

Conditionnelles et itérations

Nicolas Delestre

Nicolas.Delestre@insa-rouen.fr

Michel Mainguenaud

Michel.Mainguenaud@insa-rouen.fr



Plan...

- Rappels sur la logique
- Les conditionnelles
- Les itérations

Rappels sur la logique booléenne...

- Valeurs possibles : Vrai ou Faux
- Opérateurs logiques : non et ou
 - optionnellement ouExc lusif mais ce n'est qu'une combinaison de non , et et ou
 - $a \text{ ouExc lusif } b = (\text{non } a \text{ et } b) \text{ ou } (a \text{ et non } b)$
- Priorité sur les opérateurs : non , et , ou
- Associativité des opérateurs et et ou
 - $a \text{ et } (b \text{ et } c) = (a \text{ et } b) \text{ et } c$
- Commutativité des opérateurs et et ou
 - $a \text{ et } b = b \text{ et } a$
 - $a \text{ ou } b = b \text{ ou } a$

Rappels sur la logique booléenne...

- Distributivité des opérateurs et et ou

- $a \text{ ou } (b \text{ et } c) = (a \text{ ou } b) \text{ et } (a \text{ ou } c)$

- $a \text{ et } (b \text{ ou } c) = (a \text{ et } b) \text{ ou } (a \text{ et } c)$

- Involution

- $\text{non non } a = a$

- Loi de Morgan

- $\text{non } (a \text{ ou } b) = \text{non } a \text{ et non } b$

- $\text{non } (a \text{ et } b) = \text{non } a \text{ ou non } b$

Les conditionnelles...

- Jusqu'à présent les instructions d'un algorithme étaient **toutes** interprétées **séquentiellement**

- **Nom** : euroVersFranc2

- Rôle** : Convertisseur des sommes en euros vers le franc

- Entrée** : valeurEnEuro : Réel

- Sortie** : valeurEnFranc : Réel

- Déclaration** : tauxConversion : Réel

- début**

- tauxConversion $\leftarrow$ 6.55957

- valeurEnFranc $\leftarrow$ valeurEnEuro * tauxConversion

- fin**

- Mais il se peut que l'on veuille conditionner l'exécution d'un algorithme
 - Par exemple la résolution d'une équation du second degré est conditionnée par le signe de Δ

L'instruction **si alors sinon**...

- L'instruction **si alors sinon** permet de conditionner l'exécution d'un algorithme à la valeur d'une expression booléenne

- Sa syntaxe est :

si *expression booléenne* **alors**

suite d'instructions exécutées si l'expression est vrai

sinon

suite d'instructions exécutées si l'expression est fausse

finsi

- La deuxième partie de l'instruction est optionnelle, on peut avoir la syntaxe suivante :

si *expression booléenne* **alors**

suite d'instructions exécutées si l'expression est vrai

finsi

Exemple...

Nom : abs

Rôle : Calcule la valeur absolue d'un entier

Entrée : unEntier : **Entier**

Sortie : laValeurAbsolue : **Entier**

Déclaration : -

début

si unEntier $\geq$ 0 **alors**

 laValeurAbsolue $\leftarrow$ unEntier

sinon

 laValeurAbsolue $\leftarrow$ -unEntier

fi **nsi**

fi **n**

Exemple...

Nom : max

Rôle : Calcule le maximum de deux entiers

Entrée : lEntier1, lEntier2 : **Entier**

Sortie : leMaximum : **Entier**

Déclaration : -

début

si lEntier1 < lEntier2 **alors**

 leMaximum $\leftarrow$ lEntier2

sinon

 leMaximum $\leftarrow$ lEntier1

fi **nsi**

fi **n**

Exemple...

Nom : max

Rôle : Calcule le maximum de deux entiers

Entrée : lEntier1, lEntier2 : **Entier**

Sortie : leMaximum : **Entier**

Déclaration : -

début

leMaximum $\leftarrow$ lEntier1

si lEntier2 > lEntier1 **alors**

leMaximum $\leftarrow$ lEntier2

fi nsi

fi n

L'instruction cas...

- Lorsque l'on doit comparer une **même** variable avec plusieurs valeurs, comme par exemple :

```
si a=1 alors
    faire une chose
sinon
    si a=2 alors
        faire une autre chose
    sinon
        si a=4 alors
            faire une autre chose
        sinon
            ...
    finsi
finsi
finsi
```

- On peut remplacer cette suite de **si** par l'instruction **cas**

L'instruction cas...

- Sa syntaxe est :

cas où v vaut

$v1 : action_1$

$v2_1, v2_2, \dots, v2_m : action_2$

$v3_1 \dots v3_2 : action_3$

...

$vn : action_n$

autre : action

fi ncas

- où :

- $v1, \dots, vn$ sont des **constantes** de type **scalaire** (entier, naturel, énuméré, ou caractère)
- $action_i$ est exécutée si $v = v_i$ (on quitte ensuite l'instruction cas)
- $action$ est exécutée si $\forall i, v \neq v_i$

Exemple...

Nom : moisA30Jours

Rôle : Détermine si un mois à 30 jours

Entrée : mois : **Entier**

Sortie : resultat : **Booléen**

Déclaration :

début

cas où mois **vaut**

 4,6,9,11 : resultat ← Vrai

 autre : resultat ← Faux

fi ncas

fi n

Les itérations...

- Lorsque l'on veut répéter plusieurs fois un même traitement, plutôt que de copier n fois la ou les instructions, on peut demander à l'ordinateur d'exécuter n fois un morceau de code
- Il existe deux grandes catégories d'itérations :
 - Les itérations **déterministes** : le nombre de boucle est défini à l'entrée de la boucle
 - les itérations **indéterministes** : l'exécution de la prochaine boucle est conditionnée par une expression booléenne

Les itérations déterministes...

- Il existe une seule instruction permettant de faire des boucles déterministes, c'est l'instruction **pour**
- Sa syntaxe est :
pour *identifiant d'une variable de type scalaire* ← *valeur de début* **à** *valeur de fin*
faire
instructions à exécuter à chaque boucle
finpour
- dans ce cas la variable utilisée prend successivement les valeurs comprises entre *valeur de début* et *valeur de fin*

Exemple...

Nom : somme

Rôle : Calculer la somme des n premiers entiers positifs, $s=0+1+2+\dots+n$

Entrée : n : Naturel

Sortie : s : Naturel

Déclaration : i : Naturel

début

$s \leftarrow 0$

pour $i \leftarrow 0$ à n **faire**

$s \leftarrow s+i$

fi n**pour**

fi n

Les itérations indéterministes...

- Il existe deux instructions permettant de faire des boucles indéterministes :

- L'instruction **tant que** :

tant que *expression booléenne* **faire**

instructions

fantantque

- qui signifie que tant que l'expression booléenne est vraie on exécute les instructions

- L'instruction **répéter jusqu'à ce que** :

répéter

instructions

jusqu'à ce que *expression booléenne*

- qui signifie que les instructions sont exécutées jusqu'à ce que l'expression booléenne soit vraie

Les itérations indéterministes...

- À la différence de l'instruction *tant que*, dans l'instruction *répéter jusqu'à ce que* les instructions sont exécutées au moins une fois
- Si vous ne voulez pas que votre algorithme “tourne” indéfiniment, l'expression booléenne doit faire intervenir des variables dont le contenu doit être modifié par au moins une des instructions du corps de la boucle

Un exemple...

Nom : invFact

Rôle : Détermine le plus grand entier e telque $e! \leq n$

Entrée : $n : \mathbf{Naturel} \geq 1$

Sortie : $e : \mathbf{Naturel}$

Déclaration : $\text{fact} : \mathbf{Naturel}$

début

$\text{fact} \leftarrow 1$

$e \leftarrow 1$

tant que $\text{fact} \leq n$ **faire**

$e \leftarrow e+1$

$\text{fact} \leftarrow \text{fact} * e$

fi tantque

$e \leftarrow e-1$

fin

Explication avec n=10...

Nom : invFact

Rôle : Détermine le plus grand entier e telque $e! \leq n$

Entrée : n : Naturel ≥ 1

Sortie : e : Naturel

Déclaration : fact : Naturel

début

fact $\leftarrow$ 1

e $\leftarrow$ 1

tant que fact $\leq$ n **faire**

e $\leftarrow$ e+1

fact $\leftarrow$ fact*e

fin tant que

e $\leftarrow$ e-1

fin

	n	e	fact	fact $\leq$ n
fact $\leftarrow$ 1	10	?	1	vrai
e $\leftarrow$ 1	10	1	1	vrai
e $\leftarrow$ e+1 fact $\leftarrow$ fact*e	10	2	2	vrai
e $\leftarrow$ e+1 fact $\leftarrow$ fact*e	10	3	6	vrai
e $\leftarrow$ e+1 fact $\leftarrow$ fact*e	10	4	24	faux
e $\leftarrow$ e-1	10	3	24	faux

Un autre exemple...

Nom : calculerPGCD

Rôle : Calculer le $\text{pgcd}(a,b)$ à l'aide de l'algorithme d'euclide

Entrée : a, b : **Naturel** non nul

Sortie : pgcd : **Naturel**

Déclaration : reste : **Naturel**

début

répéter

$\text{reste} \leftarrow a \bmod b$

$a \leftarrow b$

$b \leftarrow \text{reste}$

jusqu'à ce que $\text{reste}=0$

$\text{pgcd} \leftarrow a$

fin