**Université des Sciences et de la Technologie d’Oran Mohamed Boudiaf -USTOMB**

Faculté de Génie Electrique

Master RT/II/ESE/IB

Module : Programmation Orientée Objet

##### TP2 : fonctions en C++

##### Exercice 1 :

Ecrire une fonction **NombreChiffre** de type int qui a une valeur entière N (positive ou négative) de type long comme paramètre et qui fournit le nombre de chiffres de N comme résultat en utilisant la formule suivante: 10p-1 ≤ X < 10p, sachant que ***x*** est un nombre entier de *p* chiffres.

Ecrire programme C++ qui teste la fonction **NombreChiffre**.

***Exemple d’exécution:***

Introduire un nombre: 123

Le nombre 123 a 3 chiffres.

Introduire un nombre: -580499

Le nombre -580499 a 6 chiffres.

**Exercice 2 :**

 Un nombre entier est parfait s'il est égal à la somme de ses diviseurs (sauf lui-même). Ex : 6 = 1 + 2 + 3 est parfait.

1. Ecrire une fonction ***void*** ***somme\_div(…)*** qui calcule la somme des diviseurs d'un nombre passé en paramètre mais qui ne retourne pas de valeurs.
2. Ecrire une fonction ***parfait*** qui teste si un nombre passé en paramètre est parfait et qui retourne ***true*** s'il l'est et ***false*** sinon.
3. Ecrire un programme C++ qui affiche tous les nombres parfaits inférieurs à une certaine limite.

**Exercice 3 :**

 Deux nombres M et N sont appelés nombres amis si la somme des diviseurs de M est égale à N et la somme des diviseurs de N est égale à M

1. Ecrire une fonction ***amis*** qui retourne le nombre\_amis (s’il existe) d’un nombre passé en paramètre, cette fonction utilise la fonction ***somme\_div*** de l’exercice 2.
2. Ecrire un programme C++ qui affiche tous les nombres\_amis inférieurs à une certaine limite.

**Exercice 4 :**

Ecrire un programme C++ qui construit et affiche le triangle de Pascal en calculant les coefficients binomiaux par une fonction nommée ***coeff\_binomial***qui calcule le coefficient binomial de 2 nombres ***n***et ***m***passés en paramètre, à l'aide de la formule ci-dessous et d’une autre fonction ***Fact***.

1

**Exercice 5 :**

 Dans cet exercice, on vous demande de coder et de tester une fonction de tirage aléatoire d’un nombre entier compris entre des valeurs min et max.

Cette fonction utilise des fonctions temporelles comprises dans la librairie **time.**

Faire donc un programme C++ qui :

1. Initialise les valeurs min et max.
2. Initialise le tirage par l’appel suivant : **srand(time(0)).**
3. Affiche en boucle les résultats des tirages aléatoires pour lesquels on codera une fonction nommée **i\_Alea.**
4. Permet de générer et d’afficher de façon aléatoire une lettre (minuscule) de l’alphabet en utilisant une fonction nommée **alpha**. Utiliser pour cela la formule suivante : **‘a’ + rand( ) % (‘z’ – ’a’ + 1)**

**Exercice 6 :**

 Ecrire une fonction calculant le nombre de Fibonacci d’un nombre ***n*** passé en paramètre. Rappelons que le nombre de Fibonacci F(n) est défini comme suit :

 F(0) = 1; F(1) = 1; F(n) = F(n - 1) + F(n-2)

 Comparer entre une solution récursive et une solution non récursive.

**Exercice 7 :**

 Ecrire un programme en C++ qui comporte un menu permettant de calculer 3 fonctions d’un argument *x* passé en paramètre, à savoir les fonctions ***sinus***, ***cosinus*** et ***exponentielle***. Ces fonctions se calculent en utilisant les développements limités suivants:





****

L’utilisateur devra entrer le chiffre 1 pour calculer la valeur de *sin(x)*, 2 pour *cos(x)*, 3 pour *exp(x)* et afficher un message pour tout autre valeur.

**Exercice 8 :**

**1.** Ecrire un programme en C++ permettant d'intégrer, par la méthode des ***trapèzes*** et de ***simpson***, n'importe quelle fonction f(x) (x de type double) entre deux bornes ***A*** et ***B***, par exp: a0 +a1. x + a2. x2

**2.** Intégrer sin(x) et cos(x) entre 0 et pi avec différentes partitions pour l’intervalle [0 - pi] (N = 4, 8, 16, 256, …) et comparer les résultats avec la formule du ***trapèze*** et la formule de ***Simpson***.

**Exercice 8 :**

Soient les déclarations suivantes:

**void** f (**int**, **float**); // fonction I

 **void** f (**float**, **int**); // fonction II

**int** i = 1; **char** c = 'A'; **float** x = 2.f; **double** y = 3.0;

Les appels suivants sont-ils corrects et, si oui, quelles seront les fonctions effectivement appelées et les conversions éventuelles mises en œuvre ?

**6.**  f(x, x);

**7.** f(i, y);

**8.** f(y, i);

**9.**  f(y, y);

**1.**  f(i, c);

**2.** f(c, i);

**3.** f(i, i);

**4.** f(x, c);

**5.** f(c, x);

2