TP Algorithmique 2

Les 5 règles d'écriture d'un programme :

- Mettre un entête
- Mettre des commentaires
- Respecter l'indentation
- Choisir des noms de variables explicites
- Choisir des noms de fichiers explicites

Pour chaque exercice (sauf les exercice 1, 9 et 10), il faudra :

- compléter un tableau récapitulatif des variables (=> document LibreOffice Writer);
- écrire l'algorithme (=> document LibreOffice Writer);
- écrire le programme source (fichier .cpp);
- générer l'exécutable;
- tester votre programme et le corriger en cas de problème.

Apport de connaissances :

1. Il est possible d'initialiser une variable au moment sa déclaration :

Exemple :

Algorithme :	=> En langage C++ :
Entier Num ← 5	int Num = 5;

2. Opérateurs ++ et -- en C++

En langage C++ :

```
int i;   // Déclaration d'une variable de nom i et de type int

i = 8;   // On affecte 8 dans la variable i
i++ ;    // incrémentation de la variable i (on lui ajoute 1)
         // instruction équivalente à i = i + 1;
i-- ;    // décrémentation de la variable i (on lui soustrait 1)
         // instruction équivalente à i = i - 1;
```

Exercice 1 :

- Déterminez ce que calcule l'algorithme ci-dessous.
- Réécrivez-le avec une structure itérative Tant que …

```
VARIABLES
        Entier nbre
        Entier somme
        Entier i
DEBUT
        somme ← 0
        Pour i allant de 1 à 4 par pas de 1 faire
                Afficher("Entrer une valeur numérique entière :")
                Saisir(nbre)
                somme ← somme + nbre
        Fin pour
        Afficher(" La somme vaut : ", somme)
FIN
```

Exercice 2 :

Ecrire un algorithme qui demande à l'utilisateur un nombre compris entre 0 et 3 (inclus) jusqu'à ce que la réponse convienne. Dans le cas où la réponse ne convienne pas, il faudra le signaler à l'utilisateur.

Exercice 3 :

Ecrire un algorithme qui demande un nombre compris entre 10 et 20 (inclus), jusqu'à ce que la réponse convienne. En cas de réponse supérieure à 20, on fera apparaître un message : « Plus petit ! », et inversement, « Plus grand ! » si le nombre est inférieur à 10.

Exercice 4 :

Ecrire un algorithme qui demande un nombre de départ, et qui ensuite affiche les dix nombres suivants. Par exemple, si l'utilisateur entre le nombre 17, le programme affichera les nombres de 18 à 27.

Exercice 5 :

Ecrire un algorithme qui demande un nombre de départ, et qui ensuite écrit la table de multiplication de ce nombre, présentée comme suit (cas où l'utilisateur entre le nombre 7) :

Table de 7 :
7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
…
7 x 10 = 70

Exercice 6 :

Ecrire un algorithme qui demande un nombre de départ, et qui calcule la somme des entiers jusqu'à ce nombre. Par exemple, si l'on entre 5, le programme doit calculer :

1 + 2 + 3 + 4 + 5 = 15

NB : on souhaite afficher uniquement le résultat, pas la décomposition du calcul.

Exercice 7 :

Ecrire un algorithme qui demande successivement 10 nombres à l'utilisateur, et qui lui dise ensuite quel était le plus grand parmi ces 10 nombres :

Entrez le nombre numéro 1 : 12
Entrez le nombre numéro 2 : 14
etc.
Entrez le nombre numéro 10 : 6
Le plus grand de ces nombres est : 14

Modifiez ensuite l'algorithme pour que le programme affiche de surcroît en quelle position avait été saisie ce nombre :

C'était le nombre numéro 2

Exercice 8 :

Réécrire l'algorithme précédent, mais cette fois-ci on ne connaît pas d'avance combien l'utilisateur souhaite saisir de nombres. La saisie des nombres s'arrête lorsque l'utilisateur entre un zéro.

Exercice 9 :

Soit l'algorithme suivant :

```
VARIABLES
        Entier X
        Entier Somme
DEBUT
        Somme ← 0
        Pour X allant de 1 à 10 par pas de 1 faire
                Somme ← Somme + X*1
        Fin pour
        X ← Somme - 50
        Afficher( "X = " )
        Afficher( X )
FIN
```

Exécuter cet algorithme "dans votre tête".
Quelle est la valeur de la variable X à la fin de l'exécution de ce programme ?
Traduire cet algorithme en langage C++, générer l'exécutable et exécuter le.

Exercice 10 :

Soit l'algorithme suivant :

```
ALGORITHME Tri par permutation

VARIABLES
        Réel    Nbre1, Nbre2, Nbre3
        Réel    Var_Temp
DEBUT
        Afficher ( "Nombre 1 ? " )
        Saisir ( Nbre1 )
        Afficher ( "Nombre 2 ? " )
        Saisir ( Nbre2)
        Afficher ( "Nombre 3 ? " )
        Saisir ( Nbre3 )

        Si (Nbre1 > Nbre2 ) Alors               -- Il faut permuter
                Var_Temp ← Nbre1
                Nbre1 ← Nbre2
                Nbre2 ← Var_Temp
        FinSi
        Si ( Nbre2 > Nbre3 ) Alors              -- Il faut permuter
                Var_Temp ← Nbre2
                Nbre2 ← Nbre3
                Nbre3← Var_Temp
        FinSi

        Afficher ( "Apres le Tri, cela donne : " )
        Afficher ( Nbre1 )
        Afficher (" ")
        Afficher ( Nbre2 )
        Afficher (" ")
        Afficher ( Nbre3 )
FIN
```

Exécuter cet algorithme "dans votre tête" et donner les valeurs finales des nombres Nbre1, Nbre2, Nbre3 lorsque l'utilisateur saisit :

- 1, 2, 3 (1ère exécution);
- 5, 2, 3 (2nde exécution);
- 6, 4, 1 (3ème exécution).

Cet algorithme permet-il de trier dans l'ordre croissant les 3 nombres entiers saisis par l'utilisateur?

Traduire cet algorithme en langage C++, générer l'exécutable et exécuter le afin de vérifier vos résultats.