CHAPITRE 1

ELEMENTS DE LANGAGE C++

Les exercices ont été testés avec les outils BORLAND C++ BUILDER (toute version) en mode « console » et BC5. Le corrigé des exercices et le listing de ces programmes se trouvent à la fin de chaque chapitre et sont téléchargeables.

Pour avancer un peu plus vite et aborder l'essentiel de la Programmation Orientée Objet (P.O.O.), on pourra étudier les chapitres et paragraphes marqués de ***, dans un deuxième temps.

INTRODUCTION

Le langage C++ est un langage évolué et structuré. C'est en ce sens une évolution du langage C.
Il possède en outre les fonctionnalités de la programmation orienté objet.
Le langage C++ se trouve à la frontière entre le langage C, non objet, et le langage JAVA conçu d'emblée en orienté objet.

On trouve sur le marché un grand nombre de compilateurs C++ destinés à différents microprocesseurs ou microcontrôleurs.

Le langage C++ possède assez peu d'instructions, il fait par contre appel à des bibliothèques, fournies en plus ou moins grand nombre avec le compilateur.

exemples: math.h : bibliothèque de fonctions mathématiques
iostream.h : bibliothèque d'entrées/sorties standard
complex.h : bibliothèque contenant la classe des nombres complexes.

On ne saurait développer un programme en C++ sans se munir de la documentation concernant ces bibliothèques.

ETAPES PERMETTANT L'EDITION, LA MISE AU POINT,

L'EXECUTION D'UN PROGRAMME

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

*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- **Editions 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 (commande « build all »).

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

Les compilateurs permettent en général de construire des programmes composés de plusieurs fichiers sources, d'ajouter à un programme des unités déjà compilées. On dit alors que l'on travaille par gestion de projet.

Exercice I-1: Editer (EXI_1.CPP), compiler et exécuter le programme suivant:

```
#include <iostream.h>  // sorties standards
#include <conio.h>     // les commentaires s'écrivent derrière 2 barres
    void main()
   {
      cout<<"BONJOUR";//affichage d'un message sur l'écran
      cout<<" Belle journée!!";//affichage d'un autre message sur l'écran
      cout<<"Pour continuer frapper une touche...";
   // Attente d'une saisie clavier pour voir l'écran d'exécution
      getch();
   }
```

Le langage C++ distingue les minuscules, des majuscules. Les mots réservés du langage C++ doivent être écrits **en minuscules.**

On a introduit dans ce programme la notion d'interface homme/machine (IHM).
- L'utilisateur visualise une information sur l'écran,
- L'utilisateur, par une action sur le clavier, fournit une information au programme.

Les instructions sont exécutées **séquentiellement**, c'est à dire les unes après les autres. L'ordre dans lequel elles sont écrites a donc une grande importance.

Echanger les 2 premières instructions, puis exécuter le programme.

Modifier maintenant le programme comme ci-dessous, puis le tester :

```
//les commentaires s'écrivent derrière 2 barres obliques

#include <iostream.h>  //sorties standard
#include <conio.h>

    void main()
   {
      int 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 ;  //
      cout <<" Affichage de a : "<< a<<"\n";
      cout <<" Affichage de b : "<< b<<"\n";
      cout <<" Voici le résultat : "<< calcul<<"\n";
      cout<<"Pour continuer frapper une touche...";
      getch();  // Attente d'une saisie clavier
   }
```

Dans ce programme, on introduit 3 nouveaux concepts :
- La notion de déclaration de variables : les variables sont les données que manipulera le programme lors de son exécution. Ces variables sont rangées dans la mémoire vive de l'ordinateur. Elles peuvent être déclarées au moment où on en a besoin dans le programme. Pour une meilleure lisibilité, il est conseillé de les déclarer au début (sauf peut-être pour des variables créées par commodité et qui ne servent que très localement dans le programme).
- La notion d'affectation, symbolisée par le signe =. La source de l'information est à droite du signe =, la destination à gauche.

a = 10; signifie « a prend la valeur 10 »

s = a + b; signifie « s prend la valeur a + b »

s = s + 5; signifie « la nouvelle valeur de s est égale à l'ancienne + 5 »

- La notion d'opération. Un programme informatique est exécuté séquentiellement, c'est à dire une instruction après l'autre. Lorsque l'instruction **s = a + b** est exécutée, **a** possède la valeur 10, et **b** possède la valeur 50.

LES DIFFERENTS TYPES DE VARIABLES

1- Les entiers

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

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

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.

Exemples:

Les caractères alphanumériques s'écrivent entre ‘ ‘

Le **caractère** 'b' a pour valeur 98.

Le **caractère** 22 a pour valeur 22.

Le **caractère** 127 a pour valeur 127.

Le **caractère** 257 a pour valeur 1 (ce nombre s'écrit sur 9 bits, le bit de poids fort est perdu).

Quelques constantes caractères:

CARACTERE		VALEUR (code ASCII)	NOM ASCII
'\n'	interligne	0x0a	LF
'\t'	tabulation horizontale	0x09	HT
'\v'	tabulation verticale	0x0b	VT
'\r'	retour chariot	0x0d	CR
'\f'	saut de page	0x0c	FF
'\\'	backslash	0x5c	\
'\'	cote	0x2c	'
'\"'	guillemets	0x22	"

Modifier ainsi le programme et le tester :

```
#include <iostream.h>  // sorties standard
#include <conio.h>     // les commentaires s'écrivent derrière 2 barres

    void main()
   {
      int a, b, calcul ;  // déclaration de 3 variables
      char u ,v ;
      cout<<"BONJOUR"; // affichage d'un message sur l'écran
      a = 10 ; // affectation
      b = 50 ; // affectation
      u = 67 ;
      v = 'A' ;
      calcul = (a + b)*2 ; //affectation et opérations
      cout <<" Affichage de a : "<< a<<"\n";
      cout <<" Affichage de b : "<< b<<"\n";
      cout <<" Voici le résultat : "<< calcul<<"\n";
      cout <<" Affichage de u :"<< u <<"\n";
      cout <<" Affichage de v :"<< v <<"\n" ;
      cout<<"Pour continuer frapper une touche...";
      getch(); // Attente d'une saisie 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

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.

SORTIES DE NOMBRES OU DE TEXTE A L'ECRAN

L'OPERATEUR COUT

Ce n'est pas une instruction du langage C++, mais une fonction de la bibliothèque iostream.h.

Exemple: affichage d'un texte:

```
cout <<"BONJOUR";        // pas de retour à la ligne du curseur après l'affichage
cout <<"BONJOUR\n";      // affichage du texte, puis retour à la ligne du curseur
```

Exercice I-2: Tester le programme suivant et conclure.

```
#include <iostream.h>
#include <conio.h>

    void main()
   {
      cout<<"BONJOUR " ;
      cout <<"IL FAIT BEAU\n";
      cout <<"BONNES VACANCES";
      cout <<"Pour continuer frapper une touche...";
      getch();  // Attente d'une saisie clavier
   }
```

Exercice I-3: Affichage d'une variable de type **int** ou **float:**

Tester le programme suivant et conclure.

```
#include <iostream.h>
#include <conio.h>

    void main()
   {
      int u = 1000 ;
      float s = 45.78 ;
      cout <<"Voici u (en base 10) :  " << u << "\n";
      cout <<"Voici u (en hexa) : "<< hex << u <<"\n";
      cout <<"Voici s : "<< s << "\n";
      cout <<"Pour continuer frapper une touche...";
      getch();  // Attente d'une saisie clavier
   }
```

Affichage multiple: modifier le programme précédent ainsi, et conclure.

```
#include <iostream.h>
#include <conio.h>

    void main()
   {
      int u;
      float s;
      u = 1000;
      s = 45.78;
      cout <<"Voici u (base 10) : "<< u << "\nVoici s : " << s << "\n";
      cout <<"Pour continuer frapper une touche...";
      getch();  // Attente d'une saisie clavier
   }
```

Exercice I-4:

a et b sont des entiers, a = -21430 b = 4782, calculer et afficher a+b, a-b, a*b, a/b, a%b en soignant l'interface homme/machine.
Indication: a/b donne le quotient de la division, a%b donne le reste de la division.

Exercice I-5: Affichage d'une variable de type **char :** tester le programme ci-dessous et conclure.

```
#include <iostream.h>
#include <conio.h>

    void main()
   {
      char u,v,w;
      int i;
      u = 'A';
      v = 67;
      w = 0x45;
      cout<<"Voici u : "<< u << "\n";
      cout<<"Voici v : "<< v <<"\n";
      cout<<"Voici w : "<< w <<"\n";
      i = u; // conversion automatique de type
      // pour obtenir le code ascii de la lettre A en base 10
      cout<<"Voici i : "<< i << "\n";
      // pour obtenir le code ascii de la lettre A en hexadécimal
      cout<<"Voici i : "<< hex << i << "\n";
      cout<<"Pour continuer frapper une touche...";
      getch();  // Attente d'une saisie clavier
   }
```

Exercice I-6:

Pour votre compilateur C++, la taille des entiers est de 32 bits;
Que va-t-il se passer, à l'affichage, lors de l'exécution du programme suivant ?

```
#include <iostream.h>
#include <conio.h>

    void main()
   {
      int a = 12345000, b = 60000000, somme;
      somme=a*b;
      cout<<"a*b = "<<somme<<"\n";

      cout <<"Pour continuer frapper une touche...";
      getch();  /* Attente d'une saisie clavier */
   }
```

Exercice I-7:

a et b sont des réels, a = -21,43 b = 4,782, calculer et afficher a+b, a-b, a*b, a/b, en soignant l'interface homme/machine.

LES OPERATEURS

Opérateurs arithmétiques sur les réels: + - * / avec la hiérarchie habituelle.

Opérateurs arithmétiques sur les entiers: + - * / (quotient de la division) % (reste de la division) avec la hiérarchie habituelle.

Exemple particulier: char c, d; c = 'G'; d = c+'a'-'A';
Les caractères sont des entiers sur 8 bits, on peut donc effectuer des opérations. Sur cet exemple, on transforme la lettre majuscule G en la lettre minuscule g.

Opérateurs logiques sur les entiers:

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

Exemples: p = n « 3; // p est égale à n décalé de 3 bits à gauche
p = n » 3; // p est égale à n décalé de 3 bits à droite

L'opérateur **sizeof(type)** renvoie le nombre d'octets réservés en mémoire pour chaque type d'objet.
Exemple: n = sizeof(char); /* n vaut 1 */

Exercice I-8: n est un entier (n = 0x1234567a), p est un entier (p = 4). Ecrire un programme qui met à 0 les p bits de poids faibles de n.

Exercice I-9: quels nombres va renvoyer le programme suivant ?

```
#include <iostream.h>
#include <conio.h>

    void main()
   {
      cout<<"TAILLE D'UN CARACTERE : "<<sizeof(char)<< "\n";
      cout<<"TAILLE D'UN ENTIER : "   <<sizeof(int)<< "\n";
      cout<<"TAILLE D'UN REEL : "     <<sizeof(float)<< "\n";
      cout<<"TAILLE D'UN DOUBLE : "   <<sizeof(double)<< "\n";
      cout <<"Pour continuer frapper une touche...";
      getch();  // Attente d'une saisie clavier
   }
```

INCREMENTATION - DECREMENTATION

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--;**

OPERATEURS COMBINES

Le langage C++ autorise des écritures simplifiées lorsqu'une même variable est utilisée de chaque côté du signe = d'une affectation. Ces écritures sont à éviter lorsque l'on débute l'étude du langage C++ car elles nuisent à la lisibilité du programme.

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

LES DECLARATIONS DE 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;
// ...
}
```

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 un modificateur« const » à tous les types.

2eme 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 équivalence « lexicale ».
Les constantes déclarées par #define s'écrivent traditionnellement en majuscules, mais ce n'est pas une obligation.

LES CONVERSIONS DE TYPES

Le langage C++ permet d'effectuer automatiquement des conversions de type sur les scalaires:

Exemple et exercice I-11:

```
 void main()
{
   char c=0x56,d=25,e;
   int i=0x1234,j;
   float r=678.9,s;
   j = c; // j vaut 0x0056, utilisé précédemment  pour afficher
          // le code ASCII d'un caractère
   j = r; // j vaut 678
   s = d; // s vaut 25.0
   e = i; // e vaut 0x34
}
```

Une conversion de type float --> int ou char peut-être *dégradante.*

Une conversion de type int ou char --> float est dite *non dégradante.*

CORRIGE DES EXERCICES

Exercice I-4:

```
#include <iostream.h>
#include <conio.h>

    void main()
   {
      int a,b;
      a= -21430;
      b= 4782;
      cout<<"A + B = "<< a+b <<"\n";
      cout<<"A - B = "<< a-b <<"\n";
      cout<<"A x B = "<< a*b <<"\n";
      cout<<"A sur B = "<< a/b <<"\n" ;
      cout<<"A mod B = "<< a%b <<"\n";
      cout<<"Pour continuer frapper une touche...";
      getch();  // Attente d'une saisie clavier
   }
```

Exercice I-7:

```
#include <iostream.h>
#include <conio.h>

    void main()
   {
      float a,b;

      a= -21430;
      b= 4782;

      cout<<"A + B = "<< a+b <<"\n";
      cout<<"A - B = "<< a-b <<"\n";
      cout<<"A x B = "<< a*b <<"\n";
      cout<<"A sur B = "<< a/b <<"\n" ;

      cout<<"Pour continuer frapper une touche...";
      getch();  // Attente d'une saisie clavier
   }
```

Exercice I-8:

```
#include <iostream.h>
#include <conio.h>

    void main()
   {
      int n,p,masque;

      n= 0x1234567a;
      p = 4;

      cout<<"valeur de n avant modification:"<< hex << n <<"\n";
      n = n >> p;
      n = n << p;

      cout<<"n modifié vaut:"<< hex << n <<"\n";
      cout <<"Pour continuer frapper une touche...";

      getch();  // Attente d'une saisie clavier
   }
```

Exercice I-9:

Avec le compilateur C++ utilisé :

- sizeof(char) vaut 1
- sizeof(int) vaut 4
- sizeof(float) vaut 4
- sizeof(double) vaut 8.