Structures conditionnelles [if]
Résumé de cours

Université de Haute Alsace

Unisciel algoprog Version 14 mai 2018

Table des matières

1 C++ - Résumé de cours

1.1 Définitions

Condition simple
Notée $v_1 \, \Phi \, v_2$, elle associe un opérateur de comparaison Φ (= < <= > >= <>) et deux valeurs v_1 et v_2 de même nature et délivre un résultat booléen (`Vrai` ou `Faux`).

Condition composée
Notée $b_1 \; \Psi_1 \; b_2 \; \Psi_2 \ldots$, elle associe des opérateurs logiques Ψ_i (`Et`, `Ou`, `Non`) et des valeurs booléennes b_j et délivre un résultat booléen.

Structure Si
Permet de programmer le choix binaire.

Arbre de choix
Permet de visualiser graphiquement les différents cas d'une suite de `Si`. La forme de l'arbre décrit s'il s'agit de structures `Si` imbriquées et/ou en cascade.

1.2 Conditions

Opérateurs de comparaison

Opérateur Mathématique	Signification	Équivalent Algorithmique	Équivalent C/C++
$<$	(strictement) inférieur	`a < b`	`a < b`
$\leq$	inférieur ou égal	`a <= b`	`a <= b`
$>$	(strictement) supérieur	`a > b`	`a > b`
$\geq$	supérieur ou égal	`a >= b`	`a >= b`
$=$	égalité	`a = b`	`a == b`
$\neq$	différent de (ou inégalité)	`a <> b`	`a != b`

Opérateurs logiques

Opérateur Mathématique	Signification	Équivalent Algorithmique	Équivalent C/C++
$\neg$	négation (unaire)	`Non a`	`! a`
$\wedge$	conjonction logique	`a Et b`	`a && b`
$\vee$	disjonction logique (ou inclusif)	`a Ou b`	`a || b`

Opérateurs logiques (mots clés)

Opérateur Mathématique	Signification	Équivalent Algorithmique	Équivalent C++
$\neg$	négation (unaire)	`Non a`	`not a`
$\wedge$	conjonction logique	`a Et b`	`a and b`
$\vee$	disjonction logique (ou inclusif)	`a Ou b`	`a or b`

Propriétés des opérateurs logiques

Elles sont définies par les **tables de vérité**.

x	y	$\neg$x (non x)	x $\wedge$ y (x Et y)	x $\vee$ y (x Ou y)
Vrai	Vrai	F	Vrai	Vrai
Vrai	F		F	Vrai
F	Vrai	Vrai	F	Vrai
F	F		F	F
F = Faux				

- `c1 Et c2` n'est vrai que lorsque les deux conditions sont vraies.
- `c1 Ou c2` est toujours vrai, sauf quand les deux conditions sont fausses.

Loi de De Morgan

Cette loi stipule que :

$$\text{Non(a Et b)} \Leftrightarrow \text{Non a Ou Non b}$$
$$\text{Non(a Ou b)} \Leftrightarrow \text{Non a Et Non b}$$

Priorité des opérateurs

Les opérateurs de même priorité sont regroupés sur une même ligne.

Priorité	Opérateur Algorithmique	Signification
La plus élevée	- (unaire)	Négation algébrique
	^	Puissance
	* / div mod	Multiplication, division, div. entière, modulo
	+ -	Addition et soustraction
	&	Concaténation de chaînes
	< <= > >=	Opérateurs de comparaison
	= <>	Opérateurs d'égalité
	Non	Négation logique
	Et	Et logique
La plus basse	Ou	Ou logique

Cas de combinaisons de Et et de Ou

Mettez des parenthèses :

```
(c1 Et c2) Ou c3
```

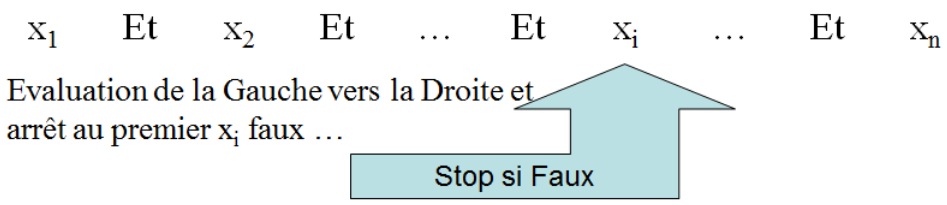
est différent de

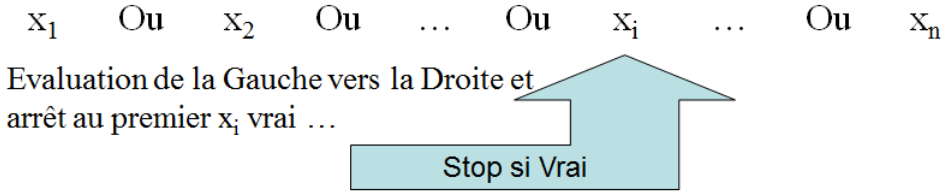
```
c1 Et (c2 Ou c3)
```

En l'absence de parenthèses, le `Et` est prioritaire sur le `Ou`.

Principe de l'évaluation paresseuse

(« lazy evaluation ») Dite aussi **évaluation court-circuitée** (« shortcut »), elle s'effectue de la **gauche vers la droite** et ne sont évalués que les conditions **strictement nécessaires** à la détermination de la valeur logique de l'expression.





Non-commutativité du Et et du Ou
L'évaluation paresseuse a pour conséquence :

`c1 Et c2`
n'est pas équivalent à
`c2 Et c1`

1.3 Sélectives Si et Si-Alors

Sélective Si

```c
if (condition)
{
  instructionsAlors;
}
else
{
  instructionsSinon;
}
```

Sélective Si-Alors

```c
if (condition)
{
  instructionsAlors;
}
```

1.4 Si-Imbriquées, Si-Cascade

Si imbriquées

```c
if (condition1)
{
  if ...
}
```

```
else
{
  if ...
}
```

C/C++ **Si en cascade**

```
if (condition1)
{
 ...
}
else
{
  if (condition2)
  {
  ...
  }
  else
  {
    if...
  }
}
```

1.5 Sélective Si-Sinon-Si

C/C++ **Sélective Si-Sinon-Si**

```
if (condition1)
{
  instructionsA1;
}
else if (condition2)
{
  instructionsA2;
}
else if...
...
else if (conditionN)
{
  instructionsAn;
}
else
{
  instructionsSinon;
}
```