Présentent

Eléments de base du langage C++

Formateur : Nene SYLLA

I. Introduction :

Le C++ a été conçu par Bjarne Stroustrup en 1980 aux Laboratoires BELL. L'idée était d'ajouter au C des possibilités sur l'orienté objet et de palier aux inconvénients du C.

Les forces du C++ :

- Il est **très répandu**. Comme nous l'avons vu, il fait partie des langages de programmation les plus utilisés sur la planète. On trouve donc beaucoup de documentation sur Internet et on peut facilement avoir de l'aide sur les forums.
- Il est **rapide**, très rapide même, ce qui en fait un langage de choix pour les applications critiques qui ont besoin de performances. C'est en particulier le cas des jeux vidéo, mais aussi des outils financiers ou de certains programmes militaires qui doivent fonctionner en temps réel.
- Il est **portable** : un même code source peut théoriquement être transformé sans problème en exécutable sous Windows, Mac OS et Linux. Vous n'aurez pas besoin de réécrire votre programme pour d'autres plates-formes !
- Il existe de **nombreuses bibliothèques** pour le C++. Les bibliothèques sont des extensions pour le langage, un peu comme des plug-ins. De base, le C++ ne sait pas faire grand-chose mais, en le combinant avec de bonnes bibliothèques, on peut créer des programmes 3D, réseaux, audio, fenêtrés, etc.
- Il est **multi-paradigmes :** Ce mot signifie qu'on peut programmer de différentes façons en C++.

Les faiblesses du C++ :

Bien entendu, le C++ n'est pas LE langage incontournable. Il a lui-même ses défauts par rapport à d'autres langages, sa complexité en particulier. Vous avez beaucoup de contrôle sur le fonctionnement de votre ordinateur (et sur la gestion de la

mémoire) : cela offre une grande puissance mais, si vous l'utilisez mal, vous pouvez plus facilement faire planter votre programme.

II. Structure d'un programme C++ :

Exemple:

```
# include <iostream >
```

```
using namespace std;

int main ()
 {
 cout << " Hello world!" << endl ; return 0 ;
}
```

Explication :

Un programme est un ensemble de déclarations et d'instructions. Le programme doit être placé entre accolades {} à la suite du nom de la fonction principale main(). En C++, toute déclaration ou fonction doit se terminer par un point-virgule **;**. ○ main () est la fonction principale. C'est le point d'entrée du programme. Elle est donnée ici sans arguments, mais on peut lui transmettre des arguments en écrivant par exemple :

main (int nbarg) où nbarg est un paramètre entier.

. ○ La première ligne #include <iostream.h> est une directive. Elle est prise en charge par le préprocesseur avant la compilation. Elle permet d'utiliser des bibliothèques C++.

III. Création d'un programme C++ :

La création d'un exécutable en C++ passe par trois étapes principales :

1. Edition du programme.
2. Compilation.
3. Edition des liens.

1. Edition du programme :

L'édition du programme consiste en la saisie du texte du programme par un éditeur. Ce texte est appelé programme source ou source tout simplement. Il est sauvegardé dans un fichier appelé fichier source. L'´éditeur de l'environnement de d´enveloppement sauvegarde le fichier source avec une dénomination possédant une extension spécifique. Pour les programmes C++, l'extension est généralement .cpp.

2. Compilation :

Le programme source est écrit dans un langage évolué. La compilation consiste à traduire le programme source en langage machine. La compilation est effectuée par un programme appelé compilateur. La compilation est précédée par le traitement par le préprocesseur.

- **Traitement par le préprocesseur :**

Le préprocesseur exécute les directives qui le concernent. Il les reconnait par le fait qu'elles commencent par le caractère #. Le résultat est un programme C++ pur, sans directives.

- **Compilation :**

Elle produit un programme en langage machine. Ce résultat est appelé module objet et porte généralement l'extension .obj. Le module objet n'est pas directement exécutable.

3. Edition des liens :

Pour s'exécuter, le module objet principal a besoin des fonctions auxquels il fait appel. L'éditeur de liens (linker) va chercher dans les bibliothèques concernées les modules objet de ces fonctions. Une bibliothèque est une collection de modules objets (fonctions compilées) organisés en plusieurs fichiers. Grâce à la compilation séparée, on peut rassembler lors de l'édition des liens plusieurs modules objet compilés de façon séparée. Le résultat de l'édition des liens est un programme exécutable qui porte généralement l'extension .exe. Le programme exécutable peut être exécuté sans faire appel à l'environnement de développement C++.

IV. Les fichiers d'entête :

Ils sont traités par le préprocesseur et sont introduits grâce à la directive **#include**.

Ces fichiers comportent les déclarations relatives aux fonctions prédéfinies. Le préprocesseur copie le contenu de ce fichier dans le programme qu'il produit à l'endroit de la directive. Le code compilé de ces fonctions est contenu dans les modules objet et il est rangé dans des bibliothèques. Le fichier d'entête peut être nommé de trois façons :

- **Par son chemin absolu** : Le préprocesseur recherche le fichier à cet emplacement même, comme le montre la ligne de code ci-dessous :

```
# include "D:\ Program Files \ CodeBlocks \ include \ entete "
```

- **À partir du répertoire courant** : Le préprocesseur recherche le fichier d'entête dans le répertoire courant, à titre d'exemple :

```
# include " header "
```

- **à partir d'un répertoire prédéfini**: C'est les répertoires standards correspondant à l'installation du compilateur, si le nom de fichier est entouré par un inférieur et un supérieur. C'est dans ces répertoires que le préprocesseur recherchera le fichier d'entête. Dans ce cas, le nom du fichier est inséré entre les symboles < et > comme dans l'exemple suivant :

```
# include <iostream >
```


V. Le préprocesseur :

Les fonctions du préprocesseur peuvent être résumées dans les points suivants :

1. Traiter le fichier source avant le compilateur.
2. Retirer les commentaires (compris entre /* et */ ou après //).
3. Prendre en compte les lignes commençant par #.

VI. Les commentaires :

Ils peuvent être faits à n'importe quel endroit du programme. Ce sont des textes explicatifs destinés aux lecteurs du programme. Ils ne sont pas pris en compte par le compilateur. On distingue deux types de commentaires :

1. Les commentaires libres (par bloc).
2. Les commentaires de ligne (de fin de ligne).

1. Les commentaires libres (par bloc) :

Ils commencent par /* et se termine par */. Ils peuvent s'étendre sur plusieurs lignes. Un exemple est donné ci-dessous :

```
/* ---------------------------------------------------------------------- |
Mon premier programme en C++ |
  ----------------------------------------------------------------------*/
```

2. Les commentaires de ligne (de fin de ligne) :

Ils sont introduits par les deux caractères //. Ils sont généralement placés en fin de ligne après une instruction. Tout ce qui se situe entre // et la fin de la ligne est un commentaire. Un exemple est donné ci-dessous :

```
Cout << "Hello \n" ; // bonjour
```


VII. Les variables :

Les variables servent à stocker les données manipulées par les programmes en mémoire. Une variable a une adresse, un type et une valeur. Le nom est appelé identificateur, quant au contenu d'une variable, il appartient à son type. Les variables peuvent être partagés en deux :

- Les variables scalaires : Ce sont les entiers, réels, pointeurs, ...
- Les variables structurées : Tableaux, structures, ...

1. Déclaration des variables :

Le langage C++ est déclaratif : Toutes les variables doivent être déclarées avant leur première utilisation. La syntaxe de la déclaration d'une variable est la suivante :

```
<Type > <identificateur >
```

2. Type des variables :

Il existe trois types de base pour les variables en C++ : entiers, flottants et caractères.

❖ Les types entiers :

Selon la taille en mémoire des variables, on distingue trois types entiers :

- **Short int (short)**
- **Int**
- **Long int (long)**

Chaque type permet de représenter les données sur un certain nombre d'octets. Le nombre d'octet définit la plage des valeurs prise par les variables de chaque type. Lorsque la plage des valeurs est strictement positive, on utilise des variables non signés (**unsigned**). On a alors les autres types suivants :

- **Unsigned short int**
- **Unsigned long int**
- **Unsigned int**

○ Les types réels ou flottants :

Un réel est composé d'un signe, d'une mantisse et d'un exposant. On distingue trois types flottants :

- **Float**
- **Double**
- **Long double**

○ Les types caractères :

Il s'agit du type char. Une variable de ce type stocke un seul caractère qui est codé sur un octet (8 bits). Il peut être signé ou non signé :

- **Char :** prend les valeurs entre -127 à +127.
- **Unsigned char :** prend les valeurs positives de 0 à 255.

Un caractère peut être une lettre, un chiffre, ou un autre caractère du code ASCII. Il peut être un caractère non imprimable (dit de contrôle). A chaque caractère du code ASCII correspond une valeur entière comprise entre 0 et 255. Ceci permet d'utiliser une variable de type char, codé seulement sur un octet, au lieu d'une variable de type int qui occupe plus de mémoire pour stocker un entier compris entre 0 et 255. Cette opération permet de réduire la mémoire utilisée.

○ Le type booléen :

Il contient deux valeurs true et false (Vrai ou Faux).

Exemple :

L'exemple suivant récapitule la déclaration de variables des différents types cités :

```
# include <iostream >

 using namespace std;

 int main ()
 {int i = 2; float x = 4.3; float y = 54e -01; double z = 5.4; unsigned char
c = 'A'; return 0;
 }
```

3. Règles d'écriture des identificateurs :

Les règles d'écritures des identificateurs sont les suivantes :

- Ne commence pas par un chiffre. ○ Il y a une différence entre majuscules et minuscules (C++ est sensible à la case ou case-sensitive).
- Ne pas utiliser les caractères accentués (é, è, ê, ù, ...).
- Ne pas utiliser les mots réservés du langage.

Le compilateur C++ n'impose aucune limite sur la longueur des identificateurs. Toutefois, certaines parties de l'environnement de développement (comme l'éditeur de liens) peuvent imposer une limitation. Cette limite peut varier en fonction des environnements utilisés.

VIII. Les constantes :

Les constantes sont déclarées avec le mot clé **const**. Comme les variables, elles possèdent un nom (identificateur) et un type. Elles ne changent pas de valeur au long du programme et doivent être initialisées au moment de leur création. Dans l'exemple suivant, on déclare une constante entière et deux constantes caractères et chaîne de caractères.

```
Const int a=1 ;// constante entière
const char c='A' ; constante caractère
Const char z [10] = "C++" ;// constante chaîne de caractères
```

1. Définition d'une constante à l'aide du préprocesseur :

Le préprocesseur effectue dans tout le code source une substitution de l'identificateur par la valeur indiquée. La syntaxe de cette déclaration est la suivante :

```
#define < identificateur > < valeur >
```

Exemple :

```
#define PI 3.14159
```

2. Les constantes énumérées :

Le mot clé **enum** permet de définir des types puis de définir des variables de ce type comme dans l'exemple suivant :

```
enum couleur {vert, rouge, bleu, noir, blanc}; couleur
c1, c2 ;
```


Les variables c1 et c2 peuvent avoir cinq valeurs possibles : vert, rouge, bleu, noir et blanc. Chaque constante énumérée possède une valeur numérique par défaut. Si aucune valeur n'est spécifiée, la première constante va avoir la valeur 0, la seconde 1, etc. . . .. Si des valeurs sont spécifiées comme dans l'exemple suivant :

```
Enum couleur {vert =2, rouge, bleu =6, noir, blanc =50};
```

Les valeurs prises par ces constantes énumérées vert, rouge, bleu, noir et blanc sont respectivement 2, 3, 6, 7, 50.

IX. Les opérateurs :

La syntaxe d'utilisation d'un opérateur est la suivante :

```
< Operande1 > < operateur ><operande2 >
```

On définira maintenant les principaux opérateurs en C++ classés par catégories.

1. Les opérateurs d'affectations :

Ils permettent de simplifier la notation des opérations arithmétiques. Chaque opérateur arithmétique possède un opérateur simplifié.

1.1 L'opérateur d'affectation :

Le symbole " = " est utilisé pour assigner (affecter) la valeur d'une expression a un identificateur.

```
j=2* i+1 ; //x reçoit le résultat de l'expression 2*i+1
```

1.2 Autres operateurs :

Opération	Résultat
x+=y;	x=x+y ;
x*=y ;	x=x*y ;
x/=y ;	x=x/y ;
x%=y ;	x=x%y ;

2. Les opérateurs arithmétiques :

Opérateur	Rôle
+	Addition
−	Soustraction
*	Multiplication
/	Division
%	Modulo (Reste de division entière)

3. Les opérateurs logiques :

Le résultat de l'utilisation d'un opérateur logique est un booléen (true ou false).

Opérateur	Rôle
&&	Et
||	Ou
!	Non (Not)

4. Les opérateurs de comparaison :

Opérateur	Rôle
==	Égal
!=	Différent
<	Strictement inférieur
<=	Inférieur ou égal
>	Strictement supérieur
>=	Supérieur ou égal

5. Les Opérateurs d'incrémentation et de décrémentation :

Opérateur	Exemple d'utilisation	Résultat
Incrémentation	`x++ ;`	`x=x+1 ;`
Décrémentation	`x-- ;`	`x=x-1 ;`

○ Dans le cas de l'opérateur préfixé, le **++** ou **--** est placé avant la variable. Dans ce cas, la variable est incrémentée (ou décrémentée) avant l'utilisation de la valeur. ○ Dans le cas de l'opérateur postfixé, le **++** ou **--** est placé après la variable. Dans ce cas, la variable est incrémentée (ou décrémentée) après l'utilisation de la valeur.

L'exemple suivant montre la différence entre les deux cas :

```
Int x=1 ; int y=0 ; y=x++ ; // dans ce
cas y=1 et x=2
X=1 ; // On réinitialise x y=++ x ; //
dans ce cas x=2 et y=2
```

X. Les entrées-sorties :

1. Les entrées : cin

Pour le moment, il est suffisant de considérer le flot cin comme une instruction permettant de saisir soit des valeurs numériques, des caractères simples ou des chaînes de caractères.

Sa syntaxe est la suivante :

```
Cin >> variable1 >> variable2 >>...>> variableN ;
```

Variable1, variable2, variableN sont les noms des variables qui vont contenir les données saisies à partir du clavier. L'exemple suivant illustre son utilisation :

```
Int i, j ;
Float z ; char s, nom [12] ;
cin >>i>>j>>z>>s>> nom;
```

2. Les sorties : cout

Le flot cout permet d'afficher le contenu des variables, mais aussi des messages à l'écran. ○ **Afficher un message :**

Le message est placé entre deux guillemets après un cout.

```
Cout << " Bonjour " ;
```

○ **Affichage d'une variable avec message :**

A titre d'exemple, le programme suivant affichera 1+2=3:


```
int i=1; int
j= 2;
cout <<i<<"+"<<j<<"="<<z;
```

Le changement de ligne se fait par \n et la tabulation par \t.