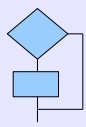

Algorithmique & Langage C

IUT GEII S1

Notes de cours (deuxième partie)

cours_algo_lgc2.04.odp



Licence



Paternité - Pas d'Utilisation Commerciale - Partage des Conditions Initiales à l'Identique 2.0 France

Vous êtes libres :

- * de reproduire, distribuer et communiquer cette création au public
- * de modifier cette création, selon les conditions suivantes :

Paternité. Vous devez citer le nom de l'auteur original.

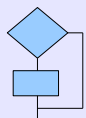
Pas d'Utilisation Commerciale.

Vous n'avez pas le droit d'utiliser cette création à des fins commerciales.

Partage des Conditions Initiales à l'Identique.

Si vous modifiez, transformez ou adaptez cette création, vous n'avez le droit de distribuer la création qui en résulte que sous un contrat identique à celui-ci.

- * A chaque réutilisation ou distribution, vous devez faire apparaître clairement aux autres les conditions contractuelles de mise à disposition de cette création.
 - * Chacune de ces conditions peut être levée si vous obtenez l'autorisation du titulaire des droits
- Ce qui précède n'affecte en rien vos droits en tant qu'utilisateur (exceptions au droit d'auteur : copies réservées à l'usage privé du copiste, courtes citations, parodie...)
voir le contrat complet sous : <http://fr.creativecommons.org/contrats.htm>



Plan du cours

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

Bases Algorithmiques

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

Vrai / Faux

VRAI : $\neq 0$

FAUX: 0

En C toute expression non nulle est dite VRAIE
La valeur 0 est dite FAUSSE

Opérateurs relationnels

Math	C
$=$	$==$
\neq	$!=$
$>$	$>$
\geq	\geq
$<$	$<$
\leq	\leq

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	!

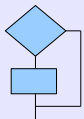
Opérateurs logiques

```
int i=3, j=4, k ;
```

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

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

Bases Algorithmiques



Haut / Bas

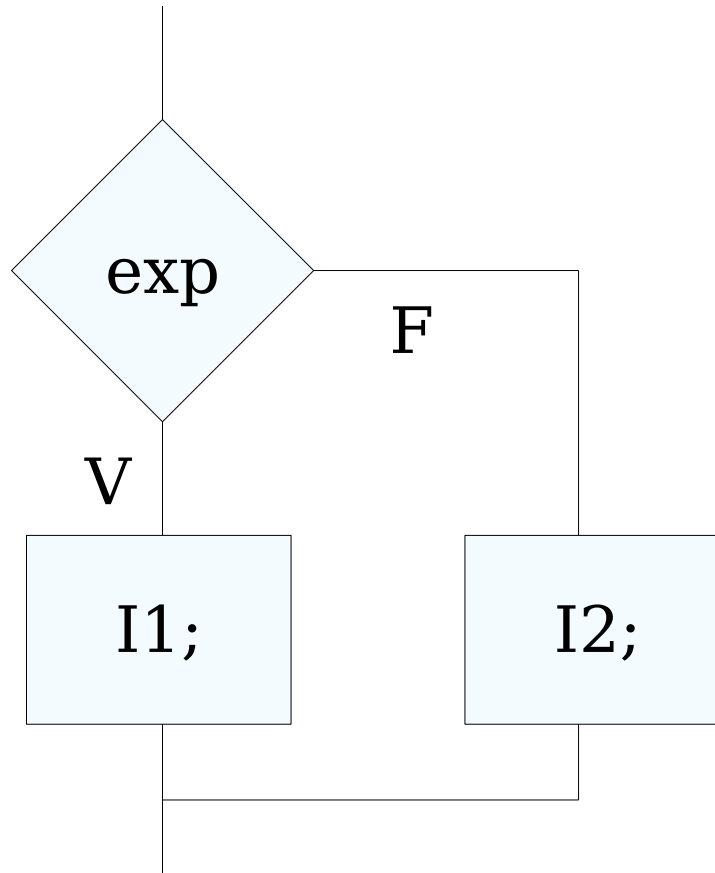
Les instructions s'exécutent de haut en bas,
les unes après les autres.

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

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

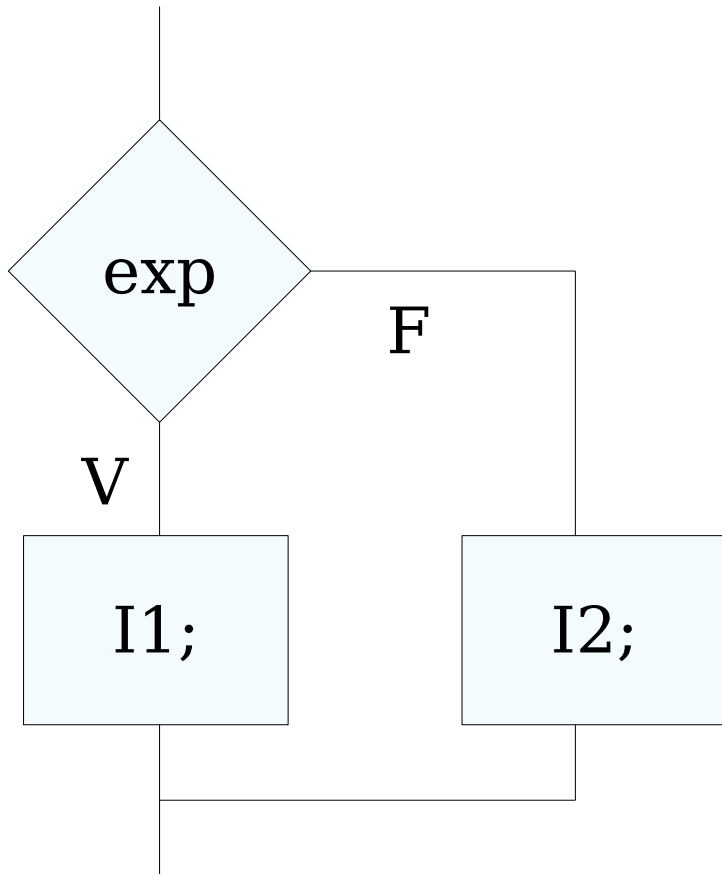
Alternative (ou choix)

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 ) ;
```

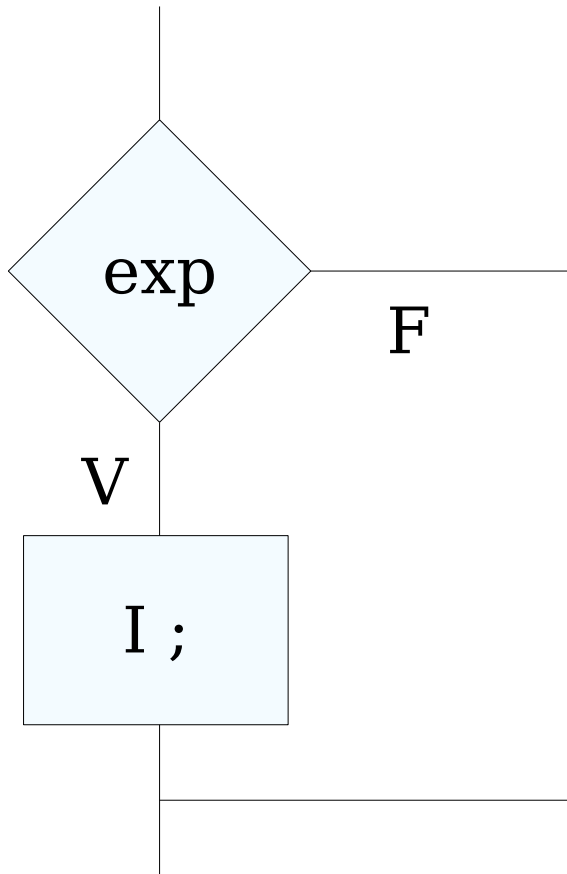
```
if ( k )  
{  
    printf ("A") ;  
}  
else  
{  
    printf ("B") ;  
}
```

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

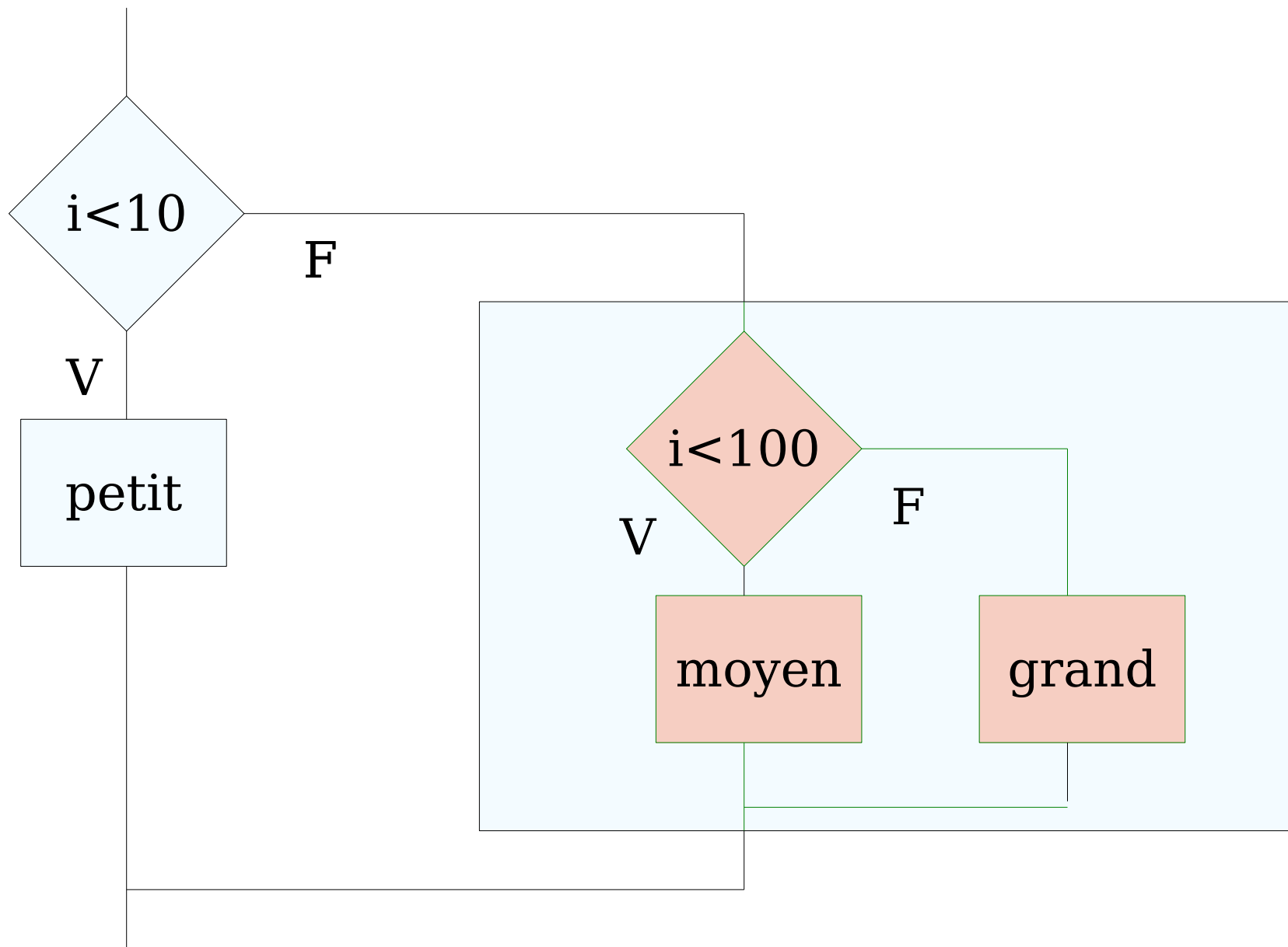


Un if/else permet de séparer deux zones.

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 8 et 12 :

```
if ( ( 8 < x ) && ( x <12 ) )  
  {  
    printf( "ok" );  
  }
```

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'incrémentatation 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 !

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 ;  
}
```