# Exercices C++ - *Semestre 1*
# *BTS SN 1ère année*

David Guilbert

September 17, 2018

Recueil d'exercices de C++.

*Conseil : Tous les exercices ne sont pas forcément à faire mais pour progresser, il est bon de s'exercer régulièrement. A bon entendeur ..*

Tous les programmes C++ doivent être développés à partir du canevas suivant :

```
#include <iostream>
//Commentaires expliquant le role du programme
//
//

using namespace std;        //utilisation de l'espace de nom std

//Fonction main : fonction principale
//Point d'entrée du programme !!
int main(int argc, char *argv[])
{
    //declarations de variables avec leur initialisation

    //corps du programme
    //...
    //...

    return 0;   //Valeur de retour de la fonction main
}
```

Figure 1: Canevas d'un programme C++ (.cpp)

Nous verrons plus tard le rôle de *argc* et *argv*

La fonction *main* est la fonction principale du programme. Le programme débute son exécution à partir de cette fonction *main*. La fin de l'exécution du programme correspond à la fin de l'exécution de la fonction *main*.
Les lignes du programme commençant par # ne sont pas des instructions du C++ ! Ce sont des directives de pré-compilation. Elles indiquent au préprocesseur (rien à voir avec le microprocesseur) un certain nombre de choses à faire avant d'entamer la compilation à proprement parlé.
Par exemple, ici, *#include <iostream>* demande au préprocesseur de remplacer cette ligne par le contenu (légèrement adapté) du fichier d'entête "iostream.h".
Les exercices qui suivent doivent permettre de se familiariser avec les notations, les types et les opérations de base du langage C++. Vous pouvez vous référer au site suivant : http://www.cplusplus.com/reference/
La manière de présenter certaines opérations sera à éviter. La règle consiste à aérer votre code, à utiliser les parenthèses pour obliger le compilateur à effectuer certaines opérations avant d'autres. Ces exercices n'ont qu'un but pédagogique!

# 1 Éléments de base du langage C/C++

## 1.1 Types de base, opérateurs et expressions

- Exercice 1

  Éliminer les parenthèses superflues dans les expressions suivantes :

$$a = (x + 5) \tag{1}$$
$$a = (x = y) + 2 \tag{2}$$
$$a = (x == y) \tag{3}$$
$$(a < b)\&\&(c < d) \tag{4}$$
$$(i++) * (n + p) \tag{5}$$

- Exercice 2

  Quelle est la valeur affectée aux différentes variables concernées par chacune des instructions suivantes :

```
int n = 5, p = 9;
int q;
float x;

q = n < p;                  //1
q = n == p;                 //2
q = p % n + p > n;          //3
x = p / n;                  //4
x = (float) p / n;          //5
x = (p + 0.5) / n;          //6
x = (int) (p + 0.5) / n;    //7
q = n * (p > n ? n : p);    //8
q = n * (p < n ? n : p);    //9
```

- Exercice 3

  Quels résultats fournira le programme suivant :

```
int n,p,q;

n = 5;
p = 2;
q = n++ > p || p++ != 3;
cout << "A : n = " << n << " p = " << p << " q = " << q << endl;

n = 5;
p = 2;
q = n++ < p || p++ != 3;
cout << "B : n = " << n << " p = " << p << " q = " << q << endl;

n = 5;
p = 2;
q = ++n == 3 && ++p == 3;
cout << "C : n = " << n << " p = " << p << " q = " << q << endl;

n = 5;
p = 2;
q = ++n == 6 && ++p == 3;
cout << "D : n = " << n << " p = " << p << " q = " << q << endl;
```

- Exercice 4

Créer un projet "Non-Qt Project/Plain C++ Application" dans QtCreator.

```
int main(int argc, char *argv[])
{
    char c,d,e,f;

    c = 0x41;
    d = 65;
    e = 'A';
    f = 2*c - d;

    cout << c << endl << d << endl << e << endl << f << endl;

    return 0;   //Valeur de retour de la fonction main
}
```

Que peut-on dire des variables c,d,e,f ?

Changer les valeurs de c,d,e,f afin d'afficher le mot "ROSE". Pour cela, trouver les valeurs manquantes dans le code suivant :

```
c = 0x__;
d = __9;
e = '_'+ 0x__;
f = 2*c - d - 0x__;
```

Pensez à supprimer les sauts de ligne entre les caractères. Vous pourrez vous aider d'une table ASCII, facilement trouvable sur internet.

Exécution en pas à pas :
Vous pouvez exécuter votre programme en pas à pas, c'est à dire instruction par instruction et vous permettre de voir l'état des variables. Placer un point d'arrêt sur la première instruction (celle effectuant l'affectation de c) :

Pour cela, cliquer sur la gauche du programme. 


Lancer le débuggage en cliquant sur le bouton triangle vert avec bug : 
Vous pouvez ouvrir une fenêtre supplémentaire permettant d'observer les valeurs des variables (*Fenêtre/Vues/local and expressions* Il est possible que cette fenêtre s'ouvre automatiquement lors du lancement en mode *debug* du programme.
A chaque appui sur F10, l'exécution continue à l'instruction suivante et les variables sont, éventuellement, modifiées.
Attention : 

signifie que l'instruction pointée par la flèche jaune n'a pas encore été exécutée. La flèche pointe donc la prochaine instruction à exécuter.

- Exercice 5

En utilisant l'opérateur *cin* >>, modifier le programme précédent pour permettre à l'utilisateur de saisir une valeur entière.
Cette valeur entière sera ajoutée aux variables c,d,e,f avant l'affichage.
Par exemple : l'utilisateur saisit la valeur 4 alors le mot affiché sera 'VSWM' au lieu de 'ROSE' Il faut par conséquent créer une variable supplémentaire. Vous choisirez le type de cette variable entre *char , int*
Que se passe-t-il lorsqu'on déclare cette variable en *char* ?

- Exercice 6

Créer un nouveau projet *non-Qt Project*
Saisir le code suivant :

```
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
    string texte;
    int longueur = 0;

    cout << "Saisir une chaine" << endl;
    cin >> texte;

    longueur = texte.length();

    for (int i=0; i<longueur; i++)
    {
        cout << texte[i];
    }

    cout << endl;

    return 0;
}
```

Mettre un point d'arrêt sur la ligne cout <<texte[i]
Observer l'évolution de la variable i dans la fenêtre "locals and expressions".

Modifier le code pour obtenir :

```
#include <iostream>

using namespace std;
int main(int argc, char *argv[])
{
    string texte;
    int longueur = 0;

    cout << "Saisir une chaine" << endl;
    cin >> texte;
    longueur = texte.length();
    for (int i=0; i<longueur; i++)
    {
        cout << texte[i]++;
    }
    cout << endl;
    cout << texte ;
    cout << endl;

    return 0;
}
```

Relancer le programme; vous observez que le texte saisi par l'utilisateur a été modifié. Pourquoi l'affichage de la chaîne dans la boucle *for* ne correspond pas à la chaîne affichée par la ligne *cout <<texte ;* ?

Pourtant on modifie son contenu en faisant *texte[i]++*

Remarque : par défaut, la saisie d'une chaîne se termine à l'appui sur entrée mais l'enregistrement dans 'texte' sera effectué jusqu'au premier espace saisi (non inclus) Autrement dit, si vous saisissez "Bonjour monsieur" seul "Bonjour" sera sauvegardé dans la chaîne 'texte'.

Pour prendre en compte les espaces, il faut remplacer la ligne *cin >>texte;* par *getline(cin,texte);*

## 1.2 Les boucles et tests

- Exercice 1

En s'inspirant des exercices précédents, réaliser un programme permettant de compter le nombre de voyelles dans un texte saisi par l'utilisateur. Le programme affichera ce nombre de voyelles et par conséquent le nombre de consonnes.
Il faudra tester chaque lettre du texte saisi; ce test devra être inséré dans la boucle for qui parcourt le texte.
On considère que le texte saisi ne comporte que des lettres de l'alphabet.

Structure du test :

```
if (texte[i] == 'o')
{
    //Faire quand le test est vrai
}
else
{
    //Faire quand le test est faux
}
```

Dans cet exemple, on teste le (i+1)ème caractère de texte afin de savoir s'il vaut 'o'.

- Exercice 2

Écrire un programme qui détermine la nième valeur $u_n$ (n étant saisie par l'utilisateur) de la suite de Fibonacci définie comme suit :

$u_1 = 1$

$u_2 = 1$

$u_n = u_{n-1} + u_{n-2} \quad pour\ n > 2$