études de cas reprennent des énoncés de problèmes de ces ouvrages et nous donnons généralement une solution distincte allant plus loin dans le problème soit parce que nous proposons un codage en Python de la solution, soit parce que nous proposons une amélioration de l'occupation mémoire pour l'algorithme sous jacent. Nous proposons donc au lecteur de lire notre approche et celle en référence bibliographique qui correspond au problème et de comprendre les différences.

Nous commençons par des exercices de logique des propositions avant d'utiliser cette logique pour prouver des programmes. Enfin nous passons à des études de cas conséquentes qui visent à dériver une solution avec de bonnes propriétés ainsi qu'un code implémentant la solution.

4.2 Logique et preuves de programme

Exercice 1 : établir que $(q \Leftrightarrow \neg p) \equiv (p \vee q) \wedge \neg(p \wedge q)$.

En utilisant les connecteurs logiques une preuve est :

$$\begin{aligned}(q \Leftrightarrow \neg p) &\equiv ((q \Rightarrow \neg p) \wedge (\neg p \Rightarrow q)) \\ &\equiv (\neg q \vee \neg p) \wedge (p \vee q) \\ &\equiv \neg(p \wedge q) \wedge (p \vee q)\end{aligned}$$

On peut aussi voir les choses de la façons suivantes: affirmez $p \Leftrightarrow q$ c'est affirmer que la proposition $p \Leftrightarrow q \equiv 1$ (une tautologie). Concernant $\neg(p \wedge q) \wedge (p \vee q)$, il est donc nécessaire que $(p \wedge q)$ soit une contradiction et $(p \vee q)$ une tautologie. Les cas qui nous intéressent sont $p = 1, q = 0$ et $p = 0, q = 1$. Pour terminer, on remarque que pour ces valeurs, dans la table de vérité de $(q \Leftrightarrow \neg p)$ les deux valeurs de sortie correspondantes sont 1 cqfd.

Exercice 2 : Prouver les énoncés $((p \Rightarrow q) \wedge (q \Rightarrow r)) \Rightarrow (p \Rightarrow r)$ et $((p \Rightarrow q) \wedge (q \Rightarrow r) \wedge \neg r) \Rightarrow \neg p$.

Règle 1 *Si, s'assurant que* A *est vraie, il est possible de trouver que* B *est vraie (par un nombre fini de pas utilisant une des règles de la déduction naturelle), alors on doit dériver aussi* $A \Rightarrow B$.

Dérivation	prémisse
1)$p \Rightarrow q$	hypothèse
2)$q \Rightarrow r$	hypothèse
3)p	hypothèse
4)q	modus ponens 1) et 3)
5)r	modus ponens 2) et 4)
6)$p \Rightarrow r$	règle 1 entre 3) et 5)

Pour cette dernière conclusion, on peut aussi la justifier en remarquant 1) $r \Rightarrow (p \Rightarrow r)$ est une tautologie et 2) r. Par application du modus ponens on déduit alors $p \Rightarrow r$.

Dérivation	justification
1)$p \Rightarrow q$	hypothèse
2)$q \Rightarrow r$	hypothèse
3)$\neg r$	hypothèse
4)$\neg q$	modus tollens 2) et 3)
5)$\neg p$	modus tollens 1) et 4)

Exercice 3 : Prouver que «si n est un entier impair alors n^2 est impair». On expliquera le raisonnement logique.

On utilise la définition de «est impair»i.e il s'agit de montrer que $n^2 = 2k + 1$ pour un certain entier k. Puisque n est impair, par définition on a $n = 2m + 1$ pour un certain entier m. On a alors que :

$$\begin{aligned} n^2 &= (2m+1)^2, \\ n^2 &= 4m^2 + 4m + 1, \\ n^2 &= 2(2m^2 + 2m) + 1 \end{aligned}$$

Donc la valeur désirée pour k est $(2m^2 + 2m)$ et puisque m est un entier $(2m^2 + 2m)$ est un entier. Ceci termine la preuve. (Vous pouvez travailler dans l'autre sens i.e. supposer que n^2 est pair $\cdots$ et arriver à n est un entier pair puis utiliser un argument sur la contraposée).

Exercice 4 : On vous demande de prouver $\neg P(a)$ à partir des connaissances suivantes : $Q(a)$ et $\forall x(P(x) \Rightarrow \neg Q(x))$.

À partir des deux hypothèses, on peut spécialiser et dériver que $P(a) \Rightarrow \neg Q(a)$. Il s'en suit que $\neg P(a) \vee \neg Q(a)$ et puisque l'on a (hyp) $Q(a)$ vraie, on déduit immédiatement que $\neg P(a)$.

Exercice 5 : Soient les propriétés suivantes : $\neg p \Rightarrow (\neg q \vee r), q \Rightarrow (p \vee \neg r), (p \vee q) \Rightarrow (q \Rightarrow \neg p), p \Rightarrow q$.
1) Est-ce que les quatres propositions peuvent être vraies simultanément ?
2) combien de propositions peuvent être vraies simultanément ?

$$\begin{aligned}
a &\equiv \neg p \Rightarrow (\neg q \vee r) \\
&\equiv p \vee \neg q \vee r \\
b &\equiv q \Rightarrow (p \vee \neg r) \\
&\equiv p \vee \neg q \vee \neg r \\
c &\equiv (p \vee q) \Rightarrow (q \Rightarrow \neg p) \\
&\equiv (\neg p \wedge \neg q) \vee \neg q \vee \neg p \\
&\equiv ((\neg q \vee \neg p) \vee \neg p) \wedge ((\neg q \vee \neg p) \vee \neg q) \\
&\equiv \neg q \vee \neg p \\
d &\equiv p \Rightarrow q \\
&\equiv \neg p \vee q \\
a \wedge b &\equiv \cdots \\
c \wedge d &\equiv \cdots \\
a \wedge b \wedge c \wedge d &\equiv \cdots
\end{aligned}$$

Exercice 6 : Coder au moyen de quantificateurs les énoncés suivants en langage naturel :

1. Il n'y a pas plus de k éléments du tableau $T[1 \cdots n]$ qui sont multiples de p ;

$$|\{x : x \in T[1 \cdots n] : x \bmod p = 0\}| \leq k$$

2. Les k premiers éléments du tableau $T[1 \cdots n]$ sont triés en ordre décroissant ;

$$(n \geq 2 \wedge (\forall i : 1 \leq i < k : T[i] > T[i+1]))$$

3. Il existe une position k dans le tableau $T[1 \cdots n]$ pour laquelle $T[k]$ est plus grand que tous les autres éléments ;

$$\exists k : 1 \leq k \leq n : (\forall i : 1 \leq i \leq n : k \neq i \Rightarrow T[k] > T[i])$$

4. Aucun des éléments de $T[1 \cdots n]$ n'est plus petit que k et plus grand que p $(k \leq p)$

$$\forall i : 1 \leq i \leq n : k \leq T[i] \leq p$$

Résolution par la manipulation du symbole somme : On veut trouver un terme en n pour la somme $T(n) = 8 + 13 + 18 + 23 + \cdots + (5n + 3), (n > 0)$ et $T(0) = 0$.

On écrit :

$$\begin{aligned} T(n) &= \sum_{i=1}^{n} (5.i + 3) \\ &= 5\sum_{i=1}^{n} + \sum_{i=1}^{n} 3 \\ &= 5\frac{n(n+1)}{2} + 3.n \\ &= \frac{5}{2}n^2 + \frac{11}{2}n \end{aligned}$$

On peut aussi résoudre le problème par la méthode «conjecturer et tester». La méthode «conjecturer puis tester»consiste en les trois phases suivantes :

- conjecturer que la somme $T(n)$ est un polynome $G(n)$ de degré p par analogie avec un résultat bien connu (ici, un polynome du 2ème degré par analogie avec $1 + 2 + \cdot + n = \mathcal{O}(n^2)$) ;
- déterminer les $p + 1$ constantes ;
- En supposant que $G(n)$ c'est $T(n)$, montrer que $G(n+1)$ c'est $T(n+1)$ (pas de récurrence) ;

On veut calculer $T(n) = 8 + 13 + 18 + 23 + \cdots + (5n + 3), (n > 0)$ et $T(0) = 0$. Supposons (conjecture = "guess") que $T(n)$ soit une expression quadratique, disons de la forme $c_1n^2 + c_2n + c_3 = G(n)$

- si $n = 0 \Rightarrow c_3 = 0$
- (1) si $n = 1 \Rightarrow 1c_1 + 1c_2 = 8$ (c_3 est nul à priori)
- (2) si $n = 2 \Rightarrow 4c_1 + 2c_2 = 13 + 8$

On multiplie (1) par 2 et on soustrait à (2). On obtient $2c_1 = 5 \Rightarrow c_1 = 2.5$ et $c_2 = 5.5$. Supposons donc que $G(n) = 2.5n^2 + 5.5n$ soit la bonne expression.

Il nous reste à prouver (par induction) que $G(n)$ c'est bien $T(n)$. On a déjà prouvé le cas de base. Supposons donc $G(n) = T(n)$ et prouvons que $G(n + 1) = T(n + 1)$

$$\begin{aligned} T(n+1) &= T(n) + 5(n+1) + 3(par\ hyp\ d'induction) \\ &= G(n) + 5(n+1) + 3 \\ &= 2.5n^2 + 5.5n + 5n + 8 \\ &= 2.5n^2 + 5n + 2.5 + 5.5n + 5.5 \\ &= 2.5(n+1)^2 + 5.5(n+1) \end{aligned}$$

d'où le théorème : $8 + 13 + 18 + 23 + \cdots + (5n + 3) = 2.5n^2 + 5.5n$

Remarque : c'est en conjecturant puis en testant (un invariant) que l'on prouve la correction d'un algorithme ce qui justifie l'utilisation ici de cette deuxième méthode (lien avec les exercices de preuve de programmes).

Preuves de programmes Soit la fonction f suivante où x, y sont des entiers :

```
int f(int x,y)
{
  int z,t;

  z=x ; t=y;
  while(t>0) {
    z = z - 1;
    t = t - 1;
  }
  return z;
}
```

Que calcule cette fonction ? Faites une preuve du programme en repérant un invariant de boucle.

La fonction f calcule $x - y$. **Preuve :** On remarque qu'à chaque itération de la boucle la différence $z - t = cte$. On prend donc comme invariant de boucle $z - t = x - y$. En effet $z_0 - t_0 = x - y$ puisque $z_0 = x$ et $t_0 = y$. Supposons maintenant que $z_n - t_n = x - y$. On a :

$$z_{n+1} - t_{n+1} = z_n - 1 - (t_n - 1) = z_n - t_n = x - y$$

par hypothèse de récurrence. En fin de boucle, $t = 0$ et donc $z = x - y$. □

Dites ce que fait le programme suivant et faites la preuve.

```
#include<stdio.h>

int f(int x, int y)
{
  int z,t;

  z=2*x ; t=2*y;
  if(t>0)
     do{
        z = z - 1;
        t = t - 2;
     }while(t>0);
  return z;
}

main(){
   int x,y;
   scanf("%d",&x);scanf("%d",&y);
   printf("Resultat (x=%d, y=%d) = %d\n",x,y,f(x,y));
}
```

Le programme calcule $z = 2x - y$ avec $y > 0$. **Preuve :** On remarque qu'à chaque itération de la boucle la différence $2.z - t = cte$. On prend donc comme invariant de boucle $2.z - t = 4.x - 2.y$. En effet $2.z_0 - t_0 = 4.x - 2.y$ puisque $z_0 = 2.x$ et $t_0 = 2.y$. La suite c'est comme on a vu précédemment.

Considérer maintenant le programme suivant :

```
program  mult;
données : x entier ≥ 0 ; y réel;

début
    {x ≥ 0}              IS : spécification d'entrée
    prod := 0 ; u:= 0 ;
    {prod = uy}          Invariant de boucle
    tantque u <> x faire
        prod := prod + y ;
        u : = u + 1 ;
    fintantque
    {prod = uy et  u = x}  condition de terminaison
    {prod = xy}            OS : spécification de sortie

    sortie de l'algorithme : prod vaut xy

fin.
```

L'étape importante dans la correction d'un algorithme est d'établir des assertions portant sur les variables de l'algorithme. On vous demande de prouver le théorème IS $\Rightarrow$ OS que l'on doit interpréter de la manière suivante :

Si IS (Input Specification) est vraie au début du programme, alors OS (Output Specification) est vraie lorsque l'algorithme termine.

Preuve : on distingue les assertions concernant la preuve et les assertions concernant l'algorithme en les entourant de { et }.

Spécification d'entrée IS : $\{x \geq 0\}$: la valeur de vérité de cette proposition dépend des valeurs des variables d'entrée de l'algorithme. Ici, on affirme que x doit être un entier supérieur ou égal à 0 (c'est une contrainte). Si l'on est plus verbeux, on écrira :

$$\{x, y : entiers \bullet x \geq 0, -\infty < y < \infty\}$$

Note : l'utilisation du terme «spécification»est naturelle parce que IS précise les types, la nature des variables (objets) en entrée du programme.

Invariant de boucle : chaque boucle doit être précédée et suivie par des assertions. Celle de tête s'appelle un *invariant* de boucle, celle de queue la *condition de terminaison de boucle*. Dans notre cas $\{prod = uy\}$ affirme que chaque fois que la boucle est répétée, la valeur courante de prod est égale à uy (u et y étant les valeurs courantes de ces variables au tour de boucle considéré).

Le problème : comment choisir un invariant de boucle ? Il n'y a pas vraiment de réponse toute faite, il faut avant tout bien examiner le problème.

La condition de terminaison est la suivante dans notre cas :

$$\{prod = uy\ etu = x\}\ \ (2)$$

que l'on aurait pu écrire :

$$\{prod = xy\}\ \ (3)$$

puisque $(2) \Rightarrow (3)$. On pose maintenant les notations suivantes :

$p_1 : x \geq 0$
$p_2 : prod = uy$
$p_3 : prod = uy\ etu = x$
$p_4 : prod = xy$

On veut montrer le théorème :

$$p_1 \Rightarrow p_4$$

Pour le monter par «déduction naturelle», il est suffisant de montrer chacun des théorèmes suivants :

$p_1 \Rightarrow p_2$
$p_2(avant) \Rightarrow p_2(apres)$
$p_2(apres) \Rightarrow p_3$
$p_3 \Rightarrow p_4$

Le deuxième théorème est utile pour montrer que si l'invariant de boucle est vrai à chaque entrée dans la boucle, il doit être vrai au dernier passage dans la boucle.

Note : ces théorèmes ont une nature un peu différente de ceux vus précédemment car ils contiennent des variables numériques.

1. lorsque l'on arrive au niveau de l'invariant de boucle, l'assertion $\{prod = uy\}$ est correcte parce que $prod$ et u viennent d'être initialisées à 0. Donc $p_1 \Rightarrow p_2$;

2. est-ce que p_2 est encore vraie au dernier passage dans la boucle ? Si l'on définit $p_2(k)$ comme étant la proposition : p_2 est vraie pour $u = k$, le théorème $p_2(avant) \Rightarrow p_2(\text{après})$ devient $p_2(k) \Rightarrow p_2(k+1)$ pour $k \in [0 \cdots x-1]$ (induction finie) :

   - cas de base : $k = 0$: on vient de le justifier ;
   - étape d'induction : soit $prod_k$ la valeur de $prod$ avant l'exécution de la boucle pour une valeur particulière $u = k$. L'hypothèse d'induction est donc : $prod_k = ky$. La valeur de $prod$ après cette exécution (et juste avant la prochaine itération) est :

$$\begin{aligned} prod_{k+1} &= prod_k + y \\ &= ky + y[par\ hyp\ d'induction] \\ &= (k+1)y \\ &= p_2(k+1) \end{aligned}$$

ce qui termine la preuve du théorème $p_2(k) \Rightarrow p_2(k+1)$;

3. la condition de terminaison de boucle est vraie parce que l'on vient de voir que $prod = uy$ à chaque fin d'itération et $u = x$ est la condition qui nous fait sortir. Donc $p_2 \Rightarrow p_3$;

4. $p_3 \Rightarrow p_4$ est vraie car la véracité de la condition de terminaison implique la véracité de la spécification de sortie OS ;

Nous avons terminé la preuve de la **correction** de l'algorithme. On doit encore s'assurer que l'algorithme **termine** i.e OS est «atteignable».

La seule chose qui peut faire que l'algorithme ne termine pas est la boucle (le reste s'enchaînant séquentiellement, il n'y a pas de problème). Puisque u et initialisée à 0 et que x est supérieur ou égal à 0 si IS est vraie ; $u \leq x$ est vraie lorsque l'on rentre pour la première fois dans la boucle. Chaque fois que la boucle est exécutée, u est augmentée de 1. Il arrivera forcément un moment où u aura la même valeur que x et la boucle s'arrêtera.

Note : la preuve n'est pas valide si IS est fausse !

Que calcule le programme suivant. Faites la preuve de correction partielle. (Vous pouvez rappeler au passage l'ensemble de la démarche). On supposera de plus que $x > 0$.

```
int f(int x, int y)
{
  int z,t,v,res;

  z=4*x ; t=2*y; v=z+t; res=0;
  if(t>0)
    do{
        res = res + z; v = v - 2;
    }while(v>0);
  return res;
}
```

```
Résultats d'exécutions :
f(1,1)=12  f(2,1)=40  f(3,1)=84
f(1,2)=16  f(2,2)=48  f(3,2)=96
f(1,3)=20  f(2,3)=56  f(3,3)=108
f(1,4)=24  f(2,4)=64  f(3,4)=120
f(1,5)=28  f(2,5)=72  f(3,5)=132

f(4,1)=144 f(5,1)=220
f(4,2)=160 f(5,2)=240
f(4,3)=176 f(5,3)=260
f(4,4)=192 f(5,4)=280
f(4,5)=208 f(5,5)=300
```

La fonction proposée calculait $8x^2 + 4xy$. Un invariant possible était :

$$res + \frac{v}{2} * z = 8x^2 + 4xy$$

Cas de base : $res = 0, v/2 = 2x + y, z = 4x$ et donc $8x^2 + 4xy = 8x^2 + 4xy$, okay.

Héritage : supposons $res_n + \frac{v_n}{2} * z = 8x^2 + 4xy$. A l'itération $n+1$ on a $res_{n+1} = res_n + 4x$ et $v_{n+1} = v_n - 2$. On a donc :

$$\begin{aligned} res_{n+1} + \tfrac{v_{n+1}}{2} * z &= res_n + 4x + \tfrac{v_n - 2}{2} * 4x \\ &= res_n + 4x + 4x * (\tfrac{v_n}{2} - 1) \\ &= res_n + \tfrac{v_n}{2} * z \\ &= 8x^2 + 4xy \quad \text{(par hypothèse d'induction)} \end{aligned}$$

En sortie de boucle : $v = 0$ et donc $res = 8x^2 + 4xy$ qui est retourné.

4.3 Le problème du triangle

Nous proposons une implantation en langage Python au problème consistant à déterminer un triangle connaissant une hauteur, un angle et une longueur de segment.

Dans le premier paragraphe nous rappelons la définition du problème ainsi que sa résolution mathématique «à la règle et au compas». Au cours de cette construction nous faisons apparaitre un point particulier M à partir duquel le triangle peut se déduire. En spécialisant ce point nous isolons des propriétés de la construction qui nous permettent ensuite d'implanter la solution de manière aisée en tracant un cercle et une droite.

Dans le deuxième paragraphe nous établissons un lien entre l'équation traditionnelle du cercle et les courbes de Bézier ce qui nous permet de dériver, dans le troisième paragraphe, des fonctions pour le tracer d'un quart de cercle puis d'un cercle.

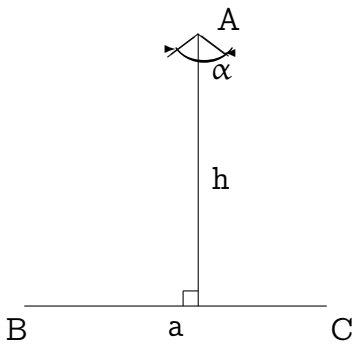
La conclusion suit et termine l'exercice. Le code source Python est ensuite donné.

4.3.1 Formulation mathématique de la solution

Les étapes élémentaires de la construction

Ce problème est dû à Polya. Notre solution est originale.

Le problème s'énonce ainsi : construire un triangle connaissant un coté a, la hauteur h perpendiculaire à a et l'angle α opposé au côté a.



On commence par nommer deux points B, C qui sont les extrémités du segment a. On ramene le problème à la construction du point A après avoir tracé le segment BC. On a donc changé à la fois l'inconnue et les données en remplaçant l'inconnue par l'une des données et vice et versa. Le nouveau problème est le suivant :

- **inconnue** : le point A
- **données** : h, α et 2 points B et C
- **condition** : la distance du point A à la droite BC mesurée perpendiculairement à cette droite doit être égal à h et $\widehat{BAC} = \alpha$.

La *condition* comprend deux parties. Si l'on néglige la 2ème partie, les points situés sur la droite parallèle à BC à la distance h de cette droite satisfont la condition. Il est ensuite facile de vérifier que le lieu géométrique des points qui satisfont la 2ème partie est l'arc de cercle dont les extrémités sont B et C. Enfin l'intersection de ces deux lieux géométriques est le point recherché.

On décompose le problème en sous problème et on montre comment recomposer le résultat final à partir des résultats intermédiaires

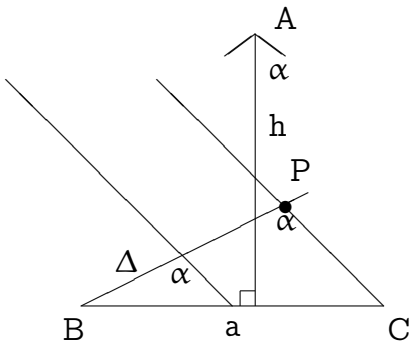


FIGURE 4.1: *Construction à la règle et au compas*

La construction du point A est alors la suivante :

1. On trace une droite quelconque Δ qui passe par B.
2. Δ coupe le cercle en un point P obtenu en faisant glisser un triangle d'angle α le long de Δ jusqu'à rencontrer C (cf Figure 4.1).

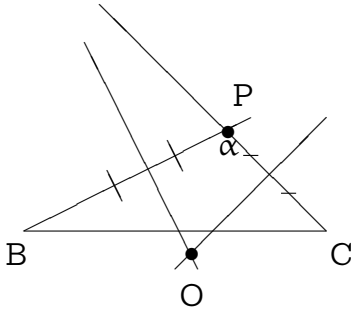


FIGURE 4.2: *Construction de O*

3. Il reste à déterminer le cercle circonscrit au triangle BPC. On dérive ainsi un nouveau problème plus simple que l'on sait résoudre. En effet, il suffit de considèrer les *médiatrices* des segments BP et PC : le point d'intersection des médiatrices est le centre O du cercle recherché (cf Figure 4.2).

Le(s) point(s) d'intersection[1] de la droite et du cercle permettent trivialement de tracer le(s) triangle(s) demandé(s) dans l'énoncé.

Spécialisation du point P

Le cas particulier de la construction que nous discutons maintenant permet de tracer le cercle de centre O uniquement en fonction de BC et α : le calcul des coordonnées de O est purement trigonométrique.

Nous invitons le lecteur à se reporter à partir de maintenant à la Figure 4.3. On considère P tel que $\widehat{BPC}$ soit isocèle puis on trace le coté PI. Pour le triangle $\widehat{BPI}$ on a $\tan(\alpha/2) = BI/PI$.

On nomme Med_{BP} la médiatrice de BP et on se place dans le repère cartésien tel qu'il est indiqué à la Figure 4.3. On peut alors considérer M (milieu de BP). On a alors :

```
XP = float(float((XC - XB) / 2) + XB)
XI = XP
XO = XP
YI = YB
YP = float(YB - float(float(XP - XB) / math.tan(ALPHA/2)))
XM = (XC - XB) / 4 + XB
YM = YB - ((YB - YP)/ 2)
YO = YM + float(float(XI - XM) * math.tan(float(ALPHA / 2)))
R  = int(YO - YP)
```

Ainsi les caractéristiques du cercle (coordonnées du centre et rayon) sont déterminées. On remarquera que la hauteur h n'intervient pas dans ce calcul. Cependant, il est clair qu'une fois le cercle construit, la solution de notre problème

[1] L'unicité de la solution sera discutée plus loin

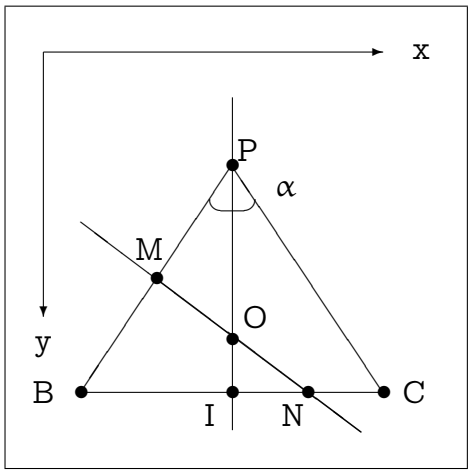


FIGURE 4.3: *Construction du cercle à partir d'un triangle isocèle*

est donnée par l'intersection du cercle avec la droite parallèle à BC à la hauteur h et donc h devra être une valeur vérifiant :

$$h \leq PI \Rightarrow h \leq \frac{XC - XB}{2 * \tan(\alpha/2)}$$

Autrement dit on devra avoir $h \leq YP$ pour avoir au moins un point d'intersection entre le cercle et la droite.

Intersections entre la droite et le cercle

Pour déterminer par le calcul les points d'intersection entre la droite de hauteur h et le cercle centré en $(X0, Y0)$ de rayon R nous égalons les équations du cercle et de la droite. Ainsi nous avons :

$$Y0 + R\sin(\theta) = Y_h$$

ce qui determine deux angles $\theta_1 = \arcsin((h - Y0)/R)$ et $\theta_2 = \pi - \theta_1$ (Examinez les relations entre $h, Y0, R$ au moment du calcul de la fontion arcsin). Les points d'intersection du cercle et de la droite ont donc pour coordonnées :

$$Q1 : \begin{cases} XQ1 = X0 + R\cos(\theta_1) \\ YQ1 = y_h \end{cases}$$

$$Q2 : \begin{cases} XQ2 = X0 + R\cos(\theta_2) \\ YQ2 = y_h \end{cases}$$

Notre programmation correspond au code :

```
#
# On passe au calcul des points solutions
#
thetau = math.asin(math.pi/180*((H - Y0) / R))
thetad = math.pi - thetau
```

```
XQ1 = XO + R * math.cos(thetau)
YQ1 = YB - H
XQ2 = XO + R * math.cos(thetad)
YQ2 = YB - H
```

ce qui clos le paragraphe consacré à la construction du triangle. Le code Python correspondant ce trouve à la fin de cette étude de cas.

4.3.2 Le tracé de courbes de Bézier

Introduction

Nous venons de voir que la solution fait intervenir le tracé d'un cercle. Nous proposons maintenant une étude visant à terme à calculer la solution en passant par des courbes particulières : les courbes de Bézier.

La courbe de Bézier que nous utilisons dans cet exercice est définie par la paire d'équations paramétrées de paramètre $t\,(0 \leq t \leq 1)$ suivante :

$$\begin{aligned} x(t) &= a_x t^3 + b_x t^2 + c_x t + x_0 \qquad \textit{coordonnées en } x \\ y(t) &= a_y t^3 + b_y t^2 + c_y t + y_0 \qquad \textit{coordonnées en } y \end{aligned}$$

Les points de contrôle sont définis par :

$$\begin{aligned} x_1 &= x_0 + c_x/3 & y_1 &= y_0 + c_y/3 \\ x_2 &= x_1 + (c_x + b_x)/3 & y_2 &= y_1 + (c_y + b_y)/3 \\ x_3 &= x_0 + c_x + b_x + a_x & y_3 &= y_0 + c_y + b_y + a_y \end{aligned}$$

L'algorithme de calcul est simple : connaissant les tangentes à l'origine, les coefficients $a_x, b_x, c_x, a_y, b_y, c_y$ sont facilement calculables et on obtient pour notre implantation en C le code suivant utilisant un tableau `coef` et les objets de type «point» $e1, e2, e_3, e4$ (reportez-vous au code source) :

```
/*       Calcul des coefficients des polynomes     */
/*       calcul de ax, bx, cx                      */
coef[2]= 3*(e2.x - e1.x);                  /* cx */
coef[1]= 3*(e3.x - e2.x) - coef[2];        /* bx */
coef[0]= e4.x - e1.x - coef[2] - coef[1] ; /* ax */
/*       calcul de ay, by, cy                      */
coef[5]= 3*(e2.y - e1.y);                  /* cy */
coef[4]= 3*(e3.y - e2.y) - coef[5];        /* by */
coef[3]= e4.y - e1.y - coef[5] - coef[4] ; /* ay */
```

Il reste à choisir un pas de calcul (dans notre implantation nous effectuons systématiquement 1001 itérations par défaut). Plus la valeur de t est petite, plus il y aura de points calculés et donc meilleure sera la résolution du tracé[2]. Puis il reste à afficher le pixel calculé : dans notre implantation la fonction de la bibliothèque s'appelle `GrPlot`. Ainsi, la portion de code correspondante est :

[2] On peut aussi choisir de ne calculer qu'un petit nombre de points et de les relier deux à deux par des droites : on obtient un tracé sous forme de lignes brisées

```
/*    Dessin point a point sur 1001 points par splines */
for(t=0;t<=1;t=t+0.001)
{
      x = coef[0]*t*t*t + coef[1]*t*t + coef[2]*t + e1.x ;
      y = coef[3]*t*t*t + coef[4]*t*t + coef[5]*t + e1.y ;
      GrPlot(x,y,180);
}
```

Tracé d'un cercle à partir d'une courbe de Bézier

Dans ce paragraphe nous allons nous intéresser plus précisément au tracé d'un quart de cercle, le tracé d'un cercle pouvant alors se déduire du tracé de 4 quarts de cercle en réajustant les valeurs des tangentes de manière appropriée.

Le problème qui se pose est donc de simuler le tracé d'un cercle au moyen d'un spline donc d'une courbe paramétrique, ici de degré 3. L'approximation que nous proposons dans ce paragraphe introduit un terme correcteur qui rendra compte (en moyenne) de l'écart entre les valeurs exactes (obtenues par les équations du cercle) et approximées (obtenues par les équations de Bézier).

Equation du cercle et approximation

On rappelle que l'équation paramétrique du cercle s'exprime par exemple de la manière suivante :

$$(1)\begin{cases} x - x_0 = R\cos(t) \\ y - y_0 = R\sin(t) \end{cases}$$

avec (x_0, y_0) les coordonnées du centre du cercle, R son rayon et $0 \leq t \leq \pi/2$[3].

De plus, on sait que les développements limités des fonctions sinus et cosinus sont :

$$(2)\begin{cases} \sin(x) = x - \frac{x^3}{3!} + \cdots + (-1)^n \frac{x^{2n+1}}{(2n+1)!} + \prime(x^{2n+3}) \\ \cos(x) = 1 - \frac{x^2}{2!} + \cdots + (-1)^n \frac{x^{2n}}{(2n)!} + \prime(x^{2n+2}) \end{cases}$$

Ici, seul l'ordre 3 nous intéresse. En effectuant le changement de variable $t \rightarrow (\pi/2)T$ pour se ramener dans le bon intervalle et en réalisant des identifications sur les coefficients des pôlynomes on obtient successivement :

$$(3)\begin{cases} x(T) = R * (1 - \frac{\pi^2 T^2}{8}) - x_0 \\ y(T) = R * (\frac{\pi}{2}T - \frac{\pi^3 T^3}{48}) \end{cases}$$

puis

$$(4)\begin{cases} a_x = 0 & a_y = \frac{-R\pi^3}{48} \\ b_x = \frac{-R\pi^2}{8} & b_y = 0 \\ c_x = 0 & c_y = \frac{R\pi}{2} \end{cases}$$

[3]Il est nécessaire de faire un changement de variable sur t afin de travailler sur l'intervalle $[0 \cdots 1]$

Correction de l'erreur d'approximation

La Figure 4.4 donne une représentation pour 1000 points calculés de la différence entre les valeurs exactes (obtenues par les équations (2)) et les valeurs approximées (obtenues par les équations (4) pour chacun des axes *pour un cercle de rayon 150 pixels*. Il s'agit des courbes repérées `x.pic` et `y.pic`. Les courbes repérées `corx` et `cory` sont les courbes qui donnent les termes correcteurs sur chacun des axes. Nous avons choisi de manière expérimentale de les définir de la façon suivante :

$$\begin{cases} corx = \exp(3.6x) - 1 \\ cory(x) = \exp(2.5x^2) - 1 \end{cases}$$

Rappel immédiat : *ces fonctions ne sont valables que pour un cercle de rayon 150 pixel.*

La Figure 4.4 est obtenue avec GnuPlot et le programme suivant :

```
corx(x)=exp(3.6*x)-1
cory(x)=exp(x*x*2.5)-1
plot [0:1] 'c:/x.pic' with lines, 'c:/y.pic' with lines, corx(x), cory(x)
```

Le code C qui permet de tracer le quart de cercle inférieur droit et qui tient compte des facteurs correctifs est maintenant le suivant (cf fichier source) :

```
for(t=0.0;t<=1.0;t=t+0.001)
{
  /* valeurs approchees (ax = cx = by = 0) */
  x = (int)(coef[1]*t*t + 320 + R + exp(3.6*t) -1);
  y = (int)(coef[3]*t*t*t + coef[5]*t + 240 + exp(t*t*2.5) - 1);
  GrPlot(x,y,250);
}
```

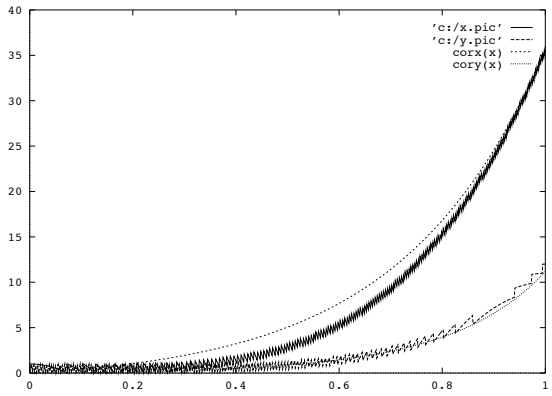


FIGURE 4.4: *Représentaion de l'erreur, sa correction*

4.3.3 Conclusion

Dans ce compte rendu nous avons proposé l'étude et la résolution (y compris la programmation en langage Python) du problème du triangle ainsi qu'une étude permettant de tracer un cercle à partir des équations d'un spline. Ces deux problèmes permettent de mieux appréhender les différentes phases de développement d'un programme qui nous font passer d'un énoncé en langage naturel à un texte dans un langage de programmation. Les solutions appellent tantôt des arguments mathématiques : c'est le cas pour la construction «formelle»de la solution du problème du triangle, tantôt à des arguments et à des choix pratiques déterminés par le contexte du problème : c'est le cas pour l'approximation du tracé d'un cercle par un spline.

Comme approfondissements nous pouvons envisager d'étudier d'une part la prise en compte du tracé d'un spline quelque soit le rayon (pour l'instant le terme correctif n'intervient que pour le rayon 150). D'autre part, il serait particulièrement intéressant d'essayer d'autres approximations (changer les polynômes) ainsi que la modélisation de l'erreur par d'autres types de fonctions (linéaire...). Il serait alors envisageable d'effecter des mesures de performance et d'étudier en particulier le temps de calcul de chacun des tracés. Pour l'instant rien ne nous permet d'affirmer que notre implantation est optimale sur ce plan là !

Le fichier source

```
#
# Programmation de l'exemple du triangle
# Christophe Cérin : 10/01/2003
#
# Note 1 : les angles a passer aux fonctions trigonométriques
# Python sont à exprimer en radians
#
# Note 2 : pour un rendu plus précis, il faudrait revoir
# les arrondis et programmer qu'avec des floats
#

from Tkinter import *
import math

#
# Fonctions permettant d'afficher le texte des boites de dialogue
#
def  resultat(self):
    print ent.get()

def  resultat1(self):
    print ent1.get()

def  resultat2(self):
```

```
    print ent2.get()

#
# On cree une nouvelle fenetre avec son titre
#
root = Tk()
root.title("Demo")

#
# La fenetre principale est composée d'un frame et d'un canvas
# Le frame est composé de plusieurs widgets (zones de saisie
# + boutons pour effacer, dessiner, quitter
#
f = Frame(root, relief='ridge', borderwidth=2)
f.grid()
l = Label(f, text='Angle: ').grid(row=0, sticky=E)
#l.grid(row=0,column=0)
ent    =  Entry(f,text="ttt",width=15)
ent.grid(row=0, column=1)
ent.insert(0,"45")
ent.config(bg="#FFEEFF",fg="#FF0000")
ent.bind('<Return>',resultat)

#
# On cree un label associé au frame f
#
l1 = Label(f, text='Hauteur h: ').grid(row=1, sticky=E)
#l1.grid(row=0,column=0)
ent1    =  Entry(f,width=15)
ent1.grid(row=1, column=1)
ent1.insert(0,"200")
ent1.config(bg="#FFEEFF",fg="#FF0000")
ent1.bind('<Return>',resultat1)

#
# On cree un label associé au frame f
#
l2 = Label(f, text='Segment BC: ').grid(row=2, sticky=E)
#l1.grid(row=0,column=0)
ent2    =  Entry(f,width=15)
ent2.grid(row=2, column=1)
ent2.insert(0,"250")
ent2.config(bg="#FFEEFF",fg="#FF0000")
ent2.bind('<Return>',resultat2)

#
# On ajoute un bouton pour quitter
#
button = Button(f, text="QUIT", fg="red", command=root.destroy)
```

```
button.grid(row=3,colum=0,sticky=E)

#
# On crée un canvas
#
cv = Canvas(root, width = 400, height=300, bg="#AAAAAA",
            relief=RIDGE, borderwidth=2)
cv.grid()
#Pour  quitter,  cliquer  dans  le  cadre
cv.bind("<Button>",quit)

def del_cv():
    """ Détruit tous les objets dessinés dans le canvas cv """
    for item in cv.find_all():
        cv.delete(item)

def draw_cv():
    """ Dessine des objets dans le canvas cv """
    cv.delete(*cv.find_all())
    if int(ent.get()) > 180:
        cv.create_text(100,10,text="ERROR:", fill=("blue"))
        cv.create_text(100,25,text="0 <= angle <= 180", fill=("blue"))
        return
    if int(ent1.get()) > 290:
        cv.create_text(100,10,text="ERROR:", fill=("blue"))
        cv.create_text(100,25,text="0 <= h <= 290", fill=("blue"))
        return
    if int(ent2.get()) > 370:
        cv.create_text(100,10,text="ERROR:", fill=("blue"))
        cv.create_text(100,25,text="0 <= BC <= 370", fill=("blue"))
        return
    a  = int(ent2.get())
    b  = 400 - a
    XB = b/2
    YB = 290
    XC = XB + a
    YC = 290
    #
    # On verifie qu'il y a une solution
    #
    H     = int(ent1.get())
    #
    # Convert angle to radian
    #
    ALPHA = float(int(ent.get())*math.pi/180.0)
    if (H > ((XC - XB) / (2 * math.tan(ALPHA / 2)))):
       cv.create_text(200,25,text="DESOLE, IL N'Y A PAS DE SOLUTION !!!",
                      fill=("blue"))
        return
```

```
#
# On y va : il y a un seul ou deux triangles qui sont solutions
#
XP = float(float((XC - XB) / 2) + XB)
XI = XP
XO = XP
YI = YB
#print "Segment BC:(",XB,YB,XC,YC,") I:",XI,YI
YP = float(YB - float(float(XP - XB) / math.tan(ALPHA/2)))
#print "YP:",YP,"XP",XP,"XB",XB,math.tan(ALPHA/2)
XM = (XC - XB) / 4 + XB
YM = YB - ((YB - YP)/ 2)
# XN = int((YB - YM) / math.tan(ALPHA / 2)) + XM
# YN = YM
# print XM,YM,XN,YN
YO = YM + float(float(XI - XM) * math.tan(float(ALPHA/2)))
R  = int(YO - YP)
#
# Le cercle est maintenant défini par son rayon R et
# son centre (XO,YO). On le dessine
#
print "Cercle de centre ",XO,YO, "et de rayon : ", R
#print XO-R/2,YO-R/2,XO+R/2,YO+R/2
oval  =  cv.create_oval((XO-R,XO-R,XO+R,YO+R),fill="#FFFAF0",
                        outline="#FF0100",width=1)
#
# On dessine aussi la hauteur H et le segment BC
#
cv.create_line(XP,YB,XP,YB-H, fill=("red"))
cv.create_line(XB,YB,XC,YC, fill=("red"))
cv.create_text(XB-8,YB,text="B", fill=("blue"))
cv.create_text(XC+7,YC,text="C", fill=("blue"))
#
# On passe au calcul des points solutions
#
thetau = math.asin(math.pi/180*((H - YO) / R))
thetad = math.pi - thetau
print "H-YO =",H-YO," (H - YO) / R =",(H - YO) / R
print "theta1=",thetau," theta2=",thetad
XQ1 = XO + R * math.cos(thetau)
YQ1 = YB - H
XQ2 = XO + R * math.cos(thetad)
YQ2 = YB - H
#
# On trace les triangles solutions et on place quelques points
#
cv.create_line(XB,YB,XQ1,YQ1, fill=("red"))
cv.create_line(XQ1,YQ1,XC,YC, fill=("red"))
cv.create_line(XB,YB,XQ2,YQ2, fill=("red"))
```

```
    cv.create_line(XQ2,YQ2,XC,YC, fill=("red"))
    cv.create_text(XQ1,YQ1-7,text="Q1", fill=("blue"))
    cv.create_text(XQ2,YQ2-7,text="Q2", fill=("blue"))
    oval1  =  cv.create_oval((float(XO-1),float(YO-1),float(XO+1), \
                 float(YO+1)),fill="#0000FF",outline="#0000FF",width=1)
    cv.create_text(XO+8,YO-6,text="O", fill=("blue"))
    cv.create_line(0,YQ1,400,YQ2, fill=("red"))
    print "-------------------------------------------------"
    print "VOICI LES SOLUTIONS :"
    print "Q1=(",XQ1,",",YQ1,")  Q2=(",XQ2,",",YQ2,")"
    print "-------Attention aux erreurs d'arrondis----------"
#
# On ajoute 2 boutons dans le frame f
#
button1 = Button(f, text="CLEAR", fg="magenta", command=del_cv)
button1.grid(row=4,colum=0,sticky=E)

button2 = Button(f, text="DRAW", fg="blue", command=draw_cv)
button2.grid(row=5,colum=0,sticky=E)

root.mainloop()
```

Un résultat d'éxécution daans une fenêtre texte donne les coordonées des points permettant de construire la solution comme suit :

```
ordinateur-de-christophe-cerin-2:~$ python triangle.py
Cercle de centre  200.0 165.207106781 et de rayon :  176
H-YO = 34.7928932188  (H - YO) / R = 0.197686893289
theta1= 0.00345029402161  theta2= 3.13814235957
-------------------------------------------------
VOICI LES SOLUTIONS :
Q1=( 375.998952403 , 90 )  Q2=( 24.0010475975 , 90 )
-------Attention aux erreurs d'arrondis----------
```

Le même exemple donne la solution graphique de la figure 4.5.

4.4 Vers un prototype gérant les formats Xfig

Le problème que nous traitons maintenant consiste à écrire une interface permettant de générer des dessins au format Xfig (http://www.xfig.org). Le programmeur écrit un code Python décrivant les objets du dessin, par exemple :

L'objectif de ce compte-rendu, sur un problème de codification pure, est de faire ressortir le travail fournit en restant général, sans rentrer dans tous les détails techniques

```
#########################################################################
# programme principal et exemple
#########################################################################

def __main__():
table = newColorTable()
```

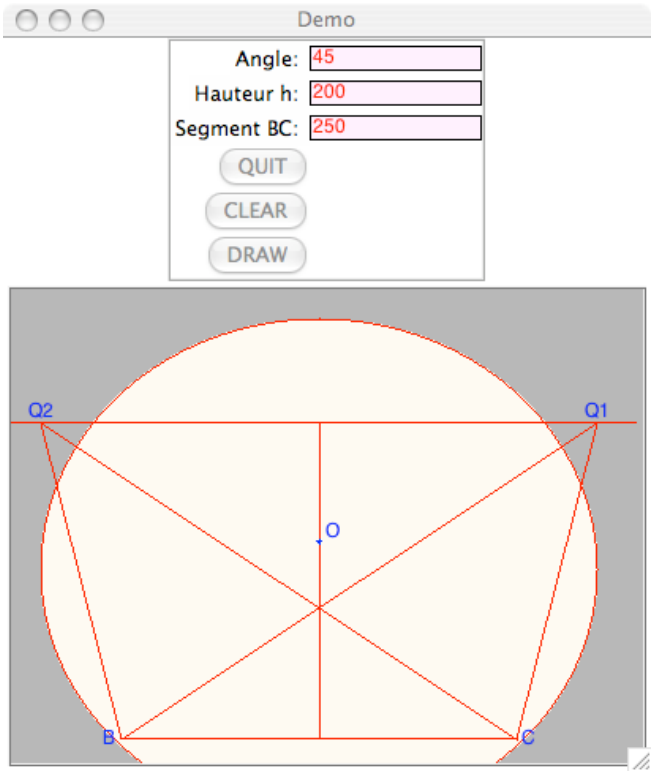


FIGURE 4.5: *Un exemple de présentation graphique de la solution*

```
red = getColor(table,colorName('RED'))
paleBlue = getColor(table,colorName('RGB',80,127,255))

basic=defaultProps()
redBricks = setAreaFill(areaFill('FILL_BRICK',None),
                    setPenColor(red,basic))

oval=ellipse(redBricks,
                    ellipsePoints('ELLIPSE_BOX',point(1700.0,300.0),
                    point(3700.0,1300.0),None))

Rempli=setAreaFill(areaFill('FILL_CROSS',None),
                    setPenColor(paleBlue,basic))
oval1 = ellipse(Rempli,
                    ellipsePoints('ELLIPSE_BOX',point(5700.0,300.0),
                    point(7700.0,1300.0),None))

basic["lt"] = lineThickness('LINE_THICKNESS',10)
Rempli2=setAreaFill(areaFill('NOT_FILLED',None),
                    setPenColor(paleBlue,basic))
oval2 = ellipse(Rempli2,ellipsePoints('ELLIPSE_CENTER_BOX',
                    point(5700.0,2300.0),point(7700.0,3300.0),None))

pageToFile('test.fig','LANDSCAPE','A4',table,[oval,oval1,oval2])
```

```
###################### PHASE DE TEST ############################

__main__()
```

Le code Python interprète les objets (ici trois ellipses et des propriétés sur les traits, la couleur du dessin) et écrit dans un fichier texte la description Xfig correspondante, ici :

```
#FIG 3.2
#PYTHON JARAXFIG
Landscape
Center
Metric
A4
100.0
Single
-1
1200 2
0 32 #507fff

1 4 -1 1 4 -1 50 0 47 10.0 0 0.0 2700 800 1000 500 1700 300 3700 1300
1 4 -1 1 32 -1 50 0 51 10.0 0 0.0 6700 800 1000 500 5700 300 7700 1300
1 3 -1 10 32 -1 50 0 -1 10.0 0 0.0 5700 2300 2000 1000 5700 2300 7700 3300
```

Lorsqu'on lance l'utilitaire Xfig sur ce fichier texte ainsi généré, nous obtenons le résultat présenté à la figure 4.6 où l'on voit le dessin des trois ellipses.

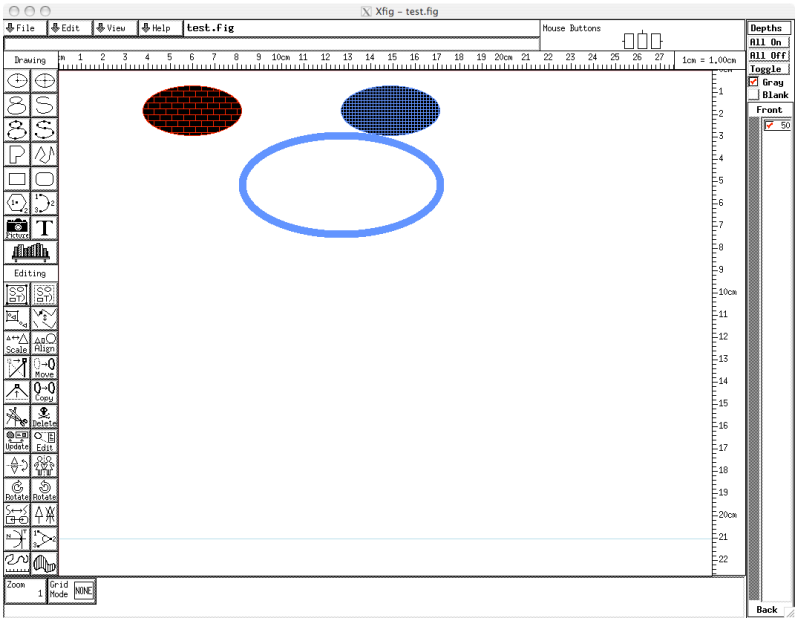


FIGURE 4.6: *Un exemple de dessin Xfig*

Pour présenter le travail réalisé, nous avons choisi de ne pas nous concentrer sur les objets Xfig et leurs propriétés (ce point technique peut être abordé en

consultant la documentation en ligne de Xfig) mais de cibler notre propos sur les grandes lignes de l'implémentation Python c'est à dire sur les choix des structures de données et de l'organisation générale du programme.

Nous sommes partis de l'outil Jara-Xfig http://www.acooke.org/jara/mllib/index.html qui propose un code SML pour réaliser l'interface et nous l'avons adapté en Python de sorte que :

1. seul l'objet ellipse est capté y compris le remplissage par un motif ;
2. la gestion des couleurs Xfig y compris celles définies par l'utilisateur est complète ;
3. la gestion des traits (forme, epaisseur) est complète ;
4. la gestion du format du papier, de l'orientation, de la mise à l'échelle est complète.

La décision d'implémentation principale a été de créer une classe Python pour chaque objets Xfig (par exemple la classe arrow implémente la notion de flêche) et pour leurs propriétés (par exemple la classe `arrowStyle` permet de définir le style des flêches).

Sur cette base nous avons aussi construit des fonctions de service (par exemple la fonction `range`) qui effectuent de la vérification sur le type des paramêtres ou les fonctions de gestion des entrées sorties dans un fichier (fonction `newLine` par exemple). A ce titre le lecteur notera que la fonction `output` qui n'est pas définie comme une classe mais elle contient la définition d'autres fonctions de service. Cela permet de masquer la gestion du transcodage vers les objets Xfig. On remarque par ailleurs à la lecture de la fonction `output`, notamment le programme principal et comme cela a déjà été dit, que le seul objet géré par notre prototype est l'ellipse.

Une autre famille de fonctions a été nécessaire pour fixer les propriétés des objets : la famille des fonctions dont le préfixe est 'set'. Dans le prototype actuel ces fonctions ne sont pas définies au niveau de la classe correspondante. Par exemple, la classe `lineStyle` a une seule méthode (initialisation succinte) et la méthode `setLineStyle` (initialisation des propriétés des lignes) est définie à l'extérieure de la classe. Nous envisageons de remodeler notre code dans le sens d'une meilleure structuration.

Notez qu'à aucun moment dans les explications nous utilisons du jargon technique

Enfin, la gestion de l'entête du fichier texte généré mérite quelques explications. L'entête permet de fixer par exemple l'orientation de la page, l'échelle, les unités de mesure, la transparence, les couleurs définies par l'utilisateur. Nous avons donc définies des classes relatives à chacun de ces attributs, puis nous avons écrit une classe 'header' dont la méthode d'initialisation récupère les attributs et les écrits dans le fichier texte. Nous en profitons également pour faire de la vérifiation de type sur les attributs. La fonction `pageToFile` qui écrit l'entête dans le fichier se résume principalement à la création d'une instance de la classe header comme suit :

```
MyHeader=header(fileN,orientation(ori),justification('CENTER_PAGE'),
                units('METRIC'),papersize(ps),
                magnification('UNIT_SCALE',None),
                singleMultiple('SINGLE_PAGE'),
                transparent('BACKGROUND',None),
                resolution('DEFAULT_RESOLUTION',None),ct)
```

Ici on résume son travail et on propose des pistes d'améliorations futures

En conclusion, notre prototype en Python pourrait maintenant être augmenté pour capter d'autres objets (par exemple les boîtes carrées, les lignes brisées). Cependant, les propriétés Xfig des traits (y compris la couleur), des motifs de remplissage sont complètes ce qui devrait permettre de se concentrer maintenant sur la génération de ces objets au format Xfig.

Le code est le suivant :

```
import math

######################################################################
# CLASS VECTOR.
# Ré-implémentation de vecteur pour ajout d'une méthode cons
######################################################################

class vector:
     def __init__(self):
          self.data=[]

     empty=[]

     def size(self):
          return self.data.__len__()

     def isEmpty(self): return (self.data==self.empty)

     def find(self,rgb):
          if self.data.__contains__(rgb):
               return self.data.index(rgb)
          else:
               return (-1)

     def cons(self,rgb):
          self.data.append(rgb)

######################################################################
#     DEFNITION DES CLASSES XFIG
######################################################################

class lineStyle:
     def __init__(self,tip):
          if ['DEFAULT_LINE','SOLID','DASHED','DOTTED','DASH_DOTTED',
              'DASH_DOUBLE_DOTTED','DASH_TRIPLE_DOTTED'].__contains__(tip):
```

```
                self.type=tip
            else:
                raise 'erreur de type (XFig - class lineStyle)'

class lineThickness:
    def __init__(self,tip,val):
        if tip=='LINE_THICKNESS' and (type(val)==int):
            self.type=tip
            self.data=val
        else:
            raise 'erreur de type (XFig - class lineThickness)'

class color:
    def __init__(self,tip,val):
        if tip=='COLOR' and (type(val)==int):
            self.type=tip
            self.data=val
        else:
            raise 'erreur de type (XFig - class color)'
class depth:
    def __init__(self,tip,val):
        if tip=='DEPTH' and (type(val)==int):
            self.type=tip
            self.data=val
        else:
            raise 'erreur de type (XFig - class depth)'
class penStyle:
    def __init__(self,tip):
        if ['UNUSED'].__contains__(tip):
            self.type=tip
        else:
            raise 'erreur de type (XFig - class penStyle)'
class areaFill:
    def __init__(self,tip,val=None):
        if ['FILL_CODE'].__contains__(tip) and (type(val)==int):
            self.type=tip
            self.data=val
        elif ['FILL_SHADE','FILL_TINT'].__contains__(tip) and
             (type(val)==float):
            self.type=tip
            self.data=val
        elif ['NOT_FILLED','FILL_FOREGROUND','FILL_BACKGROUND',
              'FILL_WHITE','FILL_30_LEFT','FILL_30_RIGHT',
              'FILL_30_CROSS','FILL_45_LEFT','FILL_45_RIGHT',
              'FILL_45_CROSS','FILL_BRICK','FILL_CIRCLES',
              'FILL_HORIZONTAL','FILL_VERTICAL','FILL_CROSS',
              'FILL_SCALES','FILL_SMALL_SCALES','FILL_OCTAGONS',
              'FILL_HORIZONTAL_TREADS',
              'FILL_VERTICAL_TREADS'].__contains__(tip) and (val==None):
```

```
                self.type=tip
                self.data=val
          else:
                raise 'erreur de type (XFig - class areaFill)'
class styleVal:
     def __init__(self,tip,val):
          if tip=='STYLE_VAL' and (type(val)==float):
                self.type=tip
                self.data=val
          else:
                raise 'erreur de type (XFig - class styleVal)'
class joinStyle:
     def __init__(self,tip):
          if ['JOIN_MITER','JOIN_BEVEL','JOIN_ROUND'].__contains__(tip):
                self.type=tip
          else:
                raise 'erreur de type (XFig - class joinStyle)'
class capStyle:
     def __init__(self,tip):
          if ['CAP_BUTT','CAP_ROUND','CAP_PROJECTING'].__contains__(tip):
                self.type=tip
          else:
                raise 'erreur de type (XFig - class capStyle)'
class radius:
     def __init__(self,tip,val):
          if tip=='RADIUS' and type(val)==int:
                self.type=tip
                self.data=val
          else:
                raise 'erreur de type (XFig - class radius)'
class direction:
     def __init__(self,tip):
          if ['CLOCKWISE','COUNTER_CLOCKWISE'].__contains__(tip):
                self.type=tip
          else:
                raise 'erreur de type (XFig - class direction)'
class arrowType:
     def __init__(self,tip):
          if ['STICK','TRIANGLE',
              'INDENTED_BUTT','POINTED_BUTT'].__contains__(tip):
                self.type=tip
          else:
                raise 'erreur de type (XFig - class arrowType)'

class arrowStyle:
     def __init__(self,tip):
          if ['HOLLOW','FILLED'].__contains__(tip):
                self.type=tip
          else:
```

```
                raise 'erreur de type (XFig - class arrowStyle)'


class arrow:
     def __init__(self,aof,at,as,r1,r2,r3):
          if aof=='ARROW_OFF':
               self.type=aof
               self.data=None
          elif aof=='ARROW_ON' and (type(r1),type(r2),type(r3))==
                                   (float,float,float):
               self.type=aof
               self.data=(arrowType(at),arrowStyle(as),r1,r2,r3)
          else:
               raise 'erreur de type (XFig - class arrow)'


class angle:
     def __init__(self,tip,val):
          if tip=='ANGLE' and type(val)==float:
               self.type=tip
               self.data=val
          else:
               raise 'erreur de type (XFig - class angle)'
class font:
     def __init__(self,tip):
          if ['DEFAULT_LATEX','ROMAN','BOLD','ITALIC','SANS_SERIF',
              'TYPEWRITER','DEFAULT_POSTSCRIPT','TIMES_ROMAN',
              'TIMES_ITALIC','TIMES_BOLD','TIMES_BOLD_ITALIC',
              'AVANTGARDE_BOOK','AVANTGARDE_BOOK_OBLIQUE',
              'AVANTGARDE_DEMI','AVANTGARDE_DEMI_OBLIQUE','BOOKMAN_LIGHT',
              'BOOKMAN_LIGHT_ITALIC','BOOKMAN_DEMI','BOOKMAN_DEMI_ITALIC',
              'COURIER','COURIER_OBLIQUE','COURIER_BOLD',
              'COURIER_BOLD_OBLIQUE','HELVETICA','HELVETICA_OBLIQUE',
              'HELVETICA_BOLD','HELVETICA_BOLD_OBLIQUE',
              'HELVETICA_NARROW','HELVETICA_NARROW_OBLIQUE',
              'HELVETICA_NARROW_BOLD','HELVETICA_NARROW_BOLD_ITALIC',
              'NEW_CENTURY_SCHOOLBOOK_ROMAN',
              'NEW_CENTURY_SCHOOLBOOK_ITALIC','NEW_CENTURY_SCHOOLBOOK_BOLD',
              'NEW_CENTURY_SCHOOLBOOK_BOLD_ITALIC','PALATINO_ROMAN',
              'PALATINO_ITALIC','PALATINO_BOLD','PALATINO_BOLD_ITALIC',
              'SYMBOL','ZAPF_CHANCERY_MEDIUM_ITALIC',
              'ZAPF_DINGBATS'].__contains__(tip):
               self.type=tip
          else:
               raise 'erreur de type (XFig - class font)'
class fontRigid:
     def __init__(self,tip):
          if ['RIGID_FONT','SCALABLE_FONT'].__contains__(tip):
               self.type=tip
```

```
          else:
               raise 'erreur de type (XFig - class fontRigid)'
class fontSpecial:
     def __init__(self,tip):
          if ['LATEX_FONT','NORMAL_FONT'].__contains__(tip):
               self.type=tip
          else:
               raise 'erreur de type (XFig - class fontSpecial)'
class fontHidden:
     def __init__(self,tip):
          if ['FONT_HIDDEN','FONT_VISIBLE'].__contains__(tip):
               self.type=tip
          else:
               raise 'erreur de type (XFig - class fontHidden)'
class fontSize:
     def __init__(self,tip,val):
          if ['POINTS'].__contains__(tip) and type(val)==int:
               self.type=tip
               self.data=val
          else:
               raise 'erreur de type (XFig - class fontSize)'

class colorName:
     couleur=['DEFAULT_COLOR','BLACK','BLUE','GREEN','CYAN','RED',
              'MAGENTA','YELLOW','WHITE','BLUE_4','BLUE_3','BLUE_2',
              'BLUE_1','GREEN_3','GREEN_2','GREEN_1','CYAN_3','CYAN_2',
              'CYAN_1','RED_3','RED_2','RED_1','MAGENTA_3','MAGENTA_2',
              'MAGENTA_1','BROWN_3','BROWN_2','BROWN_1','PINK_4',
              'PINK_3','PINK_2','PINK_1','GOLD']
     def __init__(self,tip,I1=None,I2=None,I3=None):
          if self.couleur.__contains__(tip):
               self.type=tip
               self.data=None
          elif tip=='RGB' and (type(I1),type(I2),type(I3))==(int,int,int):
               self.type='RGB'
               self.data= I1,I2,I3
          else:
               raise 'erreur de type (XFig - class font)'

######################################################################
#     DEFNITION DES CLASSES POINT, ELLIPSE ET OBJECT
######################################################################

class point:
     def __init__(self,x,y):
          self.type='POINT'
          self.data=(x,y)

class ellipseSubType:
```

```
     def __init__(self,subType):
          if ['ER','ED','CR','CD'].__contains__(subType):
               self.type=subType
               self.data=None
          else:
               raise 'erreur de type (XFig - class ellipseSubType)'

class ellipsePoints:
     def __init__(self,type,point1,point2,reel):
          if ['CIRCLE_DIAMETER','ELLIPSE_CENTER_BOX',
              'ELLIPSE_BOX'].__contains__(type) and
                          (point1.type,point2.type)==('POINT','POINT'):
               self.type=type
               self.data=(point1,point2)
          elif ['CIRCLE_CENTER'].__contains__(type) and
                            point1.type=='POINT' and type(reel)==float:
               self.type=type
               self.data=(point1,reel)
          else:
               raise 'erreur de type (XFig - class ellipsePoints)'


class object:
     def __init__(self,obj,props,subtype,point1,point2):
          if obj=='ELLIPSE' and (point1.type,point2.type)==('POINT','POINT'):
               self.type=obj
               self.data=(props,ellipseSubType(subtype),point1,point2)
          else:
               raise 'erreur de type (XFig - class object)'

##########################################################
#     FONCTION RANGE
##########################################################

def rangei(mn,mx,x):
     if (type(mn),type(mx),type(x))==(int,int,int):
          if x<mn:
               raise 'Valeur trop petite'
          elif x>mx:
               raise 'Valeur trop grande'
     else:
          raise 'erreur de type (XFig - fonction rangei)'


def range0(mx,x):
     rangei(0,mx,x)

def rangePos(x):
     range0(40000,x)
```

```
def rangeFillCode(x):
     rangei(-1,56,x)

def rangeDepth(x):
     range0(999,x)

def rangeColor(x) :
     range0(255,x)


def range2(mn,mx,x):
     if (type(mn),type(mx),type(x))==(float,float,float):
          if x<mn:
               raise 'Valeur trop petite'
          elif x>mx:
               raise 'Valeur trop grande'
     else:
          raise 'erreur de type (XFig - fonction range2)'

def range02(mx,x):
     range2(0.0,mx,x)

def rangePos2(x):
     range02(20000.0,x)

def rangeScale2(x):
     range02(1.0,x)


#####################################################################
#     FONCTION DEFAULTPROPS
#       Cette fonction permet, via un dictionnaire, de
#       retrouver les propriété d'un objet xfig
#####################################################################

def defaultProps():
     return( {"ls":lineStyle('DEFAULT_LINE'),
              "lt":lineThickness('LINE_THICKNESS',1),
              "pc":color('COLOR',-1),"fc":color('COLOR',-1),
              "dp":depth('DEPTH',50),"ps":penStyle('UNUSED'),
              "af":areaFill('NOT_FILLED',None),
              "sv":styleVal('STYLE_VAL',10.0),"cs":capStyle('CAP_BUTT'),
              "js":joinStyle('JOIN_MITER'),"rd":radius('RADIUS',10),
              "dr":direction('CLOCKWISE'),
              "fa":arrow('ARROW_OFF',None,None,None,None,None),
              "ba":arrow('ARROW_OFF',None,None,None,None,None),
              "ag":angle('ANGLE',0.0),"ft":font('DEFAULT_POSTSCRIPT'),
              "fr":fontRigid('SCALABLE_FONT'),
```

```
             "fx":fontSpecial('NORMAL_FONT'),
             "fh":fontHidden('FONT_VISIBLE'),"fs":fontSize('POINTS',12)})


############################################################
#     FONCTIONS SET :
#       Elles permettent de fixer des propriétés à des objets
############################################################

def setLineStyle(ls2,props):
     props2=props.copy()
     props2["ls"]=ls2
     return(props2)

def setLineThickness(n,props):
     props2=props.copy()
     rangePos(n)
     props2["lt"]=lineThickness('LINE_THICKNESS',n)
     return(props2)

def setPenColor(pc2,props):
     props2=props.copy()
     props2["pc"]=pc2
     return(props2)

def setFillColor(fc2,props):
     props2=props.copy()
     props2["fc"]=fc2
     return(props2)

def setDepth(n,props):
     props2=props.copy()
     rangeDepth(n)
     props2["dp"]=depth('DEPTH',n)
     return(props2)

def setPenStyle(ps2,props):
     props2=props.copy()
     props2["ps"]=ps2
     return(props2)

def setAreaFill(af2,props):
     if af2=='FILL_CODE':
          rangeFillCode(af2.data)
     elif af2=='FILL_SHADE':
          rangeScale2(af2.data)
     elif af2=='FILL_TINT':
          rangeScale2(af2.data)
```

```
     props2=props.copy()
     props2["af"]=af2
     return(props2)

def setStyleVal(n,props):
     props2=props.copy()
     rangePos2(n)
     props2["sv"]=styleVal('STYLE_VAL',n)
     return(props2)

def setCapStyle(cs2,props):
     props2=props.copy()
     props2["cs"]=cs2
     return(props2)

def setJoinStyle(js2,props):
     props2=props.copy()
     props2["js"]=js2
     return(props2)

def setRadius(n,props):
     props2=props.copy()
     rangePos(n)
     props2["rd"]=radius('RADIUS',n)
     return(props2)

def setDirection(dr2,props):
     props2=props.copy()
     props2["dr"]=dr2
     return(props2)

def setForwardArrow(fa2,props):
     props2=props.copy()
     props2["fa"]=fa2
     return(props2)

def setBackwardArrow(ba2,props):
     props2=props.copy()
     props2["ba"]=ba2
     return(props2)

def setAngle(ag2,props):
     props2=props.copy()
     props2["ag"]=ag2
     return(props2)

def setFont(ft2,props):
     props2=props.copy()
     props2["ft"]=ft2
```

```
     return(props2)

def setFontRigid(fr2,props):
     props2=props.copy()
     props2["fr"]=fr2
     return(props2)

def setFontSpecial(fx2,props):
     props2=props.copy()
     props2["fx"]=fx2
     return(props2)

def setFontHidden(fh2,props):
     props2=props.copy()
     props2["fh"]=fh2
     return(props2)

def setFontSize(n,props):
     props2=props.copy()
     rangePos(n)
     props2["fs"]=fontSize('POINTS',n)
     return(props2)

######################################################################
# FONCTION GETCOLOR :
# Gestion des couleur de l'utilisateur
######################################################################


def newColorTable():
     return vector()

def getColor(vec, rgb):
     if vec.__class__!=vector or rgb.__class__!=colorName:
          raise 'erreur de type (XFig - getColor)'
     else:
          if rgb.type=='RGB':
               rangeColor(rgb.data[0])
               rangeColor(rgb.data[1])
               rangeColor(rgb.data[2])
               i=vec.find(rgb.data)
               if i==-1:
                    vec.cons(rgb.data)
                    return color('COLOR',vec.size()+31)
               else:
                    return color('COLOR',i+32)
          else:
               return color('COLOR',colorName.couleur.index(rgb.type)-1)
```

```
########################################
#  GESTION DES FLUX d'I/O
########################################

def appString(os,s):
     os.write(s)

def newLine2(os):
     appString(os,"\n")

def appSpace(os,s):
     appString(os,s+" ")

def appI2(os,n):
     if n<0:
          string="-"+str(int(-n))
     else:
          string=str(int(n))
     appSpace(os,string)

def appR2(os,x):
     if x<0:
          string="-"+str(float(-x))
     else:
          string=str(float(x))
     appSpace(os,string)

def appB2(os,b):
     if b:
          val=1
     else:
          val=0
     appI2(os,val)

def appP2(os,x):
     val=round(x)
     appI2(os,val)

############################################################
#     FONCTION OUTPUT
#       Ecriture dans un fichier d'un objet xfig
############################################################

def output (os, obj):
     def newLine():
          newLine2(os)

     def appI(n):
          appI2(os,n)
```

```
def appR(x):
     appR2(os,x)

def appP(x):
     appP2(os,x)

def appLineStyle(ls):
     if ls.type=='DEFAULT_LINE':
          lsret=-1
     elif ls.type=='SOLID':
          lsret=0
     elif ls.type=='DASHED':
          lsret=1
     elif ls.type=='DOTTED':
          lsret=2
     elif ls.type=='DASHED_DOTTED':
          lsret=3
     elif ls.type=='DASHED_DOUBLE_DOTTED':
          lsret=4
     elif ls.type=='DASHED_TRIPLE_DOTTED':
          lsret=5
     appI(lsret)

def appLineThickness(lt):
     appI(lt.data)

def appColor(cl):
     appI(cl.data)

def appDepth(dp):
     appI(dp.data)

def appPenStyle(ps):
     appI(0)

def appAreaFill(af):
     def scaleToRange(x,mn,mx):
          mn2=float(mn)
          mx2=float(mx)
          return( round( mn2+(x*(mx2-mn2))) )
     if af.type=='FILL_CODE':
          afret=af.data
     elif af.type=='NOT_FILLED':
          afret=-1
     elif af.type=='FILL_FOREGROUND':
          afret=0
     elif af.type=='FILL_SHADE':
          afret=scaleToRange(af.data,0,20)
```

```
        elif af.type==’FILL_BACKGROUND’:
             afret=20
        elif af.type==’FILL_TINT’:
             afret=scaleToRange(af.data,20,40)
        elif af.type==’FILL_WHITE’:
             afret=40
        elif af.type==’FILL_30_LEFT’:
             afret=41
        elif af.type==’FILL_30_RIGHT’:
             afret=42
        elif af.type==’FILL_30_CROSS’:
             afret=43
        elif af.type==’FILL_45_LEFT’:
             afret=44
        elif af.type==’FILL_45_RIGHT’:
             afret=45
        elif af.type==’FILL_45_CROSS’:
             afret=46
        elif af.type==’FILL_BRICK’:
             afret=47
        elif af.type==’FILL_CIRCLES’:
             afret=48
        elif af.type==’FILL_HORIZONTAL’:
             afret=49
        elif af.type==’FILL_VERTICAL’:
             afret=50
        elif af.type==’FILL_CROSS’:
             afret=51
        elif af.type==’FILL_SCALES’:
             afret=52
        elif af.type==’FILL_SMALL_SCALES’:
             afret=53
        elif af.type==’FILL_OCTAGONS’:
             afret=54
        elif af.type==’FILL_HORIZONTAL_TREADS’:
             afret=55
        elif af.type==’FILL_VERTICAL_TREADS’:
             afret=56
        appI(afret)

   def appStyleValue(sv):
        appR(sv.data)

   def appDirection(dr):
        if dr.type==’CLOCKWISE’:
             drret=0
        elif dr.type==’COUNTER_CLOCKWISE’:
             drret=1
        appI(drret)
```

```
def appAngle(an):
     appR(an.data)

def appXy(x,y):
     appP(x)
     appP(y)

def appEllipse(subType,x1,y1,x2,y2):
     if subType=='CR':
          appXy(x1,y1)
          appXy(x2-x1,x2-x1)
          appXy(x1,y1)
          appXy(x2,y2)
     elif subType=='CD':
          r=sqrt( (x2-x1)*(x2-x1)+(y2-y1)*(y2-y1) )
          appXy(0.5*(x1+x2),0.5*(y1+y2))
          appXy(r,r)
          appXy(x1,y1)
          appXy(x2,y2)
     elif subType=='ER':
          appXy(x1,y1)
          appXy(x2-x1,y2-y1)
          appXy(x1,y1)
          appXy(x2,y2)

     elif subType=='ED':
          appXy(0.5*(x1+x2),0.5*(y1+y2))
          appXy(0.5*(x2-x1),0.5*(y2-y1))
          appXy(x1,y1)
          appXy(x2,y2)

# Programme principal de la fonction 'output'
for i in obj:
     if i.type=='ELLIPSE':
          appI(1)
          if (i.data[1]).type=='CR':
               st=1
          elif (i.data[1]).type=='CD':
               st=2
          elif (i.data[1]).type=='ER':
               st=3
          elif (i.data[1]).type=='ED':
               st=4
          appI(st)
          appLineStyle((i.data[0])["ls"])
          appLineThickness((i.data[0])["lt"])
          appColor((i.data[0])["pc"])
          appColor((i.data[0])["fc"])
```

```
                appDepth((i.data[0])["dp"])
                appPenStyle((i.data[0])["ps"])
                appAreaFill((i.data[0])["af"])
                appStyleValue((i.data[0])["sv"])
                appDirection((i.data[0])["dr"])
                appAngle((i.data[0])["ag"])
                appEllipse((i.data[1]).type,(i.data[2]).data[0],
                      (i.data[2]).data[1],(i.data[3]).data[0],
                      (i.data[3]).data[1])
                newLine()


############################################################
#     FONCTION ELLIPSE
############################################################


def ellipse(props,subType):
          if subType.type==’CIRCLE_CENTER’:
               return(object(’ELLIPSE’,props,’CR’,
                      point((subType.data[0]).data[0]+subType.data[1],
                            (subType.data[0]).data[1]),None))
          elif subType.type==’CIRCLE_DIAMETER’:
               return(object(’ELLIPSE’,props,’CD’,
                      point((subType.data[0]).data[0],
                            (subType.data[0]).data[1]),
                             point((subType.data[1]).data[0],
                              (subType.data[1]).data[1])))
          elif subType.type==’ELLIPSE_CENTER_BOX’:
               return(object(’ELLIPSE’,props,’ER’,
                      point((subType.data[0]).data[0],
                            (subType.data[0]).data[1]),
                             point((subType.data[1]).data[0],
                                   (subType.data[1]).data[1])))
          elif subType.type==’ELLIPSE_BOX’:
               return(object(’ELLIPSE’,props,’ED’,
                           point((subType.data[0]).data[0],
                                 (subType.data[0]).data[1]),
                                    point((subType.data[1]).data[0],
                                          (subType.data[1]).data[1])))

############################################################
#     DEFINITION DES CLASSES RELATIVES AU HEADER
############################################################

class orientation:
     def __init__(self,ori):
          if [’LANDSCAPE’,’PORTRAIT’].__contains__(ori):
               self.type=ori
```

```
          else:
               raise 'erreur de type (XFig - class orientation)'


class justification:
     def __init__(self,js):
          if ['CENTER_PAGE','LEFT_PAGE'].__contains__(js):
               self.type=js
          else:
               raise 'erreur de type (XFig - class justification)'


class units:
     def __init__(self,un):
          if ['METRIC','INCHES'].__contains__(un):
               self.type=un
          else:
               raise 'erreur de type (XFig - class units)'



class papersize:
     def __init__(self,ps):
          if ['LETTER','LEGAL','LEDGER','TABLOID','A','B','C','D',
              'E','A4','A3','A2','A1','A0'].__contains__(ps):
               self.type=ps
          else:
               raise 'erreur de type (XFig - class papersize)'


class magnification:
     def __init__(self,mg,p):
          if mg=='UNIT_SCALE':
               self.type=mg
          elif mg=='PERCENTAGE' and type(p)==float:
               self.type=mg
               self.data=p
          else:
               raise 'erreur de type (XFig - class magnification)'



class singleMultiple:
     def __init__(self,sm):
          if ['SINGLE_PAGE','MULTIPLE_PAGES'].__contains__(sm):
               self.type=sm
          else:
               raise 'erreur de type (XFig - class singleMultiple)'
```

```
class transparent:
     def __init__(self,tr,c):
          if tr=='NO_TRANSPARENT':
               self.type=tr
               self.data=None
          elif tr=='BACKGROUND':
               self.type=tr
               self.data=None
          elif tr=='TRANSPARENT' and c.type=='COLOR':
               self.type=tr
               self.data=c
          else:
               raise 'erreur de type (XFig - class transparent)'

class resolution:
     def __init__(self,rs,u):
          if rs=='DEFAULT_RESOLUTION':
               self.type=rs
               self.data=None
          elif rs=='UNITS_PER_INCH' and type(u)==int:
               self.type=rs
               self.data=u
          else:
               raise 'erreur de type (XFig - class resolution)'


####################################
# CLASS HEADER
####################################

class header:
     def newLine(self):
          newLine2(self.os)
     def appS(self,s):
          appString(self.os,s)
          self.newLine()
     def appR(self,x):
          appR2(self.os,x)
          self.newLine()
     def appI(self,x):
          appI2(self.os,x)
          self.newLine()
     def dumpColor(self,table):
          def toHex3(x):
               if x<10:
                    return(str(x))
               else:
                    return(str(chr(ord('a')+x-10)))
```

```
        def toHex2(x):
             return(str(toHex3(int(x/16)))+str(toHex3(int(math.fmod(x,16)))))

        def toHex((r,g,b)):
             return(toHex2(r)+toHex2(g)+toHex2(b))

        def appColor(table):
             for i in range(0,table.size()):
                  self.appS("0 "+str(i+32)+" #"+toHex(table.data[i])+"\n")

        # Appel de la méthode principale
        appColor(table)

    def __init__(self,name,ori,js,un,ps,mg,sm,tr,rs,ct):
        self.os=open(name,'w')
        self.appS("#FIG 3.2")
        self.appS("#PYTHON JARAXFIG")
        if ori.type=='LANDSCAPE':
             orientation="Landscape"
        elif ori.type=='PORTRAIT':
             orientation="Portrait"
        self.appS(orientation)
        if js.type=='CENTER_PAGE':
             justification="Center"
        elif js.type=='LEFT_PAGE':
             justification="Left"
        self.appS(justification)
        if un.type=='METRIC':
             units="Metric"
        elif un.type=='INCHES':
             units="Inches"
        self.appS(units)
        if ps.type=='LETTER':
             papersize="Letter"
        elif ps.type=='LEGAL':
             papersize="Legal"
        elif ps.type=='LEDGER':
             papersize="Ledger"
        elif ps.type=='TABLOID':
             papersize="Tabloid"
        elif ps.type=='A':
             papersize="A"
        elif ps.type=='B':
             papersize="B"
        elif ps.type=='C':
             papersize="C"
        elif ps.type=='D':
             papersize="D"
```

```
          elif ps.type==’E’:
               papersize="E"
          elif ps.type==’A4’:
               papersize="A4"
          elif ps.type==’A3’:
               papersize="A3"
          elif ps.type==’A2’:
               papersize="A2"
          elif ps.type==’A1’:
               papersize="A1"
          elif ps.type==’A0’:
               papersize="A0"
          elif ps.type==’B5’:
               papersize="B5"
          self.appS(papersize)
          if mg.type==’PERCENTAGE’:
               rangePos2(mg.data)
               magnification=mg.data
          elif mg.type==’UNIT_SCALE’:
               magnification=100.0
          self.appR(magnification)
          if sm.type==’SINGLE_PAGE’:
               singlemultiple="Single"
          elif un.type==’MULTIPLE_PAGES’:
               singlemultiple="Multiple"
          self.appS(singlemultiple)
          if tr.type==’TRANSPARENT’:
               transparent=tr.data
          elif tr.type==’NO_TRANSPARENT’:
               transparent=-2
          elif tr.type==’BACKGROUND’:
               transparent=-1
          self.appI(transparent)
          if rs.type==’UNITS_PER_INCH’:
               rangePos(rs.data)
               resolution=str(int(mg.data))+" 2"
          elif rs.type==’DEFAULT_RESOLUTION’:
               resolution="1200 2"
          self.appS(resolution)
          self.dumpColor(ct)


#########################################################################
#     FONCTION PAGETOFILE
#########################################################################

def pageToFile(fileN,ori,ps,ct,obj):
     MyHeader=header(fileN,orientation(ori),justification(’CENTER_PAGE’),
                     units(’METRIC’),papersize(ps),
```

```
                    magnification('UNIT_SCALE',None),
                    singleMultiple('SINGLE_PAGE'),
                    transparent('BACKGROUND',None),
                    resolution('DEFAULT_RESOLUTION',None),ct)
    output(MyHeader.os,obj)
    MyHeader.os.close


######################################################################
#     programme principal et exemple
######################################################################

# Reprendre le code de __main__ qui figure plus haut dans le document.
# Merci
```

4.5 Simulateur de cache

4.5.1 Présentation du problème

Ici nous commençons par un rappel complet des notions importantes avant de nous lancer dans la construction du simulateur. La section devient auto-suffisante.

Un vieux principe énoncé en architecture des ordinateurs dit que la plupart des programmes n'accèdent pas aux données de manière unifome. Ce principe connu sous le nom de *principe de localité* a conduit les architectes à considérer que via du matériel spécialisé de petite taille on pouvait construire une hiérarchie mémoire de différentes tailles et de différentes vitesses de traitement.

Qu'est ce que l'on entend par *hiérarchie*? Puisque de la mémoire rapide est chère, une hiérarchie mémoire est organisée en plusieurs niveaux – qui font correspondre des adresses mémoire (virtuelle) d'un niveau à un autre : les niveaux. De manière générale, les niveaux sont inclus les uns dans les autres dans le sens où toutes les données d'un niveau sont aussi trouvées dans le niveau inférieur et toutes les données dans un niveau inférieur se retrouve dans le niveau supérieur. De cette façon chaque niveau fait correspondre des adresses d'un grand ensemble de mémoire dans un plus petit ensemble plus rapide d'accès et plus haut dans la hiérarchie. Une autre fonction rendue par le système hiérarchique de mémoire est d'effectuer la vérification d'adresses (la donnée est elle accessible par un certain processus par exemple ou bien est-elle protégée en lecture, en écriture).

Ces remarques liminaires conduisent à la notion de *cache* : en général c'est le premier niveau de hiérarchie mémoire que l'on rencontre quand les adresses quittent le processeur. Un cache contient des *blocs* de données (généralement quelques centaines par cache) et le cache est vu comme le plus haut niveau de la hiérarchie. Naturellement, si l'on veut construire un cache efficace on doit se poser les question suivantes :

Q1 : à quel endroit du cache doit-on placer un bloc ?

Q2 : comment retrouver un bloc dans le cache ?

Q3 : quel bloc remplacer quand on a un défaut de cache (la donnée n'est pas présente) ?

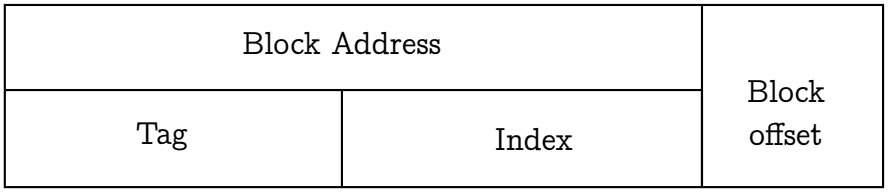
Q4 : que faire si l'on désire écrire une donnée (mise à jour du cache et/ou de la mémoire) ?

Les réponses que l'on trouve pour la question Q1 dans les processeurs actuels sont :

- Si chaque bloc a une et une seule place où il peut apparaitre dans le cache, le cache est dit *direct mapped* et la mise en correspondance se fait par la formule : (Block address) MOD (Number of Blocks in cache) ;
- Si un bloc peut être placé n'importe où dans le cache alors le cache est dit *fully associative* ;
- Si un bloc ne peut être placé que dans certaines places du cache, le cache est dit *set associative*. Un bloc est d'abord mis en correspondance avec un ensemble, puis le bloc est placé quelque part dans cet ensemble. L'ensemble est généralement choisi par la formule : (Block addess) MOD (Number of sets in cache). S'il y a n blocs dans un ensemble on dit que le placement est n-way-associative.

Pour information, la grande majorité des processeur du commerce sont soit *direct mapped*, soit *2-way-associative*, soit *4-way associatif*

Les réponses que l'on trouve pour la question Q2 (comment trouver un bloc dans le cache) sont maintenant discutées. Pour cela, le format d'une adresse vue depuis le processeur est souvent celui décrit sur la Figure ci dessous.



La première division est entre le «block address»et le «block offset». Le champs «block offset»sélectionne la donnée dans un bloc, le champs «index»sélectionne l'ensemble et le champs «tag»est comparé avec celui-ci en cas de collision.

Le format d'un bloc (adresse) vue depuis le cache est généralement constitué d'un champs «valid»pour signifier si une adresse est valide ou pas, d'un champs «Tag»qui est comparé avec celui décrit plus haut et d'un champs «data»qui contient la donnée.

L'algorithme pour une lecture est celui ci :

1. une requête de lecture est envoyée au cache ;
2. le champs «Index»permet de sélectionner un bloc ;
3. les champs «Tag»de l'adresse CPU et les champs «Tag»et «Valid»vue depuis le cache sont comparés ;

4. en cas «d'égalité», le champs «Data»du cache est envoyé au CPU ; sinon il y a un «manque de cache (miss)»envoyé au CPU.

Nous voilà maintenant face à la question [Q3] : que faire en présence d'un manque de cache ? Le contrôleur de cache doit remplacer les données du cache par la donnée que l'on recherche. Dans le cas des cache «fully associative»et «set associative»on utilise les techniques suivantes pour remplir le cache :

Random: en plus de la donnée que l'on recherche, des blocs tirés au hazard sont ramenés au niveau du cache ;

LRU: (Least Recently Used) les accès aux blocs sont enregistrés. Les blocs que l'on vire du cache sont les blocs qui ont été inutilisés depuis le plus de temps. Ainsi, si un bloc récemment utilisé s'avère être de nouveau utilisé (principe de localité des données), il sera encore dans le cache.

Il nous reste à examiner ce qui se passe sur une écriture en mémoire (question [Q4]). Il y a ici deux options généralement retenues :

Write Through (or store trough): l'information est écrite à la fois dans le bloc du cache et dans la mémoire principale ;

Write back (or copy back or store in): l'information est écrite seulement dans un bloc du cache. Le bloc modifié est écrit en mémoire principale au moment d'un remplacement.

Pour réduire la fréquence des écritures sur un remplacement dans une politique «write back», une astuce appelée «dirty bit»est employée. Ce bit d'état indique si un bloc est «dirty»(modifié dans le cache) ou «clean»(pas modifié). Si le bloc est marqué «clean», ce n'est pas la peine de l'écrire en mémoire principale lors d'un remplacement.

4.5.2 Description des différentes organisations de cache

Le cache «fully associatif»

- Si un bloc peut se retrouver n'importe où en mémoire, le cache est dit «fully associatif». Une adresse d'une information est alors composée d'un numéro de ligne et d'un offset dans la ligne. L'ensemble de ce codage implique qu'il est nécessaire d'avoir autant de comparateurs que de lignes de cache. Ainsi on peut faire toutes les comparaisons entre l'adresse émise par le processeur et toutes les adresses rangées dans le cache en parallèle

- Très coûteux à implémenter en terme de circuiterie donc peu usité.

Le cache «direct mapped»

- Chaque bloc mémoire ne peut aller que dans une seule ligne du cache dont le numéro est calculé aisément à partir du numéro de bloc mémoire par la formule *adresse bloc MOD nombre de blocs dans le cache*.

- Inconvénient : deux mots qui appartiendraient à la même ligne ne peuvent se trouver simultanément dans le cache.

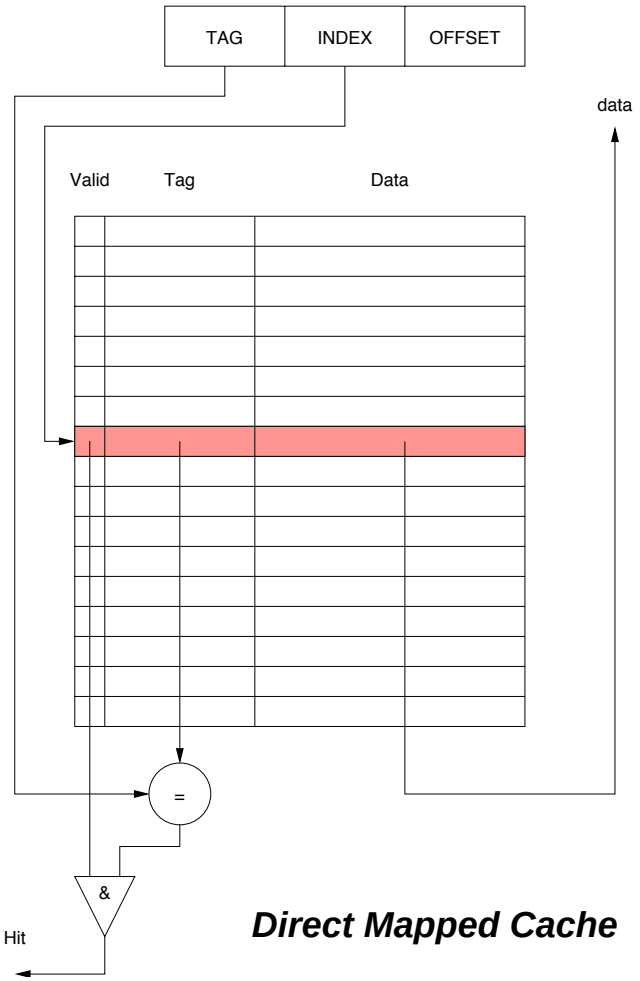


FIGURE 4.7: *Le cache direct-mapped*

Ici on a 4 blocs de 1 mot machine par ligne afin de tirer partie de la localité spaciale. Quand il y a un manque, on charge 4 mots. La sélection du mot se fait avec le champs «Index». Avantage : un seul «tag valid».

Les «set associative caches»

En français, on dit associatif par blocs.

- On associe plusieurs caches à accés direct ;

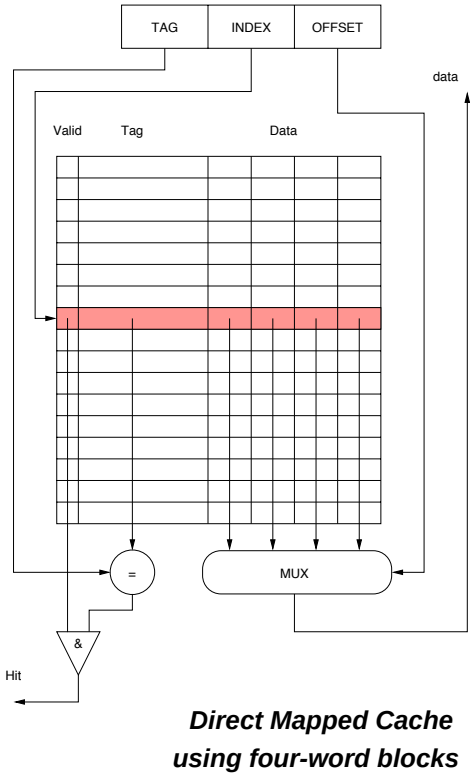


FIGURE 4.8: *Le cache direct-mapped*

- Par rapport au cache direct, plusieurs entrées sont autorisées par ligne. Si on en autorise n, on dit que le cache est un n way associative cache.
- Une information peut être placée n'importe où sur une même rangée ; Il faut donc une politique de remplacement (sur un manque, généralement le LRU) ;
- Si on considère une seule rangée, il s'agit d'un cache associatif ; si on considère un seul bloc, il s'agit d'un cache direct.
- Généralement, l'ensemble est choisi par une «bit selection»du type *block address MOD nombre d'ensemble dans le cache*

Les «set associative caches»(exemple) : Ici, une donnée peut aller dans 2 endroits du cache. Le schéma ci dessous donne l'idée générale pour retrouver une information. Dans ce schéma, le champs offset n'est pas utilisé.

- Exemple d'un 4-way associative cache : on choisit l'ensemble par le calcul du modulo, le «block address»permet alors de sélectionner 4 blocs ; le contenu des blocs est comparé à l'étiquette. Si c'est égal on envoie l'octet contenu à cette position à un multiplexeur qui permet de retourner au processeur au plus 4 bytes dans le cas où les 4 adresses correspondaient. Éventuellement, au lieu d'avoir un multiplexeur, on peut réserver des bits de l'adresse pour faire la sélection d'une unique valeur parmi 4. Ceci peut être fait par le champs offset.

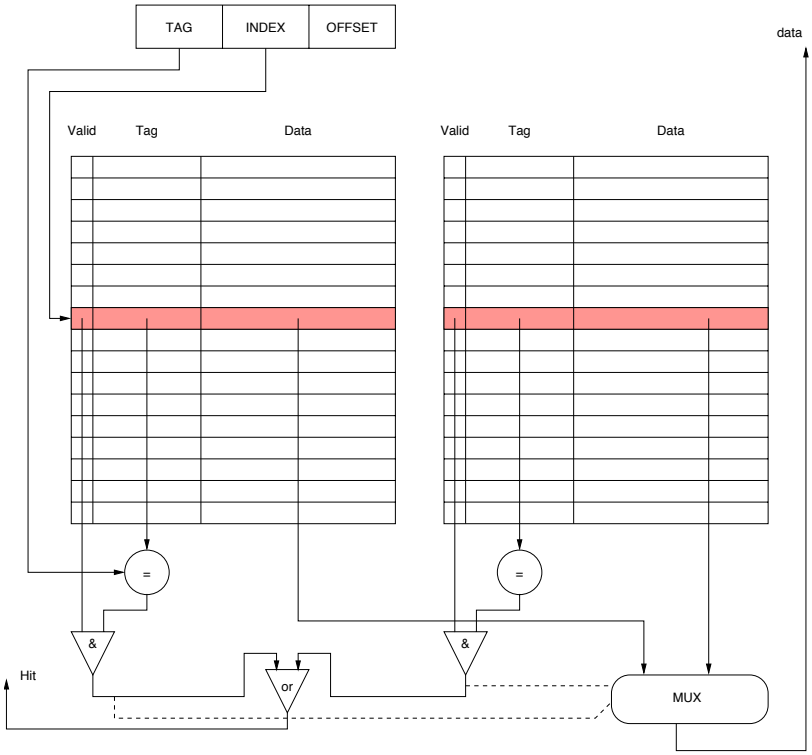


FIGURE 4.9: *Le cache 2-way associative*

Pour information, la grande majorité des processeurs du commerce sont soit *direct mapped*, soit *2-way-associative*, soit *4-way associatif*.

Le codage d'une adresse : exemple

Supposons un cache de 256K avec des blocs de 64 octets et une associativité de 4 blocs par ensemble (une ligne contient $64 * 4$ octets). Avec une adresse sur 32bits, le codage suivant est acceptable :

Déplacement (offset) est codé sur le logarithme en base 2 de $64 * 4 = 8\text{bits}$;

Ensemble (set) est codé sur le logarithme en base 2 de $(256 * 1024)/(64 * 4) = 10\text{bits}$;

Étiquette (tag) est codé sur $32 - 8 - 10 = 14\text{bits}$.

Exercice : supposons que l'adresse envoyée au cache soit 7FFFFBB8, alors l'offset vaut ? HEXA (? décimal), le set ? HEXA (? décimal) et l'étiquette ?. Pour rechercher la donnée, on pointe sur l'ensemble ? hexa, son contenu est comparé à ? ; s'il est égal c'est que la donnée est présente dans le cache alors on prend le ? ème octet. (remplacer tous les ? par leurs valeurs).

4.5.3 Ce que vous avez à faire

Vous avez à votre disposition un programme contenu dans un fichier et qui comprend des lignes de la forme suivante (ce fichier simule l'utilisation de données, pas celle d'instruction) :

- Ra : pour signifier que l'on veut lire en mémoire la valeur de la variable a (on n'en fait rien, on veut juste y accéder) ;
- Wa3 : pour signifier que l'on veut écrire la valeur 3 dans la variable a ;
- PaR : pour signifier que l'on protège l'accès à la variable a en lecture on ne peut plus la lire :
- PaW : pour signifier que l'on protège a de l'écriture ;
- PaRW : pour signifier que l'on ne peut ni lire ni écrire ;
- les trois écritures précédentes ont leurs réciproques notés respectivement AaR, AaW, AaRW dont les significations sont claires (autoriser l'accès en lecture seule, en écriture seule et en lecture/écriture pour la variable a ;

Note : l'alphabet utilisé pour les variables peut être réduit à une seule lettre parmi $[a \cdots z]$, les valeurs assignées aux variables sont les constantes numériques codés sur un seul digit $0, 1, \cdots, 9$.

Voici un exemple de programme (initialement on suppose que les variables sont accessibles en R/W) :

```
Wa3
Wb2
PaW
Ra
Wb4
Wa3
AaRW
Wa2
```

1- Construisez un simulateur de cache qui lit en entrée ce genre de programme (c'est vous qui avez fait le choix d'une politique de (rem)placement, d'identification en fonction de ce qui a été dit plus haut).

2- En fonction des programmes générés, calculer (ce qui est donné par le résultat de la simulation) le temps passé à la gestion de la mémoire. Pour cela on considèrera quatre coûts : celui de lire une donnée depuis le cache, celui de lire depuis à mémoire (pour un chargement dans le cache), celui d'une écriture dans le cache et enfin celui d'une écriture du cache vers la mémoire. Le programme affichera la somme des fifférents coûts.

4.5.4 Notre travail

L'implémentation que nous proposons pour le gestionnaire de cache ne concerne pour l'instant que l'organisation direct mapped. Nous faisons aussi l'hypothèse que l'adresse mémoire de la variable a est 0, celle de b est 1 etc et celle de z est 25. En fait nous ne gérons pas les tableaux mais uniquement les variables dites simples. Nous ne gérons pas non plus les autorisations d'accès de type A ou P mais seulement les instructions de lecture ou d'écriture (un R ou un W apparait en premier caractère sur une ligne du fichier d'entrée).

La gestion et la reconnaissance des symboles sur les lignes du fichier en entrée se fait grâce au module des expressions régulières de Python. La démarche est de définir un motif qui est compilé pour des raisons d'optimisation des performances (instruction Python `compile`) puis de demander la mise en correspondance de la ligne avec le motif (instruction Python `search`). Le motif est composé de trois zones permettant d'isoler l'opération (W, R, A ou P) puis le nom de la variable puis le reste qui est soit vide soit un digit soit soit un R soit un symbole RW. Nous pouvons récupérer chacune des trois zones par le biais de l'instruction Python `group`. C'est un fait le parenthésage dans l'expression du motif qui permet de déterminer des groupes.

Pour l'instant, le seul type de cache que nous gérons est le cache 'direct mapped' avec une seule données par ligne de cache. Cependant, le méthode de construction d'un cache comprend trois paramètre (le type du cache, le nombre de lignes et le nombre d'objets par ligne) si bien qu'elle permet de construire des caches avec plusieurs objets sur une ligne... mais on ne les gère pas.

Dans notre implémentation, un cache est vu comme une liste de tuples comme suit :

```
[
 (adresse,dirty-bit,[valeur,valeur,...]),
 (adresse,dirty-bit,[valeur,valeur,...]),
 ...
]
```

Chaque ligne a une adresse qui est donnée par la position dans la liste et qui ne doit pas être confondue avec le champs 'adresse' qui correspond à l'adresse d'un objet en mémoire, un dirty-bit et une liste de données. C'est la méthode `BuildCache` qui réalise la construction et d'initialisation d'un cache.

La classe `Cache` contient actuellement les deux méthodes de lecture et d'écriture dans le cache. Ces deux méthodes sont commentées dans le texte pour une bonne compréhension des comportements. Voici donc le texte du code Python :

```
#
# simulation d'un cache 'direct mapped' avec :
#   une seule donnée par ligne de cache
#   type de re-ecriture : Write Back
#   gestion du dirty bit
#
```

```
# Christophe Cérin
#
import fileinput, re, math, sys, random

class Cache:
     def __init__(self,montype,nbligne=4,nbobjdansligne=1):
          if ['MAPPED','FULLY','SET_ASSO'].__contains__(montype):
               self.type=montype
          else:
               raise 'erreur de type de cache :',montype,'non supporté'
          if ['MAPPED'].__contains__(montype):
               self.type=montype
          else:
               raise montype,'non supporté pour le moment'
          if  (type(nbligne)==int) and (type(nbobjdansligne)==int):
               if (nbligne >= 1) and (nbobjdansligne >= 1):
                    self.NB_LIGNE=nbligne
                    self.NB_OBJ_PAR_LIGNE=nbobjdansligne
               else:
                    raise 'erreur : les parametres de def du cache doivent être >= 1'
          else:
               raise 'erreur de type (on veut des entiers > 1)'
          # On définit ici les coûts d'acces a la mémoire
          self.ReadFromCache=1
          self.ReadFromMemory=6
          self.WriteToCache=1
          self.WriteToMemory=7
          # On peut commencer l'initialisation d'un cache 'vide'
          self.Cache=[]
          for i in range(0,nbligne):
                 inter=[]
                 for j in range(0,nbobjdansligne):
                    inter.append(0)
                 interone=(-1,0,inter)
                 self.Cache.append(interone)

     def ReadData(self,adresse):
         print 'ReadData call'
         # on commence par tenter de lire dans le cache
         # Attention : on ne gere que des lignes avec
         # un SEUL objet dans la ligne. De plus, la valeur
         # lue a l'adresse est rangée en prenant une valeur aléatoire
         MyAdr= adresse % self.NB_LIGNE
         if (self.Cache[MyAdr][0] == adresse):
            # on a trouvé l'objet dans le cache
            # car les adresses sont les memes
            return self.ReadFromCache
         else:
            # on ecrit la donnee (simulation de lecture
```

```
            # depuis la memoire) dans le cache parce que les
            # adresses de la donnée est celle dans le cache
            # ne correspondent pas. La donnée
            # ecrite dans le cache est prise par un tirage
            # aléatoire
            cycle=0
            if (self.Cache[MyAdr][1] == 1):
               # il faut vider le cache dans la memoire
               # si le dirty bit a été positionné
               cycle=cycle+self.WriteToMemory
            self.Cache[MyAdr]=(adresse,0,[random.randint(0,10000)])
            return self.ReadFromMemory+cycle

     def WriteData(self,adresse,data):
         MyAdr= adresse % self.NB_LIGNE
         print 'WriteData call'
         if (self.Cache[MyAdr][0] == adresse) or (self.Cache[MyAdr][0] == -1):
            # on a trouvé l'objet dans le cache
            # car les adresses sont les memes. On marque
            # le dirty-bit
            self.Cache[MyAdr]=(adresse,1,[data])
            return self.WriteToCache
         else:
            # on ecrit la donnee (simulation de lecture
            # depuis la memoire) dans le cache.
            cycle = 0
            if (self.Cache[MyAdr][1] == 1):
               # il faut vider le cache dans la memoire
               cycle=cycle+self.WriteToMemory
            self.Cache[MyAdr]=(adresse,0,[data])
            return self.WriteToCache+cycle

def adr(a):
    return ord(a)-97

#########################################################################
# programme principal et exemple
#########################################################################
def __main__():
# On verifie le nombre d'arguments
argc = len(sys.argv)
if argc != 2:
    print 'Usage: python cache.py file\n'
    sys.exit(0)

        # On cree un nouveau cache
        MyCache=Cache('MAPPED',4,1)
        print MyCache.Cache
        # Le nombre de cycle est initialisé à zero
```

```
        nb_cycle = 0
# Une ligne en entrée est constituée de 3 zones
# que l'on isole avec la notion de groupage
# dans l'expression rationnelle suivante :
pone = re.compile('([W|R|A|P])([a-z])([0-9]|W|RW)*')
for line in fileinput.input(str(sys.argv[1])):
    # print fileinput.lineno()
    m = pone.search(line)
    if m != None:
#print m.group(1),m.group(2),m.group(3)
                if (m.group(1) == 'R'):
                   nb_cycle=nb_cycle+MyCache.ReadData(adr(m.group(2)))
                elif (m.group(1) == 'W'):
                   nb_cycle=nb_cycle+MyCache.WriteData(adr(m.group(2)),int(m.group(3)))
                else:
                   print 'Instruction pas encore traitee'
                # print nb_cycle
    else:
print 'Probleme avec le fichier input (ligne)',fileinput.lineno()
fileinput.close()
print 'Le nombre de cycle(s) occupé(s) dans la gestion du cache est :',nb_cycle


__main__()

#######################################################
Le fichier d'exemple est :
Wa3
Wb2
PaW
Ra
Wb4
Wa3
AaRW
Wa2
Re
```

4.6 Plus grande sous suite croissante extraite d'une suite d'entiers

4.6.1 Dérivation de la solution

Ce problème fait partie du folklore et il est très utile pour montrer comment dériver une solution en maintenant un invariant. Il nous fait donc travailler sur l'induction. Notre contribution est de fournir un algorithme avec une complexité de stockage linéaire.

Soit S une séquence d'entiers distincts $\alpha_1, \alpha_2, \cdots, \alpha_n$. Une *sous suite croissante* de S est une sous suite $\alpha_{i_1}, \alpha_{i_2}, \cdots, \alpha_{i_k}$ avec $i_1 < i_2 < \cdots < i_k$ telle que pour tout $1 \leq j \leq k$ nous avons $\alpha_{i_j} < \alpha_{i_{j+1}}$. Une *plus grande sous suite croissante* de

S (pgssc) est une sous suite croissante de longueur maximale.

Problème : trouver la plus grande sous suite croissante d'une suite donnée et sa longueur.

Par exemple, si nous rayons tous les nombres sauf les 2 premiers, nous obtenons 9 44 qui est une sous suite croissante : 9 44 32 12 7 42 34 92 35 37 41 8 20 27 83 64 61 28 39 93 29 17 13 14 55 21 66 72 23 73 99 1 2 88 77 3 65 84 62 5 11 74 68 76 78 67 75 69 70 22 71 24 25 26.

Comme autre exemple, considérons la séquence $S = \{9, 44, 32, 12, 7, 42\}$. Ses sous suites croissantes sont $\{9, 32, 42\}$ et $\{9, 12, 42\}$ de longueur 3 ; $\{9, 44\}$, $\{9, 32\}$, $\{9, 12\}$, $\{9, 42\}$, $\{32, 42\}$, $\{12, 42\}$, $\{7, 42\}$ de longueur 2 et tous les singletons comme sous suites croissantes de longueur 1.

Pour cet exemple, les plus grandes sous suites croissantes de somme minimales sont $\{7\}$, $\{9, 12\}$, $\{9, 12, 42\}$. La longueur de la plus grande est donc 3. On notera qu'une sous suite croissante de longueur l n'est pas systématiquement le préfixe d'une sous suite de longueur $l+1$ (voir le contre exemple : $\{7\}$ and $\{9, 12\}$). Ce sera le cas si la suite de départ est déjà en ordre croissant. Dans le cas où la suite S de départ est en ordre décroissant, nous avons n sous suites croissantes de longueur 1 ; dans le cas où la suite de départ est en ordre croissant, nous avons $2^n - 1$ sous suites croissantes. Donc dans le cas général, le nombre N de sous suites croissantes est $n \leq N \leq 2^n - 1$.

Une première approche pour trouvr la solution est de générer puis fouiller dans toutes les combinaisons de k objets parmi n mais cela implique une complexité en temps en $\mathcal{O}(2^n)$ ce qui n'est pas raisonnable.

Pour améliorer la complexité de l'algorithme qui génére toutes les partitions, considérons d'abord les notations suivantes :

- la suite d'entrée est implanté dans le tableau S de taille n.
- $lis : integer \rightarrow integer$ est un prédicat qui donne la longueur de la plus longue sous suite croissante du tableau S entre les indices 1 jusqu'à l'indice passé en paramètre.
- $tablis$ est un tableau de taille n contenant à la position i, $1 \leq i \leq n$ une sous suite croissante de longueur i ;
- $l_{max} : integer$ sera la longueur de la plus grande sous suite croissante de S.

La propriété que l'on cherchera à maintenir à chaque iération est alors la suivante :

$$P : \forall i : 1 \leq i \leq n : l_{max} = lis(i)$$

et le corps du programme principal serait :

```
l_max := 0;
for i:=1 to n do begin
    maintenir P;
end;
return l_max;
```

Pour maintenir l'invariant P, on doit s'assurer que, étant donné une suite de taille inférieure strictement à i, on sait comment trouver toutes ses sous suites croissantes ($tablis[j]$ est connu pour tous $1 \leq j < i$).

Considérons maintenant le i^{eme} élément de S, x_i. Pour maintenir P, on doit comparer x_i avec tous les éléments terminaux des sous suites croissantes rangées dans $tablis$ et chercher si x_i peut étendre l'une d'entre elle. La tâche est facile à remplir mais elle introduit un nouveau problème qui est la recherche de **toutes** les sous suites croissantes.

Or, on peut remarquer qu'il est suffisant de garder les sous suites croissantes de taille $j : 1 \leq j < i$ qui terminent avec le plus petit élement car ce sont elles qui offrent le plus grand pouvoir «d'extension». On note par MP une telle sous suite croissante.

Finalement, nous dénotons par $tablis[l]$ la sous suite croissante de taille l qui termine avec le plus petit élément et $tablis[l].last$ son dernier élément. Nous modifions alors l'invariant de la façon suivante :

$$P : ((\forall i : 1 \leq i \leq n : l = lis(i) \wedge (\forall j : 1 \leq j \leq l : tablis[j] \; est \; connue))$$

Il nous reste à déterminer les valeurs de $tablis[j], 1 \leq j \leq l$ que l'on doit modifier. x_i étend un certain $tablis[j], 1 \leq j \leq l$ si et seulement si l'une ou l'autre des deux conditions suivantes arrive :

1. $x_i > tablis[j].last$, et ainsi x_i peut être concaténé à $tablis[j+1]$;
2. $x_i < tablis[j+1].last$, et ainsi $tablis[j]$ avec x_i au bout est meilleur que $tablis[j+1]$;

La notion de O notation est supposée connue dans cette section.

On notera enfin que la recherche de l'élément terminal plus petit que x_i qui permet de créer une nouvelle sous suite croissante peut s'effecter de manière dichotomique car les élements terminaux des différents $tablis$ sont ordonnés de manière croissante. Ainsi une recherche nous coûte $\log n$ opérations (dans le pire cas) ; et comme on répète n fois cette recherche, l'algorithme de recherche de la longueur de la plus grande sous suite croissante a une complexité en $\mathcal{O}(n \log(n)$[4]. L'emplacement mémoire nécessaire aux mémorisations des $tablis$ est en $\mathcal{O}(n^2)$, ce qui donne la complexité en place de l'algorithme.

Le lecteur notera que $tablis[j], (1 \leq j \leq l_{max} - 1)$ ne contiennent pas nécessairement les sous suites croissantes de somme minimale : c'est seulement le cas pour $tablis[l_{max}]$ (examiner l'exemple $S = 10, 15, 1, 20$).

[4]Si on lance une recherche linéraire, le pire cas est alors $\mathcal{O}(n^2)$

Expliquez les résultats suivants lorsqu'on déroule l'algorithme pour la séquence $\{10, 15, 20, 14, 9, 25\}$

Input : $\{10, 15, 20, 14, 9, 25\}$

Step	$lmax$	*tablis*
1 :	1	$\{10\}$
2 :	2	$\{10\}$, $\{10, 15\}$
3 :	3	$\{10\}$, $\{10, 15\}$, $\{10, 15, 20\}$
4 :	3	$\{10\}$, $\{10, 14\}$, $\{10, 15, 20\}$
5 :	3	$\{9\}$, $\{10, 14\}$, $\{10, 15, 20\}$
6 :	4	$\{9\}$, $\{10, 14\}$, $\{10, 15, 20\}$, $\{10, 15, 20, 25\}$

Concluez que le résultat est $\{10, 15, 20, 25\}$. Le paragraphe qui suit explique comment compresser l'information requise pour coder les différents MP

4.6.2 Une capacité de stokage linéaire

Dans la suite, le dernier élément d'un $tablis[i]$ est dénoté $Last(i)$. On rappelle que l'on a défini $lmax$ comme la longueur de la plus grande sous suite croissante courante.

Supposez que l'élement x_i écrase la plus longue sous suite croissante de longueur l (on a déjà visité $x_1, \cdots, x_{i-1}$). Au lieu de copier explicitement $MP(l-1)$ comme nouveau prefixe de la nouvelle $MP(l)$, on garde seulement un pointeur sur $MP(l-1)$. Le contenu de ce pointeur est la valeur $Last(l-1)$. Le couple $(x_i, Last(l-1))$ est formé et rangé dans un tableau *Tab-MP* à la position i. L'algorithme est maintenant le suivant :

1- Initialization : $Last(i) := \inf$;
2- step(1) : $lmax := 1$; $Last(1) := x_1$;
$Tab\text{-}MP(1) := (x_1, Nil)$;
3- step(i), $2 \leq i \leq n$:
4- $l :=$ le plus petit l tel que $Last(l) > x_i$;
5- $l > lmax \Rightarrow lmax := lmax + 1$;
6- $Last(l) := x_i$; $Tab\text{-}MP(i) := (x_i, Last(l-1))$;

Expliquez pourquoi la complexité en temps est toujours en $\mathcal{O}(n \log n)$ et pourquoi la complexité en place est maintenant devenue linéaire. Vous pouvez travailler à partir de l'exemple suivant :

Input : $\{10, 15, 20, 14, 9, 25\}$

Step	*Last*	*Tab-MP*
1 :	10	$(10, Nil)$
2 :	10, 15	$(10, Nil); (15, 10)$
3 :	10, 15, 20	$(10, Nil); (15, 10); (20, 15)$
4 :	10, 14, 20	$(10, Nil); (15, 10); (20, 15); (14, 10)$
5 :	9, 14, 20	$(10, Nil); (15, 10); (20, 15); (14, 10);$ $(9, Nil)$

6 : 9, 14, 20, 25 (10, Nil); (15, 10); (20, 15); (14, 10); (9, Nil); (25, 20)

4.7 Justification d'un texte

nous proposons maintenant de mettre en place une spécification consistant à justifier les lignes d'un texte en insérant des blancs entre les mots de telle façon que le dernier mot de chaque ligne termine dans la dernière colonne de la ligne de plus grande longueur. Par exemple, si l'on utilise le symbole # pour rendre compte des blancs, les trois lignes où les mots sont initialement separés par un seul blanc :

```
il#justifie#des#lignes#en
inserant#des#blancs#supplementaires
est#une#tache#d'un#editeur#de#textes
```

doivent être transformées en :

```
il###justifie####des####lignes####en
inserant##des#blancs#supplementaires
est#une#tache#d'un#editeur#de#textes
```

De sévères restrictions sont imposées quant à l'insertion des blancs ce qui va nous amener à introduire des variables qui devront vérifier les contraintes, en particulier à la fin du programme.

Dans le premier paragraphe nous rappelerons les définitions utiles au problème et nous traitons un exemple afin de faire apparaître les difficultés principales du problème. Le deuxième paragraphe sera consacré au calcul du nombre de caractères blancs à insérer de part et d'autre d'un mot «frontière»précédemment défini selon que l'on traite une ligne paire ou impaire. Le troisième paragraphe propose le codage en Python.

Ce problème est introduit dans l'ouvrage de D. Gries – *The Science of Programming*.

4.7.1 Définitions, exemples

Considérons que pour chaque ligne à n mots, le tableau `b[1..n]` contient les positions de début de mot. Par exemple, pour la première ligne non justifiée de notre exemple, on a `b[1]=1`, `b[2]=4`, `b[3]=13`. Initialement, une ligne d'entrée a la forme :

$$W1[1]W2[1]\cdots[1]Wn[s]$$

avec $W1[1]$ est le premier mot suivi d'un blanc, $W2[1]$ le deuxième mot suivi également d'un blanc et $Wn[s]$ le dernier mot suivi de s blancs.

Remarque : pour calculer la valeur de s de chaque ligne, il suffit de soustraire le nombre de caractères de la ligne de plus grande longueur du nombre de caractères de la ligne courante.

L'écriture de la première ligne de l'exemple précédent dans ce formalisme est $W1[1]W2[1]W3[1]W4[1]W5[1]$ avec $W1 = il$, $W2 = justifie$, $W3 = des$, $W4 = lignes$, $W5 = en$ et le tableau b défini par :

```
b[1] = 1 ; b[2] = 4 ; b[3] = 13 ; b[4] = 17 ; b[5] = 24
```

La forme de la ligne de sortie dans ce formalisme est (désignation des 3 variables entières) :

$$W1[p+1]\cdots[p+1]W_t[q+1]\cdots[q+1]Wn$$

Remarque : dans le cas où l'on a qu'un seul mot, $W1/Wn$ seront égaux à ϵ (mot vide) selon que l'on se trouve sur une ligne paire ou impaire.

On doit encore imposer des restrictions sur les variables p, q, t afin de rendre compte des exigences figurant dans l'énoncé du problème. p et q sont supérieurs ou égaux à 0 et leurs valeurs respectives doivent différer que d'une unité. Le nombre total de blancs insérés doit être égal à s et leurs positions dépend selon que l'on traite une ligne paire ou impaire. w_t est l'un des mots $W_1 \cdots W_n$.

Ainsi la précondition de la spécification est :

$Q : 0 \leq s \wedge 0 \leq n \wedge b[1 \cdots n]$ est connu au départ $\wedge$ 1 blanc entre chaque mot + s blancs

Les restrictions sont formulées par :

Q1 :	$1 \leq t \leq n \wedge$	(Wt est un des mots)
	$0 \leq p \wedge 0 \leq q \wedge$	(on ne réduit pas le nb de blancs)
	$p * (t-1) + q * (n-t) = s \wedge$	(insertion des blancs)
	$(odd(z) \wedge q = p+1 \vee even(z) \wedge p = q+1)$	(restriction ligne paire-impaire)

La postcondition de la spécification s'écrit alors :

$$R : (\forall i : 1 \leq i \leq t : b[i] := b[i] + p * (i-1)) \wedge$$
$$(\forall j : t < j \leq n : b[j] := b[j] + p * (t-1) + q(j-t))$$

On vérifie que les variables p, q, t dans R vérifient les contraintes exprimées dans la formule Q1.

Nous avons désigné une nouvelle variable, z qui est le numéro de la ligne et qui sera du type `entier`.

4.7.2 Calcul de p, q et t suivant que z soit pair ou impair

Il nous faut maintenant calculer p, q, t pour établir la proposition Q1. Examinons par raffinements successifs et par décomposition par cas ce qui peut se passer pour les valeurs de ces variables.

Nous nous intéressons pour l'instant qu'au cas où les lignes portent un numéro impair (le problème est réduit à un problème plus simple – on espère que cette étude va mettre en lumière des solutions que l'on pourra réutiliser dans le cas des lignes de numéro pair). On a donc dans ce cas, $q = p + 1$. On veut donner des valeurs à p, q, t pour que Q1 soit vérifiée. En substituant q par $p + 1$ on obtient :

$$1 \leq t \leq n \wedge 0 \leq p \wedge 0 \leq p+1 \wedge p(t-1)+(p+1)(n-t) = s$$

qui se réécrit en :

$$1 \leq t \leq n \wedge 0 \leq p \wedge p*(n-1)+n-t = s$$

Une solution évidente est $p = \lfloor s/(n-1) \rfloor$ et $n-t = s \mod (n-1)$. Et on remarque immédiatement que la division par $n-1$ n'est pas possible si $n = 1$ c'est à dire s'il n'y a qu'un seul mot sur la ligne ! Donc la spécification est inconsistante et il faudrait exprimer ce cas de base. Remarquer que l'on pourra décider de ne rien faire lorsque $n = 1$ si z est pair ou impair.

De même, un deuxième cas de base apparait lorsque $n = 0$ (ligne blanche) car on obtient une division par -1 ce qui n'a pas de sens. Il faut transformer de nouveau la spécification pour rendre aussi compte de ce cas.

Finalement, dans le cas où z est impair et $n > 1$, p, q, t sont calculées par :

$p := \lfloor s/(n-1) \rfloor$
$t := n-(s \mod (n-1))$
$q := p+1$

On vérifie aisément que ces valeurs permettent de vérifier la proposition Q1. Le cas où z est impair est résolu de la même manière ce qui nous conduit au calcul général suivant $(n > 1)$:

```
# attention, n>1 uniquement
# le symbole % est l'opérateur Python pour le modulo
if z % 2 == 0:
     # Le numéro de ligne est pair
     q = s/(n-1)
     t = 1+(s % (n-1));
     p = q+1

else:
     # Le numéro de ligne est impair
     p = s/(n-1)
     t = n -(s % (n-1))
     q = p+1
```

Mise à jour du tableau b : la nature de la proposition R nous suggère d'utiliser deux boucles pour modifier le tableau b. La première permet la mise à jour pour les cases d'indice $1 \cdots t$, la deuxième la mise à jour des cases d'indices $t+1 \cdots n$. On remarque enfin que si, dans le cas d'une ligne impaire $p = 0$ alors il n'est pas utile de rentrer dans la boucle qui va changer les valeurs de $b[1], \cdots, b[t]$.

En pseudo langage algorithmique, nous obtenons :

```
\begin{verbatim}
/*
  attention : n >1 et les valeurs de p,q,t sont connues
```

```
  et les cases utiles du tableau sont celles indiees de
  1 a n (case 0 inutilisee).
*/
e=0;
if(p>0) {
  for(i=2;i<=t;i++){ /* pas de changement pour le premier mot */
     e = e + p;
     b[i] = b[i] + e;/* qui reste a sa place                  */
  };
}
for(i=t+1;i<=n;i++) {
   e = e + q;
   b[i] = b[i] + e;
}
```

On prend du recul sur sa programmation et on suggère des pistes d'amélioration.

Finalement, pour le cas des lignes impaires nous proposons le code Python suivant ainsi qu'un exemple. Dans ce code nous utilisons les facilités de Python pour manipuler les chaînes de caractères (fonctions split, strip, len) ainsi que le mécanismes de décomposition par tranches car il n'y a pas d'interface pour copier du texte dans un chaîne en remplacement d'un texte existant.

Enfin, dans ce code, la reconstruction du résultat une fois le tableau b connu aurait pu se programmer au niveau du calcul des valeurs b[i] (puisqu'a ce moment toutes les informations nécessaires sont connues) plutôt que de le sortir dans une étape intermédiaire supplémentaire.

```
import string

# Le texte a modifier. Notez qu'il termine par 9 blancs
# et qu'il comporte 5 mots.
texteAVANT='aaa bb c dddd eee         '

# Numero de la ligne
z=1
# Calcul du nombre de mots dans la ligne
texte=string.strip(texteAVANT)
n=len(string.split(texte,' '))
# print string.split(texte,' ')
# Calcul du nombre de blancs en fin de ligne
s=string.count(texteAVANT,' ') - n + 1
# Initialisation de la liste contenant
# les indices de début des mots
b=[0]
ltexte=string.split(texte,' ')
long=len(ltexte[0])+1
for i in range(1,n):
     b.append(long)
     long=long+len(ltexte[i])+1
#b=[0,4,7,9,14]
# Calcul de p,q,t
```

```
if z % 2 == 0:
     # Le numéro de ligne est pair
     q = s/(n-1)
     t = 1+(s % (n-1));
     p = q+1
else:
     # Le numéro de ligne est impair
     p = s/(n-1)
     t = n - (s % (n-1))
     q = p+1
     print p,q,t
# modification de la liste de debut des mots
e=0
if p>0:
   for i in range(1,t):# pas de changement pour le premier mot
     e = e + p         # qui reste a sa place
     b = b[0:i]+[b[i] + e]+b[i+1:]
for i in range(t,n):
   e = e + q
   b = b[0:i] + [b[i] + e] + b[i+1:]
print b
# Il reste a construire la nouvelle chaine
# On commence par construire une chaine de blancs
texteRESUL=''
for i in range(0,len(texteAVANT)):
     texteRESUL=texteRESUL+' '
mylong = len(ltexte)
for i in range(0,mylong):
     # Inserer ltexte[i] en position b[i] dans texteRESUL
     #print texteRESUL[0:b[i]],ltexte[i],texteRESUL[b[i+1]:]
     texteRESUL=texteRESUL[0:b[i]]+ltexte[i]
     # Il reste a concatener b[i] - len(texRESUL) blancs
     if i< mylong-1:
          myblancs = ''
          #print b[i+1]-len(texteRESUL),b[i+1],len(texteRESUL)
          for j in range (0,b[i+1]-len(texteRESUL)):
                    myblancs = myblancs + ' '
          texteRESUL = texteRESUL + myblancs
print 'Avant    : |',texteAVANT,'|'
print 'Resultat : |',texteRESUL,'|'

Exécution du programme :
ordinateur-de-christophe-cerin-2:~$ python e.py
2 3 4
[0, 6, 11, 15, 23]
Avant    : | aaa bb c dddd eee          |
Resultat : | aaa   bb   c   dddd    eee |
```

Il reste à greffer dans le code précédent le cas des lignes paires.

4.8 Spécification des naturels

L'objectif dans cette étude est de faire prendre conscience que c'est parce qu'on définit les objets d'une certaine manière qu'on obtient telle ou telle propriété pour ces objets. Considérez donc maintenant la spécification des entiers naturels suivante :

```
SORTES : Nat
UTILISE : Bool
OPERATIONS :
    0 : -> Nat
    1 : -> Nat
    2 : -> Nat
    3 : -> Nat
    4 : -> Nat
    5 : -> Nat
    6 : -> Nat
    7 : -> Nat
    8 : -> Nat
    9 : -> Nat
    s _ : Nat -> Nat
     _ + _ : Nat Nat -> Nat
     _ * _ : Nat Nat -> Nat
     p _ : Nat -> Nat
     _ - _ : Nat Nat -> Nat
     _ < _ : Nat Nat -> Bool
    fact _ : Nat -> Nat
    fib _ : Nat -> Nat
    _ div _ : Nat Nat -> Nat
    _ mod _ : Nat Nat -> Nat
VARIABLES :
    x, y : Nat
AXIOMES :
    1 = s 0
    2 = s s 0
    3 = s s s 0
    4 = s s s s 0
    5 = s s s s s 0
    6 = s s s s s s 0
    7 = s s s s s s s 0
    8 = s s s s s s s s 0
    9 = s s s s s s s s s 0

    0 + y = y
    x + 0 = x
```

```
(s x) + y = s (x + y)

0 * y = 0
x * 0 = 0
(s x) * y = y + (x * y)

p(0) =  0
p(s(x)) = x

x - 0 = x
x - s y = p (x - y)

x < 0 = false
0 < (s y) = true
(s x) < (s y) =  x < y

fact(0) = 1
fact(s(x)) = (s x) * fact x

fib(0) = 0
fib(s(0)) = s(0)
fib(s(s(x))) = (fib x) + fib(s(x))

x < s y = true => x div s y = 0
x < s y = false => x div s y = s ((x - s y) div s y)

x < s y = true => x mod s y = x
x < s y = false => x mod s y = (x - s y) mod s y
```

Interprétons les opérations et les axiomes. Après la définition des 10 constantes, on trouve dans l'ordre, l'opération succ (s), l'addition, la multiplication, l'opération précédent, la soustraction, l'opérateur inférieur strictement, la fonction factorielle, fibonacci, division entière et modulo.

Le naturel 1 est défini comme le successeur de 0, le naturel 2 comme le successeur du successeur de 0...

L'axiome de l'addition `(s x) + y = s (x + y)` provoque des réduction «gauche»(opérande gauche de +). Ainsi le terme 2+3 est transformé en `s(1+3)` lui même transformé en `s(s(0+3))` lui même transformé, au moyen du premier axiome en `s(s(3))` qui vaut, par définition 5. Pour terminer, on aurait pu choisir aussi l'axiome `x + s(y) = s (x + y)` qui aurait provoqué des réductions à «droite».

On peut montrer un certain nombre de propriétés de l'addition ainsi définie. Par exemple, il est facile de monter que `0+y=y+0`. Preuve : `0+y=y=y+0` par application des deux premiers axiomes de l'addition.

On peut aussi monter que 2+1=1+2. Preuve :

$$\begin{aligned} 2+1 &= s(1)+1 \\ &= s(s(0)+1) \\ &= s(s(0+1)) \\ &= s(s(1+0)) \\ &= s(s(1)+0) \\ &= s(0+s(1)) \\ &= s(0)+s(1) \\ &= 1+2 \end{aligned}$$

On peut aussi monter, à partir du résultat précédent que $x+1=1+x$, puis que l'addition est commutative : $x+y=y+x$. Preuve : Soit $y=s^{y-1}(1)$. Comme s est une fonction monotone croissante, nous pouvons dire que :

$$s^{y-1}(x+1)=s^{y-1}(1+x) \Longleftrightarrow x+s^{y-1}(1)=s^{y-1}(1)+x \Leftrightarrow x+y=y+x$$

ce qui termine la preuve.

Compléments pour la preuve : (première partie)

$$\begin{aligned} x+1 &= s^{x}(0)+1 \\ &= s(s^{x-1}(0)+1) \\ &= s(s^{x-1}(0+1)) \\ &= s(s^{x-1}(1+0)) \\ &= s(x+0) \\ &= s(0+x) \\ &= s(0)+x \\ &= 1+x \end{aligned}$$

Deuxième partie : comme y a été pris égal à $s^{y-1}(1)$, on a par voie de conséquence que $s^{y-1}(1)=1+y-1 \Rightarrow s^{y-1}(x)=x+y-1$. De là on peut dériver $s^{y-1}(x+1)=s^{y-1}(x)+1=x+y-1+1=x+y$ et aussi que $s^{y-1}(1+x)=s^{y-1}(1)+x=y+x$.

L'axiome le plus intéressant pour la multiplication est `(s x) * y = y + (x * y)`. L'évaluation du terme 2*3 conduit aux réductions suivantes :

$$\begin{aligned} 2*3 &\Leftrightarrow s(1)*3=3+(1*3) \\ &\Leftrightarrow 3+(s(0)*3)=3+(3+(0*3)) \\ &\Leftrightarrow 3+(3+0)\ (axiome\ 1\ multiplication) \end{aligned}$$

En appliquant maintenant les axiomes pour l'addition on déduit facilement le résultat.

La définition de l'opérateur précédent nous dit que le prédécessur de 0 c'est 0 (ceci pour rester dans les naturels) et que le précédent du successeur c'est le nombre en question.

Comme conséquence des définitions nous avons que `s(p(x))=x`. Preuve : soit `y=s x`. On a `s(p(y))=s(p(s x))=s(x)=y`.

La soustraction s'effectue par des réductions "droite" et au moyen de l'opération prédecesseur. Ainsi le terme $3-2$ s'évalue :

$$\begin{aligned} 3-2 &\Leftrightarrow 3-s(1)=p(3-1) \\ &\Leftrightarrow p(3-1)=p(p(3-0) \\ &\Leftrightarrow p(p(3))=p(p(s(2)))=p(2)=p(s(1))=1 \end{aligned}$$

On notera enfin, qu'avec cette définition l'évaluation de $2-3$ conduit à la valeur 0.

L'opération inférieure est définie classiquement. Idem pour factorielle et fibonacci.

Les axiomes de l'opération de division entière suivants :

```
x < s y = true => x div s y = 0
x < s y = false => x div s y = s ((x - s y) div s y)
```

nous apprennent que dans le cas ou x est strictement plus petit que le successeur de y alors la division entière de x par le successeur de y c'est 0. Sinon c'est le successeur du résultat de la division entière de $(x-sy)$ et du successeur de y. Prenons un exemple pour être convaincu totalement :

4 div 2 $\Leftrightarrow$ $4<2=$ false donc 4 div s 1 = s((4 - s 1) div s 1)
$\Leftrightarrow$ s(2 div 2)
$\Leftrightarrow$ s(s((2 - s(1)) div s(1)))
$\Leftrightarrow$ s(s(0 div 2))
$\Leftrightarrow$ s(s(0))
$\Leftrightarrow$ 2

Attention, j'ai choisi une priorité d'évaluation plus grande de l'opération - vis à vis de div et j'ai d'abord choisi de réduire systématiquement l'opérande "gauche" de div jusqu'à obtenir un terme clos (une constante).

Une implantation en langage C est alors la suivante :

```
Nat dvi(Nat x,y)
{
  /* x,y doivent etre <> de 0 ensmebles */
  return( x<y ? 0 : s(div(x-y,y)));
}

Nat s(Nat x)
{
  return x+1;
}
```

Faites "fonctionner" les réductions (gauche – droite ?) pour l'opération `4 mod 2` avec les axiomes suivants pour le modulo :

```
x < s y = true => x mod s y = x
x < s y = false => x mod s y = (x - s y) mod s y
```

```
Nat mod(Nat x,y)
{
  return( x<y ? x : mod(x−y,y));
}
```

4.9 Le type multi-ensemble en Python

DANS LE CHAPÎTRE précédent nous avons exposé la spécification abstraite du type algébrique multi-ensemble. Nous proposons ici une implémentation Python. À titre d'exercice, le lecteur est invité à dire si cette implémentation suit la spécification et ce qu'elle implémente en plus ou en moins par rapport à ce qui est spécifié dans le précédent chapître. Autrement dit, il faut donner les différences de points de vue entre la spécification et l'implémentation. En fait, dans le texte qui suit il y a deux implémentations de la notion de multi-ensemble : une avec des listes, l'autre avec des dictionnaires. Laquelle vous parait la plus pertinente ? À vous de jouer maintenant.

Solution numéro 1 avec des listes :

```
#########################################################################
#
# Classe multiensemble implémentée avec des listes de tuples :
# la première composante du tuple est l'objet que l'on range et la
# deuxième composante donne le nombre d'occurences.
#
# Note : les fonctions add et remove (par exemple)sont en O(n)
# c.à.d linéaire selon le nombre n d'objet car on
# implemente un multiensemble par une liste (et pas un dictionnaire)
#
# Version 1.3 au 6 janvier 2003.
#
# Teste' avec Python 2.2.2. Il semblerait qu'il y ait des problemes
# avec des versions anterieures
#
#########################################################################

class multiensemble:
        def __init__(self,bool=0):
                if bool != 1:
                        x=int(raw_input("donne le nb d'elements differents :"))
                        self.L = list()
                        for y in range(x):
                                a=raw_input("donne l'element :")
                                b=int(raw_input("donne le nb d'occ element:"))
                                self.L.append((a,b))
                else:
                        self.L = list()
```

```
        def afficher(self):
                print("[")
                for (a,b) in self.L:
                        print('('+a+','+str(b)+')')
                print("]")


        def search(self,L,a):
                ind = 0
                lg = len(L)
                while(ind != lg):
                        if(L[ind][0] == a):
                                return ind
                        else:
                                ind += 1
                return -1

        def add(self,a):
                indice = self.search(self.L,a)
                #print 'indice=',indice
                if indice != -1:
                        self.L = self.L[:indice] + [(a,self.L[indice][1] +
                                    1)] + self.L[indice+1:]
                else:
                        self.L.append((a,1))

        def remove(self,a):
                indice = self.search(self.L,a)
                #print 'indice=',indice
                if indice != -1:
                        if self.L[indice][1] == 1:
                                del self.L[indice]
                        else:
                                self.L = self.L[:indice] +
                     [(a,self.L[indice][1] - 1)] + self.L[indice+1:]

##########################################################################
#
# La classe Bag hérite des methodes de la classe multiensemble
# et on ajoute une methode permettant de faire l'union de deux Bags
#
##########################################################################

class Bag(multiensemble):
        def addbis(self,a,number):
                indice = self.search(self.L,a)
                #print 'indice=',indice
                if indice != -1:
                        self.L = self.L[:indice] + [(a,self.L[indice][1] +
```

```
number)] + self.L[indice+1:]
                else:
                        self.L.append((a,number))
                        # ERROR if you code according to the next line
                        # Explain why!
                        # self.L.append((a,1))

        def union(self,a):
                # print a.L
                for (first,second) in a.L:
                        self.addbis(first,second)

        def diff(self,b):
                """ les elements de self qui ne sont pas dans b, union les
                        elements de b qui ne sont pas dans self """
                c=Bag(1)
                for (elem,occ) in self.L:
                        indice = self.search(b.L,elem)
                        if indice == -1:
                                c.L.append((elem,occ))
                for (elem,occ) in b.L:
                        indice = self.search(self.L,elem)
                        if indice == -1:
                                c.L.append((elem,occ))
                return c

        def intersection(self,b):
                """ on garde les elements communs de self et de b
                        (on somme les occurrences) """
                c=Bag(1)
                for (elem,occ) in self.L:
                        indice = self.search(b.L,elem)
                        if indice != -1:
                                c.L.append((elem,occ + b.L[indice][1]))
                return c


#########################################################################
#
# Programme principal
#
#########################################################################

l=Bag()
l.afficher()
l.add('aaa')
l.add('ccc')
l.add('aaa')
l.add('ccc')
```

```
l.add('aaa')
l.add('aaa')
l.afficher()
l.remove('aaa')
l.remove('ccc')
l.afficher()
l.remove('ccc')
l.afficher()

m=Bag()
print('Multiensemble l:')
l.afficher()
print('Multiensemble m:')
m.afficher()
l.union(m)
print('Multiensemble l union m (range dans l):')
l.afficher()

d=l.diff(m)
print('Multiensemble l diff m (range dans d):')
d.afficher()

e=l.intersection(m)
print('Multiensemble l intersection avec m (range dans e):')
e.afficher()
```

Solution numéro 2 avec des dictionnaires :

```
class Bag:
    """A Bag is a collection of items with no particular ordering,
    containing duplicates if any."""

    default_to_zero = 1      #class attribute
                             #used to turn off error handler

    def __init__(self, *args):
        self._dict = {}
        for arg in args:
            self.add(arg)

    def __repr__(self):
        import string
        l=map(lambda x,y: str(x)+'='+str(y),
                                 self._dict.keys(),self._dict.values())
        return "%s(%s)" % (self.__class__.__name__,
                                  string.join(map(lambda x: x, l), ', '))
#        return "%s(%s)" % (self.__class__.__name__,
#                           string.join(map(repr, self._dict.keys()), ', '))
```

```
    def extend(self, args):
        """Add several items at once."""
        for arg in args:
            self.add(arg)

    def add(self, item):
        """Add one item to the set. Test the occurrence number"""
        D = self._dict     #D is local to this function
                           #but points to SAME object as self.D
        try:
                D[item] = D[item]+1  #try adding 1 to the key count
        except KeyError:
                D[item] = 1          #insert new key into self.D

#        self._dict[item] = {'occ':0}
#       self._dict.__setkey__(item,1)

    def remove(self, item):
        """Remove an item from the set."""
        del self._dict[item]

    def contains(self, item):
        """Check whether the set contains a certain item."""
        return self._dict.has_key(item)

    # Higher performance member-test for python 2.0 and above
    __contains__ = contains

    def __getitem__(self, index):
        """Support the 'for item in set:' protocol."""
        return self._dict.keys()[index]

    def __len__(self):
        """Return the number of items in the set."""
        return len(self._dict)

    def items(self):
        """Return a list containing all items."""
        return self._dict.keys()

    def values(self):
        """Return a list containing all values associated to keys."""
        return self._dict.values()

    def howmany(self,key):
        """Return the number of keys 'key'"""
        try:
                return self._dict[key]
        except KeyError:
```

```
            if self.default_to_zero:
                        return 0
            else:
                   raise KeyError,"no such element"

# End class Bag()

Mybag = Bag()
Mybag.extend("abcdefcdf")
print Mybag.__repr__()
Mybag.remove("c")
print Mybag.__repr__()
print Mybag.items()
print Mybag.howmany("b")
print Mybag.contains("c")


#
# Exécution du programme
#
# [christophe@m183 christophe]$ python bag.py
# Bag(a=1, c=2, b=1, e=1, d=2, f=2)
# Bag(a=1, b=1, e=1, d=2, f=2)
# ['a', 'b', 'e', 'd', 'f']
# 1
# 0
# ====> la methode de suppression est à revoir car elle supprime
# toutes les occurences (la clé comprise)
#
```

4.10 Un problème de parenthèses

Les mots bien appariés se définissent ainsi. Vous considérez d'abord les nombres de 1 à $2n$ ou encore $2n$ points alignés dans le plan, et on veut que tous ces points soient reliés entre eux deux à deux, sans que les lignes ne se croisent.

Jusqu'ici, c'est la même chose que les mots bien parenthésés. Par exemple `(()())` code $1,2,3,4,5,6$

Mais on veut *de plus* que chaque nombre pair soit relié à un nombre impair, et que chaque nombre impair soit relié à un nombre pair. Nous avons donc besoin de deux jeux de parenthèses et [].

Ainsi {}[] est OK, mais pas {[}] (car 1 est relié à 3 dans ce dernier exemple).

{[[]{]]} est OK, mais pas {[]{]}, etc.

Avoir un algorithme rapide pour compter ces mots aiderait a comprendre une formule profonde de la physique (la formule KPZ). On vous demande donc de dériver un programme qui compte le nombre de mots bien appariés à $2n$ lettres, pour n allant de 1 à 40. Jusqu'à 22 les solutions sont :

```
1: 2
2: 8
3: 40
4: 228
5: 1424
6: 9520
7: 67064
8: 492292
9: 3735112
10: 29114128
11: 232077344
12: 1885195276
13: 15562235264
14: 130263211680
15: 1103650297320
16: 9450760284100
17: 81696139565864
18: 712188311673280
19: 6255662512111248
20: 55324571848957688
21: 492328039660580784
22: 4406003100524940624
```

Un code Python qui implémente cette solution est alors la suivante :

```
import time

class X:
    a=1
    b=1
    n=1

    def __init__(self):
        self.L = [(1,1,1)]
        #      (a,b,nb)

    def mycmp(self,x1,x2):
     print x1," - ", x2
         if (x1[0] != x2[0]):
          return x1[0] < x2[0]
     else:
          return x1[1] < x2[1]

    def update(self,k):
        temp=X()
     temp.L=[]
        res=X()
     res.L=[]
        for p in self.L:
```

```
            temp.L.append((2 * p[0] + (k & 1), p[1],p[2]))
         temp.L.append((p[0], 2 * p[1] + (k & 1),p[2]))
            if ((p[0] > 1) and ((p[0] & 1) != (k & 1))):
              temp.L.append ((p[0] / 2, p[1], p[2]));
            if ((p[1] > 1) and ((p[1] & 1) != (k & 1))):
              temp.L.append ((p[0], p[1] / 2, p[2]));
     #print temp.L
     #temp.L.sort (self.mycmp);
     temp.L.sort ();
     print "Triee ", temp.L
       acc=(0,0,0);
     #print temp.L
     t3=0
     for p in temp.L:
          #print p[0],acc[0],p[1],acc[1]
          if ((p[0]==acc[0])and(p[1]==acc[1])):
               lg=len(res.L)
               lg1=lg-1
               t1=time.time()
               inter=(res.L[lg1][0],res.L[lg1][1],res.L[lg1][2] + p[2])
               #res.L=res.L[:lg1]+inter
               res.L.__setitem__(lg1, inter)
               t2=time.time()
               t3 = t3+(t2-t1)
               #print res.L
               #print "-+-+-+-+-+-+-+-+-+-+-+-"
          else:
               acc=p
               res.L.append(acc)
               #print res.L
               #print "-----------"
     print "Time: ",t3
     print "#triplets sur la pile: ",len(res.L)
     return res;

run=X()
for k in range(1,4):
     run = run.update(k);
     s = 0;
     for p in run.L:
          print p[0],": ",p[1],": ",p[2],": "
          s = s+(p[2] * p[2])
     print k,": ",s
```

5 Travailler à son projet professionnel

Sommaire

5.1 Motivations

CE CHAPITRE aborde les questions de l'expérience professionnelle une fois sorti de l'université. Il invite en particulier les jeunes (futurs) diplômés à réfléchir à un projet, à un métier en informatique. Le chapitre aborde non seulement des notions ou des savoirs faire pratiques (par exemple la réalisation d'un entretien) mais renvoie aussi à des considérations plus générales comme par exemple l'organisation du champ professionnel, l'articulation entre diplômes et certifications. La façon dont les acquis universitaires peuvent être validés en terme de compétences cibles de l'emploi visé est une question fondamentale.

Les objectifs sont maintenant de découvrir, dans un contexte plus large que la salle de cours, les exigences de la profession, les attentes, les discours ambients. Par exemple, les professionnels ont à faire face à des problèmes d'organisation du travail et des hommes, et concernant les logiciels de savoir s'ils sont performants, simples à utiliser, iteropérables (le format des données et les modèles de communication doivent suivre les besoins de l'utilisateur), faciles à mettre à jour, sécurisés et fiables, déployables...

Nous abordons la Profession par le biais du Programme Pédagogique National (PPN) des DUT de la spécialité «Informatique»qui introduit un module intitulé

«Le Projet Personnel et Professionnel (PPP)». Le PPN précise *qu'il sagit d'un travail de fond qui doit permettre à l'étudiant de se faire une idée précise des métiers de la spécialité «Informatique»et de ce quils nécessitent comme aptitudes personnelles. Il doit amener l'étudiant à mettre en adéquation ses souhaits professionnels immédiats et futurs, ses aspirations personnelles et ses capacités afin de concevoir un parcours de formation cohérent avec le ou les métiers choisis et à devenir acteur de son orientation.*

Les objectifs affichés dans le PPN pour le module PPP sont :

1. Fournir à l'étudiant l'opportunité de se faire une idée plus précise des métiers de linformatique ;
2. Préciser le projet personnel de l'étudiant en terme de métier ;
3. Familiariser l'étudiant avec la recherche documentaire ;
4. Familiariser l'étudiant avec le monde de l'emploi ;
5. Familiariser l'étudiant avec la gestion du temps.

Les paragraphes qui suivent constituent une déclinaison du PPP telle qu'elle a été mise en place pour le DUT informatique de l'IUT de Villetaneuse. Chaque paragraphe peut être vu comme une fiche de travail pour étudiants. Les paragraphes peuvent être indépendants les uns des autres, par exemple les paragraphes «Introduction à l'informatique et à ses métiers»et «Veille technologique»ou alors peuvent s'enchaîner comme les paragraphes sur la découverte d'un métier, l'entrevue d'un professionnel et la restitution de l'entrevue.

Il s'agit d'amener les étudiants à parler d'eux-mêmes, et la difficulté est de mener un travail d'introspection, forcément individuel, en TD, sans que cela ressemble trop à une psychothérapie de groupe. . .

Les deux dimensions du projet («personnel» et «professionnel») doivent être affirmées. Il ne s'agit plus de faciliter l'insertion professionnelle d'un étudiant à l'issue de la formation, de favoriser la transition entre les études et la vie professionnelle, mais bien de préparer les étudiants à évoluer tout au long de la vie.

Le Projet Personnel et Professionnel a pour vocation d'outiller les étudiants en leur fournissant des méthodes, réflexes, principes qu'ils pourront réinvestir tout au long de leur carrière, en particulier lors des phases de transition. L'une des difficultés principales repose sur le fait que les étudiants soient capables de réutiliser ces méthodes dans un contexte quel que peu différent (préparation de l'après - IUT, recherche d'un nouvel emploi, réorientation professionnelle. . .

L'étudiant doit construire son projet à partir d'expériences construites, vécues, capitalisées et confrontées avec d'autres. Il doit être le principal acteur de la démarche. Le travail entrepris dans le cadre d'une telle démarche doit lui apprendre à mieux se connaître, à faire des choix, à hiérarchiser ses priorités, à définir ce qui lui convient le mieux, à construire des expériences et en tirer un bilan, puis à mettre en adéquation ses ambitions et sa capacité à les réaliser.

Il doit être capable de saisir les opportunités qui s'offrent à lui et adapter sa propre stratégie en conséquence. C'est parce qu'il va expliciter son projet vis à vis d'un tiers (confrontation de son propre projet avec celui d'autres étudiants, présentation de son projet à des professionnels ou des enseignants), qu'il va progressivement se l'approprier et l'enrichir. Ce projet évolue donc constamment, l'étudiant le transformant et le remettant régulièrement en question.

Pour se projeter, au moins à court et moyen termes, l'étudiant doit disposer d'informations précises sur l'environnement professionnel du diplôme qu'il prépare. C'est d'autant plus important que le DUT prépare à des palettes de métiers larges et que les licences professionnelles répondent souvent à des besoins émergeants. Or lorsque l'étudiant entre à l'IUT, qu'il soit néo-bachelier ou en réorientation, il ne dispose que d'une vision très partielle des secteurs d'activité, métiers, postes de travail, entreprises...

L'étudiant doit également apprendre à maîtriser l'information sur les métiers pour s'en faire une idée réaliste et se débarrasser de certaines représentations erronées. Un métier ce n'est pas seulement un niveau de salaire, un statut ou un niveau d'études. L'étudiant doit apprendre à contextualiser un métier, c'est à dire à prendre conscience de la pluralité de ses modes d'exercices et de la diversité de ses conditions de mise en oeuvre. Le projet personnel et professionnel relève donc d'une démarche progressive où l'image d'un métier, dans toutes ses dimensions s'affine peu à peu et donne du sens au parcours de formation qui y conduit.

La découverte des métiers peut se faire de multiples façons, recherche théorique à partir de sources documentaires, recherche pratique à partir d'entretiens auprès de professionnels... La diversité des témoignages (enseignants, professionnels, anciens, voire étudiants encore en formation) contribue à élargir la vision que les étudiants peuvent avoir des métiers. La synthèse de ces témoignages leur permet de se construire progressivement de nouvelles représentations de ces métiers.

L'animation du projet personnel et professionnel n'est pas l'affaire de spécialistes. Si longtemps l'orientation a été une affaire de famille, puis du ressort de spécialistes-experts, elle concerne aujourd'hui l'ensemble des acteurs de la formation. Pour les équipes pédagogiques qui ont la charge d'une telle démarche, il ne s'agit pas de transmettre un savoir que l'étudiant aura à restituer *in fine*, mais de favoriser *l'appropriation par l'étudiant de la construction de son propre projet.*

Les enseignants possèdent de nombreux atouts et de la méthode. Le suivi régulier et la bonne connaissance des aptitudes de leurs étudiants leur permettent de les guider, de les motiver, de les aider à révéler leur potentiel. A travers son rôle d'accompagnateur, l'enseignant apprend autant que l'étudiant. Il approfondit lui-même sa connaissance des métiers et s'approprie la diversité de leur modes d'expression. Si le rôle de l'équipe pédagogique doit rester majeur, celle-ci peut s'appuyer selon les différentes étapes du processus sur des spécialistes (enseignants de communication, responsables de bilans de compétence, experts – métiers, SCUIO, services de Formation Continue...).

Le projet personnel et professionnel ne consiste pas seulement à initier une

démarche. L'accompagnement des étudiants doit se faire tout au long de la formation et quel que soit le parcours privilégié par l'étudiant. Sa forme en revanche est évolutive. En début de formation, il s'agit d'offrir une bonne connaissance de l'environnement professionnel et d'établir un lien entre la connaissance de soi et l'émergence progressive d'un projet professionnel. Pour les spécialités qui proposent un stage dès la première année, les séances de projet personnel et professionnel doivent fournir des méthodes de travail et aider à préparer cette première expérience professionnelle. A l'issue d'un stage de découverte l'enjeu consiste à prendre du recul par rapport à l'expérience professionnelle vécue par l'étudiant et d'en tirer tous les enseignements afin de consolider ou infléchir le projet initial. Il s'agit par la suite de permettre une orientation entre les différentes voies proposées : insertion immédiate à l'issue du DUT, insertion différée par une année supplémentaire ou une poursuite d'études plus longue... La multiplication des objets crée l'acquisition de méthodes utiles tout au long de la vie.

Elle doit s'accompagner d'éléments de mesure pendant et à l'issue de la formation (enquêtes de suivi des diplômés).

Le projet personnel et professionnel ne doit être ni optionnel, ni isolé du reste de la formation. Il est par essence fortement transdisciplinaire, faisant appel à des connaissances en matière de recherche documentaire, supposant la maîtrise de méthodes d'expression, favorisant l'approfondissement de connaissances en rapport avec les milieux professionnels constituant les débouchés d'un diplôme... Il doit irriguer l'ensemble de la formation par les questions qu'il impose, ainsi que par la construction et la capitalisation d'expériences qu'il induit. Si le projet personnel et professionnel présente des enjeux en matière d'orientation, il doit également permettre une plus grande appropriation des expériences à caractère professionnalisant vécues par les étudiants (stage, projets tuteurés...).

Le projet personnel et professionnel doit s'intégrer à l'expérience déjà ancienne des IUT en matière de professionnalisation. Sa mise en place ne consiste pas à ajouter de manière mécanique un volume supplémentaire d'heures aux programmes pédagogiques nationaux. Elle vise au contraire à privilégier un certain arbitrage entre des activités déjà existantes, pouvant servir de support à son organisation (enseignements de communication ou de techniques d'expression, projets tuteurés, adaptations locales...) et l'adjonction d'initiatives nouvelles et innovantes.

Tout enseignant intervenant dans un module de PPP s'engage, vis à vis des étudiants qu'il accompagne à...

1. être essentiellement un guide suscitant le questionnement et non un dispensateur de connaissances ;

2. veiller à faire évoluer positivement sans déstabiliser ;

3. se garder de porter tout jugement de valeur sur la personne comme sur ses appartenances religieuses, culturelles, politiques, syndicales et ne pas avoir d'attitude discriminatoire ;

4. respecter la confidentialité des informations personnelles dont il aura connaissance ;
5. accepter que celui-ci refuse de communiquer certaines informations confidentielles ;
6. ne pas avoir pour fonction de résoudre les problèmes psychologiques ou se prêter à ce rôle ;
7. en cas de difficulté majeure, apparemment difficile à résoudre, s'en remettre à (ou orienter l'étudiant vers) une tierce personne, compétente pour traiter ce problème ;
8. rappeler clairement, au début de chaque module, les objectifs opérationnels à atteindre afin que le contrat PPP soit explicite ;
9. évaluer sur des données objectives le travail fourni en fonction des critères énoncés initialement (dans les objectifs) ;
10. être exigeant quant à la qualité du travail fourni, ce qui est une marque de respect et d'honnêteté (ainsi qu'un facteur de progrès).

Àprès ce cadrage, nous pouvons maintenant commencer à dérouler les différentes fiches de travail pour l'étudiant. La première fiche est une présentation générale du champ professionnel de l'informatique. Elle comporte des questions. La deuxième fiche s'intitule «J'ai de l'expérience»et propose à l'étudiant de remplir des tableaux lui permettant de mettre en lumière une expérience. Cette fiche s'accompagne de remarques à destination des évaluateurs.

Nous continuons par une fiche sur les profils et les métiers. Les paragraphes suivants sont consacrés à la préparation et à la restitution de l'interview d'un profesionnel. Notons que ce travail vise à explorer les métiers, leur contenus concrèts.

Nous proposons également un travail lié à une visite d'entreprise sous l'angle de sa préparation et de sa restitution. Là aussi, ce sont les métiers qu'il faut «creuser»dans le sens de la découverte de la réalité fine du contenu d'un métier jusqu'à sa mise en perspective en terme de carrière par exemple.

Nous proposons ensuite une fiche pour un travail autour de la veille technologique. Cette partie invite l'étudiant à préparer des exposés sur les technologies, des outils un peu novateurs. Dans notre cas, les outils qui sont à découvrir sont des outils du «monde du logiciel libre». Nous terminons par une fiche relative aux formations post-DUT. Cette fiche vise à formaliser une démarche pour s'insérer dans une formation après le DUT.

5.2 Introduction à l'informatique et à ses métiers

IMAGINONS que vous venez d'obtenir le Baccalauréat et vous intégrez une formation en informatique. Ce paragraphe introductif évoque la «profession informatique»(son organisation, ses métiers). Il est organisé en deux parties. La

première partie introduit la branche professionnelle : les champ d'intervention des entreprises, les métiers, la deuxième partie quant à elle vous invite à réfléchir sur les diplômes, l'alternance, les certifications, les qualifications.

Vous trouverez également au fil du texte des exercices (qui sont présentés de manière encadrée dans le texte) consistant principalement à utiliser Internet pour repérer de l'information. Le vocabulaire technique relatif à l'organisation du champ professionnel et des métiers est en italique dans le texte. Les définitions relatives à ces termes sont données autant que possible.

5.2.1 Organisation du champ professionnel

Les technologies de l'information et de la communication au sens large font travailler en France environ 700000 salariés (350000 environ en informatique et 350000 dans les télécom cette dichotomie est discutable.

Le site http://www.passinformatique.com donne des chiffres par «grands secteurs».

L'informatique est présente dans tous les secteurs de l'économie et les informaticiens travaillent chez :

- Les constructeurs et les fournisseurs de systèmes informatiques qui développent et vendent des produits informatiques. Exemples : Bull, IBM, HP ;
- Les prestataires de services informatiques, sociétés de services et d'ingénieries informatiques (SSII) qui conçoivent des solutions logicielles (logiciels standards ou très spécifiques) ; Exemple : Cap-Gemini, EDS, Unilog, Business Objects ;
- Les utilisateurs (qui constituent le marché de l'emploi le plus important). Les utilisateurs organisent des équipes pour développer et exploiter les applications dont ils ont besoin. Ils sont présents dans l'industrie, les banques, l'assurance, les administrations, la distribution, le transport, le tourisme. Ce sont aussi bien des grandes entreprises (regroupées au sein du Cigref - http://www.cigref.fr) que des petites ou moyennes entreprises.

Les principaux domaines d'application de l'informatique en France sont :

- l'informatique de gestion,
- l'informatique industrielle et technologique,
- les télécommunications et les réseaux,
- l'internet et le multimedia.

Exercice 1 (*Fiches métiers*) : Trouvez sur le site PassInformatique[1] une fiche métier pour chacun de ces quatre domaines. Sur le site, consultez le lien sur les

[1] http://www.passinformatique.com

domaines de l'informatique, isolez des mots clés et cherchez les dans la fiche métier. Les fiches métiers que vous avez sélectionnées correspondent elles à un diplôme BAC+2 ?

Les métiers de l'informatique sont des métiers de :

Développement : qui regroupe les activités de création, intégration de logiciels ou matériels informatiques dans l'entreprise ;

Support : qui regroupe les activités de maintenance, d'exploitation et d'aide (support) auprès des utilisateurs ;

nseil et expertise : qui regroupe les activités d'aide auprès des entreprises pour qu'elles choisissent les outils informatiques les plus adaptées ;

'information (SI) : qui regroupe les activités de gestion (direction, contrôle des coûts, GRH) de l'informatique.

Exercice 2 (*Compléments à l'exercice précédent*) : Pour les métiers que vous avez identifiés dans l'exercice précédent, dire, brièvement :

- à quoi ils servent dans l'entreprise ;
- les principales missions des personnes exerçant ce métier ;
- la formation et l'expérience nécessaires ;
- les évolutions de carrières envisageables pour une personne qui occupe ce métier ;
- les types de compétences peu nécessaires/nécessaires/très nécessaires. On rappelle que :
  - Les savoir-faire techniques sont liés à la technique informatique
  - Les savoir faire généraux regroupent les compétences en animation, conduite de projet, connaissance de l'entreprise, langue, droit, etc
  - Les savoir faire, aptitudes comportementales regroupent les compétences en résolution de problème, compétences managériales, etc.

  On pourra se référer aux sites du CIGREF (http://www.cigref.fr) et à celui du Syntec (http://www.passinformatique.com).

Dans la suite du module PPP nous allons discuter dans le détail des métiers. Pour linstant, nous pouvons avancer que les métiers de l'informatique sont en constante evolution. L'évolution des métiers est liée à l'émergence de nouvelles technologies, à la complexité croissante des services à rendre, à l'écoute et à la satisfaction des besoins des utilisateurs.

Il est certain que dans 40 ans, à la veille de votre retraite professionnelle, vous n'exercerez pas le même métier que celui de vos débuts ni même que celui-ci soit identique à ce quil est aujourdhui.

L'article du monde du 3/8/2005 écrit Yves Eudes intitulé *Dans la main invisible du Net*[2] projette dans un futur proche (2030) les tendances et programmes scientifiques en informatique déjà lancés en 2005. Dans cet article, le lecteur se retrouve dans un monde où il est en permanence connecté au Net : des réseaux de capteurs implantés sous la peau grâce aux nano-technologies surveillent sa santé, les données lui appartenant sont stockées dans le réseaux (et pas sur ce que l'on appelle aujourdhui son disque dur), son PDA (Personnel Digital Assistant) l'avertit quand un ami passe à proximité car il est relié à d'autres PDA par une technologie ad-hoc (sans antenne relais - le PDA est à la fois récepteur et transmetteur ou relais pour dautres PDA comme le permet déjà la norme WIFI) etc

Si ce qui est décrit se réalise celà a pour conséquence de développer les activités de services (par exemple le service qui transmet les données d'un capteur mesurant la pression artérielle, celui qui les analyse, celui qui les conserve), de transformer le travail (un télé-médecin est disponible instantanément et à tout moment il est alerté par une alarme il est d'astreinte en permanence - cela modifie considérablement le rapport au temps et à l'espace), la relation entre les Hommes (à la limite je ne vois pas mon docteur il adapte mon traitement automatiquement en fonction du résultat des analyses reçues des capteurs, tout ceci modulo un accord préalable).

Exercice 3 (*À préparer après lecture complète du chapître*) : Pouvez-vous identifier les secteurs et les compétences actuelles qui répondraient le mieux aux réalités de 2030 telle quelles sont décrites dans l'article du Monde ?

Ainsi, puisque l'évolution des techniques est continue en informatique (voire même imprévisible : qui peut dire ce que sera l'informatique dans seulement 10 ans ?) et transforme lacte productif, la gestion de sa carrière professionnelle (et par ailleurs d'étudiant) implique de développer de nouvelles compétences tout au long de la vie professionnelle (et universitaire). Un certain nombre de savoirs et savoirs-faire sont passés par vos enseignants qui sont là pour vous assurer un bon niveau de connaissances générales mais ceux-ci ne seront plus là pour vous évaluer après votre sortie de l'université.

Nous verrons par la suite que la profession demande des compétences qui vont au delà des compétences techniques, des savoirs faire immédiats : il faut aussi avoir des aptitudes à communiquer, à travailler en groupe.

Ainsi, la gestion de sa carrière professionnelle (et d'étudiant) nécessite de définir des objectifs prioritaires qui favoriseront sa propre mobilité, de développer des initiatives qui permettront de réfléchir à un projet, de choisir et de mobiliser des moyens pour mettre en oeuvre son projet, d'intégrer les contraintes qui pèsent sur le monde du travail (ou le monde universitaire) et ses modes de reconnaissance et d'approfondir ses connaissances.

[2] http://www.lemonde.fr/web/article/0,1-0@2-3230,36-677367@51-633431,0.html

5.2.2 Diplômes, qualifications, certifications

La formation de DUT vous conduit au bout de deux ans à un diplôme c'est à dire à un titre donné par le ministère de l'éducation nationale sur la base de programmes nationaux ; titre qui est reconnu également par les *branches professionnelles* (**définition :** une branche professionnelle est constituée par un champ professionnel caractérisé par la présence dune convention collective (voir la définition plus bas) et de représentants syndicaux et patronaux : les partenaires sociaux). Le rôle de la branche est de veiller au développement de celle-ci par des négociations régulières qui peuvent porter sur le temps de travail, les qualifications professionnelles requises...

La figure ?? présente une organisation des diplômes et des certifications. Les études liés à la médecine n'y figurent pas. Le BTS (Brevet de Technicien Supérieur) qui se prépare dans un lycée non plus. Le diplôme d'ingénieur peut également se préparer dans une université. Renseignez-vous auprès des universités.

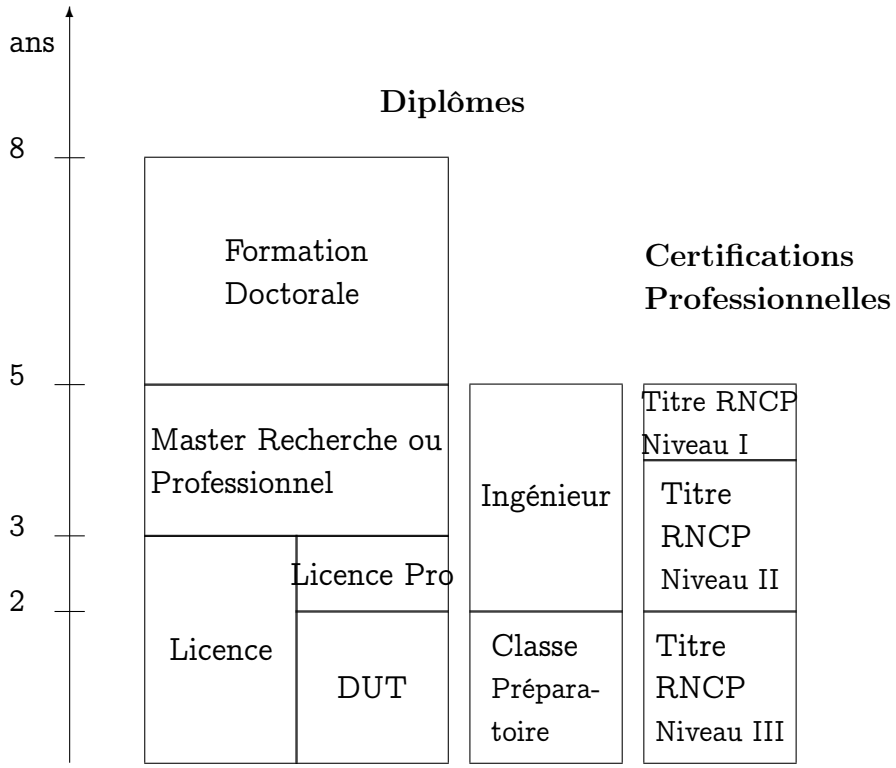


FIGURE 5.1: Carte des formations

Les barres verticales de gauche et de centre sont à priori connues des étudiants, celle de droite beaucoup moins. Sur cette barre, les mots clés sont Titre et Niveaux. Le RNCP (Répertoire National des Certifications Professionnelles[3]) est une base de données des certifications à finalité professionnelle reconnues par l'État et les partenaires sociaux, classées par domaine d'activité et/ou par niveau. Un titre de niveau III correspond à un titre de technicien supérieur et de deux années d'étude après le baccalauréat. Un titre de niveau II correspond à un niveau de cadre et

[3] http://www.cncp.gouv.fr/

le niveau I à celui de cadre supérieur (cinq ans après le baccalauréat). Le site Interneet mentionné fournit des informations sur le secteur d'activité, un résumé du référentiel d'emploi ou des éléments de compétence acquis,les modalités d'accès aux titres.

Le lecteur notera que sur la Figure ?? il n'y a pas de flèches entre les barres verticales qui matérialiseraient des correspondances entre diplômes et certifications car cela n'est pas automatique. Par exemple, une certification de qualification professionnelle n'est pas systématiquement reconnue par le RNCP comme un diplôme mais peut être considérée comme un titre.

La reconnaissance d'un titre ou d'un diplôme peut signifier en particulier que vous allez être reclassé de manière (presque) automatique dans une grille de salaires, en fonction d'une *convention collective* (**définition :** une convention collective est un texte élaboré dans le cadre de négociations collectives entre les partenaires sociaux permettant de définir un cadre législatif au travail dans la branche). Une convention collective est un «plus»pour le salarié et ne doit pas contredir le droit du travail. Les conventions collectives peuvent se trouver sur le site http://www.legifrance.gouv.fr.

Cependant, de nombreuses conventions collectives ne mentionnent pas explicitement de diplômes et c'est au demandeur de montrer au recruteur que ses compétences coincident avec le poste qu'il souhaite occuper. L'étudiant doit comprendre qu'il est demandeur et qu'il doit répondre aux attentes exprimées par le recruteur dans une fiche de poste. Les attentes ne sont pas exprimées en terme de disciplines ou de matières !

Le DUT informatique confère un diplôme de technicien c'est à dire celui qui met en œuvre de manière systématique un procédé. Un technicien informatique pourra voir son salaire fixé par la convention collective dont il dépend à partir du moment où il aura explcité ses compétences. De plus, s'il est salarié dans une entreprise utilisatrice (une banque par exemple), il ne dépendra pas de la même convention collective que s'il travaille chez Bull (qui est un constructeur). Les salaires de techniciens, suite à une première embauche, pourront être différents ! Si on vous propose un travail dans le secteur banquaire (ou dans un autre secteur), veuillez consulter la convention collective des banques pour connaître les rémunérations. Il sera peut être nécessaire de passer par un «équivalent»(de technicien) car pour le BTP par exemple, il nest pas sûr que la convention collective mentionne le terme «informaticien». Par contre elle mentionnera peut être le terme technicien.

Ce qui vient d'être dit ne concerne que la première insertion professionnelle du jeune diplômé. En effet, les branches professionnelles reconnaissent et encouragent la mobilité des salariés visant à ce qu'ils acquièrent de nouvelles compétences. Ainsi, après cinq années d'expérience professionnelle, si vous voulez une augmentation de salaire, il ne suffira pas de dire que vous avez un DUT mais il faudra défendre votre projet et les compétences acquises pendant ces cinq ans. Il s'agira alors de montrer et de justifier que vous pouvez être reclassé à tel ou tel niveau dans la convention collective.

Autrement dit, le diplôme vous permet d'accéder à un niveau de rémunération.

Accéder à un niveau supérieur passe par différentes stratégies qui peuvent être par exemple (liste non exhaustive) :

- passer un diplôme plus élevé que le DUT (Bac+2). Attention aux paliers : la notion de cadre supérieur est à partir du niveau BAC+4 pour l'APEC (Agence pour l'emploi des cadres). Cela ne veut pas dire qu'il ne faut pas faire de licence profesionnelle !
- passer un diplôme plus élevé par la voie de l'alternance, de l'apprentissage ou de la VAE (validation des acquis et de l'expérience - pour les salariés exclusivement) ;
- acquérir un CQP (Certificat de Qualification Professionnelle). Il s'agit de dispositifs reconnus par les branches (donc les entreprises). Il sont élaborés dans le cadre de négociations collectives. Par exemple, une partie des informaticiens en France dépendent de la branche de la métallurgie : veuillez rechercher sur le site http://www.uimm.fr/ les CQP de la branche. Vous devriez retrouver le métier de «Assistant(e) de projet en systèmes industriels informatisés».

Exercice 4 (*CNCP - RNCP*) : Connectez-vous au site http://www.cncp.gouv.fr/ et recherchez les BTS informatique reconnus par la commission des titres.
Exercice 5 (*APEC*) : Connectez-vous au site http://www.apec.fr/ et vérifier le niveau de diplôme requis pour accéder à une fonction de cadre (par exemple en informatique).
Exercice 6 (*Certificat de qualification professionnelle*) : À qui s'adressent ces CQP (niveau d'entrée requis ?). A quoi correspondent les termes «3ème échelon du niveau IV (échelon 285)» ?

Pour votre information, ou pour la question précédente, vous pouvez aussi consulter les sites :

- http://www.passinformatique.com/60-formationcontinue/30-20-31_metallurgie.asp et plus particulièrement le site : http://www.droitdelaformation.com/actualites/actu0508006.htm pour des détails sur le CQP Administrateur de réseau d'entreprise.
- http://www.cqpm.com/ : certificats de qualification paritaire de la métallurgie. Ces certificats concernent tous les secteurs de la métallurgie et tous les niveaux de qualification. Dans l'informatique, il existe notamment des CQPM de :
  - technicien en installation et en développement micro-informatique ;
  - chargé de projet informatique ;
  - chargé de développement de projets logiciels industriels (temps réel) ;
  - technicien de mise en oeuvre et de maintenance de microsystèmes informatiques ;

- technicien en instrumentation intelligente et en transmission de données ;
- assistant de projet informatique ;
- technicien pour le développement de l'informatique industrielle dans le secteur de la mécanique (TEDIM)
- assistant(e) de projet en systèmes industriels informatisés,
- correspondant(e) pour les technologies de l'information et de la communication ;
- technicien de maintenance réseau ;
- technicien de télécommunication, réseaux informatiques et domotiques ;
- administrateur d'infrastructures de réseaux informatiques et télécommunications.

Note : l'AFPA (Association nationale pour la formation professionnelle des adultes) qui est sous la tutelle du ministère du travail permet d'obtenir des titres de techniciens en informatique.

Exercice 7 (*AFPA*) : Veuillez consulter le site http://www.afpa.fr afin d'énumérer les titres de techniciens (en informatique) délivrés par l'AFPA. Faites une recherche sur Google pour trouver une définition de «Titre Professionnel, de Certificat de Compétences Professionnelles», en particulier en informatique.

Exercice 8 (*Convention collective des SSII*) : À partir du lien http://www.syntec.fr/convention/sommaire.php qui concerne la convention collective des sociétés de service et d'ingénierie en informatique (SSII) et les éditeurs de logiciels :

- Téléchargez l'annexe 3 pour avoir une idée de la rémunération d'un employé, technicien et agent de maitrise ;
- Le titre 7 de la convention parle de la formation : quels sont les droits du salarié en matière de congés de formation et comment ce congé est-il financé ?
- Au sujet de l'information et l'orientation tout au long de la vie, lire l'annexe 6 de la convention ; Sur quels critères types évalue-ton le salarié ? Ces critères sont-ils les mêmes que ceux pratiqués à l'université ?
- La partie 4 de l'annexe 6 parle de l'alternance à partir de la page 117. Dites quelles sont les rémunérations proposés dans le cadre d'un contrat de qualification.

5.2.3 Financement d'une formation professionnelle

Que vous soyez salarié ou demandeur d'emploi, plusieurs dispositifs législatifs vous permette de financer le coût de votre formation professionnelle. En règle général, les entreprises planifient sur l'année leurs actions de formation dans le cadre du

plan de formation. Si l'entreprise n'a pas conçu de plan de formation, vous pouvez lui demander de prendre tout de même en charge votre formation ainsi que vos frais d'hébergement et de déplacement.

Par exemple, les CQP (Certification de qualification professionnels) sont créés et délivrés par les branches professionnelles et permettent aux salariés d'acquérir une qualification opérationnelle reconnue. Ils peuvent être préparés dans le cadre du contrat de qualification, du plan de formation, du capital temps de formation ou du congé individuel de formation.

Rappelons que les CQP ont été validé de manière paritaire, ils sont reconnus dans les seules entreprises de la branche concernée. Les CQP homologués inscrits au RNCP (Répertoire Nationale des Certificats Professionnels[4])) dérogent cependant à cette règle, les titulaires pouvant en tirer parti auprès d'entreprises de branches différentes.

Le droit individuel à la formation (DIF) s'adresse à tous les salariés du secteur privé en CDI (Contrat à durée indéterminée), CDD (Contrat à durée déterminée) ou intérimaires. Depuis le 7 mai 2005, le DIF permet ux salariés, en accord avec l'employeur, de bénéficier de 20H de formation par an, cumulables sur 6 ans et financées par l'entreprise. La formation peut avoir lieu pendant ou en dehors du temps de travail suivant les accords pris par les partenaires sociaux.

Si la formation est acceptée (pour celà, présentez votre demande en soulignant que votre projet s'inscrit dans les besoins de l'entreprise), il y aura prise en charge de la formation ainsi qu'éventuellement de l'hébergement et des déplacements.

Le congé individuel de formation (CIF) s'adresse aux salariés en CDI, CDD et aux intérimaires et contient une clause d'ancienneté. Ainsi, un salarié en CDI devra avoir travaillé pendant au moins deux ans, dont un an dans l'entreprise actuelle pour en bénéficier. Le CIF permet de suivre prendant un an à temps plain ou 1200 heures à temps partiel une foration de son choix, pendant le temps de travail. La prise en charge est assurée par l'organisme paritaire collecteur agrée au titre du CIF (Fongecif ou OPCA) auquel cotise votre entreprise. Par exemple l'organisme paritaire collecteur de l'ingénierie, de l'informatique, des études, du conseil, des foires et salons et des traductions est le FAFIEC.[5]

Rappelons que toutes les entreprises sont assujetties à la formation professionnelle sous la forme d'une taxe sur la masse salariale de l'entreprise (la taxe est différente selon que l'entreprise à plus ou moins de 10 salariés).

Vous pouvez aussi valider votre expérience pour accéder à un cycle de formation sans les diplômes ou titres pré-requis : la VAE[6] (Validation des Acquis et de l'Expérience). La VAE permet de valider une partie de la formation que vous souhaitez suivre ainsi que les examens correspondants, d'obtenir la validation de la totalité du diplôme, d'accéder à une formation sans avoir le requis à l'entrée. Il faut posséder une expérience d'au moins trois ans en rapport direct avec le contenu du diplôme. Le congé pour validation des acquis de l'expérience (24 heures maximum)

[4] http://www.cncp.gouv.fr/
[5] Voir http://www.fafiec.fr
[6] Voir http://www.travail.gouv.fr/dossiers/vae/

peut être lui aussi pris en charge au titre du plan de formation de l'entreprise.

Par ailleurs, lorsque vous êtes étudiant, comme les universités sont des acteurs majeurs en terme de formations diplômantes, n'hésitez pas à prendre des avis auprès des personnes ressources : responsables de formations, responsables des stages, de la formation continue... qui pourront vous conseiller. Si vous vous interrogez sur votre avenir professionnel, ne restez pas isolés !

Enfin, le Conseil Economique et Social[7] à publié le 11/7/2005, sous la direction de Michel Muller, le rapport 2005-13 intitulé «L'insertion professionnelle des jeunes issus de l'enseignement supérieur». Il est vivement recommandé de le lire. Vous pouvez le télécharger à partir du site Internet du Conseil Economique et Social.

5.2.4 Des sites de référence

- http://www.passinformatique.com/ et http://www.syntec.fr
- http://www.cigref.fr
- http://www.anpe.fr
- http://www.fafiec.fr
- http://c2i.education.fr/ : certificat informatique et internet. Dans le but de développer, de renforcer et de valider la maîtrise des technologies de l'information et de la communication par les étudiants en formation dans les établissements d'enseignement supérieur, a donc été institué un Certificat informatique et internet (C2i).
- http://www.soc-etudes.cgt.fr/download.php?view.191 : étude complète sur les développeurs informatiques ;
- http://83.143.18.40/dossiers/dossiers.php?id_dossier=130 : accès aux référentiels des métiers de l'Informatique, et de l'Ingénierie ;
- http://www.apec.fr
- http://www.legifrance.gouv.fr
- http://www.specif.org/[8]
- http://www.education.gouv.fr/ et plus particulièrement pour le supérieur : http://www.education.gouv.fr/sup/default.htm et http://www.cpu.fr/
- Les journaux : http://www.lemondeinformatique.fr et http://www.01net.com. Voir aussi http://www.letudiant.fr. En langue anglaise : http://slashdot.org

[7] Voir http://www.conseil-economique-et-social.fr

[8] Il y a un annuaire des formations en informatique (niveau Master)

5.3 J'ai de l'expérience

LE TRAVAIL qui va maintenant être demandé change de nature. Il s'agit pour l'étudiant de remplir des fiches d'auto-analyse. Vous pouvez faire figurer dans ces fiches vos stages, vos jobs d'été, ou des activités occasionnelles (coup de main pour des travaux, déménagement, cours particuliers, installation de matériel, garde d'enfants, etc.). Prenez prioritairement celles qui vous semblent les plus importantes. Les fiches concernent donc un travail de connaissance de soi, de prise de recul par rapport à ce que l'on sait faire.

FICHE 1	**Dénomination de l'activité :**
En quoi consistait mon travail ?	
Pourquoi ai-je choisi de pratiquer cette activité ?	
Quels atouts personnels avais-je pour accomplir cette tâche ?	
Quelles compétences nouvelles ai-je acquises ?	
Quels sont les aspects de cette expérience que je voudrais retrouver dans mon futur emploi ?	
Quels sont ceux que je voudrais éviter à l'avenir ?	

FICHE 2	**Dénomination de l'activité :**
En quoi consistait mon travail ?	

Pourquoi ai-je choisi de pratiquer cette activité ?	
Quels atouts personnels avais-je pour accomplir cette tâche ?	
Quelles compétences nouvelles ai-je acquises ?	
Quels sont les aspects de cette expérience que je voudrais retrouver dans mon futur emploi ?	
Quels sont ceux que je voudrais éviter à l'avenir ?	

FICHE 3	Dénomination de l'activité :
En quoi consistait mon travail ?	
Pourquoi ai-je choisi de pratiquer cette activité ?	
Quels atouts personnels avais-je pour accomplir cette tâche ?	
Quelles compétences nouvelles ai-je acquises ?	

Quels sont les aspects de cette expérience que je voudrais retrouver dans mon futur emploi ?	
Quels sont ceux que je voudrais éviter à l'avenir ?	

Passez en suite à la fiche suivante intitulée «Mes choix de vie»

	Descriptif de cette expérience (où ? quand ? comment ?)	**Pourquoi j'ai fait ce choix** (motivations, contraintes)	**Ce que j'ai appris à cette occasion** (sur moi-même ou sur les autres)	**Points positifs ou négatifs** de cette expérience
Centres d'intérêt : Loisirs, pratique sportive, passions, activités associatives...				
Vacances, Voyages, séjours linguistiques, occupations diverses...				

Vie quotidienne : Je travaille pour financer mes études ou mes loisirs. J'habite seul, à deux, en famille. J'étudie près ou loin de chez moi...				

A partir des expériences que vous avez décrites précédemment et de l'ensemble de votre vécu, essayez de cocher tous les verbes qui correspondent aux savoir-faire que vous pensez posséder. Cette nouvelle fiche s'appelle «Mon profil d'aptitudes»

CREER	COMMUNIQUER	ORGANISER
Inventer Innover Améliorer Développer Transformer Réfléchir	Ecouter Dialoguer Informer Expliquer Clarifier Détendre l'atmosphère	Anticiper Coordonner Planifier Budgétiser Exploiter Hiérarchiser
CONSEILLER	**DIRIGER**	**DECIDER**
Comprendre les besoins Analyser Aider S'adapter Convaincre Prévoir	Assumer Commander Déléguer Manager Orienter Etre responsable	Choisir Juger Régler Réagir Trancher Etre réactif

TRAVAILLER A PLUSIEURS	SAVOIR	AGIR
	S'informer	Expérimenter
Animer	Consulter	Entreprendre
Savoir se concerter	Se mettre à niveau	Réaliser
Motiver	Etudier	Exécuter
Gérer	Découvrir	Concrétiser
S'entraider	Etre curieux	Persévérer
Savoir se compléter		

Pour chaque verbe coché, comptez un point et totalisez ensuite les points obtenus dans chaque case. Une case comptant 3 points, ou plus, constitue un pôle d'aptitudes majeur. Avec moins de 3 points, c'est un pôle mineur. Vous pouvez ainsi remplir la première colonne du tableau ci-dessous :

Aptitudes majeures (ou, à défaut, mineures)	Aptitudes préférées	Aptitudes reconnues

- Pour remplir la deuxième colonne, cherchez, parmi l'ensemble des aptitudes (majeures ou mineures) que vous avez repérées, celles que vous aimez particulièrement exercer.
- Pour remplir la dernière colonne, cherchez de la même façon quelles sont les principales aptitudes que votre entourage, personnel ou professionnel, vous reconnaît.

5.4 Vers un choix de métier

CETTE FICHE est composée de deux parties. Nous commençons par sonder les besoins et valeurs dans lesquels vous vous reconnaissez puis nous travailloins sur le plan du profil et des métiers. Ce paragraphe vise à se projeter dans l'avenir. Il s'agit pour nous de montrer combien il est important de se projeter dans l'avenir, à court, à moyen et à long terme.

5.4.1 Besoins et valeurs

La fiche suivante s'intitule «BESOINS ET VALEURS». Le choix d'un itinéraire professionnel n'est pas seulement le fait d'aptitudes : il correspond aussi à des motivations liées à la façon dont on conçoit globalement sa vie. Vous avez peut-être déjà constaté cela lorsque vous avez justifié vos choix de métiers. La question financière, le désir d'être indépendant, d'avoir du temps à soi ou des responsabilités, l'envie d'être utile, d'évoluer ou de faire ce qu'on aime entrent aussi en ligne de compte. Ce sont ces éléments que visent à repérer les tableaux ci-dessous.

MES BESOINS	Peu important	Important	Très important
Accorder du temps à mon entourage			
Avoir du temps pour moi			
Me sentir utile			
Me sentir en sécurité			
Me sentir solidaire d'un groupe			
Donner l'image de quelqu'un qui a réussi			
Gagner de l'argent			
Avoir du pouvoir			
Etre indépendant			
Etre responsable de mes actes			
Ne pas m'ennuyer, me répéter			
Autres...			

MES VALEURS	Peu important	Important	Très important
Le travail			
La réussite individuelle			
Le savoir			
La créativité, l'inventivité			
L'action			
L'épanouissement personnel			
Le respect des autres			
Le sens des responsabilités			
La solidarité, l'entraide			

L'affirmation de soi, l'autorité			
La détermination, la combativité			
Le désir d'entreprendre			
L'honnêteté, la franchise			
La liberté			
La compétition, la confrontation			
Autres...			

Après avoir rempli ces deux tableaux, pensez-vous que ces orientations personnelles vous « interdisent » certains métiers ? En quoi peuvent-ils vous permettre de préciser votre projet ?

Revenons maintenant à la projection en terme de métier. Essayons les trois cas de figure. Vous pouvez choisir des métiers en lien, ou non, avec l'informatique.

5.4.2 Projection à court terme

Imaginez que des circonstances imprévues vous contraignent tout à coup à arrêter vos études : vous devez au plus vite subvenir à vos besoins. Vos aptitudes sont celles que vous avec listées auparavant. Le marché du travail et l'urgence vous imposent un certain réalisme. Quel emploi essayeriez-vous d'obtenir en priorité ?

Métier choisi :

Raisons de ce choix :

Aptitudes personnelles pour ce métier :

5.4.3 Projection à moyen terme

Vous venez de terminer votre DUT informatique. Vous décidez de chercher immédiatement un emploi dans ce domaine.

Quelles sont les aptitudes nouvelles que vous êtes susceptible d'avoir acquises à travers le stage et la formation, et qui justifient de viser un métier différent de celui envisagé à court terme ?

Quel métier choisiriez-vous ?

Raisons de ce choix :

5.4.4 Projection à long terme

Il existe un métier qui vous fait rêver depuis toujours. Ou un projet professionnel qui vous semble idéal, bien que difficile à atteindre. Si vous manquez d'idées, essayez de voir où pourraient vous mener votre ambition et vos désirs.

Quel est ce métier ?

Pourquoi ce choix ?

À priori, quelles sont les aptitudes qu'il vous faudrait acquérir pour exercer ce métier (formation, expérience, vécu personnel) ? Vous pouvez éventuellement vous renseigner sur les attentes liées à ce métier.

5.5 Aides pour l'évaluateur

NOUS INTRODUISONS maintenant quelques point de repère à destination des évaluateurs afin que le travail d'auto-analyse se déroule bien. Il faut prévoir trois/quatre séances de travail (3/4 fois 1h30) pour le remplissage des fiches dont nous venons de parler dans les précédents paragraphes.

1. Installer la salle en rond ; ne pas hésiter à tout redéplacer pour faire des petits groupes de travail. Donner au maximum la parole, relancer, interroger, faire des tours de tables (même partiels) s'il y a des blancs, passer de groupe en groupe pour débloquer les situations et vérifier l'avancée, etc. Leur donner les fiches à la séance précédente si on veut qu'ils y réfléchissent avant d'arriver (le risque est qu'ils oublient le document). Commencer, éventuellement, par un tout de table sur le thème : pourquoi êtes-vous là ? pourquoi avez-vous choisi l'informatique. Silence et réponses vagues assurées : en tirer l'idée qu'il faut préciser son projet sans attendre la 2^e année (les connaissances arrivant peu à peu ne suffisent pas à y voir plus clair, il faut adopter dès maintenant une position de recul qui permet de se poser des questions sur ce que l'on fait et ce que l'on veut faire).

2. Rappeler en introduction les enjeux du PPP, en insistant sur l'aspect personnel de cette démarche. Voir l'introduction de ce chapitre.

3. Pour prévenir les blocages individuels ou collectifs (« je vois pas pourquoi vous avez besoin de savoir ça », « c'est personnel », « je sais pas, je peux pas répondre à la minute » etc.), mettre l'accent sur votre rôle d'accompagnateur, qui ne juge pas, et sur le temps de maturation de ces éléments (on a le droit de changer d'avis). Vous pouvez vous référer à la « charte éthique » du PPP (même source : ci-dessous). On impose, en effet, une réflexion minimale, permettant d'amorcer un processus que les étudiants devront poursuivre par eux-mêmes. On leur pose, en effet, des questions indiscrètes. Ils ne sont pas obligés de tout nous dire, c'est la prise de conscience qui compte. Mais on vérifie, en effet, que le travail demandé est fait, parce que chacun a dans son expérience des éléments pas si inavouables que ça et qui peuvent servir de base à la réflexion... En plus, on est censés jouer le rôle des tuteurs, cette année, donc il faut qu'ils nous fassent un peu confiance.

Et si c'est le temps qui manque, vous pouvez toujours vérifier le travail la semaine suivante seulement. Enfin, leur rappeler que lors des entretiens de recrutement, les employeurs n'hésitent pas à poser des questions indiscrètes, et que ce ne sera pas alors le moment d'y réfléchir !

4. Les étudiants risquent d'être assez lents, sur le remplissage des tableaux. Ce qu'on est en droit d'attendre d'eux en cours, c'est de remplir au moins certaines rubriques des tableaux, suffisamment pour montrer qu'ils ont compris l'enjeu, quitte à laisser les moins rapides finir chez eux. On peut accepter des blancs, des hésitations... Mais alors, on peut imaginer de demander à celui qui ne se sent pas prêt à remplir la grille de mettre par écrit, en une page, ou quelques lignes selon que c'est un exercice entier ou une simple rubrique, ce qui le fait hésiter.

5. Certaines grilles peuvent être remplies en groupe, à partir d'une discussion, ou d'un jeu de questions réponses (ex : expérience professionnelle; besoins et valeurs ; profil d'aptitudes)

Déroulement des 2 séances

- Séquence 1 : « J'ai de l'expérience »

L'objectif est de leur montrer que tout en étant jeunes étudiants, ils ont déjà tout un passé qu'ils peuvent mettre en valeur (perspective des CV et lettres de motivation en filigrane), et qui peut les aider à mieux cerner leurs aptitudes et leurs envies (perspective PPP). On peut remplir la « fiche 1 » de cette première fiche avec eux, en expliquant les rubriques au fur et à mesure qu'ils les complètent, et leur laisser les deux autres à finir. On peut insister sur la définition de ce qu'est l'expérience professionnelle : un emploi ? une activité rémunérée ? une activité qui suppose certaines compétences reconnues dans le monde professionnel ? On peut insister aussi sur la possibilité de valoriser une expérience, même modeste. Enfin, des leçons tirées de ces expériences, il faudrait tirer une sorte d'auto-portrait : on le fera avec la fiche « profil d'aptitudes » plus loin (ramasser ces feuilles permettra aussi de les redistribuer à la séance suivante pour cet exercice).

On passe en douceur à la deuxième fiche : « **mes choix de vie** ». C'est là surtout que ça coince, il faut donc bien expliquer l'enjeu si on n'a pas trop insisté dessus en introduction. Le but n'est pas de savoir s'ils habitent chez leurs parents ou ce qu'ils font de leur temps libre, mais bien d'essayer de cerner, à terme, leurs dispositions pour tel ou tel type de métier. Qu'ils se rappellent, par exemple, la dernière fois qu'ils ont fait un test psychologique dans un magazine. On ne va pas leur proposer une grille de réponse avec des carrés, des ronds ou des triangles, mais il faudra qu'ils essaient de s'interroger sur ce que signifient les choix accumulés qui ont abouti à leur vie actuelle (autonomie, curiosité, goût du contact, responsabilisation, capacité à décider par soi-même, cohérence, etc.). Et si certains de ces éléments ne sont pas choisis, cela peut amener à réfléchir à la façon dont on réagit face à la contrainte (passivité vs réaction, rigidité vs adaptation, etc).

On peut les inviter à mettre par écrit un auto-portrait synthétisant tout cela (et éventullement à nous le faire lire s'ils le veulent). On peut imaginer aussi de

demander à chaque étudiant de faire le portrait de l'autre à partir de ces grilles (soit sous forme rédigée, soit en s'appuyant sur la grille « profil d'aptitudes). Cela peut donner des choses très intéressantes et une véritable prise de conscience, mais cela suppose une très bonne ambiance de groupe et une envie collective de mieux se connaître.

- Séquence 2 : « Mon profil d'aptitudes »

On a vu qu'il pouvait être intégré à la séquence précédente. Même si on l'en détache, il en découle. Cette fiche est une version remaniée, et simplifiée, d'un document que nous avait donné Françoise, et qui figure en fait dans le classeur PPP, dans la fiche appelée « le cas de Julie » : vous pouvez y jeter un œil. Si le groupe est très réticent à parler de lui-même, on peut proposer de cette façon de faire un exemple sur une personne imaginaire. Ce travail peut se faire à partir des résultats de la séquence précédente, mais on n'est pas obligé de s'y limiter. Il faut tenir compte de tout ce qu'on croit savoir de soi, et éventuellement de ce que les autres en disent. On peut demander aux étudiants de se présenter narrativement, en préalable (mais à mon avis, à l'oral c'est beaucoup trop long, à l'écrit ils ne le feront peut-être pas, et on risque à nouveau de perdre du temps à vérifier). Bref, chaque étudiant doit cocher les termes qui lui semblent correspondre à son propre profil. Il faut qu'ils en cochent pas mal, sinon, on manque de matière pour la suite : il faut les aider à partir de leurs grilles.

Deux difficultés : la première est qu'il est possible que des dominantes réelles ne se dégagent pas. Il faut alors lister quend même les aptitudes latentes, voir celles qui pourraient être développées, parce qu'elles sont plus utiles, plus atteignables, plus souhaitées (se fixer des objectifs). Deuxième difficulté : pour que l'exercice soit intéressant, il faudrait pouvoir discuter du résultat avec chaque étudiant. On peut faire le choix de leur faire remplir cette grille à la maison, et de faire un débriefing collectif, ou individuel (les autres avancent sur les exercices suivants pendant ce temps).

- Séquence 3 : Besoins et valeurs

Idée conductrice : choisir un métier, c'est souvent choisir la vie qui va avec. Même si les contraintes du marché du travail, le hasard et les compétences de chacun pèsent évidemment sur nos choix, autant faire le point sur ce qu'on attend de notre avenir professionnel.

Besoins : cette partie insiste sur les conditions nécessaires à un relatif épanouissement. Ce n'est pas juste une question de confort personnel. On considère parfois qu'un individu se met réellement à remplir des missions collectives (par exemple, travailler pour le bien de l'entreprise), seulement quand il a assouvi des besoins secondaires (reconnaissance sociale, par exemple), ou primaires (se nourrir, se loger, etc.). Faire aussi le lien, déjà, entre ces besoins et des métiers possibles ou impossibles (éviter de travailler en indépendant si on a besoin de sécurité, éviter les métiers trop prenants si on a besoin de temps, etc.).

Valeurs : réflexion sur les principes qu'on souhaite mettre en œuvre dans son travail. Cela peut leur sembler plus abstrait, mais insister sur l'importance d'être cohérent avec soi-même, ou sur les rapports entre travail et éthique...

Séquence 4 : Profil et métiers

But : synthétiser tout cela en le mettant en perspective professionnellement. Mettre un peu de légèreté en même temps dans l'exercice : il est trop tôt pour choisir LE métier, mais il faut s'exercer à la projection, s'habituer à se poser ce genre de question. Essayer, si c'est possible, de donner un aspect un peu ludique à cette mise en situation, ce qui n'empêche pas une prise de conscience. Le mieux est sans doute d'amorcer l'exercice, de l'expliquer, et de leur demander de le rendre. Si on leur demande de faire référence à leurs fiches précédentes dans ce document, il peut servir d'évaluation de synthèse à l'ensemble des deux séances. Par exemple : j'ai choisi à court terme d'être garçon de café, car j'ai déjà travaillé dans ce domaine pendant un job d'été et j'ai bien aimé le contact avec les clients (fiche 1) ; je pourrai travailler à mi-temps, car j'ai un logement étudiant qui ne me coûte pas trop cher (fiche 2), et comme je suis organisé et tenace (fiche 3), j'essaierai de continuer mes études par correspondance, car je tiens à réussir et à bien gagner ma vie (fiche 4).

Difficultés : à court terme : essayer de dresser l'éventail des petits métiers sans qualification pour leur donner des pistes, et les inciter à chercher vraiment, parmi ceux-là, celui qui leur semble le plus cohérent par rapport à tout ce qu'ils auront emmagasinné pendant ces séances (ne pas choisir au hasard ou parce que c'est un job facile, et qu'ils cherchent toujours du monde, critères importants mais pas uniques. Voir aussi ce qui serait valorisable dans le métier choisi.

A moyen terme : C'est la question la plus crédible, et ils ne savent pas encore trop ce que la formation en DUT leur apprend. En amorce ou en reprise, on peut leur dire quelques mots de ce que c'est qu'être « technicien en informatique » : ça annonce la séance suivante, surtout si la reprise de cet exercice se fait à cette même séance.

A long terme : Ils peuvent partir sur autre chose que l'informatique, à la limite. L'essentiel est de bien percevoir le chemin à parcourir pour y arriver. On peut glisser un questionnement sur ce que c'est qu'un cursus (universitaire), une carrière (professionnelle), une expérience (humaine) ; sur la mobilité, la nécessité de s'adapter, se mettre à la page.

5.6 Grille métier

LA CONSTITUTION et l'exploration d'une grille métier est le but de la fiche introduite dans ce paragraphe. Ce paragraphe est à usage des étudiants, il est un support pour réaliser une grille métier selon le modèle fourni.

Le secteur d'activité qui vous intéresse : suite aux travaux précédents, quel est le secteur qui vous intéresse pour y travailler ultérieurement ?

Le type de fonction d'informaticien qui vous intéresse : Rappel, on identifie 4 grand types de fonction en informatique :

1. **Développement** : regroupe les activités de création, intégration de logiciels ou matériels informatiques dans l'entreprise
2. **Support** : regroupe les activités de maintenance, d'exploitation et d'aide (support) auprès des utilisateurs
3. **Conseil** et expert : regroupe les activités d'aide auprès des entreprises pour quelles choisissent les outils informatiques les plus adaptées
4. **Gestion** du SI (système d'information) : regroupe les activités de gestion (direction, contrôle des coûts, GRH) de l'informatique

Le type choisi :

Pourquoi :

5.6.1 Grille du métier choisi

Vous utiliserez les annexes qui suivent pour remplir cette grille.

Sa mission : A quoi il sert dans l'entreprise. *A définir en 3 points maximum*

Ses Activités : ce qu'il fait dans son travail de tous les jours. *A définir en 5 points maximum*

Sa formation : (*A définir*)

Son expérience : (*A définir en nombre d'année d'expérience*)

Son évolution possible : (*A définir*)

Ses « savoir faire » principaux : au moins 5 par famille

Techniques

	NR	N	A	M	E

Généraux

	NR	N	A	M	E

Aptitudes

	NR	U	ES

5.6.2 Annexe 1 : Les savoir faire requis des Informaticiens

Il y a 3 familles de « savoir faire » sont définies :

1. Les « savoir-faire **techniques** » sont liés à la technique informatique
2. Les « savoir faire **généraux** » regroupent les compétences en animation, conduite de projet, connaissance de l'entreprise, langue, droit, etc. (UE2 du DUT)
3. Les « savoir faire **aptitudes** comportementales » regroupent les compétences en résolution de problème, compétences managériales, etc.

Pour les 2 premières familles de « savoir faire », on utilise 5 niveaux requis en fonction du métier :

1. **NR** pour non requis
2. **N pour Notion** : connaissance des concepts de base.
3. **A pour Application** : niveau notion + être capable d'appliquer une procédure ou une connaissance sans initiative (sans sortir du cadre défini).
4. **M pour Maîtrise** : niveau application + être capable de traiter les cas d'exception, d'extrapoler, de former aux concepts de base et de sortir du cadre lorsque c'est nécessaire.
5. **E pour Expert** : niveau maîtrise + être capable de modifier significativement une procédure dans un souci d'optimisation.

Pour la famille « aptitude » on utilise :

1. NR : non requis
2. U : utile
3. ES : essentiel

5.6.3 Annexe 2 exemples de compétences par famille

Vous pouvez ajouter de nouvelles compétences, les tableaux ne sont pas complets.

Techniques

	NR	N	A	M	E
Architecture technique Architecture du SI de l'entreprise					
Composants matériels et connectique					
Sécurité informatique Charte d'utilisation et de sécurité des SI					
Télécom - Réseaux Logiciels et matériels réseaux					
Techniques d'installation physique des ordinateurs des réseaux et des télécommunications					
Architecture applicative / fonctionnelle (logiciels, applications métiers)					
Développement, méthodes et normes de paramétrages d'application					
Intégration de logiciels / Intégration de matériels Intégration de systèmes					
Techniques d'installation et de maintenance de systèmes et de réseaux					
Sécurité informatique Droits d'accès aux applications et services.					
Architecture des systèmes d'exploitation					
Algorithmique					
Langages de programmation					
Méthodes, normes et outils de développement					
Base de données					
Normes et procédures de sécurité					
Télécom - Réseaux Logiciels et matériels réseaux					
Environnement d'exploitation					
Module de progiciel (ERP)					

Généraux

	NR	N	A	M	E
Connaissances des métiers de l'entreprise et de leurs besoins informatique					
Culture générale Informatique					
Pratique des principaux logiciels bureautiques					
Vision et compréhension des technologies récentes					
Comptabilité et contrôle de gestion (coût, budget)					
Gestion de stock					
Anglais technique (lu écrit parlé)					
Communication téléphonique					
Capacité rédactionnelle					
Ergonomie et interface homme machine					
Connaissance des processus par métier					
Connaissance de la stratégie de l'entreprise					
Gestion des ressources humaines (GRH)					
Connaissances juridiques					

Aptitudes

	NR	U	ES
Compétences d'assistance et de service Pédagogie			
Compétences de résolution de problèmes Analyse / Diagnostic - modélisation / créativité / perception des enjeux			
Compétences d'efficacité personnelle Gestion de situation			
Compétences d'efficacité personnelle : pragmatisme et rigueur			
Compétences relationnelles Coopération / écoute et communication			
Négociation			

5.7 Grille métier : étude de cas

CE PARAGRAPHE à usage des enseignants est un exemple de grille métier telle qu'un étudiant devrait la construire, ici pour le métier de technicien de support par exemple. Nous avons rajouté des informations générales et des

commentaires qui renvoient au travail qui a été fait dans ce chapitre ou qui renvoie à des documents disponibles par exemple auprès des branches profesionnelles.

Une source principale a été utilisée :

- Nomenclature du CIGREF 2005 : Les emplois et métiers du système d'information dans les grandes entreprises (http://www.cigref.fr/)

Une autre source très intéressante peut être consultée :

- Le site du Syntec : http://www.passinformatique.com/

5.7.1 Les grandes fonctions des métiers de l'informatique

Le référentiel métier du CIGREF présente les « principales missions », « activités et tâches » et « compétences nécessaires » requises pour les principaux acteurs des technologies de l'information dans les entreprises françaises. Il est organisé en six familles.

On peut simplifier ces 6 familles en 4 pôles :

- **Développement** : regroupe les activités de création, intégration de logiciels ou matériels informatiques dans l'entreprise
- **Support** : regroupe les activités de maintenance, d'exploitation et d'aide (support) auprès des utilisateurs
- **Conseil** et expert : regroupe les activités d'aide auprès des entreprises pour quelles choisissent les outils informatiques les plus adaptées
- **Gestion** du SI : regroupe les activités de gestion (direction, contrôle des coûts, GRH) de l'informatique

A la sortie d'un DUT ou d'une licence professionnelle, les 2 pôles visés sont développement et support. Faire du développement signifie travailler sur des domaines techniques qui changent souvent. Faire du support signifie travailler sur des activités techniques ou plus fonctionnelle (liées au métier des utilisateurs) plus récurrentes.

5.7.2 Les secteurs

Le document «Les informaticiens»est un document d'archive qui date de 1995 mais qui est très bien fait pour introduire le champ professionnel et expliquer l'organisation des métiers. Ce document préfigure le site PassInformatique. Sa page de garde est présentée à la Figure ??.

Il est intéressant de s'imprégner du discours de l'époque et de mesurer qu'il reste d'actualité pour certaines de ces parties comme par exemple pour certaines des recommandations aux entreprises et aux partenaires de l'études. Nous les reprenons d'abord les recommandations aux entreprises :

FIGURE 5.2: La synthèse prospective formation emploi éditée à la Documentation Française en 1995

5

1. Définir les grandes orientations en matière de gestion des ressources humaines : les objectifs prioritaires, les mesures qui favoriseront la mobilité des informaticiens ;

2. Développer les initiatives qui permettront aux informaticiens de réaliser un bilan professionnel et de réfléchir à leur projet ;

3. Choisir les actions prioritaires de formation ainsi que leur modalité de mise en œuvre ;

4. Mobiliser les moyens budgétaires sur les actions prioritaires ;

5. Intégrer et reconnaître les savoir-faire acquis au cours de la réalisation d'un projet ;

6. Approfondir les actions possibles pour les informaticiens isolés.

puis celles aux partenaires de l'étude :

1. Sensibiliser les entreprises et les salariés aux évolutions de l'informatique ;

2. Élaborer des parcours de formation qui permettront à des salariés ayant un emploi sensible d'évoluer vers des emplois porteurs ;

3. Proposer des solutions en matière de validation des acquis et de formation diplômante ;

4. Inciter l'ensemble des partenaires économiques à constituer un observatoire permettant d'analyser l'évolution des technologies, des secteurs et des métiers ;

5. Encourager le dialogue social dans l'entreprise et dans la branche professionnelle afin de favoriser l'évolution des mentalités et d'encourager le changement vers les axes prioritaires ;

6. Mettre l'accent sur la culture générale technologique dans les formations initiales afin d'aider à mieux comprendre les évolutions des technologies. Encourager également les formations aux méthodes ;

7. Développer les formations de haut niveau, bac+4 en particulier. Stabiliser, voire diminuer les effectifs de niveaux bac+2 ans.

8. Renforcer les échanges entre le monde de l'éducation et les entreprises, de façon à permettre aux formations et aux méthodes pédagogiques de continuer d'évoluer ;

9. Favosier, à terme, la reconnaissance des diplômes entre pays européens ;

10. Repérer ou bien construire une offre de formation adaptéee, qui réponde aux évolutions technologiques comme à celle des métiers ;

11. Multiplier les passerelles entre formation initiale et formation continue et permettre aux informaticiens d'acquérir des diplômes par la voie de la formation continue ;

12. Encourager le développement de nouveaux types de formation, notamment les formations ouvertes, dont celles à distance.

Dans le reste du document, le lecteur notera également les métiers de l'exploitation qui sont prévus en forte baisse à l'horizon de l'an 2000 (page 21 du document). Qu'est devenu ce métier ? Sur le site PassInformatique, peut être que le métier qui s'y rapproche le plus est technicien de maintenance. Pouvez-vous comparer les descriptions faites sur le site PassInformatique et celle faite à la Figure ?? ?

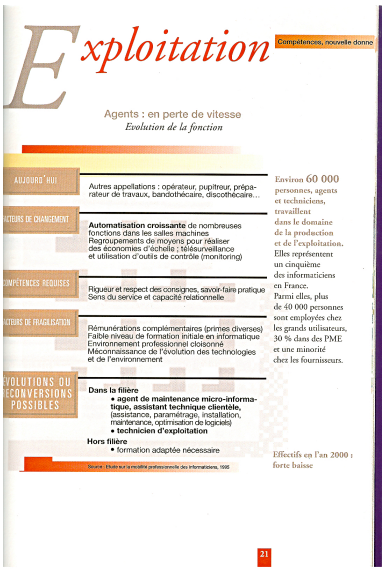


FIGURE 5.3: Le métier de technicien d'exploitation en 1995

Par ailleurs, il y a actuellement (2007) sur le site PassInformatique deux fiches métiers qui comportent le mot clé «exploitation». Lesquelles ? Ces dernières questions mettent en perspective l'idée que les métiers sont en constante évolution.

Un facteur important en informatique qui pousse à l'évolution des métiers est l'externalisation qui peut prendre la forme d'une externalisation dans des pays émergents (outsourcing offshore). L'idée d'externalisation vise à réduire les coûts de production. Nous pouvons par exemple imaginer que le codage de l'application se fait en Inde alors que l'élaboration du cahier des charges pour un client en France s'effectue en France car nous avons besoin d'être au contact du client pour bien comprendre se qu'il souhaite. Dans la pratique, il y a des conditions à remplir pour que l'externalisation soit bénéfique. Il faut s'assurer par exemple que la rotation de

personnels (turn-over) dans le pays émergent ne soit pas pénalisant, ou encore que les coûts de communication et de mise en place d'infrastructures de communication entre les parties ne soient pas prohibitifs.

5.7.3 Les savoir faire requis des Informaticiens

Il y a 3 familles de « savoir faire » qui sont définies comme suit :

- Les « savoir-faire **techniques** » sont liés à la technique informatique
- Les « savoir faire **généraux** » regroupent les compétences en animation, conduite de projet, connaissance de l'entreprise, langue, droit, etc. (UE2 du DUT)
- Les « savoir faire **aptitudes** comportementales » regroupent les compétences en résolution de problème, compétences managériales, etc.

Pour les 2 premières familles de « savoir faire », on utilise 5 niveaux requis en fonction du métier :

- **NR** pour non requis
- **N pour Notion** : connaissance des concepts de base.
- **A pour Application** : niveau notion + être capable d'appliquer une procédure ou une connaissance sans initiative (sans sortir du cadre défini).
- **M pour Maîtrise** : niveau application + être capable de traiter les cas d'exception, d'extrapoler, de former aux concepts de base et de sortir du cadre lorsque c'est nécessaire.
- **E pour Expert** : niveau maîtrise + être capable de modifier significativement une procédure dans un souci d'optimisation.

Pour la famille « aptitude » on utilise :

- NR : non requis
- U : utile
- ES : essentiel

5.7.4 Les fiches métiers

Les fiches ci-dessous sont adaptées (surtout simplifiées) de celles proposées par le Cigref.

Technicien de support (pôle support)

Sa mission :

- Il reçoit les demandes d'intervention des utilisateurs qui ont des problèmes avec l'utilisation des outils (matériels ou logiciels) informatiques.
- Il fait appel à des ressources compétentes pour trouver une solution aux difficultés ou incidents (ruptures du service habituellement rendu) déclarés par les utilisateurs.
- Il contribue, au premier niveau, à la résolution des problèmes de service.
- Il traite tout type d'incidents et n'est pas toujours présent auprès des utilisateurs (il peut travailler à distance).

Ses Activités : accueil des demandes des utilisateurs suite à des dysfonctionnements :

- Prise en compte des appels des clients

Enregistrement des demandes

- Enregistrement des incidents ou anomalies de fonctionnement signalés

Préqualification des dysfonctionnements pour orientation sur les supports ad hoc (technique, fonctionnel)

- Prédiagnostic et transfert des appels des clients aux entités compétentes

Traitement ou déclenchement des actions de support correspondantes

- Traitement de 1er niveau des incidents ou anomalies sur les postes de travail, signalés en interne ou par le SVP : diagnostic, identification, formulation, résolution

Suivi des incidents

- Suivi du traitement des appels des clients
- Exploitation de la base d'incidents : relances, consolidation, analyse de tendance
- Demandes d'actions préventives de fond

Information des clients

- Alerte, information du client

Sa formation : Bac + 2.
Son expérience : Premier poste.
Ses « savoir faire » principaux :
Techniques

	NR	N	A	M	E
Architecture technique Architecture du SI de l'entreprise		X			
Composants matériels Compétences relatives aux différents matériels Connectique			X		
Sécurité informatique Charte d'utilisation et de sécurité des SI				X	
Télécom - Réseaux Logiciels et matériels réseaux			X		
Techniques d'installation physique des éléments actifs des réseaux et des télécommunications			X		

Généraux

	NR	N	A	M	E
Connaissances des métiers de l'entreprise			X		
Culture générale Informatique			X		
Communication téléphonique				X	
Pratique du poste de travail informatique des utilisateurs (de ses différentes configurations existantes) et des principaux logiciels bureautiques					X

Aptitudes

	NR	U	ES
Compétences d'assistance et de service Pédagogie			X
Compétences de résolution de problèmes Analyse Diagnostic - modélisation Ouverture d'esprit Synthèse Perception des enjeux		X	
Compétences d'efficacité personnelle Gestion de situation			X
Compétences relationnelles Coopération / écoute et communication		X	

5.7.5 Technicien poste de travail (Pôle support)

Sa mission : dans le cadre de projets informatique, il assure l'installation et la garantie de fonctionnement des équipements informatiques et téléphoniques (matériels et logiciels). À la demande des utilisateurs, il assure la maintenance (à distance ou sur site) de ces équipements et traite les incidents.

Ses Activités :

Installation, tests et recettes

- Installation initiale des équipements informatiques et téléphoniques (applications, matériel micro, matériels de téléphonie et péritéléphonie) ;
- Installation des mises à jour ;
- Tests et recette utilisateurs des équipements informatiques et téléphoniques ;

Maintenance, administration et sécurité

- Détection et limitation des virus ;
- Suivi de l'évolution de l'équipement ;
- Administration de la messagerie (connexion, exploitation, facturation) ;
- Contrôle de la conformité des équipements avec les référentiels
- Exploitation et administration du réseau téléphonique, de la messagerie vocale, de la vidéotransmission ;
- Suivi et gestion des stocks de fournitures pour le téléphone (postes téléphoniques, cartes Minitel...) ;

Exploitation

- Traitement des incidents à distance sur micros, réseaux, messagerie ou téléphonie ;
- Diagnostic et traitement des incidents ;
- Gestion du parc informatique connecté au réseau ;

Support

- Aide à la prise en mains des équipements installés ;

Sa formation : Bac + 2.
Son expérience : Premier poste.

Ses « savoir faire » principaux :

Techniques

	NR	N	A	M	E
Architecture applicative / fonctionnelle Architecture fonctionnelle du SI de l'entreprise (logiciels, applications métiers			X		
Architecture technique Architecture du SI de l'entreprise			X		
Composants matériels Compétences relatives aux différents matériels Connectique				X	
Développement Méthodes et normes de paramétrages d'application		X			
Intégration de logiciels / Intégration de matériels Intégration de systèmes		X			
Production - Exploitation Gestion de production (optimisation des capacités) Techniques d'installation et de maintenance de systèmes et de réseaux			X		
Sécurité informatique Droits d'accès aux applications et services Normes et procédures de sécurité Informatique				X	
Télécom - Réseaux Logiciels et matériels réseaux		X			

Généraux

	NR	N	A	M	E
Compréhension de l'environnement et du fonctionnement de l'entreprise Connaissance des processus et des applications en place Connaissance de l'offre Vision et compréhension des technologies récentes			X		
Compréhension des clients de la DSI (utilisateurs fonctionnels) et de leurs besoins Connaissance de l'entreprise et de son informatique Culture générale informatique			X		
Gestion de stock			X		
Anglais technique (lu écrit parlé)			X		

Communication téléphonique			X		
Pratique du poste de travail informatique des utilisateurs (de ses différentes configurations existantes) et des principaux logiciels bureautiques					X

Aptitudes

	NR	U	ES
Compétences d'assistance et de service Pédagogie			X
Compétences de résolution de problèmes Analyse Diagnostic - modélisation Ouverture d'esprit Synthèse Perception des enjeux		X	
Compétences d'efficacité personnelle : pragmatisme et rigueur			X
Compétences relationnelles Coopération / écoute et communication			X
Négociation		X	

5.7.6 Développeur (pôle développement)

Sa mission : à la demande de la maîtrise d'œuvre (regroupe les informaticiens ici des analystes), et sur la base des spécifications fonctionnelles (définition de ce que doit faire le programme) émises par celle-ci, le développeur analyse, paramètre et code les composants logiciels applicatifs dans le respect des normes et procédures, ainsi que les évolutions souhaitées.

Ses Activités :

Analyse

- Définition de spécifications ;
- Analyse organique ;
- Adaptation et paramétrage de progiciels applicatifs ;
- Prototypage ;

Développement

- Réalisation de modules (objets et composants logiciels) ;
- Assemblage de ces éléments ;

- Rédaction de documentations ;
- Industrialisation de composants et d'applications ;

Qualification

- Élaboration de jeux d'essais (tests unitaires d'intégration) ;
- Tests ;
- Identification et traitement des dysfonctionnements ;

Maintenance

- Maintenance corrective ;
- Maintenance évolutive ;
- Administration des composants logiciels réutilisables et gestion de la nomenclature de ces composants ;

Sa formation : Bac + 2 ou 3.
Son expérience : Premier poste.

Remarque sur les évolutions actuelles

- usage croissant des progiciels, d'où importance croissante du paramétrage, de l'objet, du fonctionnel aux dépens du développement spécifique, de l'algorithmique ;
- renouvellement rapide des langages : java, langages objet ;
- importance croissante de l'ergonomie ;
- durée de vie des applications raccourcie ;
- souci de réutilisation des développements.

Ses « savoir faire » principaux

Techniques

	NR	N	A	M	E
Architecture applicative de l'entreprise (logiciels, applications métiers			X		
Architecture des systèmes d'exploitation		N			
Algorithmique				X	
Langages de programmation					X

Méthodes, normes et outils de développement					X
Base de données			X		
Normes et procédures de sécurité			X		
Télécom - Réseaux Logiciels et matériels réseaux		X			

Généraux

	NR	N	A	M	E
Compréhension des clients de la DSI (utilisateurs fonctionnels) et de leurs besoins Culture générale informatique			X		
Anglais technique (lu écrit parlé)		X			
Capacité rédactionnelle			X		
Ergonomie et interface homme machine				X	

Aptitudes

	NR	U	ES
Compétences de résolution de problèmes Analyse / méthode / logique			X
Compétences d'efficacité personnelle : adaptabilité, pragmatisme et rigueur		X	
Compétences relationnelles écoute / travail en équipe		X	
Négociation		X	

5.7.7 Paramétreur d'ERP (pôle développement)

Sa mission : à la demande de la maîtrise d'oeuvre ou de la maîtrise d'ouvrage (les utilisateurs), et sur la base des spécifications fonctionnelles émises par les chefs de projets métier ou le responsable de projets métiers, le paramétreur d'ERP analyse, prototype et paramètre les nouveaux composants progiciels applicatifs dans le respect des normes et procédures, ainsi que les évolutions souhaitées sur les composants.

Il assiste et apporte sa maîtrise sur le module dont il a l'expertise et les processus associés. Son expertise porte sur l'un des modules fonctionnels composant l'ERP (Enterprise Ressource Planning ou progiciel de gestion intégré en français).

Ses Activités :

Analyse

- Prototypage en collaboration avec l'expert fonctionnel et l'administrateur de données ;
- Justification des écarts entre le besoin et le standard de l'ERP ;
- Analyse organique des besoins en interfaces avec les produits environnants dans le système d'information de l'entreprise (bases de données, applications. . .)

Développement

- Adaptation et paramétrage des programmes applicatifs de l'ERP ;
- Modélisation et paramétrage des processus selon la méthodologie propre à l'ERP choisi ;
- Documentation (BPR – ré-ingénierie des processus -, manuel d'utilisation. . .) ;
- Participation à la réalisation des interfaces ;
- Participation à la réalisation des supports de formation des utilisateurs.

Qualification

- Élaboration de jeux d'essais (tests unitaires et d'intégration) ;
- Tests des développements internes et des solutions fournies par les éditeurs ;
- Identification et traitement des dysfonctionnements constatés aux tests.

Support et maintenance

- Assistance aux analystes support ;
- Maintenance corrective et évolutive à l'aide des outils et de l'éditeur ;
- Traçabilité en base de connaissances des solutions apportées.

Sa formation : Bac + 2 ou 3.
Son expérience : première expérience en développement ou sur un projet de paramétrage.
La certification d'un éditeur d'ERP est très demandée
Remarque sur les évolutions actuelles :

- usage croissant des progiciels, d'où importance croissante du paramétrage, de l'objet, du fonctionnel aux dépens du développement spécifique, de l'algorithmique ;
- très demandé et peu évoluer vers du conseil ;
- Il peut être difficile de changer de métiers dans l'informatique après une telle expérience ;

Ses « savoir faire » principaux :
Techniques

	NR	N	A	M	E
Architecture applicative de l'entreprise (logiciels, applications métiers et progiciels			X		
Langages de programmation					X
Méthodes, normes et outils de développement					X
Environnement d'exploitation		X			
Module ERP					X
Base de données			X		
Normes et procédures de sécurité		X			
Télécom - Réseaux Logiciels et matériels réseaux		X			

Généraux

	NR	N	A	M	E
Compréhension des clients de la DSI (utilisateurs fonctionnels) et de leurs besoins Culture générale informatique				X	
Connaissances de l'entreprises – processus				X	
Anglais technique (lu écrit parlé)				X	
Capacité rédactionnelle			X		
Ergonomie et interface homme machine				X	

Aptitudes

	NR	U	ES
Compétences de résolution de problèmes Analyse / méthode / logique			X
Compétences d'efficacité personnelle : adaptabilité, pragmatisme et rigueur		X	
Compétences relationnelles écoute / travail en équipe		X	
Négociation		X	

5.8 Réalisation d'une interview

NOUS NOUS CONCENTRONS maintenant sur une activité un peu indépendente de ce que l'on vient de traiter dans les paragraphes précédents. Il

s'agit toujours de travailler sur le plan des métiers en réalisant et en exploitant l'interview d'un professionnel mais l'essence du travail n'est plus vraiment le remplissage de grilles d'analyse. Nous visons plutôt la construction proprement dite d'une grille. L'interview d'un professionnel a un triple objectif :

1. vous permettre de *mieux connaître un métier*, et plus généralement un secteur d'activité dans lequel vous pourriez être amené à travailler plus tard ;
2. vous faire envisager ce métier sous un *angle pratique*, grâce à quelqu'un qui en a l'expérience ;
3. vous faire réfléchir aux questions que vous devrez vous-même vous poser lors de *votre propre démarche professionnelle*.

Ce travail peut être mené seul ou à deux (attention, la composition des binômes doit résulter d'une communauté d'intérêt pour la personne rencontrée). Il sera évalué à travers une présentation des résultats et la réalisation d'un poster selon des modalités qui sont communiquées ci après. Le respect du calendrier ci-dessous sera pris en compte.

4 octobre	Présentation de la démarche
3 novembre	Date limite pour s'informer, définir le projet et fournir la liste des contacts
30 novembre	Date limite pour réaliser l'interview
décembre	Mise en forme de la restitution
5 janvier	Restitution (évaluation)

5.8.1 Pistes de recherche

Après avoir rempli la grille métier, et avoir fait mon bilan personnel, j'essaie de définir un ou plusieurs métiers sur lesquels j'aimerais me renseigner davantage. Je justifie ce(s) choix.

Métiers choisis	Justifications

5.8.2 Où chercher ?

Voici quelques pistes qui pourront vous aider à trouver des contacts :

- pensez aux étudiants qui font le DUT informatique par alternance : les $2^{ème}$ année, en particulier, travaillent tous dans une entreprise, et peuvent vous aider à trouver des contacts ;
- essayez de contacter d'anciens étudiants de l'IUT, ou de formations informa-

tiques autres (en Licence pro, à l'Institut Galilée, etc.) ;

- pensez à la formation continue : elle concerne des personnes ayant déjà une expérience professionnelle. Prendre contact avec la responsable administrative de la formation continue. Voir aussi la personne responsable de l'alternance.
- des salons ont lieu régulièrement ;
- vous pouvez aussi chercher des contacts sur internet, à travers petites annonces et sites d'entreprises, ou dans la presse spécialisée ;
- vous pouvez vous rapprocher de votre mairie (service économique), les chambres de commerce, l'APEC (Association pour l'emploi de cadres), l'ANPE (Agence Nationale Pour l'Emploi) ;
- les relations (de vos proches).

5.8.3 Mes contacts

Pour mener vos recherches, vous devez repérer plusieurs entreprises où il vous semble possible de rencontrer une personne ayant le profil recherché : listez ci-dessous les contacts que vous comptez prendre (vous complèterez la dernière case à mesure que vous mènerez votre recherche).

Métier concerné	Nom et spécialité de l'entreprise	Coordonnées personne ou service à contacter	Résultat du contact

5.8.4 Retour sur l'interview

Il s'agit de réaliser l'interview d'un professionnel de votre choix, lié à votre futur secteur d'activité, afin de recueillir auprès de lui des informations sur le métier qu'il exerce, la nature exacte de ses tâches et sa vision du secteur concerné. Le but est de compléter de manière concrète la recherche personnelle que vous avez commencée en TD : votre auto-analyse, votre bilan d'aptitudes, votre travail de projection dans l'avenir professionnel, la fiche métier que vous avez réalisée forment les étapes préalables à cet entretien. Cette enquête peut être réalisée seul ou en groupe.

Les bénéfices attendus	Les erreurs à éviter

• Vous aider à vous projeter dans l'avenir et à mieux connaître votre futur secteur d'activité, les métiers et postes qui s'offriront à vous pour construire votre projet professionnel • Vous faire connaître le monde de l'entreprise et en particulier une société utilisant certaines des compétences que vous allez acquérir au cours de votre formation. • Vous permettre de communiquer avec un interlocuteur professionnel. • Vous créer un point de vue original sur une entreprise ou un métier, et vous permettre des contacts que vous pourrez utiliser pour vos futures recherches d'emploi et de stage. • Vous préparer à présenter une activité professionnelle et un secteur d'activité (en vue du rapport de stage) • Vous apprendre à conduire un projet en établissant et en respectant un cahier des charges ; • Vous apprendre (éventuellement) à travailler dans un groupe comme c'est le cas dans la vie professionnelle	*Dans la préparation, l'organisation* : • Tarder à chercher un professionnel • Mal organiser le partage des tâches • Mal planifier le travail • Ne pas respecter les délais *Dans le contenu* • Contenu trop riche, informations mal triées • Contenu trop pauvre ou impersonnel - sur l'entreprise (copier/coller), - sur le service, les collègues (titres, formation, fonction, liens avec l'interlocuteur), - sur le travail exact de votre interlocuteur, la méthode suivie dans le travail, le matériel, etc. *Dans la présentation orale* : • Problèmes de clarté et d'organisation logique (plan) • Problèmes d'expression (incorrections, oral pas assez répété, mal

Réunissez votre groupe le plus vite possible et recherchez le plus tôt possible votre interlocuteur pour prendre un rendez-vous.

Comment réaliser l'interview ?

Prendre contact avec un éventuel interlocuteur ne se fait pas à la légère. Il faut d'abord avoir choisi judicieusement la personne à interroger : n'oubliez pas que l'objectif est de **vous** informer, et que vous aurez à justifier votre choix (profil métier, secteur, type d'entreprise, etc.).

Il faut ensuite se renseigner sur l'entreprise et la profession de l'interlocuteur choisi : il ne doit pas avoir l'impression que vous l'avez choisi au hasard, mais que vous savez ce que vous recherchez, et que vous êtes motivé par une meilleure connaissance des particularités propres à son environnement professionnel.

Il faut enfin réfléchir à la façon dont vous allez lui présenter les choses. Quel que soit le mode de prise de contact que vous choisirez (mail, téléphone ou contact direct), vous allez devoir lui expliquer brièvement ce que vous attendez de lui. Cela suppose de savoir vous présenter (vous, votre IUT, votre discipline), de pouvoir expliquer le projet (le cadre général du PPP et les objectifs précis de cette interview), et de motiver votre demande (pourquoi lui, pourquoi cette entreprise ?).

Bien sûr la politesse et la conscience des impératifs professionnels sont nécessaires (il aura peut-être peu de temps à vous accorder, et pas exactement sur les créneaux que vous auriez souhaité : à vous de faire preuve de souplesse). N'oubliez pas que vous êtes en position de demandeur : à tout moment, votre interlocuteur peut décider de ne pas poursuivre...

La prise de contact n'est pas toujours facile : il faudra sans doute passer plusieurs coups de fil, mais même si vous essuyez quelques échecs, persévérez. Gardez trace des démarches effectuées (nom de l'entreprise, personne contactée, etc.), et allez voir votre enseignant de TD si vous avez besoin de conseils ou d'aide matérielle.

Poser les bonnes questions : il y a plusieurs façons de conduire un entretien :

L'ENTRETIEN NON- DIRECTIF	L'ENTRETIEN SEMI DIRECTIF	L'ENTRETIEN DIRECTIF

L'objectif est de permettre à l'interviewé de s'exprimer librement, avec une certaine liberté, sur les thèmes que lui-même a choisis. L'intervie-weur est dans une attitude d'écoute totale et n'intervient que pour relancer l'autre.	L'intervieweur est dans une grande attitude d'écoute mais intervient pour introduire certains thèmes ou demander qu'ils soient approfondis.	C'est l'intervieweur qui mène l'entretien en fonction de ses objectifs, de sa curiosité, de son système de valeurs : il pose les questions qu'il choisit en fonction de sa curiosité et de ses objectifs.

L'interview d'un professionnel est un mélange de directif et semi-directif. L'objectif de cet entretien est de **développer une véritable attitude d'écoute** qui va au delà de la politesse : il s'agit de tout faire pour permettre à l'autre de s'exprimer Cela suppose notamment de bien utiliser toute la palette des questions :

LES QUESTIONS FERMEES : Elles appellent des réponses assez brèves.	Questions informatives : demandent une précision, une identification (qui, où quand, combien etc.) Questions alternatives : réponse par OUI ou NON Questions à choix multiples : elles délimitent et canalisent la réponse (vous allez en vacances à la mer, à la montagne, à l'étranger ?)
LES QUESTIONS OUVERTES	Elles donnent à réfléchir, forcent à mobiliser ses idées, à se découvrir. (Quel est votre projet dans ce domaine ? Quelle est au fond votre opinion ?) On ne les pose qu'en cours d'entretien, quand la confiance est installée
LES QUESTIONS RELAIS	Elles permettent d'approfondir ou de détailler les réponses pour guider et soutenir l'entretien.

LES QUESTIONS MIROIRS	Elles retournent au locuteur une phrase, une idée, un mot pour mettre en évidence une erreur, un préjugé. Elles permettent à l'interviewé de revenir sur ce qu'il a dit sans entrer dans une polémique. (« - C'était l'unique problème. - L'unique problème ? »)
LES QUESTIONS BILAN	Elles permettent de synthétiser, de reformuler ce qui a été dit et compris. (« Comment donc résumeriez-vous cette expérience ?)

Avoir préparé les questions

Vous trouverez ci-joint quelques pistes pour mener l'entretien : **à vous de personnaliser et de compléter ces questions** afin de vous adapter au mieux au profil de votre interlocuteur et à vos propres interrogations.

Regroupez vos questions par thème, mais pensez aussi à les classer par ordre d'importance : si votre temps d'entretien est limité, mieux vaut poser d'abord les questions les plus importantes, puis revenir sur les points secondaires si on a le temps.

Et, sur le moment, n'hésitez pas à improviser si de nouveaux thèmes intéressants sont abordés par votre interlocuteur (mais attention au hors-sujet).

5.8.5 Préparer la restitution

Le but de l'interview est de **garder trace de la discussion**, et de **faire partager le contenu informatif aux autres étudiants du groupe**.

- Cela suppose sur le moment de **prendre des notes**, afin de mémoriser les réponses de l'interlocuteur. Si vous êtes plusieurs, l'un de vous peut se charger de ce travail. Si vous êtes seul, prenez le temps d'écrire, sinon, vous risquez d'oublier beaucoup d'éléments importants. Dès que l'entretien est fini, vous devez relire ces notes, afin de les clarifier et de les compléter à chaud, si nécessaire : c'est votre base de travail, elle doit être claire et utilisable. Si vous avez travaillé à plusieurs, faites une photocopie, afin que chacun en ait un exemplaire.

- Vous pouvez si vous en avez la possibilité, **enregistrer** votre interlocuteur (demandez son autorisation). Cela vous permettra notamment de pouvoir le citer fidèlement. Attention toutefois : cela ne dispense pas de prendre des notes, car un

incident technique est vite arrivé, et car il est très long et fastidieux, ensuite, de réécouter toute une bande pour la synthétiser.

- Au TD du 30 novembre, **n'oubliez pas d'apporter ces notes**, nous travaillerons dessus pour les mettre en forme et préparer le poster correspondant.

- Chez vous, commencez à réfléchir à ce que vous voulez dire à l'oral, et à ce qui pourrait être mis en valeur dans le poster. N'oubliez pas que vous devez **synthétiser** ce que vous avez appris. **Critères à prendre en compte :**

- Intérêt des thèmes abordés, pour vous, et/ou pour les autres étudiants ;

- Pertinence des réponses de votre interlocuteur (qu'elles confirment vos suppositions ou qu'au contraire elles remettent en cause les idées reçues) ;

- Aspect concret, précis, des informations récoltées ;

- Bilan personnalisé de l'entretien : que vous a-t-il apporté ?

Remarque : Il est possible que votre interlocuteur demande à relire la retranscription de l'entretien ou la synthèse que vous en ferez, et s'autorise éventuellement à y apporter des rectifications. A vous de voir ce que vous êtes prêts à accepter : d'un côté, cela suppose un effort supplémentaire de rédaction et de respect des délais ; de l'autre, cela peut permettre de poursuivr l'échange, et c'est parfois une condition à la réalisation de l'entretien. La présence de votre interlocuteur le jour de la restitution est aussi envisageable, s'il le souhaite. Ne vous laissez pas déstabiliser en tout cas : vous pouvez toujours dire, si vous h ésitez : « il faut que j'en parle à mon responsable ».

Quelques pistes pour les questions :

Votre interlocuteur : son parcours, son insertion dans l'entreprise

- Quelle est votre formation ?
- Avez-vous déjà changé de travail ? Combien de fois ? Pour quelles raisons ?
- Depuis combien de temps êtes-vous dans la vie professionnelle ? Dans cette société ?
- Selon quelle procédure avez-vous été recruté ?
- Quel était le profil de carrière qui vous était présenté ? S'est-il révélé conforme ?
- Qu'est-ce qui a été déterminant dans votre acceptation (salaire, nature du travail, profil de carrière, réputation de l'entreprise, mobilité, etc.) ? Quel bilan feriez-vous aujourd'hui ?
- Quelles ont été vos premières impressions en entrant dans cette entreprise?
- Quel regard jetez-vous aujourd'hui sur ce choix ? Pensez-vous qu'il serait encore possible pour quelqu'un qui démarrerait maintenant ?

Nature du travail

- Travaillez-vous plutôt dans la recherche, la production, le technico-commercial ?
- Quelle est la dénomination exacte de votre fonction ?
- En quoi consiste précisément votre fonction ? (Vous devrez en savoir assez pour l'expliquer)
- Pouvez-vous décrire vos tâches quotidiennes en quelques points ?

• Avez-vous eu à modifier le contenu de votre travail depuis votre entrée en fonction ?

• Etes-vous seul dans votre travail ou avez-vous la responsabilité d'un groupe de personnes ? d'un service ? Combien de personnes constituent ce service ?

• Quelles sont les fonctions et les objectifs de votre service, son mode de fonctionnement interne et ses relations avec les autres services ?

• Comment se prennent les décisions d'orientation ? Y participez-vous ?

• Votre formation était-elle bien adaptée au métier que vous exercez ? Quels sont les aspects de cette formation qui vous ont été les plus directement utiles ? Y a-t-il des lacunes qui vous ont paru préjudiciables ?

• Vos stages étaient-il en rapport avec votre profession actuelle ?

• Le métier que vous exercez aujourd'hui correspond-il à celui que vous envisagiez lorsque vous étiez étudiant ?

• A-t-il été nécessaire pour vous de suivre une formation complémentaire (à l'intérieur ou à l'extérieur de l'entreprise) ?

• Quelles sont les aptitudes qui vous semblent nécessaires pour exercer ce type de travail ? Quels besoins satisfait-il ? Quelles valeurs s'y rattachent ? (voir les fiches sur ces thèmes).

Carte d'identité de l'entreprise

• Présentation de la société (brièvement, présentez ceux de ces éléments qui vous semblent utiles : chiffre d'affaire annuel, capital, nombre de salariés, filiale, organigramme général de la société, historique, secteurs d'activité)

• Y a-t-il un comité d'entreprise ? Quelle est l'importance de la vie syndicale ?

• Quels sont les horaires ? les rythmes de travail ?

• Y a-t-il dans votre société des possibilités pour les employés d'influencer l'organisation du travail ?

5.9 Restitution de l'interview

NOUS PASSONS maintenant à la valorisation de l'interview. Cette phase est en deux parties. Premièrement il y a un travail préparatoire lié à l'organisation de la présentation orale (la restitution s'effectue en effet par le biais d'un exposé oral), deuxièmement il est demandé de réaliser un poster synthétisant l'interview. Des consignes pour régiger un poster sont donc passés dans le deuxième sous-paragraphe.

5.9.1 Organisation des séances de travail

- séance 1 : consignes et préparation à la restitution ;
- séance 2 : préparation du poster qui sera présenté en TD ;
- séance 3 : suite des conseils et validation des posters ;
- séance 4 : préparation de l'oral et finalisation du poster

- séance 5 : présentation orale en TD, devant les $2^{ème}$ année.

La présentation dure 10 minutes, que vous soyez seul ou en groupe (si vous êtes plusieurs, on attend un effort qualitatif supplémentaire, sur le contenu et les documents visuels).

Elle doit être organisée selon un plan clair et efficace. Vous avez le choix – quelques exemples : *plan thématique* (différentes rubriques correspondant aux différentes dégagées par l'entretien), *plan partant du général* (l'informatique) *pour aller vers le particulier* (ce métier dans cette entreprise, dans ce service, vu par cet individu), ou *allant des faits vers leur analyse*, ou encore mettant en valeur les avantages et les inconvénients de ce métier...N'oubliez pas de faire une place à votre opinion personnelle, en conclusion, et/ou par touches successives dans le développement.

Chaque exposé sera synthétisé sur un poster soigné et bien utilisé, sur lequel s'appuiera la présentation.

Chaque intervention sera suivie de questions et d'une discussion avec les étudiants présents : n'hésitez pas à lancer vous-même le débat.

N'oubliez pas vos objectifs : récolter et transmettre des informations ; Vous devez montrer votre capacité à :

- *aller chercher l'information* (pendant l'interview, poser les bonnes questions, être curieux ; avant et après : s'informer pour préparer et compléter l'entretien),
- *la synthétiser* (trier, hiérarchiser, organiser les informations récoltées),
- *la transmettre* (souci de clarté et d'utilité, mise en forme agréable et efficace).

N'oubliez pas vos objectifs : montrer une réflexion personnelle.

Il s'agit pour vous de faire le bilan de ce que cet entretien vous a apporté. C'est une démarche personnelle qui doit vous amener à vous poser des questions sur le métier que vous souhaitez exercer plus tard. N'hésitez pas à souligner ce qui vous a étonné, heureusement surpris ou déçu. La rencontre a-t-elle été enrichissante ou pas ? Pourquoi ? Voyez comment vos *a priori* sont confirmés ou contredits, montrez en quoi telle information vous pousse à porter un autre regard sur le métier, le secteur informatique ou votre avenir. Faites la part du pour et du contre. De manière concrète, pensez aussi à la façon dont vous pourriez orienter vos études ou vos recherches de jobs, de stages...

N'oubliez pas vos objectifs : rendre compte d'une rencontre.

Vous avez rencontré une personne particulière : on s'attend aussi à ce que votre restitution soit personnalisée : essayez de cerner les spécificités de son métier, de son entreprise, mais aussi insistez sur son parcours individuel, sa vision des choses, les conseils qu'elle a pu vous donner, faites-nous partager l'ambiance du rendez-vous. Cette personne vous a consacré du temps : demandez-vous quel enjeu cette rencontre peut avoir pour vous et pour elle.

5.9.2 Construire un poster

Cette fiche de travail vous indique comment rédiger un poster. L'objectif concret est le suivant dans le cadre du PPP : vous avez rencontré un « professionnel de la profession » pour parler de son métier et vous voulez en rendre compte. La fiche est divisée en deux parties qui introduisent les qualités de fond et de forme que peut prendre un poster. Le rédaction d'un poster pose la difficulté majeure de synthétiser l'ensemble du travail, dont la démarche, dans une forme encore plus concise par rapport à l'exposé oral. La forme ou les formes peuvent être un graphique, un texte mêlée à un dessin... Le poster est au bout de la chaîne de communication quand on regarde un travail sous la forme d'un article (un compte rendu de projet), puis d'un exposé oral puis du poster... et il y a de moins en moins de caractères graphiques quand on passe du premier au dernier !

Nous allons exposer comment le rédiger en étant aussi général que possible (pour pouvoir réutiliser les idées dans un autre contexte) et en tentant de l'isoler des deux autres formes bien que les conseils restent valables bien souvent pour la rédaction d'un compte rendu ou d'un exposé oral. Certains éléments de cette fiche ont déjà été plus ou moins présentés dans le chapitre lié à la typographie. Des exemples terminent cette fiche.

Les qualités de fond du poster

Généralement, le poster doit reprendre les points traités dans l'article (le compte rendu). Avant de rédiger le poster, dans votre cas, vous devez avoir en tête ou sur papier l'ensemble des idées que vous vous êtes faites à l'issue de l'entretien : cela vous servira pour la présentation du poster. L'objectif principal du poster est de faire comprendre à des non-spécialistes (et pas seulement à nous les enseignants du PPP) l'objet du travail et de dégager clairement l'intérêt des travaux réalisés. Il aborde succinctement l'aspect historique (ou scientifique selon le cas) et méthodologique pour préférer la problématique des travaux.

La session pendant laquelle vous allez présenter le poster doit permettre la discussion entre vous et votre auditoire : cette session ne doit pas être un long monologue. A la limite elle doit être interactive. Les personnes présentant un poster doivent donc perdre le moins de temps possible en explications savantes sans fin. Par conséquent, le poster doit contenir toutes les informations principales et être compris sans explication ou presque.

Travail préparatoire : avant de commencer la rédaction, il faut identifier le niveau de votre auditoire. Dans notre cas, visez un auditoire qui n'a jamais entendu parlé de votre problème et du PPP. Éventuellement, lire les instructions données par les organisateurs (dimension du support,...). Ensuite, pensez à faire une phrase de 25 mots définissant le comment, le pourquoi, et le message principal du poster. Ce sera le FIL CONDUCTEUR qu'il faudra garder à l'esprit tout au long de la préparation.

Pour faciliter la compréhension, le poster doit être linéaire c'est à dire commencer d'un point de départ A bien défini et aller vers une arrivée Z également

bien définie. Le plus simple c'est de faire que A corresponde sur le poster au mot clef Introduction et que Z corresponde au mot clef Conclusion. Vous pouvez faire des variantes du genre : A= « La commande » ; Z= « Les apports du travail ».

La structure du poster doit être en colonnes : lorsque plusieurs personnes se trouvent en même temps devant le poster, cela permet un mouvement général par translation de la gauche vers la droite sans gêne, sans zigzags et sans croisements. Pensez y !

L'organisation générale du poster peut suivre l'architecture suivante mais cela vous conduit à avoir 5 à 6 pages au format A4 (au moins 2 ou 3 pages pour le point 3 ci dessous) :

TITRE : il est énorme et tient sur toute la zone supérieure. Il doit évoquer les 25 mots de votre fil conducteur. Il comprend les noms et prénom des auteurs ainsi que leurs adresses (courriel par exemple). Le titre doit être accrocheur et évocateur mais pas racoleur !

INTRODUCTION : bien replacer l'étude dans le CONTEXTE et identifier clairement l'OBJECTIF en mettant un titre «OBJECTIF».

METHODES ET RESULTATS : les METHODES doivent être abrégées au maximum sauf si l'objectif est d'améliorer une technique (mais c'est pas votre cas ici). Les RESULTATS doivent suivre la présentation de la méthode correspondante. Cette partie «Méthodes et Résultats» doit représenter pas loin des 2/3 du poster. La seule lecture des titres doit permettre de comprendre les résultats. Par ailleurs, veuillez numéroter les figures (s'il y en a) ce qui renforce l'ordre de lecture.

CONCLUSION : elle est en bas à droite: identifier clairement la CONCLUSION avec un titre «CONCLUSION» puis éventuellement les PERSPECTIVES ET/OU APPLICATIONS et/ou APPORTS PERSONNELS selon le cas.

REMERCIEMENTS : lister les personnes avec qui vous avez interagit, le cas échéant vos financiers ainsi que quelques sites Web de référence ou pour trouver plus d'information sur votre travail.

Remarque : si vous utilisez un format beaucoup plus grand que le A4, par exemple un seul feuillet, tachez de répartir les points précédents en les « enchaînant » par un lien graphique (flèche, pictogramme, numérotation) le plus approprié possible, par exemple en le liant à votre contexte. Dans votre cas, vous sortez de l'université pour aller faire une interview dans une entreprise puis vous faites un brainstorming collectif puis etc etc. Dans ce cas, utiliser un pictogramme « université », un pictogramme « entreprise » et enfin un pictogramme « tête qui fume ». Ne le faite pas, il fallait avoir l'idée avant !

Le poster doit être simple ce qui veut dire qu'il contient un minimum d'éléments inutiles. Le but d'un support visuel est d'être utile, de venir en appui à un discours et non artistique, car le lecteur doit retenir le contenu (les messages que vous passez) et non le contenant.

Il ne faut pas essayer de transmettre trop d'informations en même temps. Il vaut mieux faire passer quelques informations bien expliquées que des dizaines d'informations qui ne seront pas retenues. La quantité n'impressionnera pas l'auditoire. Un petit « truc » un peu inattendu, personnel c'est beaucoup mieux !

En résumé, soyez humble. À lire votre poster ou a vous écouter on ne doit pas en conclure « comme il a bien travaillé, comme c'est beau... mais j'ai rien compris ».

Enfin, il faut préparer le commentaire du poster. Si possible faites une présentation vivante, soyez souriant, enthousiaste. Ne ronronnez pas. Utiliser des phrases courtes, des mots simples, la voix active et non la voix passive. Ne donnez pas l'impression que vous récitez un discours par coeur. Éviter également les «euh» et les tics divers, un discours haché. Ne mettez pas les mains dans les poches ; parlez en étant debout face à votre auditoire.

Poser une question et y répondre est un mode de présentation très dynamique. En toute fin de votre présentation donnez quelques mots sur ce que ce travail vous a apporté en faisant du lien avec les éléments de votre conclusion. Ce sera un plus ! Soignez bien cette dernière partie : la mise en perspective du projet entier x'est à dire ce que vous avez pu tirer comme enseignements de ce projet.

Faire des pauses après les points importants, en particulier après l'exposition de l'objectif.

Choisir les mots avec précaution et les utiliser de façon appropriée : pas de jargon technique, n'est-ce pas les informaticiens ?

Et enfin, faites plusieurs répétitions.

Les qualités de forme du poster

L'aspect esthétique ne doit pas être négligé : les degrés d'attraction, de lisibilité et de clarté seront des critères d'appréciation de votre poster. Attention, les effets artistiques ne remplacent pas un bon contenu et de plus prennent beaucoup de temps de préparation. C'est même une affaire de spécialistes : préférez une présentation sobre aux effets contre productifs des surcharges visuelles.

Chaque étape du travail doit être clairement identifiée par des signaux visuels. L'introduction et la conclusion seront sur un fond d'une couleur et les éléments intermédiaires dans une autre couleur. Utiliser également des icônes pour le codage de l'information sur l'organisation (introduction... conclusion). Si vous racontez plusieurs « histoires en parallèle », elles doivent être visuellement organisées en parallèle (une couleur et/ou un type d'icône par histoire).

Les codes choisis (couleurs, flèches, symboles, ...) doivent être les mêmes partout (un type de flèche ne doit pas signifier à un endroit: «ceci implique cela»; et à un autre «ceci interagit avec cela»). Au maximun il faut utiliser 3 types de flèches dans tout le poster. Les couleurs de fond ne doivent pas être trop vives pour ne pas attirer l'attention au détriment des résultats... et si vous imprimez en couleur.

Utiliser des phrases courtes, des mots simples, alignement à gauche (non justifié). Attention aux césures : évitez les !

Limiter le texte au profit des illustrations... ce qui règle le problème précédent. Chaque figure doit avoir un objectif clairement identifié et ne pas être une simple collection de données. Soignez le titre de la figure. Le message principal de chaque figure doit attirer immédiatement l'attention. Il est préférable que chaque figure contienne un message unique, toutes les données étant organisées autour du thème central unique.

Les éléments visuels qui retiennent le plus l'attention du lecteur sont, dans l'ordre décroissant : photo > dessin > schéma > tableau > mots

Ce qui suit ne concerne pas à proprement parlé l'exercice proposé dans le PPP (quoique) mais cela peut s'avérer utile pour la suite de vos études.

Si vous choisissez de présenter un tableau, enlevez toutes les lignes et colonnes inutiles et faites ressortir les données principales par un mécanisme d'icônes, de couleur, de gras. N'utilisez jamais le souligné : il peut être confondu avec du barré.

Les figures doivent pouvoir être lues à au moins 1 mètre de distance (en cas d'attroupement et pour les lecteurs qui n'ont pas une bonne vision de loin).

Toutes les légendes doivent être écrites horizontalement et non verticalement. Les légendes des séries de données sont directement sur le graphe et non pas sur le côté. Les couleurs les plus contrastées/vives sont utilisées pour les informations principales et non pas pour les informations annexes (axes, légendes,...).

Les échelles doivent être choisies de manière à ne pas déformer la réalité ou à laisser entendre une interprétation fausse. Considérons par exemple les valeurs suivantes :

Valeurs en x	2	4	8	16
Valeurs en y=f(x)	32	64	128	256

Vérifiez que si vous choisissez une échelle logarithmique pour les x et les y ou bien une échelle linéaire pour les x et logarithmique pour les y (et toutes les autres combinaisons), vous n'obtenez pas la même impression visuelle pour la courbe y=f(x). Il est donc important de mentionner les échelles et les unités !

Concernant les polices de caractères, un exposé ou un poster se rédige avec une police sans-sérif (on dit encore sans empattement). La police Times est une police avec empattement (créée en 1932 par M. Morisson pour le journal The Times – c'est donc pas une police pour faire un poster). Vous utiliserez avec modération le gras qui dans un article ne s'utilise pas pour renforcer une idée (ceci se fait avec de l'italique) mais pour les titres (de paragraphe). Sur un poster, jouer sur la chasse (la largueur des caractères), l'espacement des caractères, leurs contrastes (c.à.d une police spéciale qui n'est pas le gras de la police X). Éviter les polices fantaisistes du genre Cowboy : sobriété avant toute chose.

Exemple

L'exemple de la Figure ?? est un poster d'un projet de recherche. On remarque tout de suite que l'auteur a voulu centrer sur quare points qui forment, semble

t'il, les quatre grandes thématiques et spécificités du projet.

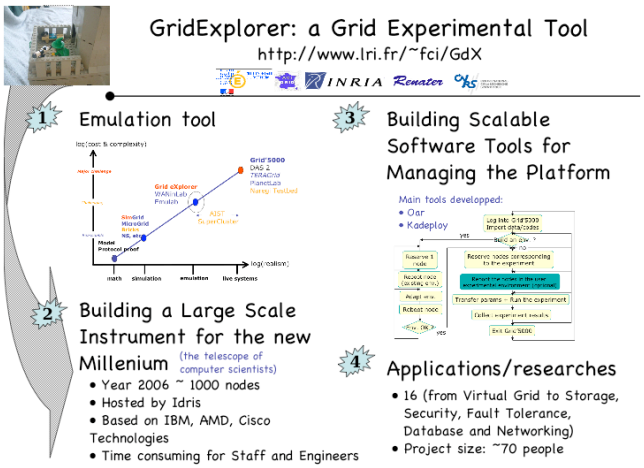


FIGURE 5.4: Exemple de réalisation d'un poster.

Il y a en haut à gauche une image d'un téléscope en LEGO pour rappeler le texte dans l'item 3 qui dit que GirdExplorer se veut « le téléscope du chercheur en informatique ». Ceci évoque immédiatement l'aspect expérimental (plus exactement : un outil pour faire de la science informatique expérimentale) du projet. Ce n'est pas un projet pour faire de la production mais plutot pour étudier (ici ce que l'on appelle les Grilles). Le quatrième point évoque les applications qui sont déployées dans le projet alors que le point 3 évoque les outils d'administration et de déploiment d'une application sur cette platforme.

Conclusion

En résumé, pensez que le poster vient en appui à un discours qu'il vous faut prépaper. C'est un exercice interactif où votre auditoire peut vous poser des questions à tout moment. Alors, décontraction (mais pas de relachement).

Un oral n'est pas un résumé du travail écrit mais un exposé sur l'ensemble du projet. Il s'écarte des détails pour s'attacher à la synthèse.

Une soutenance exprime de manière concise et synthétique le contexte, les raisons d'une recherche, puis ses méthodes, ses résultats et ses conclusion principales puis les discute, en imagine les prolongements. Il convient donc de cibler 3 à 5 points importants, pas plus. Ne cherchez pas à raconter les deux semaines passées sur le projet, tout le monde va s'endormir (maitrisez votre temps – si on vous dit 5 minutes, faites 5 minutes). Allez à l'essentiel. Dites ce que les résultats présentés vous ont permis de déduire / comprendre sur votre sujet. Mettez ces réflexions en perspective et apportez votre opinion personnelle.

5.10 La visite en entreprise

UNE ACTIVITÉ supplémentaire peut être proposée aux étudiants afin de s'imerger un peu plus dans le monde du travail : la visite en entreprise. Le sous paragraphe qui suit est un cadrage écrit pour les étudiants visant à rappeler quelques consignes de bon déroulement de la visite et aussi des conseils pour la préparer.

5.10.1 Remarques générales et importantes

Vous allez participer à une visite en entreprise, dans une société en informatique ou en centre universitaire. Nous attendons de vous un effort de ponctualité, de politesse et d'attention. Vous véhiculez en effet l'image de votre formation.

Les personnes qui vous reçoivent sont là pour répondre à vos questions. N'hésitez donc pas à interagir et à vous montrer curieux.

Nous vous conseillons vivement de prendre des notes sur place. Elles vous seront utiles pour préparer le compte-rendu.

En effet, un compte rendu de la visite est à préparer ainsi qu'un exposé. Cet exposé de 10 minutes devra être structuré (le plan sera annoncé). Il devra notamment comporter :

- une brève présentation de l'entreprise ;
- une présentation des métiers de l'informatique découvert à cette occasion et leur utilité pour l'entreprise.

Nous attendons également une réflexion personnelle sur ce que vous avez appris et les questions qui restent en suspens.

Si vous allez visiter une SSII, préparez des questions en relation avec les métiers, la façon de travailler dans les SSII. Le site PassInformatique (déjà mentionnés plusieurs fois) énonce des profils de postes et des compétences associées à ce poste. Vous pouvez élaborer des questions pour vérifier si la personne que vous avez en face de vous pendant la visite confirme ou insiste sur un point, une compétence particulière. Vous pourrez ainsi conclure votre compte rendu par une phrase de type «le métier Z nécessite bien de fortes compétences en Y«ou alors que tel ou tel aspect n'est pas, à votre avis, représentatif de l'activité de votre interlocuteur.

5.10.2 Exemple de compte rendu de visite

Le texte qui qui est le reproduction d'un texte de compte rendu de visite d'entreprise effectuée par un étudiant de première annéee du DUT informatique de Villetaneuse. Le lecteur est invité à examiner les éléments factuels et ceux qui montrent du recul par rapport à ce qui a été entendu lors de la visite. Le lecteur est aussi invité à examiner la conclusion qui montre ce que la visite a apportée à l'étudiant.

Compte rendu de la visite d'entreprise chez NVIDIA France

Benoît Villa - étudiant première année -

Elle s'est déroulée mardi 12 décembre 2006 pour un groupe de 12 étudiants du DUT Informatique, première année, Semestre 1, et dans les locaux de Levallois-Perret (Hauts de Seine).

Lors de cette visite, nous avons rencontré :

- Pauline Frieh – Manager Marketing ;
- Olivier – Responsable des ventes en France ;
- Stéphane Quentin – Responsable relations presse ;
- Cyril - Groupe DevTech ;
- Vincent Nguyen et Sébastien du groupe FAE : relations et support technique avec les OEM (Original Equipement Manufacturer).

Présentation générale d'NVIDIA par Mme Frieh

La société NVIDIA a été fondée en 1993 et son siège est à Santa Clara CA USA. Elle conçoit des puces graphiques et des chipsets qui sont intégrés aux cartes graphiques aux cartes mères.Cette intégration n'est par réalisée par NVIDIA mais par des partenaires tels qu'Abit, Asus, Gigabyte, Msi.

La fabrication des puces NVIDIA est sous-traitée à des entreprises spécialisées dans ce domaine. NVIDIA emploi 4500 à 4800 personnes dans le monde dont :

- 3000 ingénieurs (répartis dans les services matériel, logiciel et relations éditeurs)
- 40 personnes en Europe
- 4 à 20 personnes en France (4 personnes travaillent exclusivement pour la France, les 16 autres sont amenées à se déplacer pour couvrir plusieurs pays européens.)

On nous projette une vidéo retraçant les évolutions majeures des capacités des processeurs graphiques NVIDIA au fil des années puis nous avons droit à trois démonstrations - en avant-première - des nouvelles puces graphiques GeForce 8800.

Olivier – Responsable des ventes en France

Il occupe des fonctions commerciales, son travail est donc de promouvoir les produits d'NVIDIA auprès des partenaires (les OEM). Il nous donne quelques chiffres concernant la pénétration des marchés par NVIDIA :

- Cartes graphiques : 57% des PC embarquent des processeurs NVIDIA ;
- Cartes graphiques haut de gamme : 75% des sytèmes embarquent des processeurs NVIDIA ;

- Circuits graphiques intégrés aux cartes mères : 53% sont des circuits de chez Intel.

Il nous précise que NVIDIA effectue des vérifications de conformité en Chine, à Taïwan et aux Etats-Unis sur les cartes graphiques produites par les sociétés partenaires. A titre d'exemple la conception des puces GeForce 8800 a nécessité 4 ans de travail et 45 millions de $. La conception des chipsets dure en moyenne 36 mois.

Stéphane Quentin – Responsable relations presse

M. Quentin est en charge des relations avec la presse en France et en Belgique, il prépare et assemble les PC destinés aux testeurs ou aux salons mondiaux. Il participe aussi à ces salons (ex : le CeBIT d'Hanovre).

Les relations presses sont importantes car c'est via les magazines « grand public » que les utilisateurs connaissent les produits et la marque NVIDIA.

Il transmet aux équipes concernées les remarques des journalistes et utilisateurs qui effectuent des tests sur le matériel NVIDIA afin d'améliorer les produits à venir.

M. Quentin à tenu à nous parler de son parcours professionnel atypique. En effet, il a fait ses études dans une école de commerce puis s'est engagé dans l'armée. Etant passionné par les jeux, il a ensuite été journaliste pour un magazine de jeux PC et enfin recruté par NVIDIA surtout pour sa passion et ses connaissances dans le domaine de la presse et des jeux vidéo.

Cyril – Groupe DevTech

Son cursus est le suivant. Il est titulaire d'une maîtrise et d'un diplôme d'ingénieur. Son profil a intéressé NVIDIA car c'est un passionné du monde de l'informatique et des jeux vidéo et que son précédent emploi était développeur chez un éditeur de jeux vidéo.

Il appartient au groupe DevTech et sa mission est d'être en relation avec les développeurs de solutions logicielles pour les processeurs graphiques NVIDIA. Ces solutions sont à 95% des jeux vidéo, 3% de la CAO et 2% des logiciels pour films de synthèse, mais aussi des logiciels financiers suite à l'ouverture des processeurs graphiques à des utilisations plus généralistes.

Le groupe DevTech, qui comporte 20 personnes dans le monde pour être en contact avec les développeurs locaux, réalise aussi des présentations lors des conférences pour les graphistes.

Le groupe DevTech est intégré au département marketing car c'est grâce à ce groupe que les jeux et autres applications tirent partie des capacités des processeurs NVIDIA (en effet, les ingénieurs qui composent le groupe DevTech peuvent être amenés à concevoir des petites parties de code source pour des problèmes bien spécifique rencontrés par les développeurs).

Etant en relations avec les développeurs, les ingénieurs DevTech font remonter leurs remarques pour améliorer les nouveaux produits et pilotes.

A propos des nouveaux jeux, notre interlocuteur nous à précisé qu'ils étaient testés dans un laboratoire NVIDIA dédié, basé à Moscou, pour détecter d'éventuels problèmes et transmettre l'information pour que les corrections soient effectués. Ainsi si les problèmes se produisent uniquement sur du matériel NVIDIA, ces

problèmes seront transmis à l'équipe de développement des pilotes ; si les défauts sont communs à tous les matériels alors les développeurs du jeu seront alertés pour qu'ils effectuent des modifications nécessaires.

Suite à une de nos questions, nous avons appris qu'NVIDIA développe bien des pilotes pour Linux mais que la société préfère donner la priorité aux pilotes Windows car la majorité des utilisateurs finaux n'utilise pas Linux mais Windows. De plus NVIDIA ne souhaite pas rendre public le code source de ses pilotes Linux car cela permettrait de découvrir une partie de l'architecture interne des processeurs NVIDIA, ce qui n'est pas souhaitable à cause des concurrents.

Vincent Nguyen – FAE

M. Nguyen est en relation avec les OEM afin de les assister pour la fabrication de produits comportant des composants NVIDIA. Ainsi il peut être amené à effectuer des tests de compatibilité avec des cartes graphiques ou des cartes mères, réaliser des intégrations de matériel pour le compte des OEM ou bien effectuer des tests avec des logiciels (pilotes, jeux, ...)

Il est notamment en relation en France avec Acer, HP et Packard Bell et peut aussi bien travailler directement chez ces OEM, qu'être joint par téléphone ou e-mail pour des problèmes techniques.

Son travail se découpe en trois principales phases :

1. Avant vente : convaincre et aider l'OEM à choisir parmi les produits NVIDIA.

2. « Design win » : s'assurer avec l'OEM que les puces NVIDIA vont être intégrées dans de bonnes conditions.

3. Après vente : fournir une assistance aux OEM s'il survenait un incident sur la chaîne de production des puces NVIDIA.

Concernant ses compétences, il nous a précisé qu'il est titulaire d'un diplôme d'ingénieur et que son travail fait appel à 20% d'électronique, 60% d'architecture logicielle et 20% de gestion de projet.

Sébastien – FAE

Il est actuellement étudiant en deuxième année d'un BTS « Informatique de gestion » qu'il a choisi de suivre en alternance. Sébastien nous a dit être très content de son stage chez NVIDIA ; il épaule M. Nguyen.

Visite du « laboratoire »

Ce laboratoire est la pièce où sont préparés les PC pour les tests de journalistes (par Stéphane Quentin), mais aussi où sont réalisés des quelques essais avec le matériel des partenaires OEM.

Comme vous pouvez vous en douter cette pièce est remplie de matériel informatique de tout genre (boîtiers, écrans, claviers, cartes graphiques, cartes mère, ventilateurs, ...). Sur les plans de travail se trouve aussi de nombreux câbles et au centre de la pièce un monticule de cartons qui contiennent les machines de démonstration des salons et du NVIDIA Reality Tour 06.

« Avez-vous des questions ? »

Notre visite s'est terminée par la traditionnelle séance de questions/réponses. L'un d'entre nous a demandé quels étaient les salaires de nos interlocuteurs. Les réponses que nous avons obtenues nous ont laissé sur notre faim : « Au début les salaires ne sont pas très intéressants mais plus vous progressez et plus votre salaire progresse »

Au cours de la conversation et plus généralement au cours de cette visite, nos interlocuteurs ont insisté sur la nécessité d'un intérêt extra-scolaire pour le domaine de l'entreprise et de l'informatique professionnelle.

Ainsi Sébastien à obtenu son stage chez NVIDIA grâce à ses connaissances et sa passion dans le monde des cartes graphiques et des produits NVIDIA (SLI, GeForce, NForce). Il s'agit en effet du critère qui a permis à M. Nguyen de choisir un stagiaire parmi les postulants.

Nous remercions nos professeurs de PPP ainsi que NVIDIA France pour avoir accepté de nous recevoir afin de nous présenter les missions, l'organisation interne, les contraintes professionnelles d'une société majeure dans le domaine des processeurs graphiques.

5.11 Assurer une veille technologique

CETTE novelle fiche de travail vous propose des pistes de réflexion sur la veille technologique. Il s'agit d'un travail différent par rapport à ce qui vient d'être fait. «La veille technologique est l'observation et l'analyse de l'environnement scientifique, technique et technologique et des impacts économiques présents et futurs, pour en déduire les menaces et les opportunités de développement». On se limitera ici à quelques aspects (connaissance de l'environnement scientifique et technique, en fait découverte de quelques technologies) qui visent à vous faire acquérir une démarche intellectuelle qui devra prendre sa place tout au long de votre carrière professionnelle.

En effet, une carrière professionnelle se construit par exemple en faisant fructifier vos connaissances et compétences auprès de votre employeur. Sur le champ de l'informatique il est bien connu que les techniques évoluent rapidement : il faut donc maintenir à niveau son arrière plan technologique. Pour les plus expérimentés d'entre vous, au cours de votre carrière professionnelle, il faudra réussir à faire le tri entre ce qui présente un intérêt pour la profession et ce qui relève des "buzz words" comme on le dit dans le monde anglo-saxon. Le lien avec le PPP est donc le suivant : vous amener à une réflexion sur l'émergence de techniques qui pourraient modifier l'environnement professionnel.

Le travail à réaliser pour cette fiche peut s'effectuer dans le cadre de groupes de discussions entre étudiants et un encadrant. Il s'agit de présenter une technologie émergente ou incontournable (sans être un spécialiste bien entendu).

5.11.1 Quelques points de repère

Plus concrètement, la veille technologique consiste en la surveillance permanente, proactive et ciblée de l'environnement pour y déceler dès que possible les technologies émergentes qui seront peut être prépondérantes demain. Elle est utile aux :

- responsables stratégique et marketing ;
- chercheurs (Recherche et développement (R&D) ou autres) ;
- consultants ;
- cadres et aux décideurs ;
- ingénieurs, techniciensldots...

Elle peut se mettre en place :

- dans la vie quotidienne ;
- par la lecture de la presse spécialisée ;
- par le tissage d'un réseau de relations (professionnelles, personnelles) ;
- par la formation continue et/ou en alternance ;
- par de la curiosité.

5.11.2 Veille technologique passive

Il s'agit de répondre à un besoin particulier. C'est relativement facile à mettre en place par la maitrise de quelques techniques ; pas cher car la recherche est très ciblée mais aussi incertaine (l'échec est possible). Il est indispensable d'élaborer un cahier des charges où l'on précise ce que l'on doit faire.

5.11.3 Veille technologique active

Elle vise à anticiper. Elle est difficile à mettre en place (où chercher ? que chercher ?). Elle peut être chère à mettre en place car il faut assez fréquemment remettre en cause des choix. Cependant c'est un passage obligé pour de vraies évolutions.

Un exemple célèbre est celui de Steve Jobs (patron d'Apple) qui, en 1980, visitant les laboratoires Xerox, découvre l'Alto, une machine crée en 1971 comprenant une interface graphique avec des fenêtres, un petit boitier avec des roulettes (l'ancêtre de la souris) et un langage de programmation : SmallTalk. C'est alors le début de l'aventure Apple.

5.11.4 Veille technologique en entreprise

Toute entreprise (technologique) doit se consacrer au développement des technologies avec tout ce que cela comporte en terme de découvertes scientifiques (recherche fondamentale et recherche appliquée :
http://fr.wikipedia.org/wiki/Recherche_scientifique)
d'innovation de produits ou de services, d'évolution des procédés de fabrication, d'apparition de nouveaux matériaux ou concepts, de constitution de filières ou de sophistication des systèmes d'information.

C'est avec cette démarche qu'elle va créer/développer de nouveux marchés. «La prise en compte de la technologie dans l'élaboration de la stratégie exige dans un premier temps que, partant du principe selon lequel tout produit repose sur un ensemble fini de technologies distinctes, on identifie l'ensemble des technologies élémentaires qui entrent dans la réalisation d'une activité. Le recensement de l'ensemble des technologies mises en oeuvre dans les diverses activités de l'entreprise, ainsi, éventuellement, que des technologies non encore exploitées, permet de dresser l'inventaire du patrimoine technologique de l'entreprise»(Stratégor, 1988).

Une entrepreprise doit commencer par faire une étude interne pour déceler, parmi les technologies émergentes les technologies clés de son secteur d'activité. Une classification propose trois catégories :

- Les technologies de base : elle sont très présentes dans la réalisation d'une activité et bien maitrisée par l'entreprise (autour de son coeur de métier) ;
- Les technologies clés : ce sont celles qui donnent à l'entreprise un avantage concurrentiel grâce à une compétence distinctive ;
- Les technologies émergentes : ce sont celles qui sont encore en développement mais dont on peut craidre qu'elles changeront un jour les règles du jeu. Les entreprises commencent parfois à peine à les utiliser et n'en n'ont peut être pas encore mesuré toutes les applications possibles et les avantages qu'elles peuvent en tirer.

Une liste de pointeurs Web possible pour aller plus loin est alors la suivante :

- http://www.fsa.ulaval.ca/personnel/vernag/PUB/veille.html
- http://www.veille.com/
- http://www.enpc.fr/~michel-j/publi/JM318.html
- http://www.01net.com/

5.11.5 Le travail à réaliser par l'étudiant

Choisir un pointeur Web dans la liste du paragraphe suivant et préparez un mini-exposé d'au plus 10 minutes où vous présenterez la technologie devant vos camarades. La grille d'évaluation du travail comporte les questions suivantes (que le

correcteur se pose) : Qu'est-ce que j'ai compris de cette technologie (est-ce que j'ai appris quelque chose) ? Quels problèmes résoud t'elle (et ne résoud pas) ? Qu'elle est son domaine d'intervention ? Qui l'utilise ? Quels sont les technologies similaires et en quoi est elle différente ? Est elle accessible (libre) ? Est-ce une technologie pérenne ? Est-ce encore un projet ou existe t'il des outils logiciels (invention ou innovation ?) pour Linux, Windows, MacOSX ? Les sources d'information exploitées se recoupent elles ? Quelles sont ces sources ? Enfin, si vous avez eu le temps, avez vous expérimenté cette technologie et quel est votre avis personnel (avant, vous avez donné le point de vue des autres ; donnez le votre maintenant).

En tout état de cause, votre présentation sera accessible aux non spécialistes (n'utilisez pas de jargon technique). D'ailleurs, vous évaluateurs potentiels ne sont pas tous des informaticiens : il vous faut alors passer les messages essentiels, sans être un expert, et les convaincre avec un ton et des idées claires. Votre travail est avant tout un travail de «revue de presse»et nous ne vous demandons pas d'être un expert du domaine traité mais d'être convainquant.

Voici maintenant un exemple de texte pour le mini-exposé. Ce texte n'utilise aucun jargon technique. Le sujet est de parler du site http://gcc.gnu.org/.

«GCC est une suite (un ensemble) de compilateurs pour les langages de programmation C, C++, fortran... Il produit donc un fichier exécutable par une machine. Le rôle du compilateur est de transformer un code écrit dans un langage haut niveau vers un code écrit dans un langage «bas niveau».

GCC est un outil libre. On peut récupérer les sources (la distribution complète) et installer les nouvelles versions avec un compilateur C. GCC est «multiplateforme»(cross-platform en anglais) c'est à dire qu'il est développé afin de pouvoir l'installer sur tous types de machines (PC, Mac, IBM. . .) et de système d'exploitation.

C'est aussi un système expérimental dans le sens ou les concepteurs n'hésitent pas à intégrer des innovations dans la manière de générer le code exécutable. Il n'est pas supporté dans le sens ou aucun support (en ligne) via une société de service n'est disponible. Pour se tenir au courant des problèmes et des évolutions, il faut consulter le Web et lire les forums des utilisateurs.

Dans le manuel en ligne on lit que gcc s'utilise avec des options. Par exemple on peut contrôler l'optimisation du code executable produit avec l'option `-floop-optimize2` qui a pour rôle l'optimisation des boucles c.à.d les constructions itératives du programme soumis. On note également que GCC permet d'utiliser certaines ressources des processeurs qui font du calcul sur 128 bits (classiquement un processeur fait du calcul sur 32 ou 64 bits). Cela a pour rôle, encore une fois, d'optimiser le code exécutable.

C'est donc un outil élaboré (note : on ne dit pas «puissant»), utilisé dans de nombreux domaines d'application. Il concurrence serieusement les compilateurs Intel ou PathScale par exemple sur le plan des performances mais surtout parce qu'il est installable et disponible sur tous les systèmes d'exploitation.»

5.11.6 Liste de sites à explorer

- http://ant.apache.org/
- http://jakarta.apache.org/
- http://struts.apache.org/
- http://moinmoin.wikiwikiweb.de/
- http://www.spip.net/
- http://fr.wikipedia.org/wiki/Jabber
- http://xml.apache.org/
- http://www.w3.org/Amaya/
- http://www.w3.org/Style/CSS/
- http://www.w3.org/MarkUp/
- http://www.w3.org/International/
- http://www.w3.org/2004/OWL/
- http://www.w3.org/RDF/
- Concernant la sécurité, les liens suivants abordent certains aspetcs : http://directory.fst.org/libsafe.html, http://www.certa.ssi.gouv.fr, http://www.openssl.org
- http://www.w3.org/AudioVideo/
- http://www.w3.org/Graphics/SVG/
- Eclipse : http://fr.wikipedia.org/wiki/Eclipse_%28logiciel%29
- Asynchronous JavaScript and XML: http://fr.wikipedia.org/wiki/Asynchronous_JavaScript_and_XML
- Action Script: http://fr.wikipedia.org/wiki/Action_script
- Xul: http://fr.wikipedia.org/wiki/XUL
- http://www.gnu.org/software/emacs/emacs.html
- http://tapestry.apache.org/
- http://tomcat.apache.org/
- http://www.lyx.org/
- http://fr.wikipedia.org/wiki/Openoffice.org
- http://directory.fsf.org/text/wordproc/spell.html

- http://www.gimp.org/
- http://fr.wikipedia.org/wiki/Blender
- http://www.imagemagick.org/script/index.php
- http://de.samba.org/samba/
- http://directory.fsf.org/GNU/gnuPG.html
- http://fr.wikipedia.org/wiki/MediaWiki
- http://fr.wikipedia.org/wiki/CinePaint http://www.ruby-lang.org/fr/
- http://www.microsoft.com/net/default.mspx (.NET by Microsoft). Voir aussi http://www.dotgnu.org/
- http://spamassassin.apache.org/
- http://www.zdnet.fr/entreprise/service-informatique/poste-client/0,50007192,39286339,00.htm : Ajax et les Web Services

5.12 Formation post-DUT en Ile de France

CETTE nouvelle fiche de travail vous propose des pistes de réflexion sur les formations post-DUT. L'idée est la suivante. Nous avons préparé une liste de formations post DUT et une liste d'établissements franciliens proposant ces formations. Les étudiants choisiront, par groupes de quatre, une formation sur laquelle ils feront un exposé et ils rédigeront un document de 3 ou 4 pages sur cette formation. Nous avons prévu dans la fiche des contraintes qui imposent d'aller au moins dans un des établissements concernés pour se renseigner (chiffres de réussite des DUT dans l'établissement, nom des personnes rencontrées. . .).

5.12.1 Liste des formations

1. Ecoles d'ingénieurs ;
2. Licences professionnelles et licences classiques.
3. Autres

5.12.2 Liste des établissements

Groupe A1 : ESCI et ESIGETEL (Avon, 77), IUT Université de Marne-la-Vallée (Champs-sur-Marne, 77), Bercy Institute (Paris, 75), ENS (Paris), ENSAD (Paris) ;

Groupe A2 : INSEAD et ISIGE (Fontainebleau, 77), CFA UPMC – CFA universitaire Pierre-et-Marie-Curie (CFA UPMC) – 4, place Jussieu, case 232, 75252 Paris cedex 05, http://www.cfa.upmc.fr, (Co), ENSMP, ENSTA, ENSAM Paris (Ecole nationale supérieure d'arts et métiers) (Paris 75)

Groupe A3 : ISTY (Versailles, 78), INT Management Groupe des écoles des télécommunications (Evry 91), Ecole nationale supérieure d'informatique pour l'industrie et l'entreprise (ENSIIE) Université d'Evry-Val d'Essonne (91), Institut universitaire de technologie d'Evry (Evry, 91), Institut des techniques informatiques. Ecole supérieure d'informatique, réseaux et systèmes d'information (Cergy - 95), Université de Paris XI (la licence classique) ;

Groupe B1 : Institut Galilée (Villetaneuse), Paris 6 (la licence classique), Licence Pro Mention Réseaux et Télécommunications, spécialité Intégrateur de Systèmes Distribués et Réseaux Numériques (ISDRN) de l'université de Versailles St Quentin, Licence professionnelle santé, spécialité statistique et informatique décisionnelles pour la santé de Paris 5, université de Paris XI : http://www.ifips.u-psud.fr, L'AFPA en Île de France est présente sur le site http://www.idf.afpa.fr/, rechercher comment certifier et valider vos compétences (secteur FONCTIONS OU TECHNIQUES DU SECTEUR TERTIAIRE) ;

Groupe B2 : Licence Domaine Sciences et Technologies, Mention Informatique de l'université de Versailles St Quentin., Ecole nationale supérieure du pétrole et des moteurs (ENSPM Rueil-Malmaison), IFIS - Université de Marne-la-Vallée, Licence professionnelle réseaux et télécommunications, spécialité administration et transmission de l'information de UTEC informatique (Université d'enseignement et de technologie consulaire industrie et nouvelles technologies, Bd Olof-Palme, Emerainville, 77436 Marne-la-Vallée cedex 2), UFR mathématiques, informatique, technologies, sciences de l'information et de la communication (UFR MITSIC) UNIVERSITE VINCENNES-SAINT-DENIS - PARIS 8, et enfin le Centre d'études supérieures industrielles (CESI Paris) 19-21, rue du 8-Mai-1945, 94110 Arcueil

5.12.3 Travail à réaliser

Nous vous proposons de travailler dans l'ordre suivant :

1. formez le groupe de quatre étudiants et entendez-vous sur un type de formation ;

2. recherchez dans la liste des établissements, celui qui vous intéresse (pour cause de proximité géographique, connaissance, lecture d'une publicité. . .)

3. préparer un questionnaire sur la formation. Vous pouvez vous inspirer du travail réalisé sur l'interview d'un professionnel (en partie), par exemple pour demander le positionnement de l'établissement par rapport à son entourage, son histoire. Une autre question incontournable sera : «qu'est ce qui peut bien inciter un étudiant à venir chez vous ?»ce qui renvoie à des sous questions de relance du type : équipement matériel, encadrement enseignant et administratif, réseau d'entreprises. . . . Il faut bien sûr aussi évoquer les taux d'insertion professionnelle, le réseau des anciens (que fait-il de particulier ?).

Veuillez bien distinguer les types de formation : informatique, autres champs techniques connexes (réseaux, télécommunications voire d'autres profils ingénieurs ou des champs disciplinaires sensiblement différents (gestion)).

Nous vous conseillons d'innover et de trouver d'autres questions pertinentes. Veuillez rédiger complètement le questionnaire ;

4. Prendre contact avec l'établissement pour un rendez-vous physique (ou par téléphone si vous sentez qu'il va y avoir des problèmes pour vous recevoir) ; Ciblez bien l'interlocuteur que vous souhaitez (étudiant, enseignant, responsable administratif. . .)

5. Réaliser l'entretien en déroulant lettre à lettre le questionnaire ;

6. Remettre en ordre les réponses du questionnaire (si nécessaire) et rédigez un paragraphe de synthèse sur ce que vous avez appris, éventuellement une recommandation (ou, si vous ne voulez pas vous mouiller, un avis neutre sur la poursuite d'étude dans cet établissement)

7. Faites une répétition de la restitution qui commencera par une introduction d'une ou deux phrases pour dégager les contraintes demandées, et résumera le travail réalisé.

Votre exposé et le document de 3 ou 4 pages que vous avez à rédiger pourront reprendre des éléments du questionnaire ci dessus et feront en particulier toute la lumière sur les points suivants :

- quelles sont les formations proposées par un établissement de la liste dans les domaines de l'informatique ?

- l'approfondissement, pour au moins l'une des formations proposées par l'établissement, du contenu, du déroulement, des exigences, des débouchés. . . Cet approfondissement devra s'appuyer sur la prise de contact (et si possible la rencontre) soit d'un cadre, soit d'un enseignant, soit d'un étudiant de l'établissement en question ;

- le niveau exigé à la sortie de l'IUT pour ce type de formation (combien d'étudiants de l'IUT ont suivi la formation, quels étaient leurs résultats, qu'ont-ils fait après ?).

6 Anglais, droit, modèles économiques de l'Internet

Sommaire

6.1 Motivations

NOUS AVONS regroupé dans ce chapitre trois thématiques qui complètent le corpus de connaissances de base que nous estimons nécessaires dans des études d'informatique. Premièrement, l'anglais semble une évidence. Il peut même être demandé à des étudiants de préparer une partie de leurs soutenances orales en anglais (les remerciement et l'introduction ou bien encore la conclusion). C'est un bon exercice. Nous passons en revue quelques ficelles pour bien préparer ces étapes.

Deuxièmement le droit. Nous commençons par présenter les enjeux liés au commerce électronique. Nous introduisons ensuite la notion de licence libre qui est propre à l'informatique.

Troisièmement les modèles économiques liés à l'Internet. Plus personne ne se pose la question de savoir pourquoi avec Internet on ne paye pas l'envoi des courriels et pourquoi on a préféré un modèle qui revient à louer la ligne. En fait, en y regardant de plus prêt on peut suggérer comme éléments de réponse que comme le trafic qui sort de sa machine est infiniment moins important que ce qui rentre, il serait indécent de poser un timbre sur les envois ! Par ailleurs, comme avec le facteur quand il vient déposer un courrier dans notre boîte aux lettres, il ne s'agit pas de taxer les réceptions. Ces observations lancent donc le débat sur la question «quels modèles de rénumération»pour Internet ? C'est ce que nous expliquons ici

et nous espérons avoir contribué dans ce chapitre à faire comprendre le monde qui nous entoure et globalement, après la lecture de ce dernier chapître à pousser le lecteur à entreprendre des études en informatique.

6.2 Anglais technique

CETTE PARTIE se veut auto-formative car elle renvoie sans doute à des connaissances en partie enfouies qu'il convient de raviver et à des liens Internet de première importance pour travailler son anglais technique. Les resources proposées en référence sont également des vidéos qui renvoient soit à des textes fondateurs, soit à un concept récent, soit à un concept maintenant devenu d'usage courant.

Nous n'abordons pas de front les questions de grammaire mais celles liées au vocabulaire, en particulier au vocabulaire technique et à celles liées à l'expression orale ou écrite en informatique.

6.2.1 Vocabulaire

Commençons par rappeler que certains noms/adjectifs anglais ressemblent beaucoup à ceux du français sauf que l'orthographe varie un peu mais ils ont les mêmes significations : language (langage en français), program (programme), measure (mesure), example (exemple), recommend (recommander) alors que d'autres ont une orthographe similaire (modulo les accents) : dependant/independant (dépendant / indépendant en francais), entrepreneur (intrepreneur), addition (addition en français), precedent (précédent), arrangement (arrangement), tolerant (tolérant – comme dans 'fault tolerant'). En cas de doute, reportez vous à un dictionnaire.

Par ailleurs il y a les *anglicismes* qui sont des mots anglais employés en français et critiqués comme emprunt abusif ou inutile. Voir un des sites de Wikipedia[1] pour des détails sur les différentes formes d'anglicismes.

Ensuite il faut faire très attention aux faux amis.

Les données du site http://villemin.gerard.free.fr/Langue/FauxAmi1.htm ont été simplifiées pour se concentrer sur des termes que l'on peut retrouver dans un texte d'informatique.

Traduction du faux ami anglais	Mot anglais faux ami	Mot français avec lequel il peut y avoir confusion	Traduction en anglais du mot français objet de la confusion
Insulter, injurier	Abuse	Abuser	Go too far, overstep
Loger	Accommodate	Accommoder	Prepare
Réaliser, atteindre (une performance)	Achieve	Achever	Finish

[1] http://fr.wikipedia.org/wiki/Anglicisme

Réalisation, exploit, réussite	Achievement	Achèvement	Finish
Réel	Actual	Actuel	Current
En fait, en réalité (tic de langage US)	Actually	Actuellement	Currently, presently, at the present time
Aborder	Address	Adresse	Address
Discours, allocution	Address	Adresse	Address
Ordre du jour	Agenda	Agenda	Diary, appointment book
Être d'accord	Agree	Agréer	Accept, to welcome
Accord	Agreement	Agrément	Pleasantness, amenity
Changer	Alter	Altérer	Impair, distort, deteriorate
Antenne hertzienne	Antenna	Antenne acoustique	Acoustic array
Présenter ses excuses	Apologise	Apologie (faire l')	Praise, justify, vindicate
Excuses	Apology	Apologie	Praise, vindication
Rendez-vous	Appointment	Appointements	Salary, wage
Reconnaître, comprendre	Appreciate	Apprécier	Appreciate, estimate, be grateful
Prendre de la valeur (finance)	Appreciate	Apprécier	Appreciate, estimate, be grateful
Se disputer	Argue	Arguer	Allege, put forward as a pretext
Dispute	Argument	Argument	Argument
Armement d'un pays (rare)	Armament	Armement	Arms, weapons
Arriver à destination	Arrive	Arriver (se produire)	Happen
Aider, assurer le support	Assist	Assister	Attend, take part in
Équilibre	Balance	Balance	Scale

Équilibrer, tenir en -	Balance (to)	Balancer	Swing, rock
Scrutin, bulletin de vote	Ballot	Ballot, paquet	Bundle
Orchestre, bande	Band	Bande	Band, Gang / Strip / Tape
Tendance / Biais (math)	Bias	Biais (en)	At an angle
Attribuer un tort	Blame	Blâmer	Reprimand, censure
Analyse, calcul infinitésimal	Calculus	Calcul	Calculation, computation, reckoning
Étalonnage	Calibration	Calibration	?
Franc, sincère	Candid	Candide	Innocent, pure
Majuscule	Capital	Capital	Capital
Carte électronique (rare)	Card	Carte électronique	Electronic board, module
Cartes de jeux	Card	Carte géographique	Map
Entourloupe, attrape/ prise, loquet	Catch	Catch	Professional wrestling
Prudence	Caution	Caution	Guarantee, bail
Risque, hasard, occasion			
y take a chance?	Chance	Chance	Luck
Monnaie	Change	Change (taux de)	Currency exchange rate
Personnage, phénomène, original	Character	Caractère	Font / Temper
Prix à payer / accusation	Charge	Charge	Load, burden, charge
Extralucide	Clairvoyant	Clairvoyant	Clear-sighted, shrewd
S'engager	Commit	Commettre	Perform, commit
Effectif, équipage	Complement	Complément (en)	Addition (in)
Achever, parachever	Complete	Compléter, remplir	Fill up, out

ə

Gratuit, de faveur	Complimentary	Complémentaire	Complementary
Complet, exhaustif	Comprehensive	Compréhensible	Understandable
Imaginer, concevoir (idée)	Conceive	Concevoir	Design, devise, plan
Confiance	Confidence	Confidence	Confidence
Compatible, logique avec	Consistent with	Consistant	Sound, substantial
Diriger, commander, faire fonctionner	Control	ContrÃ´ler	Inspect / Monitor , supervise, command (rare)
Direction centrale	Corporate	Corporation	Trade guild
Organisme public	Corporation	Corporation	Trade guild
Route d'un bateau	Course	Course	Race
Collision / En urgence	Crash	Crash	Crash landing
Actuellement	Currently	Couramment	Commonly / Fluently (langue)
Trompeur	Deceiving	Décevant	Disappointing
Supercherie, tromperie	Deception	Déception	Disappointment
Remettre à plus tard	Defer	Déferrer au parquet	Bring to trial
Retard	Delay	Délai	Deadline, time limit, delivery time, lead time, period, cut-off date, closing date
Livrer	Deliver	Délivrer	Free, release, issue
Exiger	Demand	Demander	Ask
Exigeant, difficile	Demanding	Demander	Ask
Dense / Obtus, lent, stupide	Dense	Dense	Dense, heavy

Concevoir	Design	Désigner	Point out, name, refer
Pauvre	Destitute	Destituer	Dismiss
Dissuasion (de)	Deterrent	Déterrer	Dig up
N'existe pas	Devise	Devise	Motto/ Currency
Imaginer, concevoir	Devise	Deviser	Converse
Désaccord	Disagreement	Désagrément	Annoyance, unpleasantness
Désaccord	Dispute	Dispute	Argument, quarrel
Ce qui détourne l'attention	Distraction	Distraction	Entertainment, leisure
Spectaculaire	Dramatic	Dramatique	Dramatic, tragic
Préparer pour la publication, corriger	Edit	Éditer	Publish, issue
Rédacteur	Editor	éditeur	Publisher
Efficace	Effective	Effectif	Real, actual
Émission de gaz, émanation	Emission	Émission radar	Transmission
Mettre l'accent sur	Emphasise	Emphase	Bombast, pomposity
Employer des personnes	Employ	Employer	Use, utilise
Affaiblir	Enervate	Énerver	Irritate, annoy
Moteur	Engine	Engin	Machine, instrument, vehicle, gear
Avoir le droit / Intituler	Entitle	Appeler	Name, label
Plat de résistance	Entree	Entrée	First course
Certainement à la longue	Eventually	Éventuellement	Possibly, maybe
Preuve	Evidence	Évidence	Obviousness
Visible	Evident	Évident	Obvious
Montrer clairement	Evince	Évincer	Evict, oust

ə

Signer pour entrer en vigueur	Execute	Exécuter	Perform, carry out
Savoir-faire	Expertise	Expertise	Expert' opinion
Exploiter une situation	Exploit	Exploiter	Run (entreprise / Operate (système)
Atténuer (circonstances)	Extenuate	Exténuer	Exhaust
Textile	Fabric	Fabrique	Factory
Installations	Facilities	Facilités	Easiness
Délicat, difficile à contenter	Fastidious	Fastidieux	Tedious
Chiffre / Silhouette	Figure	Figure	Piece of art, chart, illustration, art work
Compagnie	Firm	Ferme	Firm, hard
Fixe (prix)	Fix	Fixe	Set time, basic salary
Réparer	Fix	Fixer à	Fasten to
Falsifier	Forge	Forger	Forge
Autrefois	Formerly	Formellement	Formally
Mobilier	Furniture	Fournitures	Supplies
Mondial, universel	Global	Global	Total, overall, comprehensive
Satisfaction	Gratification	Gratification	Gratuity
Gratification	Gratuity	Gratuit	Free, complimentary
Risque, danger	Hazard	Hasard, par hasard	Chance, fate / Random by chance
Clément	Humane	Humain	Human
Ignorer délibérément	Ignore	Ignorer	Not to know, pay no attention
Qui importe, essentiel	Important	Important	Considerable, sizeable
Qui mérite attention	Interesting	Intéressant	Attractive, advantageous, beneficial, worthwhile
Présenter quelqu'un	Introduce	Introduire	Insert (object) / LAUNCH (product)

De grande valeur, sans prix	Invaluable	Sans valeur	Of no value
Invoquer / Appeler – You can also invoke with symbolic input	Invoke	Invoquer	Invoke, put forward, call for
Édition / un sujet à traiter – There's three issues to be addressed	Issue	Issue	Exit, way out, outlet
Voyage	Journey	Journée	Day
Conférence	Lecture	Lecture	Reading matter
Bibliothèque	Library	Librairie	Bookshop
Permis de conduire	Licence	Licence	Licence
Endroit, site	Location	Location	Rental, location
Engin, machine	Machine	Machine de calcul	Processor, computer
Question, sujet	Matter	Matière	Matter, material
Numéro, nombre (quantité)	Number	Nombre, chiffre	Figure
Observation	Observation	Observation	Remarks (pluriel)
Délit	Offence	Offense	Misdeed
Autoritaire	Officious	Officieux	Unofficial
Pétrole, mazout	Oil	Huile	Oil
Difficile, lourd à supporter	Onerous	Onéreux	Costly, expensive
Fonctionner	Operate	Opérer, réaliser	Carry out, implement
Occasion	Opportunity	Opportunité	Timeliness
Réussir	Pass an exam	Passer un examen	Take, sit an exam
Brevet	Patent	Patente	Trading licence
Amende (infraction)	Penalty	Penalty	Penalty kick

ə

Exécution d'un travail	Performance	Performance	Exploit, feat, achievement
Autoriser	Permit, authorise	Permettre	Allow, enable, make possible, provide
Peu usité	Precise (to)	Préciser	Clarify, be more specific
Minutie, exactitude	Precision	Précision	Accuracy
Présenter, offrir	Present	Présenter une image	Display a picture
Imaginaire	Pretend	Prétendu	Supposed, alleged
Faire semblant	Pretend (to)	Prétendre	Claim
Empêcher	Prevent	Prévenir	Warn, inform
Avancer, se mettre à, continuer	Proceed	Procéder	Carry out
Taux	Rate	Rate	Spleen
Réel (existe)	Real	Réel	Actual
Réaliser une ambition	Realise	Réaliser	Implement a plan, Execute
Se rendre compte	Realise	Réaliser	Perform, achieve
Calculer, compter / estimer	Reckon	Reconnaître	Recognise, acknowledge, admit
Estime, respect	Regard	Regard	Look
Loyer	Rent	Rente	Pension
Réagir	Respond	Répondre	Answer
Réaction	Response	Réponse	Answer
Reprendre	Resume	Résumer	Summarise
CV (Curriculum Vitae)	Résumé	Résumé	Summary
Attacher, fixer (avec des vis?)	Secure	Sécuriser	Give security
Sensé, raisonnable	Sensible	Sensible	Sensitive
Trier	Sort	Sort	Fate
Norme	Standard	Standard	Common, standard
Objet	Subject	Sujet	Topic
Soutien, service, servitude	Support	Supporter	Stand, endure, tolerate

Nom de famille	Surname	Surnom	Nickname
Ombrageux, sensible	Susceptible	Susceptible	Likely to
Compatissant, compréhensif	Sympathetic	Sympathique	Charming, congenial
Avoir lieu	Take place	Prendre place, Prendre une grande place	Take a seat, Play a major role, to mean a lot
Mettre fin à, résilier	Terminate	Terminer	Finish, end
Épouvantable	Terrible	Terrible, formidable	Terrific, wonderful
Grâce à (?Dieu)	Thanks to	Grâce à, du fait de	Due to
Insignifiant, banal	Trivial	Trivial	Trivial, vulgar
Polyvalent	Versatile	Versatile	Moody, fickle, erratic

ə

6.2.2 Remercier

On peut vous demander lors d'une soutenance de stage de faire l'introduction ou la conclusion en Anglais. C'est l'occasion de remercier celles et ceux qui ont participé activement à votre travail. Ce n'est bien sûr pas la seule chose à faire dans une introduction puisqu'il vous faut donner également les motivations, le plan de votre exposé et rappeler les apports, les résultats obtenus dans la conclusion. Voila un exemple de texte de remerciement concernant toutes les personnes (la liste est longue) ayant participé à l'organisation d'une conférence internationale. Ne passez pas plus d'une minute dans un exposé oral à remercier.

We would like to take this opportunity to thank everyone involved with the organization of GPC 2007. First, we would like to thank all the authors for their submissions to the conference as well as for travelling some distance to participate in the conference. Second, we would like to thank the Program Committee members and external reviewers for their superb job in selecting a set of excellent papers that reflect the current research and development states of grid and pervasive computing.

Third, we would like to thank Franck Cappello (INRIA, France), Jean-Luc Gaudiot (University of California at Irvine), and Hai Jin (Huazhong University of Science and Technology, Wuhan) for their valuable comments during the year. Our appreciation also extends to Alfred Hofmann and Anna Kramer, both from Springer, for their helpful comments in strengthening the conferences. We will continue to improve further, in particular with the selection of the Program Committees and other scientific issues. We are also

grateful to Christine Nora and Cyril Drocourt from IEEE France for the secure Web payment and for managing the finances. Jean-Christophe Dubacq (Paris XIII) was busy with the review system, the Web server, registration, and many other important issues regarding the technical program. Catherine Girard from the INRIA Office of the Colloqium did a superb job once again with the organization and the INRIA sponsorship. It is always a pleasure to work with Catherine Girard and her high level of professionalism is highly appreciated.

GPC2007 was sponsored by Hewlett Packard through the strong support of Franck Baetke, Philippe Devins, and Jean-Luc Assor, by INRIA and the University of Paris XIII through the Conseil Scientifique, and also through Laboratoire de Recherche en Informatique de Paris Nord (LIPN - UMR CNRS 7030).

Last but not least, we express our gratitude to François and Ludivine from Dakini Conseil for their help in organizing accommodation for conference atten- dees, finding a venue for the conference and also for its banquet. We would also like to thank Severine Bonnard from MGEN for allowing us to rent the beautiful MGEN building with all the services that a speaker dreams to find on a site (e.g., comfortable rooms, a restaurant for the gourmets, etc.) in the center of Paris.

Pour travailler l'introduction, nous vous conseillons de lire des articles des magazines Spectrum[2] (plus particulièrement les onglets Magazine et Computer) ou Computer Magazine [3]. Pour ces deux sites, le lecteur est donc invité à délimiter dans un article la partie relative à l'introduction, à trouver les motivations, le plan, l'articulation générale de l'article.

6.2.3 Le style

Le style n'a rien a voir avec l'utilisation correcte de la grammaire mais comment le texte rentre en communion avec le lecteur. Les conventions de le style sont importantes car la conformité à des usages de style réduit les chances d'obtenir un texte, une présentation ennuyeuse. Ceci est bien entendu vrai pour l'anglais comme pour le français.

Dans un des tous premiers chapitres relatif au «comment écrire», nous avons vu que chaque phrase doit être nécessaire dans une discipline scientifique. La phrase est généralement écrite au présent de l'indicatif et plus elle est courte, meiux c'est. On préfère toujours la voix active. Un sujet, un verbe, un complément sont bien souvent suffisants. Les nuances, les ambiguités, les métaphores, les circonvolutions n'ont rien à faire avec un texte scientifique et technique.

Pour l'anglais plus spécifiquement, le lecteur est maintenant invité à consulter http://www.bartleby.com/141/strunk.html qui renvoie à "William Strunk, Jr. (1869 - 1946). The Elements of Style" paru en 1918.

[2] Voir http://www.spectrum.ieee.org/
[3] Voir http://www.computer.org/portal/site/computer/

Pour la grammaire, le lecteur est invité à lire les pages de l'université de Chicago[4] dont le lien qui renvoie à "English for non-native speakers (ESL)". Il y a également des liens sur le thème des "Science and Technical Writing Guides".

Le site http://www.rbs0.com/tw.htm, très complet donne des conseils et les conventions pour "Technical Writing". Dans la même veine, nous vous recommandons le site OWL[5] de l'université de Purdue.

Enfin le site de Jack Lynch[6] est une mine d'information de même que le site http://alt-usage-english.org/ et le forum :
http://www.linguistic-funland.com/mele.faq.html

6.2.4 Videos and lectures resources

Nous vous conseillons également les ressources suivantes disponibles en ligne pour travailler votre compréhension de l'anglais oral ou écrit :

Donald Knuth : http://scpd.stanford.edu/knuth/ (vidéos d'un acteur majeur de l'informatique théorique)

Edsger W. Dijkstra http://www.cs.utexas.edu/users/EWD/video-audio/video-audio.html. Voir également la page principale du site précédent qui renvoie à des articles importants de notre discipline. Par exemple, il faut lire "Co-operating sequential processes", "Solution of a problem in concurrent programming control", "A constructive approach to the problem of program correctness", "Towards correct programs", "Structured programming"

Tony Hoare : http://research.microsoft.com/~thoare/

Table ronde en accès libre à propos du concept de "Pervasive Grids" : http://www-lipn.univ-paris13.fr/~cerin/RoundTable.htm

6.3 Droit et Internet

DANS cette partie nous abordons un cadrage général quant au droit lié au commerce électronique puis nous introduisons la notion particulière de licence libre qui illustre une évolution importante de la notion de droit d'auteur pour les œuvres logicielles. Par ailleurs, comment lutter contre le téléchargement illégal de musiques, de films ou de séries télévisées sans froisser les internautes qui considèrent la Toile comme un espace de liberté, tout en garantissant aux auteurs une rémunération juste ? Cette problématique n'est pas abordée ici, bien que touchant au droit d'auteur et nous préférons renvoyer le lecteur à la notion de «contribution créative», théorisée par le chercheur Philippe Aigrain dans Internet et création (In Libro Veritas, 2008) - l'ouvrage est en accès libre et téléchargeable[7]

[4] Voir http://writing-program.uchicago.edu/resources/grammar.htm
[5] Voir http://owl.english.purdue.edu/
[6] Voir http://andromeda.rutgers.edu/~jlynch/Writing/links.html
[7] http://www.inlibroveritas.net/lire/oeuvre20460-chapitre100789.html

6.3.1 Principes généraux du droit lié au commerce électronique

Wikipedia désigne le commerce électronique ou la vente en ligne, comme l'échange de biens et de services entre deux entités sur les réseaux informatiques, notamment Internet. En France, les professionnels du secteur sont rassemblés au sein de la fédération du commerce électronique et de la vente à distance (FEVAD).

Le Parlement européen et le Conseil ont adopté le 8 juin 2000 une directive européenne sur le commerce électronique (Directive 2000/31/CE du Parlement européen et du Conseil relative à certains aspects juridiques des services de la société de l'information, et notamment du commerce électronique, dans le marché intérieur). Celle-ci a été transposée en France par la loi pour la confiance dans l'économie numérique de 2004.

Les éléments touchant au droit sont alors les suivants :

- les sources du droit du commerce électronique (internationales, Nations Unies, convention de Vienne sur la vente internationale de marchandises, droi français)

- les règles juridiques qui encadrent la visibilité du commerçant et la constitution de sa clientèle ; Par exemple, le commerçant doit choisir un nom de domaine pour son site web et donc le déclarer pour qu'il puisse être atteignable ;

- la publicité sur Internet ; Il y a bien sûr inderdiction de publicité trompeuse ;

- les règles qui régissent l'établissement d'un contrat entre le vendeur et l'acheteur ; C'est la loi n°2004-575 du 21 juin 2004 pour la confiance dans l'économie numérique ("LCEN") qui définit la notion de contrat de commerce électronique, rappelle les diverses obligations d'information qui pésent sur le vendeur, impose au vendeur de nouvelles obligations lors de la conclusion et de l'exécution du contrat, et renforce la protection du consommateur.

- les règles de preuves (pour attester qu'il y a bien une une transaction) ; Précédemment, la loi nÂ? 2000-230 du 13 mars 2000 à l'origine des articles 1316 à 1316-4 du code civil reconnaissait que l'écrit électronique avait la même valeur probante que l'écrit sur support papier. Toutefois, cette loi ne traitant que de la preuve, elle se gardait bien d'établir la validité d'un contrat conclu sous forme électronique lorsqu'un écrit était exigé par loi.

  Le consommateur conservant rarement la preuve de ces échanges, d'éventuels problèmes de preuve auraient pu survenir en cas de litige avec le commerçant. C'est pourquoi l'article 27 de la LCEN a inséré le nouvel article L. 134-2 du code de la consommation.

  Cet article prévoit une obligation à la charge du cybervendeur professionnel de conservation de l'écrit constatant le contrat, dès lors que celui-ci porte sur une somme supérieure ou égale à un montant fixé par décret.

- le paiment en ligne ; Pour le commerçant, l'essentiel est d'être payé. Il lui importe peu que le payeur soit le client. Il convient donc de distinguer le paiement en ligne et l'authentification de l'acheteur. L'envoi du numéro de carte bleu se fait dorénavent en contexte chiffré. Le chiffrage intervient soit à l'initiative du client qui utilise alors des mécanismes ou des fonctionnalités de son navigateur, soit à l'initiative du commerçant qui renvoie généralement sur un site de confance, sécurisé (une banque ou un organisme dédié) ;

Nous venons d'introduire quelques points de droit lié au commerce éléctronique. Chacun de ces points est amené à être développé dans un enseignement disciplinaire spécifique.

6.3.2 Licence et logiciel libres

Introduction

Dans ce paragraphe nous allons introduire des notions propres à l'informatique, liées au droit d'auteur et qui sont poussées en particulier par l'association FSF (Free Software Foundation). L'association est à l'origine des quatre règles fondatrices suivantes :

- La liberté d'exécuter le programme, pour tous les usages ;
- La liberté d'étudier le fonctionnement du programme, et de l'adapter à ses besoins. L'accès au code source est donc une condition requise.
- La liberté de redistribuer des copies, donc d'aider son voisin ;
- La liberté d'améliorer le programme et de publier des améliorations, pour en faire profiter toute la communauté. Pour ceci l'accès au code source est une condition requise.

Prenons l'exemple du point 3 relatif à l'accès au code source du logiciel. Il vous parait peut être plus judicieux, pour éviter qu'on vous copie, de ne pas divulger votre secret de fabrication et d'enfermer vos codes sources dans une banque ! Pour un logiciel, il en va tout autrement : c'est l'algorithme implémenté dans un certain langage qui code les propriétés désirées. Si on n'a pas d'accès aux sources, on ne peut pas vérifier si les propriétés énoncées sont vraies ou pas. L'idée de rendre libre l'accès aux sources est donc de permettre à n'importe qui de vérifier les affirmations (commerciales).

Comme autre exemple, nous pouvons considérer que de nombreux protocoles cryptographiques sont libres d'accès (openssl par exemple) c.à.d qu'ils assurent un niveau de protection (confidentialité des données qui transitent d'un point A à un point B) par l'algorithme qu'ils mettent en œuvre. C'est à nouveau l'algorithme qui assure les propriétés et pas le fait de cacher le code source. La confiance est plus forte lorsque l'on voit tout, que rien n'est caché !

Licence libre

Le site Wikipedia définit le terme dela façon suivante : «Une licence libre est une licence s'appliquant à une œuvre de l'esprit par laquelle l'auteur concède tout ou partie des droits que lui confère le droit d'auteur, en laissant au minimum les possibilités de modification, de rediffusion et de réutilisation de l'œuvre dans des œuvres dérivées. Ces libertés peuvent être soumises à conditions, notamment l'application systématique de la même licence aux copies de l'œuvre et aux œuvres dérivées, principe nommé copyleft.»

Une licence libre est donc un contrat juridique. La personne qui modifierait un code source et le vendrait s'expose à des poursuites s'il ne fait pas attention.

Cette notion de «libre»a pu se développer parce qu'elle a été longtemps synonyme d'innovation technologique. Ainsi nous avons assisté pendant les dix dernières années à l'expansion du nombre de projets communautaires qui consistent à lancer une idée puis de fédérer les énergies autour de l'idée initiale en faisant appel à des contributeurs qui travaillent souvent de manière bénvole et qui interagissent via Internet.

Il est moins clair en 2012 que le libre continuera à se développer autant parce qu'on assiste à une certaine concentration des services à travers la notion de Cloud Computing. Le Cloud vise à offrir à travers un service unique l'accès à tous les services et à tout moment. En fait, le cloud computing, informatique en nuage est un concept qui consiste à déporter sur des serveurs distants des traitements informatiques traditionnellement localisés sur des serveurs locaux ou sur le poste client de l'utilisateur comme Microsoft Word par exemple. Au lieu d'acheter un logiciel et sa licence, l'utilisateur paye à l'usage par exemple ou sur la base d'un forfait mensuel.

La principale idée qui a motivé le concept de logiciel libre était de rendre indépendants les utilisateurs de logiciel du bon vouloir des développeurs originaux de ceux-ci. Avec le Cloud, l'utilisateur devient prisonnier d'un fournisseur qui décide quand il le veut s'il met à disposition une nouvelle version du logiciel.

Un des aspects souvent mal compris du principe de licence libre concerne le fait que celui-ci ne traite pas de la valeur marchande de la diffusion des œuvres. Une œuvre sous licence libre n'est pas nécessairement disponible gratuitement, de même une œuvre disponible gratuitement n'est pas nécessairement libre (cette confusion est entretenue par le double-sens du mot anglais free).

Le principe de licence libre n'interdit pas en effet de faire payer l'accès à l'œuvre, il garantit juste des libertés sur l'œuvre une fois celle-ci obtenue.

Des variantes de licences libres sont apparus au fil du temps et elles s'inspirent donc du même mécanisme, mais ne sont pas motivées par les mêmes aspirations d'équité. Ainsi elles peuvent par exemple interdire la modification ou l'utilisation dans certains contextes (commercial, militaire, politique, etc.). Nous examinons maintenant certaines de ces licences bien connues des informaticiens.

Principe de la licence GPL

La GPL ne donne pas à l'utilisateur des droits de redistribution sans limite. Le droit de redistribuer est garanti seulement si l'utilisateur fournit le code source de la version modifiée. En outre, les copies distribuées, incluant les modifications, doivent être aussi sous les termes de la GPL. Cette condition est connue sous le nom de copyleft.

La GPL met en œuvre la notion de copyleft. Le bien connu copyright garantit exclusivement les droits de l'auteur, le copyleft s'attarde tout particulièrement sur les droits des utilisateurs, et vise à préserver la liberté d'utiliser, d'étudier, de modifier et de diffuser le logiciel et ses versions dérivées.

La GNU GPL a une licence sœur, la LGPL (GNU Lesser General Public License), une version modifiée pour être moins contraignante quant à son utilisation dans un contexte de cohabitation avec des logiciels propriétaires. La GNU GPL a une autre licence sœur, la GFDL (GNU Free Documentation License) qui est applicable aux manuels, livres ou autres documents écrits plutÃ´t qu'aux logiciels ; cependant on peut très bien choisir à sa place la GPL, qui est tout à fait applicable à un livre, article sur le Web ou autre création.

Certaines licences libres ne peuvent pas se combiner avec la licence GPL. Les contraintes imposées par chaque licence sont parfois incompatibles et il est alors interdit de les mélanger. La Free Software Foundation (FSF) publie la liste des licences considérées comme compatibles avec la GPL.

Le tribunal de grande instance de Paris a jugé applicable la licence GPL (version 2) en France le 28 mars 2007.

Variation française

Les licences CeCILL ont été mises en place afin de permettre à des établissements publics de publier leurs travaux logiciels sous licence libre rédigée selon le droit français. La licence CeCILL est compatible, depuis sa version 2, avec la licence publique générale GNU. CeCILL est l'abréviation de «CEA CNRS INRIA logiciel libre».

Elle a été créée pour plusieurs raisons : garantir aux créateurs et aux utilisateurs de logiciel libre le respect du droit français en matière de responsabilité civile ; en matière de droit de la propriété intellectuelle ; et de garantir le respect des principes du logiciel libre, à savoir libre accès au code source, libre utilisation, libre modification, libre redistribution.

6.4 Modèles économiques de l'Internet

Cette partie a été rédigée par Ahmed Elleuch, Ahmed.Elleuch@ensi.rnu.tn, *Laboratoire Cristal/ENSI*, Université de La Manouba, Tunisie.

6.4.1 Introduction

DEPUIS LE MILIEU des années quatre vingt dix, l'Internet a commencé à s'imposer en tant que média mondial et de masse à coté des médias traditionnels. Les avancées rapides des technologies de l'information et de la communication (TIC), dont l'Internet est l'objet mais aussi le support, et la multiplication des activités économiques sur l'Internet ont donné naissance à la *nouvelle économie* ou plus précisément à *l'économie de l'Internet*.

Cette économie correspond à une nouvelle ère où pratiquement chaque organisation se doit d'utiliser l'Internet comme un media majeur à travers lequel les affaires et les activités commerciales sont développées. Fort de cet état de fait, en promettant des gains substantiels, plusieurs entreprises en phase de construction, dites *start-ups* ou *jeunes pousses*, ont ainsi attiré des capitaux risques considérables rien qu'en s'introduisant en bourse et en profitant d'opérations de rachat. Cependant, l'engouement des investisseurs pour les secteurs liés aux TIC et particulièrement à l'Internet a conduit, en mars 2000, à l'éclatement de la *bulle Internet*.

Parmi les causes de cet éclatement, nous citons les mauvaises estimations, à long terme, relatives à la pression des taux d'intérêt et à la consommation du capital, qui ont induit en erreurs des investisseurs crédules. De plus, la nature des produits et des services offerts a été aussi et sans doute un autre facteur à l'origine de l'échec de certaines start-ups. Boo, Vizzavi ou ZeBank sont des exemples de start-up qui se sont soldés par des pertes importantes de capitaux. Néanmoins, malgré la crise par laquelle est passée l'économie de l'Internet, l'usage de l'Internet n'a pas cessé de se généraliser à un rythme soutenu, comme en témoignent des exemples de start-ups devenus rapidement rentables tels que Ebay, Amazon ou Boursorama.

Il existe différentes activités qui s'apparentent à l'économie de l'Internet. Nous allons donc tenter de cerner ces activités et surtout chercher à comprendre l'essentiel des modèles économiques régissant ces activités. En s'appuyant sur une décomposition en couches de l'économie de l'Internet, la section 6.4.2 identifie les acteurs au centre de cette économie. L'absence d'un opérateur qui contrôle le réseau Internet, la gratuité inhérente à la non prise en compte de l'utilisation des ressources de ce réseau et enfin les problèmes liés à la qualité des services, nous amènent à nous focaliser sur les modèles économiques qui régissent l'infrastructure de l'Internet. C'est l'objet de la section 6.4.3. Bien plus qu'une infrastructure, l'Internet est le média autour duquel se développent, à un autre niveau, des activités économiques d'intermédiation. La section 6.4.4 donne un aperçu sur les principaux modèles appliqués à ce niveau.

6.4.2 Caractérisation de l'économie de l'Internet

Une étude[8] conduite par le CERC (*Center for Research in Electronic Commerce*) de l'université du Texas a tenté de caractériser l'économie de l'Internet

[8] Voir http://cism.mccombs.utexas.edu/works/articles/internet_economy.pdf

en reprenant la démarche conceptuelle de décomposition des réseaux en couches. Quatre niveaux ont été ainsi dégagés. Afin de repérer les activités économiques, l'étude a identifié les différents types d'acteurs propres à chacun de ces niveaux.

- Le premier niveau est celui de *l'infrastructure Internet*. Ce niveau inclut les constructeurs d'équipements de réseaux et d'ordinateurs, les fournisseurs de dorsales et d'accès à l'Internet, les développeurs de logiciels de réseaux et les sociétés de service assurant le support pour le déploiement des systèmes, des réseaux et des solutions de sécurité.

- Le second niveau est relatif aux *applications Internet*. Il s'agit de tous les produits et services bâtis au dessus de l'infrastructure Internet et qui rendent possible l'activité commerciale en ligne. Nous retrouvons dans cette couche les consultants Internet, les applications de commerce électronique, les applications multimédia, les logiciels de développement du Web, les moteurs de recherche et les bases de données Web.

- Le troisième niveau est celui des *intermédiaires de l'Internet* qui mettent en relation les acheteurs et les vendeurs et qui catalysent les transactions entre eux à travers l'Internet. Ce niveau comporte les plateformes de e-sourcing permettant de retrouver de nouveaux fournisseurs ayant de meilleures offres, les sites d'enchères, les fournisseurs de contenu, la publicité, les agences de voyage et les courtiers en ligne.

- Le quatrième niveau reflète le *commerce sur l'Internet* autrement dit la vente en ligne de produits et de services. Ce niveau est relatif aux commerçants électroniques, aux industriels, aux compagnies aériennes ...

Cette décomposition en couches est purement logique et n'empêche pas qu'une même firme intervienne à différents niveaux, tel est le cas de Microsoft qui intervient particulièrement dans les trois premières couches.

La même étude - ayant retenu ces différentes couches dans l'objectif de dégager des indicateurs de mesure de l'économie de l'Internet - souligne que les deux premières couches représentaient à l'époque, en 1998, la moitié de l'économie de l'Internet et que les activités d'intermédiation ont de plus en plus une importance énorme dans la croissance de l'économie de l'Internet. Depuis, l'économie de l'Intermédiation n'a pas cessé de s'étendre avec de nouveaux acteurs. Nous allons donc nous focaliser, dans la suite, sur les concepts et les modèles liés à cette économie.

Néanmoins, vu le caractère particulier de la tarification des réseaux et sans chercher à décrire toutes les activités économiques qui relèvent de la couche infrastructure, nous allons aussi cerner les particularités de l'économie des opérateurs de l'Internet et les enjeux associés.

6.4.3 Economie des opérateurs d'infrastructure de l'Internet

Allocation des ressources et tarification

Au départ avec les réseaux longues distances tels que les réseaux téléphoniques, les réseaux RNIS ou encore les réseaux de transmission de données, l'économie des réseaux pouvait supposer qu'un réseau était la propriété d'un même opérateur. Les efforts n'allaient pas dans le sens de l'interopérabilité de ces réseaux. L'étude économique devait alors optimiser la rentabilité d'un réseau et donc l'allocation des ressources en appliquant une tarification appropriée à ce réseau.

Avec l'Internet, le réseau des réseaux, l'interconnexion de réseaux hétérogènes est devenue une réalité. Les différents acteurs de l'Internet, particulièrement les fournisseurs de dorsales et d'accès, ont adhéré au principe d'interconnexion et d'interopérabilité. De nouvelles méthodes de tarification ont été ainsi utilisées. Sans pouvoir garantir une bande passante de bout en bout, chaque fournisseur d'accès ou de dorsale devait savoir dimensionner le réseau de façon à ce que, d'une part, l'infrastructure déployée soit rentable et que, d'autre part, les situations de congestion ne soient pas une rai son d'insatisfaction des abonnés. Cependant, il s'est avéré que les surcharges n'étaient pas rares.

Devant cet état de fait, les économistes pensent qu'il faudrait différencier les services en fonction des besoins et appliquer des modèles de tarification basés sur l'utilisation. Parmi les modèles proposés, nous citons :

- les modèles de tarification basée sur une réservation de bande passante pour les trafics qui nécessitent des garanties de qualité de service,
- les modèles de tarification sans garanties de service où les réseaux les plus chers devraient être moins encombrés,
- les modèles avec enchères pour la bande passante,
- les modèles à priorités avec un prix fixe ou dynamique en fonction de l'encombrement.

D'autres continuent à estimer qu'il faudrait tout simplement augmenter la capacité des voies de communication et garder le principe de tarification tel qu'il est, car c'est l'une des principales raisons du succès de l'Internet.

Tarification pour les usagers

Jusqu'à maintenant, la tarification pour les usagers de l'Internet a été marginale et va jusqu'à la gratuité. Les fournisseurs d'accès à l'Internet proposent généralement, différentes offres possibles basées sur les modèles d'abonnement, de facturation suivant la consommation ou encore basées sur la gratuité de l'accès. Les offres généralement proposées sont les suivantes :

- Abonnement avec un accès illimité et une tarification forfaitaire, seules les communications sont aux frais de l'abonné. D'autres services sont en plus

offerts, en particuliers des boites de courriers électroniques et des pages Web personnelles.

- Abonnement avec un accès limité en nombre d'heures et avec un mode de payement comparable au précédent, sauf que l'abonnement est moins cher et que le prix est majoré par tranche de temps supplémentaire.

- Accès sans abonnement mais payement d'un coût de la communication à la minute. Ce mode est prévu pour les personnes qui se connectent rarement à l'Internet.

- Accès gratuits, les fournisseurs qui offrent un tel accès tirent leurs bénéfices de la publicité ou en fournissant d'autres services et en appliquant d'autres modèles économiques qui s'apparentent à l'économie de l'intermédiation.

La diversification des services offerts et le recours aux modèles économiques de l'intermédiation deviennent de plus en plus une nécessité pour les fournisseurs d'accès. Ces derniers sont alors confrontés à la difficulté de trouver le meilleur rapport entre les revenus provenant des utilisateurs et les revenus provenant des partenaires pour lesquels des services d'intermédiation sont rendus.

Tarification pour les fournisseurs

Du coté des fournisseurs d'accès et de dorsales , la tarification se base sur des accords de "*peering*" ou de "*transit*". Dans le cas d'un accord de peering, deux fournisseurs d'accès (ou plus), de tailles comparables, peuvent établir une liaison directe entre eux afin d'échanger directement des paquets sur cette liaison. Un tel accord permet de ne pas payer des redevances à un fournisseur tiers. Les accords de peering sont le plus souvent gratuits. Lorsqu'il s'agit d'un accord de transit, chaque fournisseur accepte de laisser passer les paquets qui ne lui sont pas destinés. Un tel accord est généralement payant.

Le peering a dominé dans les premiers accords. Sa mise en place ne pose pas de difficultés techniques. Depuis plusieurs années, il a été remis en cause par les grands fournisseurs à cause de l'asymétrie qui peut exister au niveau des échanges et à cause du problème des "passagers clandestins" qui cherchent à utiliser un réseau pour transiter vers un autre à travers une liaison de peering. A ceci se rajoute le fait qu'en l'absence d'un cadre législatif adapté, les différents se règlent, en général, en fonction de la pression qu'un fournisseur peut appliquer sur l'autre.

Le succès d'un opérateur est donc tributaire de la part du marché qu'il détient. Plus un opérateur arrive à étendre cette part, plus il lui sera possible de faire évoluer ses propres infrastructures tout en mutualisant les investissements conséquents.

6.4.4 Economie de l'intermédiation

Les enjeux de l'intermédiation

A l'origine l'intermédiation était une activité qui consistait à collecter des fonds et à les mettre à la disposition des tiers. Les banques sont des exemples d'intermédiaires. L'intermédiation se généralise à d'autres secteurs et consiste à optimiser *la chaîne de valeur* en évitant le recours aux intermédiaires traditionnels tels que les distributeurs.

Avec les possibilités de communication directe entre acheteurs et vendeurs à travers l'Internet, des phénomènes de *désintermédiation* et de *réintermédiation* sont entrain de conduire à la disparition de certains intermédiaires et en contre partie à l'apparition de nouveaux intermédiaires.

Cette réaffectation des marges de distribution a permis à des cybercommerçants de trouver les moyens de proposer des réductions de prix significatives. Il est clair que la réussite des intermédiaires tient à leur aptitude à apporter une réelle valeur ajoutée aux consommateurs et aux fournisseurs. A cet effet, un intermédiaire a besoin de développer une bonne connaissance des désirs et des comportements des consommateurs. Ils doit aussi développer et rentabiliser son concept en adoptant le modèle économique et plus précisément le modèle d'affaire "*business model*" le plus approprié.

Il existe de nombreux modèles d'affaire décrits dans la littérature[9]. Dans ce qui suit, nous allons passer en revue les principaux modèles propres à la couche d'intermédiation.

Le modèle marchand

Ce modèle fait intervenir des grossistes ou des détaillants qui assurent la vente en ligne en se basant sur des prix de catalogue ou sur la vente aux enchères. Les distributeurs *"Click & Mortar"* sont ceux qui ont ajouté des activités en ligne à leurs activités classiques fondées sur des points de vente physiques. Les *marchands virtuels* sont des marchands en ligne sans point de vente (Amazon).

A l'encontre de ce modèle, il existe le *modèle manufacturier* suivant lequel les producteurs cherchent à vendre directement leurs produits à travers le Web. Ils espèrent ainsi éviter les intermédiaires et donc les surcoûts des circuits de distribution.

Le modèle publicitaire

En plus des contenus propres qu'un site Web a pour rôle de fournir, il pourrait entretenir des bannières publicitaires qui seraient une source majeure ou unique

[9] Voir http://www.centor.ulaval.ca/cda/IndustrieMusique/SituationActuelle.aspx
http://digitalenterprise.org/models/models.html#Infomediary
http://www.pch.gc.ca/progs/pcce-ccop/reana/pubs/economic_model/5_f.cfm#3

de ses revenus. Comme contenus, souvent accompagnés par ces bannières, nous citons particulièrement les articles (01net) et les moteurs de recherche (Yahoo).

Un des indicateurs clés en publicité est le CPM: le Coût par Mille. Il correspond au montant perçu à chaque fois que mille bannières publicitaires sont visualisées. A coté du CPM, il existe le CPC et le CPA. Le CPC est le Coût par Clic sur un bouton ou une bannière. Le CPA est le Coût par Action, c'est-à-dire, lorsque le visiteur finit par transformer sa visite en un achat, en une inscription à un service ou en toute autre action.

Le modèle de l'affilié

Ce modèle rejoint le modèle publicitaire dans le sens où un affilié fournit un lien vers un annonceur. Au cas où ce dernier réalise des ventes à travers ce lien, un pourcentage des ventes est versé à l'affilié. Ce modèle est basé sur la performance des ventes ainsi réalisées "*pay-for-performance*"

Le modèle infomédiaire

Le terme infomédiaire provient de la contraction des mots "information" et "intermédiaire". Un infomédiaire cherche à recueillir et à vendre de l'information stratégique sur les acheteurs potentiels, comme par exemple les habitudes des utilisateurs. Bien souvent, les infomédiaires amènent les utilisateurs à souscrire à un service, éventuellement gratuit, et par la même collecter les informations utiles (NetZero). L'examen de la navigation Web des consommateurs est une autre source d'information. Grâce à l'ensemble de ces informations, un infomédiaire arrive à orienter un annonceur afin que ce dernier puisse atteindre un public visé.

Inversement, l'infomédiaire peut aussi orienter les consommateurs en collectant des informations sur les fournisseurs, les produits et les services. L'objectif est alors de faciliter la tâche aux consommateurs pour qu'ils puissent repérer les meilleures offres (*e-sourcing*). Cette fonction de l'intermédiaire peut se présenter sous la forme d'un annuaire ou d'un site de recherche de biens ou de services. Un autre procédé appliqué par les infomédiaires est de monter des forums pour que les utilisateurs apportent leurs expériences et leurs avis sur certains produits ou services (ePinion).

L'infomédiaire est donc amené à collecter des informations décrivant les consommateurs et/ou les fournisseurs et à agréger ces informations de façon à les rendre plus facile à exploiter. Afin d'être plus efficace, les infomédaires ont tendance à se spécialiser en visant les acteurs d'un même secteur. Ainsi, d'un commerce *B2C* (*Business To Consumer*) les infomédiaires s'orientent davantage vers un commerce entre professionnels *B2B* (*Business to Business*).

La réussite d'un infomédiaire tient à sa capacité à produire une valeur ajoutée aux informations brutes. Sa fonction d'aiguilleur lui permet de s'insérer dans une chaîne de valeurs sous la forme d'un portail dit de *commutation* ou de *routage*. Bien plus, les infomédaires ont tendance à devenir des courtiers qui facilitent les transactions entre les acheteurs et les vendeurs sans, forcément, une implication

dans les échanges réels de marchandises ou de services. Le modèle de courtage est davantage développé dans la section suivante.

Le modèle de courtage

Ce modèle s'appuie sur un courtier dont le rôle est de rapprocher les fournisseurs et les clients afin de favoriser et de faciliter les transactions. Les bénéfices du courtier proviennent de commissions ou de rémunérations forfaitaires généralement appliquées aux transactions réalisées.

Dans le cas d'une place de marché en ligne, le courtier a pour rôle d'offrir une gamme complète de services couvrant l'évaluation, la négociation et l'accomplissement d'un marché (ChemConnect).

Un autre modèle consiste à créer un site commercial virtuel qui présente les produits de différents commerçants (Amazon). Le courtier facture alors les commerçants sous forme d'honoraires mensuels ou d'honoraires d'installation.

Il existe aussi un modèle pour la vente aux enchères, eBay, Auctionet et OnSale sont des exemples de courtiers utilisant ce modèle. Certains courtiers appliquent la vente aux enchères inversée : les acheteurs décrivent leurs besoins et combien ils sont prêts à payer et le courtier cherche à trouver les offres qui répondent à ces demandes. Priceline est un exemple où la vente aux enchères inversée est appliquée pour l'acquisition de billets d'avion.

Mercata a fait breveter un modèle de courtier qui consiste à rassembler le plus possible d'acheteurs pour que le courtier puisse commander une grande quantité et profiter ainsi d'une remise substantielle.

D'autres modèles de courtage existent, même s'ils diffèrent dans leur approche, le rôle du courtier est toujours celui de *facilitateur d'affaires* (ou "*market-makers*").

Le modèle communautaire

La viabilité de ce modèle est basée sur la fidélité des utilisateurs. Chaque membre d'une communauté devrait s'investir activement pour un projet donné. Les revenus peuvent provenir de la vente de produits auxiliaires, de frais d'abonnement ou des contributions de volontaires. Les revenus peuvent aussi provenir de la publicité. Les développements de logiciels et de contenus libres sont des exemples appliquant ce modèle.

6.4.5 Conclusions

Aucun des modèles économiques présentés n'est propre à l'Internet et chacun d'eux a déjà fait ses preuves dans le monde classique des affaires. Cependant, l'application de ces modèles à l'économie de l'Internet nécessite des adaptations qui tiennent compte de certaines spécificités du réseau Internet. Particulièrement, la difficulté d'apprécier la valeur ajoutée d'un contenu, l'absence d'un contact

physique et la facilité de monter des sites frauduleux suscitent la méfiance. Or la confiance est un pré requis à toute transaction commerciale.

Il ne suffit pas d'attirer la clientèle, l'intermédiaire doit aussi inspirer confiance et savoir fidéliser sa clientèle. C'est pourquoi les sites peu connus trouvent des difficultés à s'introduire dans le marché et à pouvoir se mettre en valeur. En revanche, les sites correspondant à des marques bien connues profitent d'une solide réputation déjà consolidée.

Pour gagner la confiance de la clientèle, les entreprises inconnues ont tendance à monter des partenariats avec des entreprises bien réputées. Autrement , ces petites entreprises sont souvent vouées à la faillite ou du moins aux rachats par de plus grandes entreprises. En outre, la crise de financement qu'a connu l'économie de l'Internet a amené les entreprises à s'unir davantage et à concentrer leurs moyens et à élargir leur clientèle. Cette tendance s'applique aussi bien pour les opérateurs d'infrastructure que pour les intermédiaires.

Enfin, il convient de revenir sur la légitimité et le bien fondé du principe de gratuité qui a caractérisé la plus part des services sur l'Internet. Avant de conquérir le grand public, l'Internet s'est développé tout d'abord dans les milieux universitaires sur la base d'infrastructures dédiées qui, une fois installées, ne nécessitaient pas des coûts directs d'utilisation, mis à part les frais de maintenance. C'est ainsi que le principe de gratuité des services Internet s'est instauré.

Suivant ce principe, les frais de mise en place et de maintenance des infrastructures et des services devraient être pris en charge par des institutions publiques ou privées et non pas par les personnes affiliées à ces institutions. On a même cru que la gratuité pourrait être assurée grâce aux revenus publicitaires. Cependant, l'éclatement de la bulle Internet n'a pas donné raison à cette croyance. Hormis les organismes publics ou subventionnés, il n'est plus possible pour les institutions et les entreprises, qui fournissent des services sur l'Internet, de continuer à exister que si elles arrivent au moins à combiner les services non payants avec d'autres services payants.

Liste de sites à explorer

- http://www.edubourse.com/guide/guide.php?fiche=economie-internet
- http://barthes.ens.fr/atelier/articles/serval99.html
- http://www.educnet.education.fr/ecogest/veille/economie/eco12.htm
- http://cktse.eie.polyu.edu.hk/eie417-2001/lecture13-14.html
- http://mission-dti.inria.fr/Rapport/economique.html
- http://www.chemconnect.com/
- http://travel.priceline.com/
- http://www.ebay.com/

- http://www.amazon.com/
- http://www.netzero.com/
- http://www.epinions.com/
- http://www.boursorama.com/