IN100 : Cours 3 – Embranchement : `if`

Sandrine Vial
`sandrine.vial@uvsq.fr`

Septembre 2015

Les instructions

- L'ordre d'exécution des instructions est séquentiel.

```
int a, b;

a = 10;
b = a * 2;
```

- L'instruction vide existe

```
int a;
a = a * 2;;;
```

- Un bloc d'instructions `{ ... }`

```
int a, b;

a = 2;
{
  b = a * 2;
  a = 12;
}
```

Et si on selectionnait ?

- Toutes les lignes de code sont exécutées séquentiellement.
- Embranchement : Permet de choisir les lignes de codes à éxécuter en fonction d'une condition.

`if` simple

- Si une condition est remplie exécuter des instructions
- `if (condition) { un bloc d'instructions }`
- `condition` doit être vraie pour que le bloc d'instructions soit exécuté.

`if ... else`

- Si une condition est remplie exécuter des instructions sinon exécuter d'autres instructions
- `if (condition) { un bloc d'instructions 1 } else { un bloc d'instructions 2 }`
- `condition` doit être vraie pour que le bloc d'instructions 1 soit exécuté et fausse pour que le bloc d'instructions 2 soit exécuté.

Exemple 1

```
#include "graphics.h"
int main()
{
     POINT p;

     init_graphics(900,600);

     p = wait_clic();
     if (p.x < 450)
     {
          draw_fill_circle(p,50,rouge);
     }

     wait_escape();
     exit(1);
}
```

Exemple 2

```
#include "graphics.h"
int main()
{
     POINT p;

     init_graphics(900,600);

     p = wait_clic();
     if (p.x < 450)
     {
          draw_fill_circle(p,50,rouge);
     }
     else
     {
          draw_fill_circle(p,50,vert);
     }
     wait_escape();
     exit(1);
}
```

Exemple 3

```
#include "graphics.h"
int main()
{
     POINT p;

     init_graphics(900,600);

     p = wait_clic();
     if (p.x % 2 == 0)
     {
          draw_circle(p,50,rouge);
     }
     else
     {
          draw_circle(p,100,bleu);
     }
     wait_escape();
     exit(1);
}
```

Exemple 4

```
#include "graphics.h"
int main()
{
     int i,j;
     int min,max;

     init_graphics(900,600);
     i = lire_entier_clavier();
     j = lire_entier_clavier();
     if (i > j)
     {
         min = j;
         max = i;
     }
     else
     {
         min = i;
         max = j;
     }
     write_text("Le plus grand entier est"); write_int(max); writeln();
     write_text("Le plus petit entier est"); write_int(min); writeln();

     wait_escape();
     exit(1);
}
```

Les Conditions

- Les conditions testées dans le `if` sont booléennes.

```
BOOL b;
int i,j;

i = lire_entier_clavier();
j = lire_entier_clavier();
b = i < j;

if (b == TRUE)
{
    write_text("i est plus petit que j");
}
```

Les Conditions

- Les conditions testées dans le `if` sont booléennes.

```
BOOL b;
int i;

i = lire_entier_clavier();
b = (i % 2 == 1);

if (b == FALSE)
{
    write_text("i est ....");
}
```

Les Conditions

- Les conditions testées dans le `if` sont booléennes.

```
BOOL b;
int i;

i = lire_entier_clavier();
b = (i % 2 == 1);

if (b == FALSE)
{
    write_text("i est pair");
}
```

Les Conditions

- Les conditions testées dans le `if` sont booléennes.

```
BOOL b;
int i;

i = lire_entier_clavier();
b = (i % 2 == 0);

if (b == TRUE)
{
    write_text("i est pair");
}
```

Les Conditions

- Les conditions testées dans le `if` sont booléennes.

```
BOOL b;
int i;

i = lire_entier_clavier();
b = (i % 2 == 0);

if (b)
{
    write_text("i est pair");
}
```

Egalité à `TRUE` n'est pas nécessaire

Tests Simples

- Tous les opérateurs de comparaisons sont utilisables dans une condition
- `if (a < 10) ...`
- `if (b >= 12) ...`
- `if (i == j) ...`

Attention ! Piège !

- L'expression `if (a = valeur_non_nulle) ...` est une expression toujours **vraie**

```
POINT p;

p.x = 100;
p.y = 200;

if (p.x = 200)
  {
    draw_fill_circle(p,50,vert);
  }
```

Attention ! Piège !

- L'expression `if (a = valeur_non_nulle) ...` est une expression toujours **vraie**
- L'expression `if (a = 0) ...` est une expression toujours **fausse**

```
POINT p;

p.x = 0;
p.y = 200;

if (p.x = 0)
  {
    draw_fill_circle(p,50,vert);
  }
```

Conditions plus complexes

- On peut combiner les conditions
  - Les deux conditions sont vraies en même temps : `ET` noté `&&`
  - Une des deux conditions soit vraie : `OU` noté `||`

Conditions plus complexes

- On peut combiner les conditions
  - Les deux conditions sont vraies en même temps : `ET` noté `&&`
  - Une des deux conditions soit vraie : `OU` noté `||`

A && *B*	TRUE	FALSE
TRUE	**TRUE**	*FALSE*
FALSE	*FALSE*	*FALSE*

Conditions plus complexes

- On peut combiner les conditions
  - Les deux conditions sont vraies en même temps : `ET` noté `&&`
  - Une des deux conditions soit vraie : `OU` noté `||`

A && *B*	TRUE	FALSE
TRUE	**TRUE**	*FALSE*
FALSE	*FALSE*	*FALSE*

A || *B*	TRUE	FALSE
TRUE	**TRUE**	**TRUE**
FALSE	**TRUE**	*FALSE*

Exemple 5

```
#include "graphics.h"
int main()
{
     POINT p;

     init_graphics(900,600);

     p = wait_clic();
     if (p.x < 450 && p.y > 300)
     {
          draw_fill_circle(p,50,rouge);
     }
     else
     {
          draw_fill_circle(p,50,vert);
     }
     wait_escape();
     exit(1);
}
```

Exemple 6

```
#include "graphics.h"
int main()
{
     POINT p;

     init_graphics(900,600);

     p = wait_clic();
     if (p.x < 450 && p.y > 300)
     {
          draw_fill_circle(p,50,rouge);
     }
     else
     {
          if (p.y < 300)
              draw_fill_circle(p,50,vert);
          else
              draw_fill_circle(p,100,bleu);
     }
     wait_escape();
     exit(1);
}
```

Exemple 7

```
#include "graphics.h"
int main()
{
     POINT p;

     init_graphics(900,600);

     p = wait_clic();
     if (p.x < 450 || p.y > 300)
     {
          draw_fill_circle(p,50,rouge);
     }
     else
     {
         draw_fill_circle(p,50,vert);
     }
     wait_escape();
     exit(1);
}
```