Programmation en C++
GI2-DAI

A. SERGHINI

Chapitre 1
Généralités sur le langage C++.

1- Présentation du langage C++

- **Le langage C++ a été conçu à partir de 1982 par Bjarne Stroustrup ingénieur au laboratoire Bell.**

- **L'objectif principal de B. Stroustrup était d'ajouter des classes au langage C et donc, en quelque sorte, de greffer sur un langage de programmation procédurale classique des possibilités de programmation orienté objet (en abrégé P.O.O).**

- **Dés 1982, C++ a connu plusieurs versions, jusqu'à sa normalisation par l'ANSI (American National Standards Institute) en 1998.**

2- Étapes permettant l'édition, la mise au point, l'exécution d'un programme en C++

1- ***Édition du programme source***, à l'aide d'un éditeur (traitement de textes). Le nom du fichier contient l'extension .CPP, exemple: EXI_1.CPP.

2- ***Compilation du programme source***, c'est à dire création des codes machine destinés au microprocesseur utilisé.

- Le compilateur indique les erreurs de syntaxe mais ignore les fonctions-bibliothèque appelées par le programme.
- Le compilateur génère un fichier binaire, non éditable en mode « texte », appelé fichier objet: EXI_1.OBJ (commande « compile »).

3- *Éditions de liens*: Le code machine des fonctions-bibliothèque est chargé, création d'un fichier binaire, non éditable en mode texte, appelé fichier exécutable: EXI_1.EXE.

4- *Exécution du programme* (commande « Run »).

Exemple :

```
//les commentaires s'écrivent derrière 2 barres obliques
#include <iostream> //sorties standard
using namespace std;
int main(int argc, char* *argv)
{
Ïint a, b, calcul ; //déclaration de 3 variables
cout<<"BONJOUR";//affichage d'un message sur l'écran
a = 10 ; // affectation
b = 50 ; // affectation
calcul = (a + b)*2 ; // affectation
cout <<" Affichage de a : "<< a<< endl;
cout <<" Affichage de b : "<< b<<endl;
cout <<" Voici le résultat : "<< calcul<<endl;
return  0;
}
```

3- Espaces de noms C++

- **Lorsque plusieurs bibliothèques sont utilisées, deux fonctions différentes peuvent avoir le même nom sans faire la même tâche. Pour éviter ce problème, le C++ définit les espaces de noms :**

```
namespace MaBiblio{
        fonc1() ;
        fonc2() ;
        ...}
```

- **Un espace de nom à un comportement similaire à un répertoire pour les systèmes d'exploitations.**

- **Ce qui est défini dans un espace de noms n'existe pas en dehors.**

- **Pour utiliser une fonction définie dans un chemin d'accès il faut utiliser le chemin d'accès :**

```
int main(){
        fon1() // erreur fonc1() n'existe pas
        MaBiblio::fonc2() // ok
        ...}
```

- **Pour éviter d'alourdir le code, il est possible de préciser que les membres d'un espace de noms seront utilisés directement grâce aux mot clef using.**
- **Il y a deux méthodes :**

1- Accéder à tout l'espace de noms

```
using namespace MaBiblio
        fonc1() // ok
        fonc2() // ok
```

2- Préciser les fonctions à utiliser

```
using Mabiblio::fonc1( )
        fonc1() // ok
        fonc2() // fonc2 n'existe pas en dehors
                // de l'espace de nom
```

- **La plupart des fonctions et classes de la librairie standard du C++ sont déclarées dans un espace de nom.**
- **L'espace de nom std contient la grande majorité des fonctions de la librairie standard.**
- **Habituellement, il est nécessaire d'utiliser « using namespace std » au début de chaque programme C++ qui utilise cette librairie. Certains compilateurs permettent de ne pas le faire.**

4- La gestion de la mémoire

La mémoire de l'ordinateur se comporte de trois zones d allocation :

1- La pile:

- **La pile contient les variables qui sont déclarées à l'intérieur des fonctions ou à des blocs ({…})**
- **Ces variables sont appelées variables locales ou automatiques**
- **Elles sont créés à l'appel de la fonction et détruites à la sortie de la fonction (respectivement à l'entrée et à la sortie du bloc)**
- **Leurs durées de vie est donc la durés de vie de la fonction.**

2- La zone statique:

- **La zone d'allocation statique contient les variables qui sont déclarées en dehors de toutes fonction ou à l'intérieur d'une fonction mais avec le qualifiant static.**
- **Ces variables sont appelées variables globales ou statiques**
- **Elles sont créés à l'exécution du programme et détruites à la fin de celui-ci**
- **Leurs durées de vie est donc la vie du programme.**

3- Le tas:

- **Le tas contient les variables qui sont créés par le programme au cours de son exécution. On parle d'allocation dynamique.**
- **Ces variables sont appelées variables dynamiques**
- **Leurs durées de vie est donc variable.**

5- Types prédéfinis

5-1 Les différents types de variables :

- Une variable est une abstraction d'une portion de mémoire
- Une variable est caractérisée par :
(Nom, adresse, valeur, type, durée de vie, portée)
- L'emplacement de la déclaration décide de sa portée.
- Son adresse est déterminée pendant l'exécution et sa valeur dépend des instructions dans lesquelles elle apparaît.

- **Le C++, est un langage typé, chaque entité doit disposer d'un type.**

Void	**:**	**type vide (pas de type)**
Bool	**:**	**type logique (true/false) (C++)**
Char	**:**	**caractères**
int	**:**	**entiers**
float	**:**	**réels**
double	**:**	**réels en double précision**
Tableaux		
Pointeurs		
wchar_t	**:**	**caractères longs (C++)**

5-2 Types –Tailles :

1- Les entiers

Le langage C++ distingue plusieurs types d'entiers:

TYPE	DESCRIPTION	TAILLE MEMOIRE
int	entier standard signé	4 octets: - 2^31 ≤ n ≤ 2^31-1
unsigned int	entier positif	4 octets: 0 ≤ n ≤ 2^32
short	entier court signé	2 octets: - 2^15 ≤ n ≤ 2^15-1
unsigned short	entier court non signé	2 octets: 0 ≤ n ≤ 2^16
char	caractère signé	1 octet : - 2^7 ≤ n ≤ 2^7-1
unsigned char	caractère non signé	1 octet : 0 ≤n ≤ 2^8

Numération:

- **En décimal les nombres s'écrivent tels que,**
- **En hexadécimal ils sont précédés de 0x.**

exemple: 127 en décimal s'écrit 0x7f en hexadécimal.

Remarque:

En langage C++, le type char possède une fonction de changement de type vers un entier:

- **Un caractère peut voir son type automatiquement transformé vers un entier de 8 bits**
- **Il est interprété comme un caractère alphanumérique du clavier.**

2- Les réels:

- **Un réel est composé :**
  - **d'un signe,**
  - **d'une mantisse,**
  - **d'un exposant,**
- **Un nombre de bits est réservé en mémoire pour chaque élément.**
- **Le langage C++ distingue 2 types de réels:**

TYPE	DESCRIPTION	TAILLE MEMOIRE
Float	réel standard	4 octets
double	réel double précision	8 octets

5-3 Les initialisations :

- **Le langage C++ permet l'initialisation des variables dès leurs déclarations:**

```
char c;          est équivalent à          char c = 'A';
c = 'A';
 int i;          est équivalent à           int i = 50;
 i = 50;
```

- **Cette règle s'applique à tous les nombres, char, int, float ...**

Pour améliorer la lisibilité des programmes et leur efficacité, il est conseillé de l'utiliser.

5-4 La portée d'une variable

C++ permet de déclarer des variables n'importe où et de les initialiser dans un bloc. La porté d'une variable dépend de l'endroit ou il est placé:

- ***Globale:*** **Les variables globales sont déclarées en dehors de toute fonction Elle est allouée et initialisée automatiquement avant l'entrée dans la fonction 'main'.**
  Elle est nettoyée et libérée automatiquement après la sortie de la fonction 'main'.
- ***Locale:*** **(dite également automatique) est visible dans le bloc d'instructions dans lequel elle à été déclaré et ce, à partir de sa déclaration. Elle est allouée et initialisée.**
  Elle est nettoyée et libérée automatiquement, lorsqu'on sort du block d'instructions
- ***Dynamique:*** **est totalement contrôlée par le programmeur.**

5-5 Classe de stockage :

Le C/C++ dispose des classes de stockage qui permettent de spécifier la portée de la variable :

- ***auto :*** **la classe de stockage par défaut. Les variables ont pour portée le bloc d'instructions où sont définis. Leur durée de vie est restreinte à ce bloc.**

- ***static :*** **permet de créer des variables dont la portée est le bloc d'instructions en cours, mais qui, contrairement aux variables auto, ne sont pas détruites lors de la sortie de ce bloc. À chaque fois que l'on rentre dans ce bloc d'instructions, les variables statiques existeront et auront pour valeurs celles qu'elles avaient avant que l'on quitte ce bloc.**

- ***const :*** **la variable désigne en fait une constante ; aucune modification n'est autorisée dans le programme.**

• ***extern*** **: est utilisée pour signaler que la variable peut être définie dans un autre fichier.**

• ***register*** **: on peut demander qu'une variable de type entier, caractère ou pointeur soit implantée dans un registre, ce qui est souvent utile quand on veut aller vite.**
Attention : **seule une variable automatique peut être de type registre. De plus, le mot-clé register, ne donne qu'une indication au compilateur; on ne garantit pas que la variable sera bien en registre, le compilateur n'ayant à sa disposition qu'un nombre limité de registres.**

• ***volatile*** **: un objet déclaré volatile peut être modifié par un événement extérieur à ce qui est contrôlé par le compilateur (exemple : variable mise à jour par l'horloge système).**

6- Les Flots cout et cin

- **Un flot peut être considéré comme un « canal » :**
  - recevant de l'information : flot de sortie ;
  - fournissant de l'information : flot d'entrée.
- **Un flot peut être connecté à un périphérique ou un fichier. Par convention, le flot prédéfini cout est connecté à ce que l'on nomme la « sortie standard ». De même, le flot prédéfini cin est connecté à ce que l'on nomme « entrée standard ».**
- **On peut dire qu'un flot est un objet d'une classe prédéfinie, à savoir ;**
  - ✓ **ostream pour un flot de sortie ;**
  - ✓ **istream pour un flot d'entrée.**

Exemple 1:

```
#include <iostream>
using namespace std;
int main(int argc, char** argv)
{
int u;
float s;
u = 1000;
s = 45.78;
cout <<"Voici u (base 10) : "<< u << endl <<"Voici s : " << s << endl;
return 0;
}
```

Exemple 2:

```
#include <iostream>
using namespace std;
int main(int argc, char** argv)
{
char u,v,w;
int i;
u = 'A';
v = 67;
w = 0x45;
cout<<"Voici u : "<< u << endl;
cout<<"Voici v : "<< v <<endl;
cout<<"Voici w : "<< w <<endl;
i = u; // conversion automatique de type
 // pour obtenir le code ascii de la lettre A en base 10
cout<<"Voici i : "<< i << endl;
 // pour obtenir le code ascii de la lettre A en hexadécimal
cout<<"Voici i : "<< hex << i << endl;
return 0 ;
}
```

Exemple 3:

```
#include <iostream>
using namespace std;
int main(int argc, char** argv)
{
char c;
int u;
cout<<"ENTRER UN CARACTERE : ";
cin >> c;
u = c; //conversion automatique de type
cout<<"VOICI SON CODE ASCII : "<< u << "\n";
return 0 ;
}
```

7- Les opérateurs

- **Opérateurs arithmétiques sur les réels:**

  + - * / avec la hiérarchie habituelle.

- **Opérateurs arithmétiques sur les entiers:**

  + - * / (quotient de la division) % (reste de la division).

- **Opérateurs relationnels**

  > >= <= < comparaisons
  == != égalité et inégalité
  ! négation (opérateur unaire)
  && ET relationnel
  || OU relationnel

- **Opérateurs logiques sur les entiers:**

  & ET
  | OU
  ^ OU EXCLUSIF
  ~ COMPLEMENT A UN
  « DECALAGE A GAUCHE
  » DECALAGE A DROITE.

- Incrémentation- Décrémentation

Le langage C++ autorise des écritures simplifiées pour l'incrémentation et la décrémentation de variables de type entier (int, char, long)

```
i = i+1; est équivalent à i++;
i = i-1; est équivalent à i--;
```

- Opérateurs combinés

Le langage C++ autorise des écritures simplifiées lorsqu'une même variable est utilisée de chaque côté du signe = d'une affectation.

```
a = a+b;     est équivalent à     a+= b;
a = a-b;     est équivalent à     a-= b;
a = a & b;   est équivalent à     a&= b;
```

8- Les déclarations des constantes

Le langage C++ autorise 2 méthodes pour définir des constantes.

1ere méthode:

déclaration d'une variable, dont la valeur sera constante pour toute la portée de la fonction main.

Exemple :

```
void main()
{
const float PI = 3.14159;
float perimetre, rayon = 8.7;
perimetre = 2*rayon*PI;
}
```

float rayon

- **Dans ce cas, le compilateur réserve de la place en mémoire (ici 4 octets), pour la variable pi, on ne peut changer la valeur.**
- **On peut associer «const » à tous les types.**

2 ème méthode:

Définition d'un symbole à l'aide de la directive de compilation #define.

Exemple:

```
#define PI = 3.14159;
void main()
{
float perimetre,rayon = 8.7;
perimetre = 2*rayon*PI;
//  ....
}
```

- **Le compilateur ne réserve pas de place en mémoire, on définit ainsi une equivalence « lexicale ».**
- **Souvent, les constantes déclarées par #define s'écrivent en majuscules, mais ce n'est pas une obligation.**

Exercice 1:

Saisir un caractère au clavier, afficher son code ASCII à l'écran.

Exercice 2:

Dans une élection, I est le nombre d'inscrits, V le nombre de votants, P = 100V / I le pourcentage de votants, M = V / 2 le nombre de voix pour obtenir la majorité.
Écrire un programme qui demande à l'utilisateur de saisir I et V, puis calcule et affiche P et M.

Exercice 3:

Saisir 3 entiers, calculer et afficher leur moyenne.