TD 3 – Fonctions et procédures

Objectifs :

- *Créer un projet avec des fichiers sources et des fichiers en-tête*
- *Utiliser correctement des fonctions et des procédures déjà écrites*
- *Définir et appeler correctement des fonctions et des procédures*
- *Utiliser correctement des tableaux*

Pour tous les exercices de ce TD, vous devrez écrire vos fonctions dans un fichier nommé 'mesFonctions.c', créer le fichier 'mesFonctions.h' et avoir enfin un fichier 'programmePrincipal.c' qui ne contiendra que votre main.

Exercice 1

Pour chacun des sous-programmes (SP) suivants, complétez le tableau en déterminant la catégorie du sous-programme (fonction ou procédure), en trouvant un nom au sous-programme, en déterminant le type de retour dans le cas d'une fonction et en identifiant les paramètres nécessaires au sous-programme :

1. Saisir 1 nombre entier.
2. Afficher 3 nombres entiers.
3. Calculer la factorielle d'un nombre.
4. Calculer N à la puissance P.
5. Calculer le PGCD de 2 nombres.
6. Savoir si un nombre Nb1 est multiple d'un nombre Nb2.

Fonction ou Procédure	Nom du SP	Type de retour (si fonction)	Liste des paramètres

Exercice 2

Vous disposez dans Moodle de deux fichiers : MesBibliothèques.h et MesBibliothèques.c .

Dans MesBibliothèques.h, vous avez les prototypes de sous-programmes suivants :

```
#ifndef MESBIBLIOTHEQUES_H
#define MESBIBLIOTHEQUES_H

void setColor(int ForgC) ;
int  minimum (int, int) ;
int  maximum (int, int) ;
int  saisieEntier (int, int) ;
void afficherEntier (char [], int) ;

#endif // MESBIBLIOTHEQUES_H
```

Dans MesBibliothèques.c, vous avez les codes suivants :

```c
#include <stdio.h>
#include <stdlib.h>
#include <windows.h>
#include <dos.h>
#include <dir.h>

/// ////////////////////////////////////////////////////////////////////////// ///
/// Procédure pour changer l'affichage du texte dans la console                ///
/// 0 : noir        4 : rouge       8  : gris foncé     12 : rouge clair       ///
/// 1 : bleu        5 : magenta     9  : bleu clair     13 : magenta clair     ///
/// 2 : vert        6 : brun        10 : vert clair     14 : jaune             ///
/// 3 : cyan        7 : gris clair  11 : cyan clair     15 : blanc             ///
/// source de cette fonction : https://askcodez.com/comment-changer-la-        ///
/// couleur-du-texte-et-de-la-console-de-couleur-dans-codeblocks.html          ///
/// ////////////////////////////////////////////////////////////////////////// ///
void setColor(int ForgC)
{
    WORD wColor;

    HANDLE hStdOut = GetStdHandle(STD_OUTPUT_HANDLE);
    CONSOLE_SCREEN_BUFFER_INFO csbi;

    ///We use csbi for the wAttributes word.
    if(GetConsoleScreenBufferInfo(hStdOut, &csbi))
    {
        ///Mask out all but the background attribute, and add in the forgournd color
        wColor = (csbi.wAttributes & 0xF0) + (ForgC & 0x0F);
        SetConsoleTextAttribute(hStdOut, wColor);
    }
}
```

```c
int minimum (int nb1, int nb2)
{
    if (nb1<=nb2)
        return nb1 ;
    return nb2 ;
}

int maximum (int nb1, int nb2)
{
    if (nb1>=nb2)
        return nb1 ;
    return nb2 ;
}
```

```c
int saisieEntier (int borneInf, int borneSup)
{
    int nb ;
    printf("Veuillez saisir un entier compris entre %d et %d\n", borneInf, borneSup) ;
    scanf ("%d", &nb) ;
    while ( (nb<borneInf) || (nb>borneSup) )
    {
        printf("Veuillez saisir un entier compris entre %d et %d\n", borneInf, borneSup) ;
        scanf ("%d", &nb) ;
    }
    return nb ;
}

void afficherEntier (char message [], int val)
{
    printf("%s %d\n", message, val) ;
}
```

Pour cet exercice, vous devez exclusivement utiliser les fonctions et procédures qui vous sont fournies dans les fichiers qui vous sont fournis.

Créez un projet qui inclue ces 2 fichiers et créez un 3ème fichier 'programmePrincipal.c dans lequel vous allez :

- Déclarer 3 entiers A, B et C,
- Saisir une valeur pour ces 3 entiers
- Afficher en bleu le maximum entre A et B
- Afficher en rouge le minimum entre A et C.

Exercice 3

Vous allez reprendre les exercices réalisés dans le TD n°2 pour l'affichage de lignes et de rectangles en utilisant des procédures. Commencez par écrire une procédure **Ligne** qui affiche une ligne remplie d'un symbole dans une couleur donnée. Le nombre de symboles, le symbole à afficher ainsi que la couleur doivent être des paramètres de la procédure. La procédure Ligne **ne doit pas comporter de retour à la ligne**.

Écrivez le programme principal qui appelle cette procédure d'affichage d'une ligne.

Exercice 4

En utilisant la procédure ligne, écrivez les *procédures* qui permettent d'afficher un rectangle et un triangle rectangle avec une couleur différente pour chaque ligne. La couleur doit être générée aléatoirement. Appelez ces procédures dans un programme principal.

Exercice 5

Écrivez une *fonction* puissance qui élève un nombre entier à une puissance donnée. Le nombre et la valeur de la puissance doivent faire partie des paramètres de cette fonction.

Écrivez un programme principal demande un nombre *Nb* et une puissance *P* à l'utilisateur, qui récupère dans une variable le résultat de Nb à la puissance P et qui affiche cette variable.

Exercice 6

À partir de maintenant, nous allons utiliser des fonctions particulières déjà définies dans le langage C qui permettent de tester vos programmes. Pour cela vous devez inclure la bibliothèque assert.h. La fonction que vous allez utiliser sera la fonction assert.

Exemple d'utilisation de la fonction assert pour tester la fonction minimum :

```
assert (minimum (4, 2) == 2) ;
```

Utilisez la fonction assert pour vérifier que le code que vous avez écrit dans l'exercice précédent est correct.

Exercice 7

Écrivez une *procédure* qui permet d'afficher le signe d'une multiplication sans la calculer mais en testant les valeurs des 2 nombres. Écrivez également le prototype de cette procédure, ainsi que les instructions dans le programme principal qui permettent d'appeler correctement cette procédure.

Exercice 8

Écrivez une *fonction* qui détermine si un nombre est premier ou non. Vous devez également écrire le prototype de cette fonction dans votre fichier .h, écrire le programme principal qui appelle cette fonction et tester également cette fonction avec l'instruction assert.

Exercice 9

Un nombre est parfait s'il est égal à la somme de ses diviseurs stricts (exemple : 6 = 1 + 2 + 3).

Écrivez une *fonction* qui détermine si un nombre est un nombre parfait ou non. Si le nombre est parfait, votre fonction devra renvoyer la valeur 1, et s'il ne l'est pas votre fonction retournera la valeur 0. Écrivez également le prototype de votre fonction ainsi que les instructions dans le programme principal qui permettent d'appeler et de tester cette fonction.

Exercice 10

Dans les prochains exercices, on souhaite écrire un programme qui permet de saisir des notes d'étudiants. Dans un premier temps, écrivez une *fonction* qui retourne une note correcte, c'est-à-dire une note comprise entre 0 et 20. La valeur de la note doit être lue dans la fonction à l'aide d'un scanf.

Écrivez le prototype de votre fonction ainsi que les instructions dans le programme principal qui permettent de l'utiliser correctement.

Exercice 11

Écrivez une *fonction* moyenneNotes qui va demander à l'utilisateur de saisie N notes correctes et qui retourne la moyenne de ces notes.

Écrivez le prototype de votre fonction ainsi que les instructions dans le programme principal qui permettent de l'utiliser correctement.

Exercice 12

À la naissance de Marie, son grand-père Nestor, lui ouvre un compte bancaire. Ensuite, à chaque anniversaire, le grand père de Marie verse sur son compte 100 €, auxquels il ajoute le double de l'âge de Marie. Le jour de sa naissance, il lui verse 100 €, lorsqu'elle a 1 an il lui verse 102 €, lorsqu'elle a deux ans, il lui verse 104 €, etc. Écrire une *fonction* qui permette de déterminer quelle somme aura Marie sur son compte lors de son *n-ième* anniversaire.

Écrivez le prototype de votre fonction ainsi que les instructions dans le programme principal qui permettent de l'utiliser et de la tester correctement.

Exercice 13

La population des Sims Alpha est de *nbA* d'habitants et elle augmente de *plusA* habitants par an. Celle des Sims Beta est de *nbB* habitants et elle augmente de *pourcentB* % par an. Écrire une *fonction* permettant de déterminer dans combien d'années la population de Sims Beta dépassera celle des Sims Alpha.

Écrivez le prototype de votre fonction ainsi que les instructions dans le programme principal qui permettent de l'utiliser correctement.

Exercice 14

Écrire une *procédure* affichant tous les nombres inférieurs à *Nb* égaux à la somme des cubes de leurs chiffres.

Écrivez le prototype de votre procédure ainsi que les instructions dans le programme principal qui permettent de l'utiliser correctement.

Exemple : 153 = 1 + 125 + 27

Exercice 15

Écrivez une *procédure* qui, à partir d'un nombre entier compris entre 1 et 365, affiche la date dans l'année (on suppose que c'est une année non bissextile) sous la forme JJ/MM.

Exemple :

- 1 -> 1/1
- 32-> 1/2
- 365 -> 31/12