Leçon 1 : Généralité sur le langage C++

Table des matières

1- Structure du langage C++

1.1 - Présentation

En 1972, *Dennis Ritchie* a conçu un langage de programmation structurée pour développer une version portable du système d'exploitation UNIX dans les « Bell Labs » : *Langage C.*

En 1982, Bjarne Stroustup à intégré la programmation orientée objet au langage C dans les laboratoires d'AT&T Bell, d'où le successeur de C : *le langage C++* qui est en fait une surcouche du *C (C++ signifie une incrémentation du C)*.

C++ n'était pas le premier langage de programmation qui a intégré la programmation orientée objet, en effet il y avait :

- *Simula* (Simple universal language) est le langage qui a introduit le paradigme orienté objet en programmation, en 1960, et il est donc considéré comme le premier langage à objet et le prédécesseur de langages ultérieur tels que Smalltalk et C++.
- *Smalltalk* est un des premiers langages de programmation objet. Il a été créé en 1972.

1.2 - Architecture du langage C++

Un programme simple se compose de plusieurs parties :

- Des directives de pré-compilation

- Les différentes fonctions éventuelles dont l'une s'appelle obligatoirement main() et à l'intérieure de laquelle est défini le programme principal

- La déclaration de toutes les variables et constantes

- Les différentes instructions du programme principal

La fonction main() est utilisée pour définir le programme principal, cette fonction commence par une accolade ouvrante «{»et se termine par une accolade fermante «}».

Exemple :

```
main()
{
Corps du programme
}
```

1.3 - Les directives de pré-compilation du langage C++

Encore appelées fonctions bibliothèques, les directives commencent par le symbole #.

ci-dessous quelques exemples de bibliothèques :

#include <iostream> : elle contient un certain nombre de définitions d'objets intervenant dans les entrées/sorties du programme, c'est-à-dire dans l'affichage à l'écran ou dans des fichiers. La définition de cout se trouve dans ce fichier;

#include <cmath>: elle permet d'utiliser les fonctions mathématiques.

#include <conio.h>: elle permet de manipuler la console Windows (par exemple effacer l'écran, changer la couleur de police de console etc.).

#include <string>: elle permet l'utilisation des fonctions liées aux chaînes de caractères.

Remarque

using namespace std : Cette instruction permet de mettre à disposition un certains nombres de fonctionnalités , elle est appelée toujours après les directives.

Application 1 :

II

Exercice

Quelle est la structure d'un programme en C++ :

- main(){}
- {main()}
- main({})

Exercice

La directive permettant de manipuler les chaîne de caractères est :

2- Les variables et les types en langage C++

2.1- Les variables

Comme nous l'avons vu en algorithmique, Une variable est un espace en mémoire qui va recevoir une information à un moment donné du traitement.

Syntaxe de déclaration :

TYPE nomvariable; Ou TYPE nomvariable[Taille] pour les variables de type chaîne de caractères

2.2- Les constantes

Contrairement à la variable, une constante ne change pas de valeur durant le déroulement du programme. Il existe deux méthodes de définition d'une constante (soit avec Const ou la #define).

Sa syntaxe est la suivante :

Méthode 1: Const Type nomconstante = valeur;

Méthode 2: #define nomconstante valeur;

2.3- Les types

Le type de données spécifie la taille occupée par les données en mémoire, les données qui lui sont applicables ainsi que l'intervalle de données autorisé.

Les différents types en langage pascal sont consignés dans le tableau ci-dessous :

Type	*Description*
bool	Une valeur parmi deux possibles, vrai (true) ou faux (false)
char	Un caractère
int	Un nombre entier.
double	Un nombre à virgule.
string	Une chaîne de caractères, c'est-à-dire un mot ou une phrase.
unsigned int	Un nombre entier positif ou nul.

Application 2 :

IV

Exercice

Donner la déclaration de la constante pi qui a la de 3,14 :

3.14

Exercice

Donner la déclaration d'un tableau temp d'entier de taille 10 :

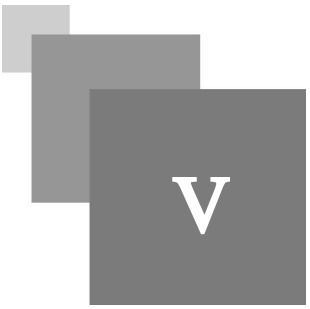
Exercice

B = A>C ;

Faites la déclaration de B.

;

3- Les opérateurs en langage C++



Définition : 3.1 Définition

Un opérateur est un signe qui relie au moins deux valeurs, pour produire un résultat. On distingue à cet effet :

- Les opérateurs arithmétiques
- Les opérateurs de comparaisons ou relationnels
- Les opérateurs logiques
- Les opérateurs de concaténations

3.2 Les opérateurs arithmétiques

Opérateur Arithmétique	*Représentation en C++*
Addition	+
Soustraction	-
Multiplication	*
Division	/
Modulo	%
Incrémentation	++
La décrémentation	--

3.3 Les opérateurs de comparaisons ou relationnels

Ces opérateurs donnent toujours un résultat de type *Booléen (0 ou 1)*.

Opérateur comparaisons	*Représentation en C++*
Supérieur	>
Inférieur	<
Égalité	==
Supérieur ou égal	>=
Inférieur ou égal	<=
Différent	!=

3.4 Les opérateurs logiques

Opérateur logiques	*Représentation en C++*
La négation (non)	!
L'intersection (et)	&&
L'union (ou)	||

Remarque :

- Une condition est une expression de type logique. Ils lui correspondent deux valeurs possibles *false (0) et True (1)*
- *La valeur que retourne la condition est de type entier.*

Exemple : Tableau d'évaluation des opérateurs logiques

Soit A et B deux conditions.

- Opérateur de négation

A	*! A*
True	False
False	True

B	*! B*
True	False
False	True

- Opérateur d'intersection

A	*B*	*A && B*
True	True	True
False	False	False
True	False	False
False	True	False

- Opérateur d'union

A	*B*	*A || B*
True	True	True
False	False	False
True	False	True
False	True	True

3.5 Les opérateurs d'assignation

=	L'affectation
+=	La somme et affectation
-=	La soustraction et l'affectation
/=	La division et l'affectation
*=	Le produit et l'affectation
%=	Le modulo et l'affectation

Exemple :

A = 5 et B =4

A+=B <=> A = A+B = 5+4 = 9

Application 3

VI

Exercice

Donner les réponses des opérations suivantes :

1. 15 == 2 =
2. 15 > 14 =
3. 15 % 2=
4. A = 5 ;

   B = 6 ;

   A= A++ =

   B= B-- =

4- Les instructions entrées-sorties en langage C++

VII

4.1- L'instruction de sortie : cout

Cout est un flux de sortie du programme (Console Output : sortie console). Ce flux de sortie est envoyé par défaut vers l'écran. Il va nous servir à afficher des messages à l'écran en utilisant l'opérateur <<.

Syntaxe :

cout<<"Message"; ou *cout<<variable*; ou *cout<<"Message"<<variable;*

Exemple 1 :

cout<<"Bonjour chers étudiants"; // Cette instruction affiche *Bonjour chers étudiants* à l'écran.

Remarque :

endl (End of Line) permet de passer à la ligne suivante.

Exemple 2 :

cout<<"Bonjour chers étudiants"<<endl; // Cette instruction affiche *Bonjour chers étudiants* à l'écran et met le curseur à la ligne suivante.

4.2- L'instruction d'entrée : cin

L'instruction d'entrée *cin* permet de récupérer une information saisie au clavier par l'utilisateur en utilisant l'opérateur >>.

cin>>variable;

Application 4 :

VIII

Exercice

Quel résultat obtient on dans chacun des cas ci-dessous :

A = 5 ; B = 6 ; cout<<A++ ; = ; cout<<B--;= ;

soit A une variable, donner l'instruction qui permettra à A de recevoir la valeur saisie par l'utilisateur en revenant à la ligne : ;

soit val2 = 15 + 9, donner le résultat de cout<<"val2 = "<<val2 :

Travaux Pratique 1

IX

Énoncé :

Écrire un programme qui permet de calculer le périmètre et l'aire du rectangle.

```
#include <iostream> //permet d'utiliser le cout
using namespace std; // permet d'utiliser le endl

 main() // début du programme
{
  float lon,lar,peri,aire; //déclaration des variables
  cout << "Donner la longueur du rectangle " << endl; // envoi le message et retourne à la ligne
  cin >> lon;  // récupère la longueur dans la variable lon
  cout << "Donner la largeur du rectangle " << endl; // envoi le message et retourne à la ligne
  cin >> lar; // récupère la largeur dans la variable lar
  peri = (lon+lar)*2; // calcule le périmètre
  aire = lon*lar; // calcule l'aire
  cout << "Le périmètre du rectangle est " << peri << endl; // envoi le message + la valeur de peri et retourne à la ligne
  cout << "L'aire du rectangle est " << aire << endl; // envoi le message + la valeur de aire et retourne à la ligne

}

```