

# **Leçon 2 : Les structures de contrôle en C++**

UVCI 2018

# Table des matières



<b>I - 1- Les Structures Conditionnelles</b>	<b>3</b>
<b>II - Application 1 :</b>	<b>8</b>
<b>III - 2- Les Structures Itératives</b>	<b>13</b>
<b>IV - Application 2 :</b>	<b>16</b>
<b>V - Travaux Pratique 2</b>	<b>17</b>

# 1- Les Structures Conditionnelles

I

## 1.1- La Structure Conditionnelle Simple

Syntaxe :

*If(condition)*

{

*Bloc d'instructions ;*

}

Exemple :

Écrire un programme qui vérifie si un nombre entré est égal à 2.

```
1 #include <iostream> //permet d'utiliser le cout
2 using namespace std; // permet d'utiliser le endl
3 main()
4 {
5     int val1; //Déclaration de la variable val1
6     cout<<"Entrez un nombre entier svp!"<<endl; //affiche le message entre les
       doubles griffes
7     cin>>val1; //récupération de la valeur saisie par l'utilisateur
8     if(val1==2) //vérification si la valeur de la variable val1 est égale 2
9     {
10     cout<<"la valeur saisie est correcte"<<endl; //exécutée si condition est vraie
11     }
12 }
13
```

## 1.2- La Structure Conditionnelle Alternative

Syntaxe :

*If(condition)*

{

*Bloc d'instructions1 ;*

}

*else*

{

*Bloc d'instructions2;*

}

*Exemple :*

Écrire un programme qui vérifie si un nombre entré est égal à 2 ou pas.

```

1 #include<iostream> //permet d'utiliser le cout
2 using namespace std; // permet d'utiliser le endl
3 main()
4 {
5 int val1; //Déclaration de la variable val1
6 cout<<"Entrez un nombre entier svp!"<<endl; // affiche le message entre les
    doubles griffes
7 cin>>val1; //récupération de la valeur saisie par l'utilisateur
8 if(val1==2) //vérification si la valeur de la variable val1 est égale 2
9 {
10 cout<<"la valeur saisie est correcte"<<endl; //exécutée si condition est vraie
11 }
12 else //vérification si la valeur de la variable val1 est différente de 2
13 {
14 cout<<"la saisie est incorrecte"<<endl; //exécutée dans le cas contraire (a
    valeur de la variable val1 est différente de 2) ;
15 }
16 }
17

```

### 1.3- La Structure Ternaire

*Syntaxe :*

(Condition) ? action\_si\_vrai : action\_si\_faux;

*Action\_si\_vrai* correspond à l'instruction à exécuter si la condition posée est vérifiée

*Action\_si\_faux* est l'instruction à réaliser dans le cas contraire

*Example :*

Écrire un programme qui affiche la valeur absolue d'un nombre.

```
1 #include<iostream> //permet d'utiliser le cout
2 using namespace std; // permet d'utiliser le endl
3 main()
4 {
5     int val1, val2; //Déclaration de variables val1 et val2
6     cout<<"Entrez un nombre entier svp!"<<endl; // affiche le message entre les
7     doubles griffes
8     cin>>val1; //récupération de la valeur saisie par l'utilisateur
9     val2=val1; //affectation de la valeur de val1 à val2
10    val2=(val2>=0) ? val2 : -1 *val2; // si la valeur de val2 est supérieure ou égale
11    à 0 alors val2 reçoit la valeur de val2 sinon val2 reçoit ( -1*val2)
12    cout<<"la valeur absolue de "<<val1<<" est "<< val2; // affichera la valeur
13    absolue de la valeur de val1 est la valeur de val2.
14 }
```

### 1.3- Les Structures Conditionnelles Imbriquées

Il n'existe pas de syntaxe figée pour les structures conditionnelles imbriquées, cependant nous pouvons présenter quelques-unes.

*If(condition)*

```
{
Bloc d'instructions1 ;
}
```

*else if(condition2)*

```
{
Bloc d'instructions2;
}
```

*else*

```
{
Bloc d'instructions n;
}
```

Autre syntaxe :

*If(condition)*

```
{
If(condition)
{
Bloc d'instructions;
}
```

*else*

```
{
Bloc d'instructions;
}
}
```

*Exemple :*

Écrire un programme qui permet de dire si un nombre est pair ou impair. Si le nombre est pair il faut aussi spécifier s'il est positif ou pas.

```
1 #include<iostream> //permet d'utiliser le cout
2 using namespace std; // permet d'utiliser le endl
3 main()
4 {
5 int nb; // déclaration de la variable nb
6 cout<<"saisissez un nombre svp!"<<endl; // affiche le message entre les griffes
7 cin>>nb; // récupère la valeur saisie dans la variable nb
```

```

8 if (nb % 2 == 0) // vérifie si le reste de la division de nb par 2 égal à 0 (% =
  modulo).
9 {
10 // la condition est vérifiée (nb % 2 == 0)
11 if (nb >= 0) // vérifie encore si la valeur de nb est supérieure ou égale à 0
12 {
13 cout << nb << " est un nombre pair positif" << endl; // affiche la valeur de la
  variable nb est un nombre pair positif.
14 }
15 else
16 {
17 cout << nb << " est un nombre pair négatif" << endl; // affiche la valeur de la
  variable nb est un nombre pair négatif.
18 }
19 }
20 else
21 {
22 // la condition n'est pas vérifiée (nb % 2 <> 0)
23 cout << nb << " est un nombre impair" << endl; // affiche la valeur de la variable nb
  est un nombre impair.
24 }
25 }
26

```

### 1.4- La Structure de choix

Syntaxe :

Switch(variable)

```

{
case valeur1 : instruction1;
break;
case valeur2 : instruction 2;
break;
.
.
.
default: instruction_par_défaut;
break;
}

```

Exemple :

Écrire un programme qui dans un premier temps un menu de saisie pour afficher la salutation dans l'une des langues de choix.

- Il demande par la suite que l'utilisateur fasse son choix.
- Selon que la valeur saisie par ce dernier vaille 1, 2, 3 ou 4, une salutation est retournée.
- Si aucune de ces 4 valeurs n'est saisie, un message par défaut est retourné.

```

1 #include<iostream> //permet d'utiliser le cout
2 using namespace std; // permet d'utiliser le endl
3 main()
4 {
5     int choix; // déclaration de la variable choix
6     cout<<"***** MENU SALUTATION *****"<<endl; // affichera ***** MENU SALUTATION
7     // ***** et ira deux fois à la ligne
8     cout<<"1 - FRANCAIS"<<endl<<"2 - ANGLAIS"<<endl<<"3 - ESPAGNOL"<<endl<<"4 -
9     ALLEMAND"<<endl; // affichera
10    1 - FRANÇAIS
11    2 - ANGLAIS
12    3 - ESPAGNOL
13    4 - ALLEMAND
14
15    cout<<"Faites votre choix svp!"<<endl; // affichera Faites votre choix svp! Et
16    ira à la ligne
17    cin>>choix; //récupère la valeur saisie dans la variable choix
18    switch(choix) // selon la valeur de choix il affichera un message de salutation
19    si la valeur est comprise en 1 et 4 sinon il affichera La valeur saisie est incorrecte
20    {
21    case 1: cout<<"SALUT!"<<endl;
22    break;
23    case 2: cout<<"HELLO!"<<endl;
24    break;
25    case 3: cout<<"HOLA!"<<endl;
26    break;
27    case 4: cout<<"TAG!"<<endl;
28    break;
29    default: cout<<"La valeur saisie est incorrecte"<<endl;
30    }
31    }

```

# Application 1 :





## Exercice

Énoncé 1 :

Écrire un programme permettant de donner le jour à partir d'un nombre saisi par l'utilisateur.

Solution :

```

[ ]
[ ]
[ ]
{
[ ] nb;
[ ] Entrer un nombre compris de 1 à 7. [ ] endl;
[ ]
[ ]
{
case [ ]
[ ] Lundi [ ] endl;
[ ]
case [ ]
[ ] Mardi [ ] endl;
[ ]
case [ ]
[ ] Mercredi [ ] endl;
[ ]
case [ ]
[ ] Jeudi [ ] endl;
[ ]
case [ ]
[ ] Vendredi [ ] endl;
[ ]
case [ ]
[ ] Samedi [ ] endl;
[ ]
case [ ]
[ ] Dimanche [ ] endl;
[ ]

```

Application 1 :

Erreur la valeur saisie est incorrecte endl;

}

}

## Exercice

Énoncé 2 :

Écrire un programme permettant de donner le grade à laquelle appartient une valeur donnée. La grille se présente comme suit :

de 0 à 40 : le grade est F ;

de 41 à 60 : le grade est C ;

de 61 à 70 : le grade est B ;

de 71 à 80 : le grade est A ;

plus de 81 : le grade est A+ ;

Solution :

```

[ ]
[ ]
[ ]
{
[ ] pts;
[ ] Enter le point : [ ] endl;
[ ]
[ ] ( [ ] 81)
{
[ ] Grade = A+ [ ] endl;
}
[ ]
[ ] ( [ ] 71)
{
[ ] Grade = A [ ] endl;
}
[ ]
[ ] ( [ ] 61)
{
[ ] Grade = B [ ] endl;
}
[ ]
[ ] ( [ ] 41)
{

```

Application 1 :

```
    Grade = C    endl;  
}  
  
{  
    Grade = F    endl;  
}  
}
```

# 2- Les Structures Itératives

III

## 2.1- La Structure while

Syntaxe :

Initialisation

while(condition)

{

Bloc d'instructions ;

variation ;

}

Exemple :

Écrire un programme qui affiche les 10 premiers nombres.

```
1 #include <iostream> //permet d'utiliser le cout
2 using namespace std; // permet d'utiliser le endl
3 main()
4 {
5     int val1; //Déclaration de la variable val1
6     val1=1; //initialisation de la variable ( Initialisation )
7     while(val1<=10) //vérification de la condition
8     {
9         cout<<val1<<" |"; //affichera 1 |2 |3 |4 |5 |6 |7 |8 |9 |10 |
10        val1++; //incrémentement de la variable ( la variation )
11    }
12 }
13
```

## 2.2- La Structure do .. while

Syntaxe :

do

{

Bloc d'instructions1 ;

}

while( condition );

*Exemple :*

Écrire un programme qui affiche les 10 premiers nombres.

```
1 #include <iostream> //permet d'utiliser le cout
2 using namespace std; // permet d'utiliser le endl
3 main()
4 {
5     int val1; //Déclaration de la variable val1
6     val1=1; ///initialisation de la variable ( Initialisation )
7     do
8     {
9         cout<<val1<<" |"; //affichera 1 |2 |3 |4 |5 |6 |7 |8 |9 |10 |
10        val1++; //incréméntation de la variable
11    }
12    while(val1<=10) //vérification de la condition
13 }
14
```

### 2.3- La Structure for

*Syntaxe :*

*for(initialisation; condition; compteur)*

*{*

*Bloc d'instructions;*

*}*

- *Initialisation* : ce paramètre est exploité une seule fois (juste avant la première itération ) car il permet d'initialiser la variable qui sera vérifiée dans le paramètre suivant (à savoir la condition) avant l'exécution du bloc d'instructions.
- *Condition* : ce paramètre vérifie la condition d'arrêt de la boucle. Avant la première itération, cette condition est vérifiée juste après l'initialisation de la variable. Après cela, la condition est toujours vérifiée après l'exécution du compteur. Ainsi la condition d'arrêt précède toujours l'exécution du bloc d'instructions.
- *Compteur* : ce paramètre est toujours sollicité après l'exécution du bloc d'instructions et juste avant que la condition ne soit vérifiée.

*Exemple :*

Écrire un programme qui affiche les 10 premiers nombres.

```
1 #include <iostream> //permet d'utiliser le cout
2 using namespace std; // permet d'utiliser le endl
3 main()
4 {
5     int val1; //Déclaration de la variable val1
6     for(val1=1;val1<=10;val1++) //initialisation de val1, condition et incréméntation
7     {
8         cout<<val1<<" |"; //affichera 1 |2 |3 |4 |5 |6 |7 |8 |9 |10 |
9     }
10 }
11 /**
```

```
12 Explication : La variable val1 est initialisée à 1 dans un premier temps, la  
   condition est ensuite vérifiée avant que l'instruction ne soit exécutée. Juste après,  
   val1 est incrémentée et la condition est à nouveau vérifiée avant l'exécution de  
   l'instruction et le cycle se poursuit jusqu'à ce que val1 soit supérieur à 10 avant  
   de sortir de la boucle. **/  
13
```

# Application 2 :

IV

## Exercice

Énoncé 1 :

Écrire un programme qui affiche la table de multiplication d'un nombre saisi par l'utilisateur pour des multiplicateurs de 1 à 10.

De la forme (par exemple) si on saisit 7 alors :

$$1 \times 7 = 7$$

$$2 \times 7 = 14$$

:

$$10 \times 7 = 70$$

NB : Le nombre doit être obligatoirement positif.

Solution :

```

[ ]
[ ]
[ ]
{
[ ] nbre, [ ] ;
do
[ ]
[ ] Enter le nombre [ ] endl;
[ ]
}
[ ]
[ ] ( [ ] ; [ ] <= [ ] ; [ ] )
{
[ ] i [ ] * " [ ] " = [ ] nbre*i [ ] endl;
}
}

```



# Travaux Pratique 2

V

*Énoncé :*

Écrire un programme en langage C++ qui contraint l'utilisateur à saisir un nombre pair avant l'affichage du triple de ce nombre.

```
1 #include <iostream> //permet d'utiliser le cout
2 using namespace std; // permet d'utiliser le endl
3 main()
4 {
5     int nb; // déclaration de la variable nb
6     do
7     {
8         cout<<"Saisissez un nombre pair svp!"<<endl; // affichera : Saisissez un nombre
           pair svp!
9         cin>>nb; // récupère la valeur saisie
10    }
11    while(nb%2!=0); // vérifie si le nombre est impair
12    cout<<"Le triple de "<<nb<<" est: ";
13    nb*=3;
14    cout<<nb;
15 }
16
```