

# Structures conditionnelles [if]

## Résumé de cours

Université de Haute Alsace

Unisciel 

algotprog 

Version 14 mai 2018

## Table des matières

<b>1 C - Résumé de cours</b>	<b>1</b>
1.1 Définitions . . . . .	1
1.2 Conditions . . . . .	2
1.3 Sélectives Si et Si-Alors . . . . .	4
1.4 Si-Imbriquées, Si-Cascade . . . . .	4
1.5 Sélective Si-Sinon-Si . . . . .	5

## 1 C - Résumé de cours

### 1.1 Définitions



#### Condition simple

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



#### Condition composée

Notée  $b_1 \Psi_1 b_2 \Psi_2 \dots$ , elle associe des opérateurs logiques  $\Psi_i$  (**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

C/C++

### Opérateurs de comparaison

Opérateur Mathématique	Signification	Équivalent	
		Algorithmique	C/C++
<	(strictement) inférieur	<code>a &lt; b</code>	<code>a &lt; b</code>
≤	inférieur ou égal	<code>a &lt;= b</code>	<code>a &lt;= b</code>
>	(strictement) supérieur	<code>a &gt; b</code>	<code>a &gt; b</code>
≥	supérieur ou égal	<code>a &gt;= b</code>	<code>a &gt;= b</code>
=	égalité	<code>a = b</code>	<code>a == b</code>
≠	différent de (ou inégalité)	<code>a &lt;&gt; b</code>	<code>a != b</code>

C/C++

### Opérateurs logiques

Opérateur Mathématique	Signification	Équivalent	
		Algorithmique	C/C++
¬	négation (unaire)	<code>Non a</code>	<code>! a</code>
∧	conjonction logique	<code>a Et b</code>	<code>a &amp;&amp; b</code>
∨	disjonction logique (ou inclusif)	<code>a Ou b</code>	<code>a    b</code>



### Propriétés des opérateurs logiques

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

x	y	¬x (non x)	x ∧ y (x Et y)	x ∨ 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 :

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

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



### 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
	Ou	Ou logique
La plus basse		



### 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**.

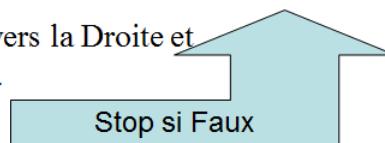


### Principe de l'évaluation paresseuse

(« 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.

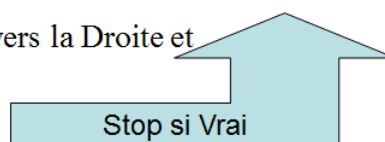
$x_1$  Et  $x_2$  Et ... Et  $x_i$  ... Et  $x_n$

Evaluation de la Gauche vers la Droite et arrêt au premier  $x_i$  faux ...



$x_1$  Ou  $x_2$  Ou ... Ou  $x_i$  ... Ou  $x_n$

Evaluation de la Gauche vers la Droite et arrêt au premier  $x_i$  vrai ...



### 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

C/C++

#### Sélective Si

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

C/C++

#### Sélective Si-Alors

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

### 1.4 Si-Imbriquées, Si-Cascade

C/C++

#### Si imbriquées

```
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;  
}
```