Chapitre 6

Les structures de contrôle

- Dans tous les tests de structures de contrôle, l'expression utilisée lors du test peut avoir le type « int » (valable aussi en C mais pas en Java) ou bien « bool » (valable aussi en Java).

1. L'instruction if

- La syntaxe générale est comme suit :

```
if (expression)
    instruction
```

- Elle est traduite comme suit :
  - D'abord « expression » est évaluée.
  - Si elle est vraie, on exécute « instruction ».
  - Sinon on saute « instruction » pour exécuter la suite du programme.

- On peut avoir aussi les variantes suivantes :

```
if (expression)
    instruction1
else
    instruction2
```

```
if (expression1)
    instruction1
else
    if (expression2)
       instruction2
    else
       instruction3
```

- L'instruction if/else, signifie que si « expression » est vraie, on exécute alors « instruction1 » sinon (else), dans le cas contraire, on exécute « instruction2 ».

- À noter que « instruction » peut-être formée aussi d'un bloc { } ou d'autres combinaisons de if/else.

```
if (x<0)
    cout << "x est négatif\n";
else
    if (x>0)
       cout << "x est positif\n";
else
    cout << "x est nul\n";
```


2. L'instruction while

- Elle permet d'exécuter un bloc d'instructions tant que la condition de test est vraie.
- Ainsi le bloc d'instructions est exécuté 0 fois ou plus.

```
int x = -5 ;
while (x<0)
    cout << "je ne fais qu'afficher que x est négatif \n";

int z = 10;
while (z>5 && z<15){
    cout << "La valeur de z est comprise entre ]5,15[ \n" ;
    --z ;
}
```

On vérifie d'abord que la valeur de z est supérieure à 5 mais inférieure à 15.
Si c'est le cas, on répète l'opération qui consiste à d'abord afficher la valeur de z, puis la décrémenter et cela, tant que z a une valeur comprise entre 5 et 15.

La condition qui consiste à dire que est-ce que x est négatif ? Elle est toujours vraie.
Le test « while » est donc une boucle infinie !

3. L'instruction do while

- Elle permet d'exécuter un bloc d'instructions tant que la condition de test est vraie. Seulement le test intervient après l'exécution du bloc d'instruction.
- Ainsi, le bloc d'instructions est exécuté au moins une fois.

```
int x = -5 ;
do {
    cout << "je ne fais qu'afficher que x est négatif \n";
    cout << "Opération garantie que lors du 2nd passage \n";
} while (x<0);

int z = 10;
do {
    cout << "La valeur de z est comprise entre ]5,15[ \n" ;
    cout << "Opération garantie que lors du 2nd passage \n";
    --z ;
} while (z>5 && z<15);
```


4. L'instruction for

- La syntaxe générale est comme suit :

- Une définition = entête + corps

```
for (initialisation ; condition ; mise à jour)
     instruction ou bloc d'instructions
```

- D'abord il y a évaluation de « initialisation ». Elle sert généralement à initialiser les variables de la boucle.
- Puis la « condition » est testée.
  - Si elle est vraie, « l'instruction » ou « le bloc d'instructions » est exécuté.
  - La « mise à jour » est effectuée.
  - Le contrôle est remis par la suite à la boucle « for » sauf que cette fois-ci on saute l'étape « d'initialisation ».
  - « L'instruction » ou « le bloc d'instructions » est exécuté tant que la « condition » n'est pas satisfaite.
- Dans le cas contraire, on sort de la boucle « for ».

- En C++ et Java, on peut déclarer des variables dans la partie « initialisation ».
- La visibilité de ces variables sera le bloc « for » délimité par « { » « } ».

```
for ( ;;) {} // boucle infinie

for (int i=0 ; i<10 ; i++)
     cout << i ;

for (int i=0, j=0 ; i<10 ; i++,j++)
     cout << i+j ;
```


5. L'instruction switch

- Elle permet un choix multiple en fonction de l'évaluation d'une expression.

- La syntaxe est comme suit :

```
swith (expression){
case e1 : instruction1;
case e2 : instruction2;
……
case e9 : instruction9;
default : instruction_par_defaut
}
```

- Évaluer d'abord « expression » : elle doit donner un résultat « int ».
- Les expressions « e1 » à « e9 » sont des expressions constantes qui doivent être des entiers uniques de type « int » ou « char ».
- La valeur de « expression » est recherchée successivement parmi les valeurs des expressions constantes « e1 » à « e9 ».
    - Si l'on trouve une correspondance, on exécute l'instruction associée jusqu'à une instruction break.
    - Dans le cas contraire, on exécute l'instruction associée à l'étiquette « default », si elle existe.

```
switch (caractere){
case 'a' :
case 'A' :
case 'b' :
case 'c' : j++; break;
case 'E' : k++;
           break;
case ' ' :
default : cout << "Je ne fais rien!\n";
}
```

6. L'instruction break

- Cette instruction est utilisée dans les structures de contrôle du type « while », « do-while », « for » ou « switch ».
- Elle permet de quitter les structures de contrôle précédemment mentionnées et de passer à l'instruction qui suit immédiatement le corps de ces structures de contrôle.

```
for (int i=0 ; i<10 ; i++){
     if (x==5) break;
     cout << x ;
}
cout << "en dehors de la boucle" ;
```

Si la condition est vérifiée, le grand saut!

7. L'instruction continue

- Cette instruction est utilisée dans les structures de contrôle du type « while », « do-while » ou « for ».
- Elle permet de passer à l'itération suivante

```
for (int i=0 ; i<10 ; i++){
     if (x==5) continue;
     cout << x ;
}
cout << "en dehors de la boucle" ;
```

Si la condition est vérifiée, on passe à l'itération suivante.
Le chiffre 5 ne sera jamais affiché.