0-M02 – Introduction à la programmation
Introduction à l'algorithmique

Cours n°1
Variables, tests et structures de contrôle

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

IUT de Villetaneuse, département R&T

2011 – 2012

Site web

Slides des cours, versions électroniques des polys, TD, TP...
`http://www.lipn.fr/~coti/cours`

Introduction

Étymologie du mot "informatique"

Formé de la contraction 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 programme indépendamment d'un langage de programmation en particulier.

Les algorithmes sont décrits en **pseudo-code**, compréhensible par le lecteur humain mais assez précis pour transcrire 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)

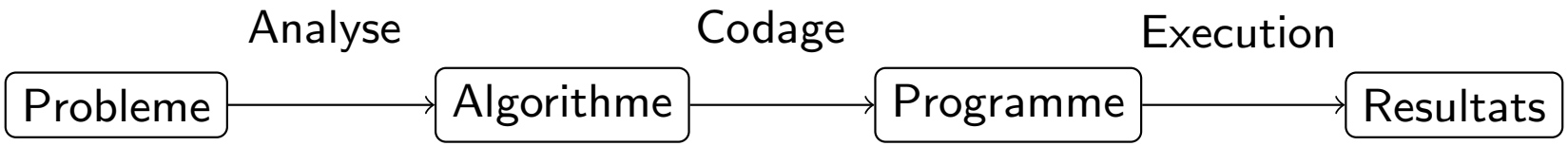
L'étude algorithmique d'un problème est indispensable

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

Généricité 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 contiguë 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
```

Booléen

Booléen

Un booléen est une variable à deux états : **Vrai** ou **Faux**. On représente aussi parfois ces deux états par 1 et 0.

Les opérateurs de comparaison renvoient un booléen. Exemple :

- $1 > 0$ renvoie Vrai
- $1 == 0$ renvoie Faux

Lorsqu'une condition est évaluée, on regarde sa valeur (booléen). Exemple :

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

- $maVar$ est initialisé à 0
- On teste $maVar < 5$
  - Équivalent à tester $0 < 5$
- La condition vaut **Vrai**
- Donc on exécute le bloc d'instruction après le mot-clé **alors**

Un peu d'algèbre de Boole

On peut évaluer des expressions booléennes en utilisant des opérateurs :

Opérateurs logiques

- ET logique : .
- OU logique : $+$
- OU exclusif : $\oplus$
- Négation : $^-$ ou !

Exemples :

- $a.b$
- $a + b$
- $a \oplus b$
- $\overline{a + b}$ =!$(a + b)$
- $a + \overline{b} = a$+!b

Tables de vérité :

.	0	1
0	0	0
1	0	1

+	0	1
0	0	1
1	1	1

$\oplus$	0	1
0	0	1
1	1	0

La **négation** transforme un Vrai en Faux et inversement :

- $\overline{Vrai} = Faux$
- $\overline{Faux} = Vrai$

Un peu d'algèbre de Boole (suite)

Prenons $a = Vrai$ et $b = Faux$.
Exemples d'expressions booléennes :

- $a + b = Vrai + Faux = Vrai$
- $a + \overline{b} = Vrai + \overline{Faux} = Vrai + Vrai = Vrai$
- $\overline{a.\overline{b}} = \overline{Vrai.\overline{Faux}} = \overline{Vrai.Vrai} = \overline{Vrai} = Faux$

Attention aux parenthèses !

- $(a + b).(c + d)$

Exemples de compositions d'expressions booléennes sur des variables en algorithmique :

- $(a > b) ET (a > 0)$
  - Si $a = 1$ et $b = 2$: $(a > b) = Faux$ donc l'expression vaut $Faux$
  - Si $a = 3$ et $b = 2$: $(a > b) = Vrai$ et $(a > 0) = Vrai$ donc l'expression vaut $Vrai$
- $(a > b) OU (a > 0)$
  - Si $a = 1$ et $b = 2$: $(a > b) = Faux$ et $(a > 0) = Vrai$ donc l'expression vaut $Vrai$
  - Si $a = 3$ et $b = 2$: $(a > b) = Vrai$ et $(a > 0) = Vrai$ donc l'expression vaut $Vrai$

Un peu d'algèbre de Boole (suite)

Prenons $a = Vrai$ et $b = Faux$.
Exemples d'expressions booléennes :

- $a + b = Vrai + Faux = Vrai$
- $a + \overline{b} = Vrai + \overline{Faux} = Vrai + Vrai = Vrai$
- $\overline{a.\overline{b}} = \overline{Vrai.\overline{Faux}} = \overline{Vrai.Vrai} = \overline{Vrai} = Faux$

Attention aux parenthèses !

- $(a + b).(c + d)$

Exemples de compositions d'expressions booléennes sur des variables en algorithmique :

- $(a > b)ET(a > 0)$
  - Si $a = 1$ et $b = 2$: $(a > b) = Faux$ donc l'expression vaut $Faux$
  - Si $a = 3$ et $b = 2$: $(a > b) = Vrai$ et $(a > 0) = Vrai$ donc l'expression vaut $Vrai$
- $(a > b)OU(a > 0)$
  - Si $a = 1$ et $b = 2$: $(a > b) = Faux$ et $(a > 0) = Vrai$ donc l'expression vaut $Vrai$
  - Si $a = 3$ et $b = 2$: $(a > b) = Vrai$ et $(a > 0) = Vrai$ donc l'expression vaut $Vrai$

Un peu d'algèbre de Boole (suite)

Prenons $a = Vrai$ et $b = Faux$.
Exemples d'expressions booléennes :

- $a + b = Vrai + Faux = Vrai$
- $a + \overline{b} = Vrai + \overline{Faux} = Vrai + Vrai = Vrai$
- $\overline{a.\overline{b}} = \overline{Vrai.\overline{Faux}} = \overline{Vrai.Vrai} = \overline{Vrai} = Faux$

Attention aux parenthèses !

- $(a + b).(c + d)$

Exemples de compositions d'expressions booléennes sur des variables en algorithmique :

- $(a > b) ET (a > 0)$
  - Si $a = 1$ et $b = 2$: $(a > b) = Faux$ donc l'expression vaut $Faux$
  - Si $a = 3$ et $b = 2$: $(a > b) = Vrai$ et $(a > 0) = Vrai$ donc l'expression vaut $Vrai$
- $(a > b) OU (a > 0)$
  - Si $a = 1$ et $b = 2$: $(a > b) = Faux$ et $(a > 0) = Vrai$ donc l'expression vaut $Vrai$
  - Si $a = 3$ et $b = 2$: $(a > b) = Vrai$ et $(a > 0) = Vrai$ donc l'expression vaut $Vrai$

Un peu d'algèbre de Boole (suite)

Prenons $a = Vrai$ et $b = Faux$.
Exemples d'expressions booléennes :

- $a + b = Vrai + Faux = Vrai$
- $a + \overline{b} = Vrai + \overline{Faux} = Vrai + Vrai = Vrai$
- $\overline{a.\overline{b}} = \overline{Vrai.\overline{Faux}} = \overline{Vrai.Vrai} = \overline{Vrai} = Faux$

Attention aux parenthèses !

- $(a + b).(c + d)$

Exemples de compositions d'expressions booléennes sur des variables en algorithmique :

- $(a > b) ET (a > 0)$
  - Si $a = 1$ et $b = 2$: $(a > b) = Faux$ donc l'expression vaut $Faux$
  - Si $a = 3$ et $b = 2$: $(a > b) = Vrai$ et $(a > 0) = Vrai$ donc l'expression vaut $Vrai$
- $(a > b) OU (a > 0)$
  - Si $a = 1$ et $b = 2$: $(a > b) = Faux$ et $(a > 0) = Vrai$ donc l'expression vaut $Vrai$
  - Si $a = 3$ et $b = 2$: $(a > b) = Vrai$ et $(a > 0) = Vrai$ donc l'expression vaut $Vrai$

Un peu d'algèbre de Boole (suite)

Prenons $a = Vrai$ et $b = Faux$.
Exemples d'expressions booléennes :

- $a + b = Vrai + Faux = Vrai$
- $a + \overline{b} = Vrai + \overline{Faux} = Vrai + Vrai = Vrai$
- $\overline{a.\overline{b}} = \overline{Vrai.\overline{Faux}} = \overline{Vrai.Vrai} = \overline{Vrai} = Faux$

Attention aux parenthèses !

- $(a + b).(c + d)$

Exemples de compositions d'expressions booléennes sur des variables en algorithmique :

- $(a > b) ET (a > 0)$
  - Si $a = 1$ et $b = 2$: $(a > b) = Faux$ donc l'expression vaut $Faux$
  - Si $a = 3$ et $b = 2$: $(a > b) = Vrai$ et $(a > 0) = Vrai$ donc l'expression vaut $Vrai$
- $(a > b) OU (a > 0)$
  - Si $a = 1$ et $b = 2$: $(a > b) = Faux$ et $(a > 0) = Vrai$ donc l'expression vaut $Vrai$
  - Si $a = 3$ et $b = 2$: $(a > b) = Vrai$ et $(a > 0) = Vrai$ donc l'expression vaut $Vrai$

Un peu d'algèbre de Boole (suite)

Prenons $a = Vrai$ et $b = Faux$.
Exemples d'expressions booléennes :

- $a + b = Vrai + Faux = Vrai$
- $a + \overline{b} = Vrai + \overline{Faux} = Vrai + Vrai = Vrai$
- $\overline{a.\overline{b}} = \overline{Vrai.\overline{Faux}} = \overline{Vrai.Vrai} = \overline{Vrai} = Faux$

Attention aux parenthèses !

- $(a + b).(c + d)$

Exemples de compositions d'expressions booléennes sur des variables en algorithmique :

- $(a > b) ET (a > 0)$
  - Si $a = 1$ et $b = 2$: $(a > b) = Faux$ donc l'expression vaut $Faux$
  - Si $a = 3$ et $b = 2$: $(a > b) = Vrai$ et $(a > 0) = Vrai$ donc l'expression vaut $Vrai$
- $(a > b) OU (a > 0)$
  - Si $a = 1$ et $b = 2$: $(a > b) = Faux$ et $(a > 0) = Vrai$ donc l'expression vaut $Vrai$
  - Si $a = 3$ et $b = 2$: $(a > b) = Vrai$ et $(a > 0) = Vrai$ donc l'expression vaut $Vrai$

Un peu d'algèbre de Boole (suite)

Prenons $a = Vrai$ et $b = Faux$.
Exemples d'expressions booléennes :

- $a + b = Vrai + Faux = Vrai$
- $a + \overline{b} = Vrai + \overline{Faux} = Vrai + Vrai = Vrai$
- $\overline{a.\overline{b}} = \overline{Vrai.\overline{Faux}} = \overline{Vrai.Vrai} = \overline{Vrai} = Faux$

Attention aux parenthèses !

- $(a + b).(c + d)$

Exemples de compositions d'expressions booléennes sur des variables en algorithmique :

- $(a > b)ET(a > 0)$
  - Si $a = 1$ et $b = 2$: $(a > b) = Faux$ donc l'expression vaut $Faux$
  - Si $a = 3$ et $b = 2$: $(a > b) = Vrai$ et $(a > 0) = Vrai$ donc l'expression vaut $Vrai$
- $(a > b)OU(a > 0)$
  - Si $a = 1$ et $b = 2$: $(a > b) = Faux$ et $(a > 0) = Vrai$ donc l'expression vaut $Vrai$
  - Si $a = 3$ et $b = 2$: $(a > b) = Vrai$ et $(a > 0) = Vrai$ donc l'expression vaut $Vrai$

Un peu d'algèbre de Boole (suite)

Prenons $a = Vrai$ et $b = Faux$.
Exemples d'expressions booléennes :

- $a + b = Vrai + Faux = Vrai$
- $a + \overline{b} = Vrai + \overline{Faux} = Vrai + Vrai = Vrai$
- $\overline{a.\overline{b}} = \overline{Vrai.\overline{Faux}} = \overline{Vrai.Vrai} = \overline{Vrai} = Faux$

Attention aux parenthèses !

- $(a + b).(c + d)$

Exemples de compositions d'expressions booléennes sur des variables en algorithmique :

- $(a > b) ET (a > 0)$
  - Si $a = 1$ et $b = 2$: $(a > b) = Faux$ donc l'expression vaut $Faux$
  - Si $a = 3$ et $b = 2$: $(a > b) = Vrai$ et $(a > 0) = Vrai$ donc l'expression vaut $Vrai$
- $(a > b) OU (a > 0)$
  - Si $a = 1$ et $b = 2$: $(a > b) = Faux$ et $(a > 0) = Vrai$ donc l'expression vaut $Vrai$
  - Si $a = 3$ et $b = 2$: $(a > b) = Vrai$ et $(a > 0) = Vrai$ donc l'expression vaut $Vrai$

Un peu d'algèbre de Boole (suite)

Prenons $a = Vrai$ et $b = Faux$.
Exemples d'expressions booléennes :

- $a + b = Vrai + Faux = Vrai$
- $a + \overline{b} = Vrai + \overline{Faux} = Vrai + Vrai = Vrai$
- $\overline{a.\overline{b}} = \overline{Vrai.\overline{Faux}} = \overline{Vrai.Vrai} = \overline{Vrai} = Faux$

Attention aux parenthèses !

- $(a + b).(c + d)$

Exemples de compositions d'expressions booléennes sur des variables en algorithmique :

- $(a > b)ET(a > 0)$
  - Si $a = 1$ et $b = 2$: $(a > b) = Faux$ donc l'expression vaut $Faux$
  - Si $a = 3$ et $b = 2$: $(a > b) = Vrai$ et $(a > 0) = Vrai$ donc l'expression vaut $Vrai$
- $(a > b)OU(a > 0)$
  - Si $a = 1$ et $b = 2$: $(a > b) = Faux$ et $(a > 0) = Vrai$ donc l'expression vaut $Vrai$
  - Si $a = 3$ et $b = 2$: $(a > b) = Vrai$ et $(a > 0) = Vrai$ donc l'expression vaut $Vrai$

Un peu d'algèbre de Boole (suite)

Prenons $a = Vrai$ et $b = Faux$.
Exemples d'expressions booléennes :

- $a + b = Vrai + Faux = Vrai$
- $a + \overline{b} = Vrai + \overline{Faux} = Vrai + Vrai = Vrai$
- $\overline{a.\overline{b}} = \overline{Vrai.\overline{Faux}} = \overline{Vrai.Vrai} = \overline{Vrai} = Faux$

Attention aux parenthèses !

- $(a + b).(c + d)$

Exemples de compositions d'expressions booléennes sur des variables en algorithmique :

- $(a > b)ET(a > 0)$
  - Si $a = 1$ et $b = 2$: $(a > b) = Faux$ donc l'expression vaut $Faux$
  - Si $a = 3$ et $b = 2$: $(a > b) = Vrai$ et $(a > 0) = Vrai$ donc l'expression vaut $Vrai$
- $(a > b)OU(a > 0)$
  - Si $a = 1$ et $b = 2$: $(a > b) = Faux$ et $(a > 0) = Vrai$ donc l'expression vaut $Vrai$
  - Si $a = 3$ et $b = 2$: $(a > b) = Vrai$ et $(a > 0) = Vrai$ donc l'expression vaut $Vrai$

Un peu d'algèbre de Boole (suite)

Prenons $a = Vrai$ et $b = Faux$.
Exemples d'expressions booléennes :

- $a + b = Vrai + Faux = Vrai$
- $a + \overline{b} = Vrai + \overline{Faux} = Vrai + Vrai = Vrai$
- $\overline{a.\overline{b}} = \overline{Vrai.\overline{Faux}} = \overline{Vrai.Vrai} = \overline{Vrai} = Faux$

Attention aux parenthèses !

- $(a + b).(c + d)$

Exemples de compositions d'expressions booléennes sur des variables en algorithmique :

- $(a > b)ET(a > 0)$
  - Si $a = 1$ et $b = 2$: $(a > b) = Faux$ donc l'expression vaut $Faux$
  - Si $a = 3$ et $b = 2$: $(a > b) = Vrai$ et $(a > 0) = Vrai$ donc l'expression vaut $Vrai$
- $(a > b)OU(a > 0)$
  - Si $a = 1$ et $b = 2$: $(a > b) = Faux$ et $(a > 0) = Vrai$ donc l'expression vaut $Vrai$
  - Si $a = 3$ et $b = 2$: $(a > b) = Vrai$ et $(a > 0) = Vrai$ donc l'expression vaut $Vrai$

Un peu d'algèbre de Boole (suite)

Prenons $a = Vrai$ et $b = Faux$.
Exemples d'expressions booléennes :

- $a + b = Vrai + Faux = Vrai$
- $a + \overline{b} = Vrai + \overline{Faux} = Vrai + Vrai = Vrai$
- $\overline{a.\overline{b}} = \overline{Vrai.\overline{Faux}} = \overline{Vrai.Vrai} = \overline{Vrai} = Faux$

Attention aux parenthèses !

- $(a + b).(c + d)$

Exemples de compositions d'expressions booléennes sur des variables en algorithmique :

- $(a > b)ET(a > 0)$
  - Si $a = 1$ et $b = 2$: $(a > b) = Faux$ donc l'expression vaut $Faux$
  - Si $a = 3$ et $b = 2$: $(a > b) = Vrai$ et $(a > 0) = Vrai$ donc l'expression vaut $Vrai$
- $(a > b)OU(a > 0)$
  - Si $a = 1$ et $b = 2$: $(a > b) = Faux$ et $(a > 0) = Vrai$ donc l'expression vaut $Vrai$
  - Si $a = 3$ et $b = 2$: $(a > b) = Vrai$ et $(a > 0) = Vrai$ donc l'expression vaut $Vrai$

Un peu d'algèbre de Boole (suite)

Prenons $a = Vrai$ et $b = Faux$.
Exemples d'expressions booléennes :

- $a + b = Vrai + Faux = Vrai$
- $a + \overline{b} = Vrai + \overline{Faux} = Vrai + Vrai = Vrai$
- $\overline{a.\overline{b}} = \overline{Vrai.\overline{Faux}} = \overline{Vrai.Vrai} = \overline{Vrai} = Faux$

Attention aux parenthèses !

- $(a + b).(c + d)$

Exemples de compositions d'expressions booléennes sur des variables en algorithmique :

- $(a > b)ET(a > 0)$
  - Si $a = 1$ et $b = 2$: $(a > b) = Faux$ donc l'expression vaut $Faux$
  - Si $a = 3$ et $b = 2$: $(a > b) = Vrai$ et $(a > 0) = Vrai$ donc l'expression vaut $Vrai$
- $(a > b)OU(a > 0)$
  - Si $a = 1$ et $b = 2$: $(a > b) = Faux$ et $(a > 0) = Vrai$ donc l'expression vaut $Vrai$
  - Si $a = 3$ et $b = 2$: $(a > b) = Vrai$ et $(a > 0) = Vrai$ donc l'expression vaut $Vrai$

Un peu d'algèbre de Boole (suite)

Prenons $a = Vrai$ et $b = Faux$.
Exemples d'expressions booléennes :

- $a + b = Vrai + Faux = Vrai$
- $a + \overline{b} = Vrai + \overline{Faux} = Vrai + Vrai = Vrai$
- $\overline{a.\overline{b}} = \overline{Vrai.\overline{Faux}} = \overline{Vrai.Vrai} = \overline{Vrai} = Faux$

Attention aux parenthèses !

- $(a + b).(c + d)$

Exemples de compositions d'expressions booléennes sur des variables en algorithmique :

- $(a > b) ET (a > 0)$
  - Si $a = 1$ et $b = 2$: $(a > b) = Faux$ donc l'expression vaut $Faux$
  - Si $a = 3$ et $b = 2$: $(a > b) = Vrai$ et $(a > 0) = Vrai$ donc l'expression vaut $Vrai$
- $(a > b) OU (a > 0)$
  - Si $a = 1$ et $b = 2$: $(a > b) = Faux$ et $(a > 0) = Vrai$ donc l'expression vaut $Vrai$
  - Si $a = 3$ et $b = 2$: $(a > b) = Vrai$ et $(a > 0) = Vrai$ donc l'expression vaut $Vrai$

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 structuré 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
```

Les boucles

Condition d'arrêt

La condition d'arrêt d'une boucle est une condition (une expression booléenne) qui détermine le moment où une boucle doit arrêter d'exécuter le bloc d'instructions.

Une boucle sert à **répéter** un bloc d'instructions tant qu'une condition de continuation est satisfaite ou que la condition d'arrêt n'est pas satisfaite.
Exemples d'utilisation :

- Parcours d'un tableau, calcul itératif...

Une boucle utilise un bloc d'instructions : c'est tout le bloc d'instructions correspondant qui est répété.

Il est possible que le bloc d'instructions ne soit pas exécuté du tout (si la condition d'arrêt est déjà satisfaite) ou un nombre infini de fois (souvent un bug).

Boucle Pour

On définit un compteur et :

- Une initialisation de ce compteur
  - $i \leftarrow 0$
- Une condition d'arrêt pour sortir de la boucle
  - **à** 9
- Un pas qui modifie le compteur **à la fin** de chaque itération
  - **pas** 1

On termine le bloc avec **finpour**

Exemple : algorithme de remplissage d'un tableau de 10 cases.

```
1 début
2 |   tab[10] : Tableau d'entiers
3 |   pour i ⟵ 0 à 9 pas 1 faire
4 |   |   tab[i] = 2 * i
5 |   finpour
6 fin
```

Boucle **Pour** (suite)

Le **pas** est n'importe quel modificateur sur le compteur : il peut être négatif, non linéaire...

```
1 début
2 |   pour i ⟵ 10 à 0 pas −2 faire
3 |   |   afficher( i )
4 |   finpour
5 fin
```

Affichage par le programme :

10
8
6
4
2
0

```
1 début
2 |   pour i ⟵ 1 à 35 pas ∗2 faire
3 |   |   afficher( i )
4 |   finpour
5 fin
```

Affichage par le programme :

1
2
4
8
16
32

La boucle s'exécute tant que i est inférieur à 35 : on s'arrête à 32.

Boucle Tant que... faire

La boucle *Tant que* exécute un bloc d'instructions tant qu'une condition est vraie : c'est la **condition de boucle**.

- La condition de boucle est définie après le mot-clé **Tant que**
- L'action à effectuer est donnée entre les mot-clés **faire** et **fintq** : on définit un bloc d'instructions qui est répété

Attention aux boucles infinies !

- Il faut que la condition de boucle finisse par être invalidée...

Exemple : algorithme de calcul des puissances de 2 inférieures à 50.

```
1 début
2 |   puissance : Entier
3 |   puissance ⟵ 1
4 |   tant que puissance < 50 faire
5 |   |   afficher( puissance )
6 |   |   puissance ⟵ puissance * 2
7 |   fintq
8 fin
```

Affichage :

```
1
2
4
8
16
32
```

Boucle **Tant que... faire** (suite)

```
1 début
2 |   puissance : Entier
3 |   puissance ← 1
4 |   tant que puissance < 50 faire
5 |   |   afficher( puissance )
6 |   |   puissance ← puissance * 2
7 |   fintq
8 fin
```

Détail de l'exécution :

- $puissance$ vaut 1
- $puissance$ est-il inférieur à 50 ? oui donc on exécute le bloc
- Affichage : 1
- $puissance$ vaut 2
- Retour à la ligne 4 : $puissance$ est-il inférieur à 50 ? oui donc on exécute le bloc
- Affichage : 2
- $puissance$ vaut 4
- ...
- Lorsque $puissance$ prend la valeur 64 : la condition ligne 4 n'est plus satisfaite et on sort de la boucle

Équivalence entre les boucles **for** et **tant que ... faire**

On peut écrire une boucle **faire ... tant que** équivalente à une boucle **for** :

- Le compteur est initialisé avant d'entrer dans la boucle **faire ... tant que**
- La condition de boucle est la même que la condition d'arrêt de la boucle **for**
- Le compteur est modifié à la fin du bloc d'instruction exécuté par la boucle **faire ... tant que**

```
1 début
2 |   pour i ⟵ 0 à 9 pas 1 faire
3 |   |   tab[i] = 2 * i
4 |   finpour
5 fin
```

```
1 début
2 |   i : entier
3 |   i ⟵ 0
4 |   tant que i < 10 faire
5 |   |   tab[i] = 2 * i
6 |   |   i ⟵ i + 1
7 |   fintq
8 fin
```

Boucle **Faire ... tant que**

- La boucle **Faire ... tant que** exécute un bloc d'instructions **puis** évalue une condition de boucle
- Le bloc d'instructions est répété si la condition de boucle est satisfaite

```
1 début
2     puissance : Entier
3     puissance ← 1
4     faire
5         afficher( puissance )
6         puissance ← puissance * 2
7     tant que puissance < 20 ;
8 fin
```

Détail de l'exécution :

- $puissance$ vaut 1
- Affichage : 1
- $puissance$ vaut 2
- $puissance$ est-il inférieur à 20 ? oui donc on ré-exécute le bloc : retour à la ligne 4
- Affichage : 2
- $puissance$ vaut 4
- ...
- Lorsque $puissance$ a pris la valeur 32 : la condition ligne 7 n'est plus satisfaite et on sort de la boucle

Différence entre les boucles **Faire ... tant que** et **Tant que ... faire**

- La boucle **Faire ... tant que commence par évaluer** la condition de boucle
  - **Puis** elle exécute le bloc d'instructions si la condition de boucle est validée
- La boucle **Tant que ... faire** exécute le bloc d'instructions **puis elle évalue** la condition de boucle

Algorithme d'attente à un stop :

```
1 début
2 |   vitesse : Entier
3 |   faire
4 |   |   vitesse ⟵ 0
5 |   tant que voituresArrivent == Vrai ;
6 |   vitesse ⟵ 50
7 fin
```

Algorithme d'attente à un feu rouge :

```
1 début
2 |   vitesse : Entier
3 |   tant que couleurFeu == rouge faire
4 |   |   vitesse ⟵ 0
5 |   fintq
6 |   vitesse ⟵ 50
7 fin
```

Avec un stop on s'arrête, puis on regarde si on peut avancer. À un feu de circulation, on s'arrête si le feu est rouge ; si le feu n'est pas rouge, on avance.