Algorithmique
&
Langage C
IUT GEII S1

2 Notes de cours

cours_algo_lgc2.09.odp

deuxième partie : Bases Algorithmiques

Licence

Plan du cours

- Opérateurs relationnels
- Opérateurs logiques
- Expressions avec les opérateurs relationnels et logiques

Bases Algorithmiques

- Haut -> Bas
- Alternative : if ... else


Vrai / Faux

VRAI : ≠ 0

FAUX: 0

En C toute expression non nulle est dite VRAIE

La valeur 0 est dite FAUSSE


Opérateurs relationnels

Math	C
=	==
≠	!=
>	>
≥	>=
<	<
≤	<=

Une expression formée avec ces opérateurs vaut :

0 si elle est fausse
1 si elle est vraie

Opérateurs relationnels

```
int i = 2 , j = 3 , k ;

k = ( i==j ) ;                                  k vaut  :

k = i >0 ;                                      k vaut  :

k = ( (i+1) == 3 ) ;                            k vaut  :

k = ( i >= j ) ;                                k vaut  :

k = 2 * ( j > i ) + ( (i+j) == (2*i + 1) ) + (i==j) ;   k vaut  :
```

Opérateurs relationnels

Ils s'appliquent à la plupart des types.

On peut mélanger certains types dans les expressions.

Ils retournent toujours une valeur entière (0 ou 1).

(Attention à ne pas comparer des valeurs signées
à des valeurs non signées.)

float x = 3.7 ; int i= 2 , j ;

j = (i < x) ;

Attention à cause de la précision limité, l'égalité de deux réels est rarement vraie.


Opérateurs logiques

ET	&&
OU	||
NON	!


Opérateurs logiques

int i=3, j=4, k ;

k = (i<6) && (j>=i) ; k vaut :

k = !((i>2) || (j < 3)) ; k vaut :


Bases Algorithmiques


Haut / Bas

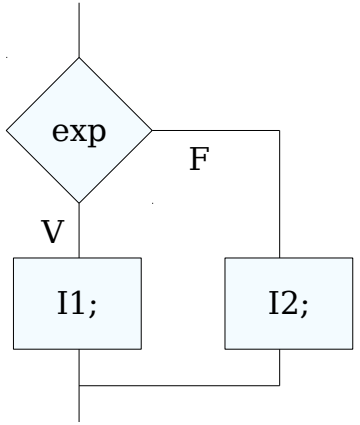
Les instructions s'exécutent de haut en bas,
les unes après les autres.

Dès qu'une instruction est exécutée, la suivante commence.

La vitesse est déterminée par les performances de l'ordinateur.

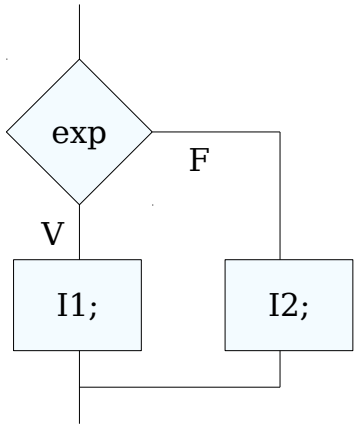
Alternative (ou choix)

Permet d'exécuter une séquence d'instructions ou une autre selon la valeur vraie ou fausse d'une expression.



Si exp est vraie la séquence I1; est exécutée.
Sinon (si exp est fausse) c'est la séquence I2; qui est exécutée.

Alternative C : if...else




```
if ( exp )
 {
   I1;
 }
else
 {
   I2 ;
 }
```

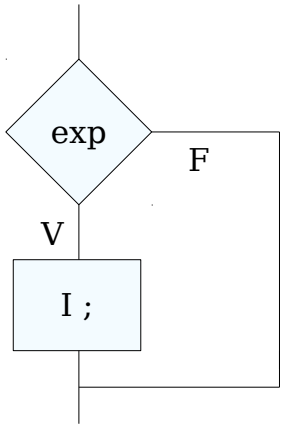

Exemple : if ... else

```
int i=3 , j = 5 , k ;

k = ( i>j ) ;

if ( k )
{
  printf ("A") ;
}
else
{
printf ("B") ;
}
```

Exemple : if ... else

```
int i=3 , j = 5 ;

if ( i > j )
{
  printf ("A") ;
}
else
{
printf ("B") ;
}
```


if sans else




```
if ( exp )
{
     I ;
}
```

Partage en trois

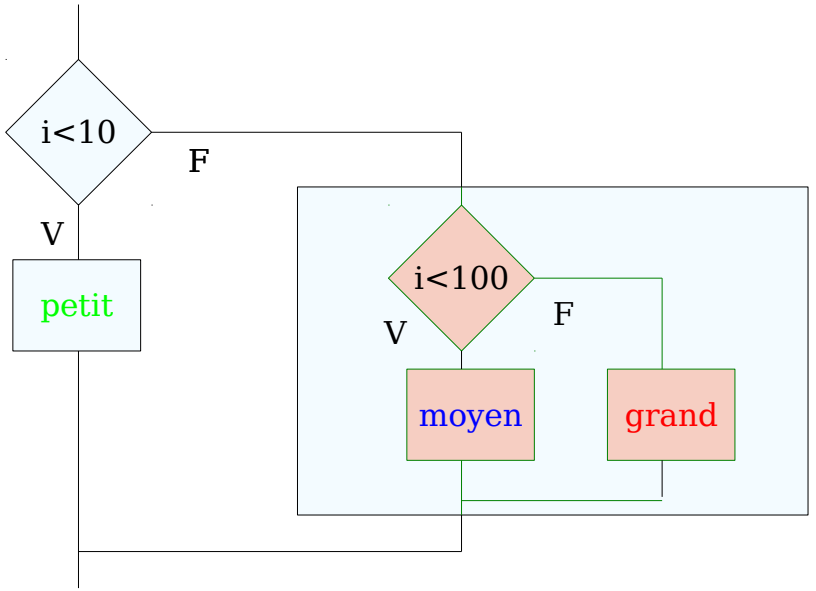
On imbrique des instructions if/else.

Exemple :

Afficher petit, moyen ou grand en fonction de la valeur de x

Un if/else permet de séparer deux zones.
Par exemple : petit et moyen/grand.
Dans la seconde zone, une autre if/else va permettre de séparer moyen et grand.

Partage en trois



Partage en trois

```
if ( i < 10 )
{
   printf( "petit" );
}
else
{
   if ( i < 100 )
   {
      printf( "moyen" );
   }
   else
   {
      printf( "grand" );
   }
}
```


Encadrements

Exemple : Afficher ok si x est compris entre -3 et 12 :

```
if ( ( -3 < x ) && ( x <12 ) )
  {
     printf( "ok" );
  }
```

Attention en C l'expression -3<x<12 n'est pas vraie si x est compris entre -3 et 12 !

(-3<x<12 est une expression toujours vraie !
En effet : (-3<x) vaut 0 ou 1 . Mais 0<12 et 1<12 donc -3<x<12 vaut 1 quelque soit x !
De même x<12 vaut 0 ou 1 or -3<1 et -3<0 donc -3<x<12 vaut 1 quelque soit x !)

Factorisation

Lorsqu'une instruction (ou une partie d'instruction) est identique dans la clause if et dans la clause else,
c'est qu'elle ne dépend pas de l'expression évaluée dans le if().

Elle peut donc être placée soit avant soit après le if/else :
C'est la *factorisation*.

Essayer de factoriser vos programmes.

Factorisation => Programmes souvent plus simples
Programmes plus concis
Programmes plus faciles à modifier


Factorisation

Non factorisé :

```
if ( x>0 )
{
   x = x+3 ;
   printf("x vaut %d",x);
}
else
{
   x = -x+3 ;
   printf("x vaut %d",x);
}
```

Première factorisation :

```
if ( x>0 )
{
   x = x+3 ;
 }
else
{
   x = -x+3 ;
}
printf("x vaut %d",x);
```


Factorisation

deuxième factorisation :

```
if ( x < 0 )
{
   x = - x ;
 }
x = x + 3 ;
printf("x vaut %d",x);
```

Si on doit modifier ce programme pour modifier la valeur d'incrémentation ou modifier la langue d'affichage, il n'y a à chaque fois qu'une seule ligne à modifier.


Ne pas remplacer if/else par deux if !

On est parfois tenté de remplacer un if/else par deux if successifs.

Outre le fait que ce n'est pas efficace car l'expression doit être évaluée deux fois, c'est parfois *dangereux*.

En effet :

```
if ( exp ) { I1; }
     else  { I2; }
```

n'est pas toujours équivalent à :

```
if ( exp ) { I1; }
if (!exp)  { I2; }
```

Le contraire d'une expression n'ait pas toujours trivial. Laissez le compilateur évaluer !exp !!


Ne pas remplacer if/else par deux if !

Soit le problème suivant : Une valeur x doit être multipliée par 2 si elle est inférieure ou égale à 15 et doit être multipliée par 4 si elle est strictement supérieure à 15.

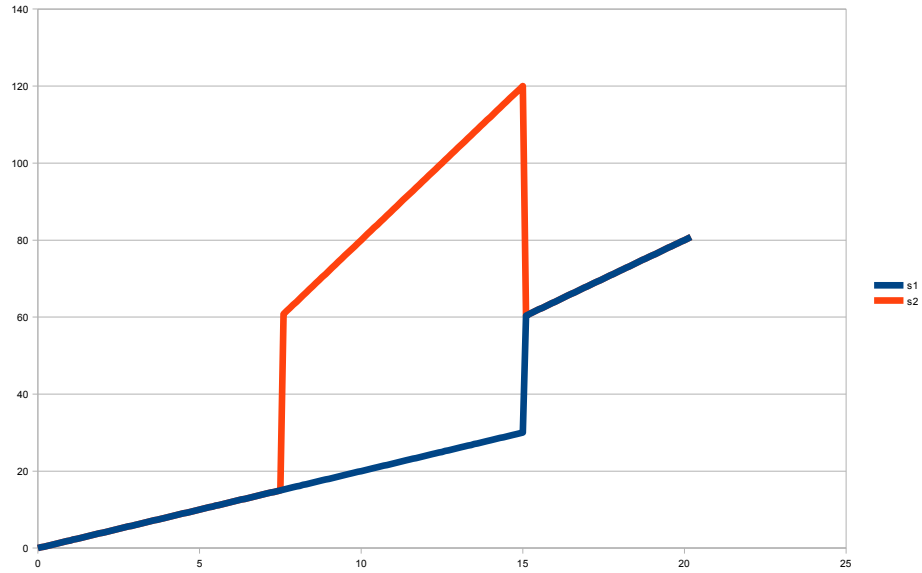
```
if ( x <= 15 )
{
   x = x * 2 ;
}
if ( x > 15 )
{
   x = x * 4 ;
}
```

Programme faux

Programme correct :

```
if ( x <=15 )
{
   x = x * 2 ;
}
else
{
  x = x * 4 ;
}
```


Ne pas remplacer if/else par deux if !



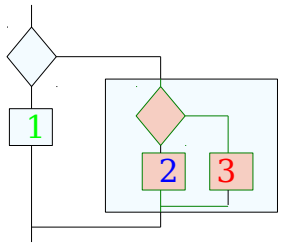
Ne pas remplacer if/else par deux if !

Il faut donc reconnaître :

- les situations ou les actions sont mutuellement exclusives et pour ces cas il faut absolument utiliser des if/else imbriqués.

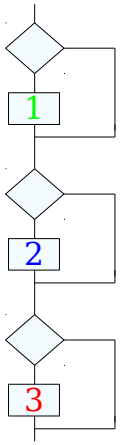
- les situations où les actions peuvent être éventuellement effectuées simultanément et dans ce cas une suite de if sans else convient.

Utilisez la structure qui convient à la situation !




```
if (...)
{
  1 ;
}
else
{
 if(...)
 {
   2 ;
 }
 else
 {
   3 ;
 }
}
```

Une seule instruction 1 ou alors 2 ou alors 3 est toujours effectuée quelque soit les valeur des expressions dans les if.




```
if
(...)
{
  1 ;
}
if
(...)
{
  2 ;
}
if
(...)
{
  3 ;
}
```

Selon les valeurs des expressions dans les if, on effectue éventuellement 1
et éventuellement 2 et éventuellement 3 ou aucune instruction.