I3 – Algorithmique et programmation
Introduction à l'algorithmique

Cours n°1
Variables et tests

Camille Coti
camille.coti@iutv.univ-paris13.fr

IUT de Villetaneuse, département R&T

2011 – 2012

Introduction

Étymologie du mot "informatique"

Formé de la contration des mots **information** et **automatique**.

- L'informatique est un outil de traitement automatique de l'information.
- On doit alors **définir** comment des informations vont être traitées (automatiquement) par l'ordinateur.

La science informatique n'est pas plus la science des ordinateurs que l'astronomie n'est celle des télescopes – Edsger Dijkstra

- L'ordinateur est un **outil** qui traite l'information comme le programme lui dit de la traiter.

Pourquoi l'algorithmique

Qu'est-ce qu'un algorithme ?

- Un algorithme définit ce que fait un programme
- Il définit quel comportement suivre selon la situation rencontrée

Algorithme : définition

Série d'instructions qui doit être exécutée par un programme.

Définition de Wikipedia :

- Processus systématique de résolution d'un problème permettant de décrire les étapes vers le résultat;
- Suite finie et non-ambiguë d'instructions permettant de donner la réponse à un problème.

Comment décrire un algorithme

Définition de Wikipedia :

- L'**algorithmique** est l'ensemble des règles et des techniques qui sont impliquées dans la définition et la conception d'algorithmes.

Algorithmique

Formalisme permettant de décrire la série d'instructions exécutée par un porgramme indépendamment d'un langage de programmation en particulier.

Les algorithmes sont décrits en **pseudo-code**, compréhensible par le lecteur humain mais assez précis pour transcirre les structures et les instructions de l'algorithme.

Par quoi est constitué un algorithme

Un algorithme permet d'obtenir un résultat à partir de données d'entrée. Il est donc constitué des éléments suivants :

- Un début et une fin ;
- Un nom ;
- Des données d'entrée ;
- Des données de sortie, qui sont le résultat du calcul effectué par l'algorithme ;
- Un ensemble d'instructions exécutées par l'algorithme.

Exemple : algorithme de calcul d'une valeur absolue

```
1 début fonction abs( i: Entier ): Entier
2 |   si i > 0 alors
3 |   |   r ⟵ i
4 |   sinon
5 |   |   r ⟵ −i
6 |   finsi
7 |   retourner r
8 fin fonction
```

Importance d'un bon algorithme

Compréhension et modélisation du problème

- L'algorithme modélise le comportement du programme

Correction du résultat

- Algorithme faux $\rightarrow$ résultat du calcul faux

Efficacité du calcul

- Coût d'un calcul = nombre d'opérations et quantité d'information manipulée
- Ces quantités sont définies par l'algorithme
- Algorithme efficace $\rightarrow$ calcul efficace (et inversement)

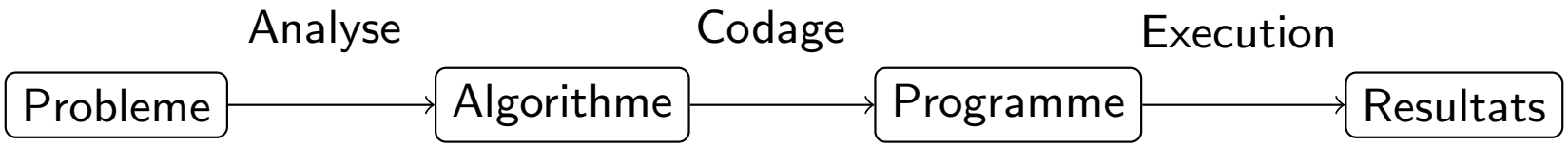
L'étude algorithmique d'un problème est indispensable

La bonne conception d'un algorithme est fondamentale et absolument nécessaire préalablement à l'écriture d'un programme informatique.

Généricté des algorithmes

Les algorithmes sont écrits dans un langage de descriptions des algorithmes (pseudo-code)

- Indépendant du langage de programmation
  $\rightarrow$ Un algorithme peut être implémenté dans n'importe quel langage



Représentation des données

Définition

Une **variable** correspond à l'emplacement mémoire d'une donnée. Elle sert à stocker une valeur d'un certain **type** à un instant donné de l'exécution du programme.

Les variables d'un algorithmes sont :

- D'entrée : données d'entrée de l'algorithme
- De sortie : résultat du calcul effectué par l'algorithme
- Interne à l'algorithme : ni d'entrée ni de sortie mais utilisée à l'intérieur de l'algorithme

Type de variables

Définition

Le **Type** d'une variable est le type de donnée qui pourra être contenu dans cette variable.

Exemples :

- Entier : tout nombre entier
- Booléen : vrai ou faux
- Caractère
- Nombre réel
- Chaîne de caractères
- Tableau...

On ne peut pas mettre une donnée d'un type dans une variable d'un autre type

- Exception : un transtypage est parfois possible (entier dans réel...)

Affectation

Affectation d'une valeur à une variable

Pour écrire une valeur dans une variable, on dit que l'on **affecte** cette valeur à la variable. On le note de la façon suivante :

$$variable \longleftarrow valeur$$

```
1 début
2 |   maVariable : Entier
3 |   maVariable ⟵ 42
4 fin
```

- On **déclare** le type de la variable avant de l'utiliser
- Les déclarations sont généralement rassemblées au début de l'algorithme

Les tableaux

Tableaux

Un tableau est un type particulier de variable. Il contient un **ensemble de variables** de même type, stockées de façon contigüe en mémoire.

Exemple : tableau d'entiers

3	5	2	1	12	5	2	9

Caractéristiques d'un tableau

- Un tableau a une taille fixe
- Il contient un certain type de données

Déclaration d'un tableau

- On le déclare avec le type de données qu'il contient et sa taille

```
1 début
2 |   monTableau[10] :
  |   Tableau d'Entiers
3 fin
```

Les tableaux (suite)

Tableau à plusieurs dimensions

- On donne la taille dans chaque dimension
  - Exemple en 2D (ordre C) : d'abord le nombre de lignes, puis le nombre de colonnes

Accès aux données d'un tableau

- Les cases du tableau sont numérotées de 0 à $N-1$ (si N est la taille du tableau)
- On accède aux données d'un tableau en utilisant l'indice dans ce tableau

```
1 début
2 |   /* Variables d'entree */
3 |   maMatrice[10][10] : Tableau d'Entiers
4 |   /* Variables de sortie */
5 |   i : entier
6 |   /* Affectation */
7 |   i ⟵ maMatrice[2][3]
8 fin
```

Les tests

Un test est une **structure conditionnelle** : les instructions exécutées dépendent de la réalisation ou non d'une condition.

Définition

Un **test** définit une condition et un comportement à suivre si elle est réalisée. Optionnellement, il peut définir un comportement à suivre dans le cas contraire.

Syntaxe :

- La **condition** est donnée entre les mot-clés **si** et **alors**
- L'**action réalisée si la condition est vérifiée** est donnée après le mot-clé **alors**
- Si on donne une **action à réaliser si la condition n'est pas vérifiée**, elle est introduite par le mot-clé **sinon**
- Le test est terminé par le mot-clé **finsi**

```
1 si condition alors
2 |   action1
3 sinon
4 |   action2
5 finsi
```

Condition d'un test

La condition d'un test est une **expression booléenne**

- Valeurs possibles : VRAI ou FAUX

Elle est évaluée pour décider quel bloc d'instructions exécuter.

On peut tester :

- l'égalité entre deux variables : $var1 == var2$
- la non-égalité entre deux variables : $var1 != var2$
- une relation d'ordre entre deux variables : $var1 > var2$
- ou toute expression renvoyant $Vrai$ ou $Faux$

```
1 début
2 |   maVar : Entier
3 |   maVar ⟵ 0
4 |   si maVar < 5 alors
5 |   |   maVar ⟵ maVar + 1
6 |   finsi
7 fin
```

On peut combiner des expressions booléennes en utilisant les opérateurs logiques ET et OU (attention aux parenthèses) :
$((var1 == var2)\ ET\ (var1 > 0))\ OU\ (var2 < 0)$.

Blocs d'instructions

Un algorithme est structué par **blocs d'instructions** contenant plusieurs instructions à exécuter séquentiellement.

- Exemple : les instructions à exécuter si la condition d'un test est réalisée
  - Les lignes 5 et 6 sont un bloc
  - La ligne 8 est un bloc
- Des blocs peuvent être **imbriqués**
  - Un bloc d'instructions peut être inclus dans un autre bloc.
  - Les lignes 2 à 9 sont un bloc
  - Les deux blocs dans la condition sont des blocs imbriqués dans ce bloc

```
1  début
2      maVar : Entier
3      maVar ⟵ 0
4      si maVar < 5 alors
5          maVar ⟵ maVar + 1
6          afficher( mavar )
7      sinon
8          maVar ⟵ 0
9      finsi
10 fin
```

- Des instructions d'un bloc sont décalées vers la droite au même niveau
- Un bloc est indiqué par une ligne verticale sur la gauche

Tests imbriqués

On peut **imbriquer** des tests, c'est-à-dire qu'un test peut être effectué dans le corps d'un autre test.

- On effectue un test ligne 4
- Si la condition est réalisée, on exécute le bloc situé entre les lignes 5 et 8
  - On effectue alors un autre test ligne 6
  - Si la condition est réalisée, on exécute le bloc ligne 7
- Sinon, on exécute le bloc ligne 10.

Le test situé entre les lignes 6 et 8 est imbriqué dans le test situé entre les lignes 4 et 11.

```
1  début
2      maVar : Entier
3      maVar ⟵ 0
4      si maVar < 5 alors
5          maVar ⟵ maVar + 1
6          si maVar > 2 alors
7              afficher( mavar )
8          finsi
9      sinon
10         maVar ⟵ 0
11     finsi
12 fin
```