Programmation C++ (débutant)/Instructions if, if...else et switch

Le cours du chapitre 3 : le if, le if...else et le switch

Les structures de contrôle

Ce sont des structures permettant d'exécuter certaines instructions de manière conditionnelle ou répétitive. Nous verrons dans ce chapitres les structures de contrôles de type conditionnelles :

- le if,
- le if else,
- le switch.

Le if

Cette structure de contrôle permet d'exécuter une instruction ou une suite d'instructions seulement si une condition est vraie.

Syntaxe :

```
if (condition) instruction;
```

On évalue la condition :

- si elle est vraie on exécute l'instruction et on passe à l'instruction suivante,
- si elle est fausse on passe directement à l'instruction suivante.

L'instruction peut être remplacée par une suite d 'instructions entre accolades

Les conditions

Les conditions habituelles sont utilisables :

- if (a>b)… strictement supérieur à
- if (a>=b)… supérieur ou égal à
- if (a<b)… strictement inférieur à
- if (a<=b)… inférieur ou égal à
- if (a==b)… test d'égalité
- if (a!=b)… différent de

Une erreur classique

Pour effectuer un test d'égalité, il faut utiliser 2 fois le symbole =.

Par exemple :

```
if(a==b)…
```

Une erreur classique consiste à écrire :

```
if(a=b)…
```

Exemple 1 : utilisation du if

Exemple 1 : utilisation du **if**

```
#include <iostream>
using namespace std;

int main()
{
  int a;
  cout << "Tapez la valeur de a : ";
  cin >> a;
  if (a > 10) cout << "Gagné !" << endl;
  cout << "Le programme est fini" << endl;
  return 0;
}
```

- Ce programme demande à l'utilisateur de saisir une valeur entière a.
- Si la valeur tapée est strictement supérieure à 10 on affiche "Gagné" puis "Le programme est fini" et le programme s'arrête.
- Dans le cas contraire, on affiche uniquement "le programme est fini" et le programme s'arrête.

Exécution n°1 de l'exemple 1

Lorsqu'on exécute notre programme, il s'affiche à l'écran :

```
Tapez la valeur de a : 80
Gagné
Le programme est fini
```

Exécution n°2 de l'exemple 1

Lorsqu'on exécute notre programme, il s'affiche à l'écran :

```
Tapez la valeur de a : 8
Le programme est fini
```

Exécution n°3 de l'exemple 1

```
Tapez la valeur de a : 10
Le programme est fini
```

Le if ... else

Cette structure de contrôle permet d'exécution soit l'instruction1, soit l'instruction 2 en fonction du résultat d'une condition.

Syntaxe :

```
if (condition) instruction1;
else instruction2;
```

1. On évalue la condition,
2. si elle est vraie, on exécute l'instruction1 et on passe à l'instruction suivante,
3. si elle est fausse, on exécute l'instruction2 et on passe à l'instruction suivante.

L'instruction1 ou l'instruction2 peuvent être remplacées par une suite d'instructions entre accolades.

Exemple 2 :Utilisation du if ...else

Exemple 2 : utilisation du **if... else**

```
#include <iostream>
using namespace std;

int main()
{
int a;
cout<<"Tapez la valeur de a : ";
cin>>a;
if(a>10)cout<<"Gagné !"<<endl;
          else cout<<"Perdu"<<endl;
cout<<"le programme est fini"<<endl;
return 0;
}
```

- Ce programme demande à l'utilisateur de saisir une valeur entière a.
- Si la valeur tapée est strictement supérieure à 10, on affiche "Gagné" puis "Le programme est fini" et le programme s'arrête.
- Dans le cas contraire, on affiche "Perdu" puis "Le programme est fini" et le programme s'arrête.

Exécution n°1 de l'exemple 2

Lorsqu'on exécute notre programme, il s'affiche à l'écran :

```
Tapez la valeur de a : 80
Gagné
Le programme est fini
```

Exécution n°2 de l'exemple 2

Lorsqu'on exécute notre programme, il s'affiche à l'écran :

```
Tapez la valeur de a : 5
Perdu
Le programme est fini
```

Exécution n°3 de l'exemple 2

Lorsqu'on exécute notre programme, il s'affiche à l'écran :

```
Tapez la valeur de a : 10
Perdu
Le programme est fini
```

Les conditions complexes

Il est souvent nécessaire d'écrire des conditions assez compliquées. Il faudra alors utiliser le ET logique, le OU logique et le NON logique. Il faudra une certaine habitude pour ne pas confondre ces différents opérateurs.

Le ET logique

Syntaxe

```
condition1 && condition2
```

Exemple 3 : utilisation du ET logique

Exemple 3 : utilisation du **ET logique**

```
#include <iostream>
using namespace std;
int main()
{
int a;
cout<<"Tapez une valeur entière : ";
cin>>a;
if(a>10 && a<20)cout<<"GAGNÉ"<<endl;
       else cout<<"PERDU"<<endl;
return 0;
}
```

- Dans ce programme, on demande à l'utilisateur de taper une variable entière a.
- Si la valeur tapée est comprise entre 11 et 19 bornes incluses on affiche "GAGNÉ".
- Dans le cas contraire, on affiche "PERDU".
- **Exécution de l'exemple 3 n°1**
  Lorsqu'on exécute notre programme, il s'affiche à l'écran :
  Tapez une valeur entière : 8
  PERDU
- **Exécution de l'exemple 3 n°2**
  Lorsqu'on exécute notre programme, il s'affiche à l'écran :
  Tapez une valeur entière : 15
  GAGNÉ
- **Exécution de l'exemple 3 n°3**
  Lorsqu'on exécute notre programme, il s'affiche à l'écran :
  Tapez une valeur entière : 20
  PERDU

Le OU logique

Syntaxe : condition1 || condition2
Rappel sur le OU logique:
VRAI OU VRAI = VRAI
VRAI OU FAUX = VRAI
FAUX OU VRAI = VRAI
FAUX OU FAUX = FAUX

Exemple 4 : Utilisation du OU logique

Exemple 4 : utilisation du **OU logique**

```
#include <iostream>
using namespace std;

int main()
{
int a;
cout<<"Tapez une valeur entière : ";
cin>>a;
if( a<3 || a>20 )cout<<"GAGNÉ"<<endl;
  else cout<<"PERDU"<<endl;
return 0;
}
```

- Dans ce programme, on demande à l'utilisateur de taper une variable entière a.
- Si la valeur tapée est strictement plus petite que 3 on affiche "GAGNÉ".
- Si la valeur tapée est strictement plus grande que 20 on affiche "GAGNÉ".
- Dans le cas contraire, on affiche "PERDU".
- **Exécution de l'exemple 4 n°1**
  Lorsqu'on exécute notre programme, il s'affiche à l'écran :
  Tapez une valeur entière : -6
  GAGNÉ
- **Exécution de l'exemple 4 n°2**
  Lorsqu'on exécute notre programme, il s'affiche à l'écran :
  Tapez une valeur entière : 3
  PERDU
- **Exécution de l'exemple 4 n°3**
  Lorsqu'on exécute notre programme, il s'affiche à l'écran :
  Tapez une valeur entière : 21
  GAGNÉ

Le NON logique

Syntaxe : !(condition)
Rappel sur le NON logique :
! VRAI=FAUX
!FAUX=VRAI

Exemple 5 : Utilisation du NON logique

Exemple 5 : utilisation du **NON logique**

```
#include <iostream>
using namespace std;
int main()
{
int a;
cout<<"Tapez une valeur entière : ";
```

```
cin>>a;
if(!(a<3 || a>20))cout<<"GAGNÉ"<<endl;
    else cout<<"PERDU"<<endl;
return 0;
}
```

- Dans ce programme, on demande à l'utilisateur de taper une valeur entière a.
- Si la condition assez complexe (!(a<3 || a>20)) est vraie on affiche "GAGNÉ" sinon on affiche "PERDU".
- En fait, on affichera "GAGNÉ" si a est compris entre 3 et 20 bornes incluses.
- **Exécution de l'exemple 5 n°1**
  Lorsqu'on exécute notre programme, il s'affiche à l'écran :
  Tapez une valeur entière : 2
  PERDU
- **Exécution de l'exemple 5 n°2**
  Lorsqu'on exécute notre programme, il s'affiche à l'écran :
  Tapez une valeur entière : 20
  GAGNÉ
- **Exécution de l'exemple 5 n°3**
  Lorsqu'on exécute notre programme, il s'affiche à l'écran :
  Tapez une valeur entière : 50
  PERDU

Exemple 6 : Mettre plusieurs instructions dans un if

Exemple 6 : Mettre plusieurs instructions dans un if

```
#include <iostream>
using namespace std;

int main()
{
  int a;
  cout << "Tapez une valeur entière : ";
  cin >> a;
  if (a==12) {
    cout << "GAGNÉ" << endl;
    a = a+1;
  }
  cout << "La valeur finale de a vaut " << a << endl;
  return 0;
}
```

- Dans ce programme, on demande à l'utilisateur de saisir une valeur entière *a*.
- Si *a* vaut 12 alors on va effectuer dans un premier temps deux instructions : on affiche "GAGNÉ" et on incrémente *a* de 1. On affiche ensuite la valeur de *a* qui vaudra 13.
- Si *a* est différent de 12, on affiche directement la valeur de *a* qui n'aura pas été modifiée.
- **Exécution de l'exemple 6 n°1**
  Lorsqu'on exécute notre programme, il s'affiche à l'écran :

```
Tapez une valeur entière : 50
```

```
La valeur finale de a vaut 50
```

- **Exécution de l'exemple 6 n°2**
  Lorsqu'on exécute notre programme, il s'affiche à l'écran :

```
Tapez une valeur entière : 12
GAGNÉ
La valeur finale de a vaut 13
```

Exemple 7 : Mettre plusieurs instructions dans un if ...else

Exemple 7 : Mettre plusieurs instructions dans un if...else

```
#include <iostream>
using namespace std;

int main()
{
int a;
cout<<"Tapez une valeur entière : ";
cin>>a;
if(a!=10){cout<<"GAGNÉ"<<endl;a=a+1;}
    else {cout<<"PERDU"<<endl;a=a-1;}
cout<<"La valeur finale de a vaut "<<a<<endl;
return 0;
}
```

- Dans ce programme, on demande à l'utilisateur de saisir une valeur entière a.
- Si a est différent de 10 alors on va effectuer dans un premier temps 2 instructions : on affiche "GAGNÉ" et on incrémente a de 1. On affiche ensuite la valeur de a.
- Si a est égal à 10 alors on va effectuer dans un premier temps 2 instructions : on affiche "PERDU" et on décrémente a de 1. On affiche ensuite la valeur de a.
- **Exécution de l'exemple 7 n°1**
  Lorsqu'on exécute notre programme, il s'affiche à l'écran :
  Tapez une valeur entière : 7
  GAGNÉ
  La valeur finale de a vaut 8
- **Exécution de l'exemple 7 n°2**
  Lorsqu'on exécute notre programme, il s'affiche à l'écran :
  Tapez une valeur entière : 10
  PERDU
  La valeur finale de a vaut 9

Le type bool

Il s'agit d'un type de base prédéfini du langage C++. Il permet de stocker une valeur booléenne pouvant prendre 2 valeurs : soit true soit false. Le résultat d'une condition peut être stocké dans un bool.

Exemple 8 : utilisation du type bool

Exemple 8 : utilisation du type bool

```
#include <iostream>
using namespace std;

int main()
{
int a; bool c,d;
cout<<"Tapez une valeur entière : ";
cin>>a;
c=(a<3);
d=(a>20);
if(c||d)cout<<"GAGNÉ"<<endl;
   else cout<<"PERDU"<<endl;
return 0;
}
```

- Dans ce programme, nous définissons une variable entière a et 2 variable booléennes c et d.
- On demande à l'utilisateur de saisir la variable entière a.
- Dans c, on affecte true si a est strictement plus petit que 3. c vaudra false dans le cas contraire.
- Dans d, on affecte true si a est strictement plus grand que 20. d vaudra false dans le cas contraire.
- Si c ou d est vrai, c'est à dire si a est strictement plus petit que 3 ou si a est strictement plus grand que 20 alors on affiche "GAGNÉ". On affiche perdu dans le cas contraire.
- **Exécution de l'exemple 8**
  Lorsqu'on exécute notre programme, il s'affiche à l'écran :
  Tapez une valeur entière : -6
  GAGNÉ

Le switch

Syntaxe :

```
switch(identificateur)
{
case c1:instruction1;break;
case c2:instruction2;break;
case c3:instruction3;break;
...
default: instruction;break;
}
```

Sémantique du switch :

On teste la variable définie par l'identificateur. On la compare successivement aux constantes c1, c2, c3,…etc… Si la variable vaut c1 alors on exécute l'instruction1 et on passe à l'instruction suivante. Si elle vaut c2, on exécute

l'instruction2 et on passe à l'instruction suivante. Idem s'il vaut c3. Si elle ne vaut ni c1, ni c2, ni c3 alors on exécute l'instruction après default et on passe à l'instruction suivante. Le default est facultatif. On peut remplacer les instructions instruction1, instruction2, instruction3 par des suites d'instructions sans mettre d'accolades. Les valeurs c1, c2,c3 .. sont obligatoirement des constantes.

Exemple 9 : utilisation du switch

Exemple 9 : utilisation du switch

```
#include <iostream>
using namespace std;

int main()
{
int i;
cout<<"Tapez un entier entre 1 et 3 bornes incluses :";
cin>>i;
switch(i)
  {
  case 1: cout<<"GAGNÉ"<<endl;
          i=i+99;
          break;
  case 2: cout<<"PERDU n° 2"<<endl;
          i=0;
          break;
  case 3: cout<<"PERDU n°3"<<endl;
          i=0;
          break;
  }
cout<<"La valeur finale de i est "<<i<<endl;
return 0;
}
```

- On commence par saisir un entier i.
- Il y a ensuite une switch qui permet d'effectuer une série d'instructions en fonction de la valeur de i.
- Les 3 valeurs de i pour lesquelles on effectue une série d'instructions sont 1, 2 et 3.
- Si i vaut une autre valeur le switch ne fait rien car il n'y a pas de default.
- **Exécution 1 de l'exemple 9**
  Tapez un entier entre 1 et 3 bornes incluses :1
  GAGNÉ La valeur finale de i est 100
- **Exécution 2 de l'exemple 9**
  Tapez un entier entre 1 et 3 bornes incluses :2
  PERDU n°2
  La valeur finale de i est 0
- **Exécution 3 de l'exemple 9**
  Tapez un entier entre 1 et 3 bornes incluses :3
  PERDU n°3 La valeur finale de i est 0

Exemple 10 : un autre exemple de switch

Exemple 10 : un autre exemple de switch

```
#include <iostream>
using namespace std;

 int main()
{
int i;
cout<<"Tapez un entier entre 1 et 3 bornes incluses :";
cin>>i;
switch(i)
   {
   case 1:
   cout<<"GAGNÉ"<<endl;
   i=i+99;
   break;

   case 2 :
   cout<<"PERDU n° 2"<<endl;
   i=0;
   break;

   case 3 :
   cout<<"PERDU n°3"<<endl;
   i=0;
   break;

   default :
   cout<<"J'ai dit entre 1 et 3 !!!"<<endl;
   i=-1000;
   break;
   }
cout<<"La valeur finale de i est "<<i<<endl;
return 0;
}
```

- On commence par saisir un entier i.
- Il y a ensuite une switch qui permet d'effectuer une série d'instructions en fonction de la valeur de i.
- Les 3 valeurs de i pour lesquelles on effectue une série d'instructions sont 1, 2 et 3.
- Si i vaut une autre valeur , on exécute l'instruction après default.
- **Exécution 1 de l'exemple 10**
  Tapez un entier entre 1 et 3 bornes incluses :1
  GAGNÉ
  La valeur finale de i est 100
- **Exécution 2 de l'exemple 10**
  Tapez un entier entre 1 et 3 bornes incluses :2
  PERDU n°2

La valeur finale de i est 0

- **Exécution 3 de l'exemple 10**
  Tapez un entier entre 1 et 3 bornes incluses :3
  PERDU n°3
  La valeur finale de i est 0
- **Exécution 4 de l'exemple 10**
  Tapez un entier entre 1 et 3 bornes incluses :5
  J'ai dit entre 1 et 3 !!!
  La valeur finale de i est -1000

Exercices

EXERCICE 1

Ecrire un programme qui résout l' équation AX+B=0. Bien évidemment, on n'oubliera pas tous les cas particuliers (notamment les cas "tout x est solution" et "pas de solution").

EXERCICE 2

Ecrire un programme qui demande à l'utilisateur de taper 5 entiers et qui affiche le plus grand. Le programme ne devra utiliser que 2 variables.

EXERCICE 3

Ecrire un programme qui résout l'équation $ax^2+bx+c=0$ en envisageant tous les cas particuliers.

EXERCICE 4

Ecrire un programme qui demande à l'utilisateur de saisir les coordonnées de 4 points A, B, C et D puis qui affiche les informations suivantes :
- si A et B sont confondus, on affiche 'A et B sont confondus'
- si C et D sont confondus, on affiche ' C et D sont confondus'
- si A et B ne sont pas confondus et si C et D ne sont pas confondus , on affiche soit 'AB et CD sont parallèles', soit 'AB et CD sont confondues', soit 'AB et CD sont sécantes'. Dans ce dernier cas, on affiche les coordonnées de l'intersection de AB et de CD.

EXERCICE 5

Ecrire un programme qui demande à l'utilisateur de saisir un entier X et qui affiche la valeur absolue de X.

EXERCICE 7

Ecrire un programme qui demande à l'utilisateur de saisir 3 entiers A, B et C et qui indique si C est compris entre A et B, bornes incluses.

EXERCICE 8

Ecrire un programme qui demande à l'utilisateur de saisir 4 entiers A, B, C et D, puis qui indique quelle est l'intersection des intervalles [AB] et [CD].

EXERCICE 9

Ecrire un programme qui demande à l'utilisateur de saisir un entier A puis qui affiche "ERREUR" si A n'est pas un nombre impair compris entre 83 et 101 bornes incluses. Dans le cas contraire, on affiche "PAS D'ERREUR".