TP : Les Fonctions

Il faut pour tous les exercices effectuer le codage en C++.

Vous ne devez **pas utiliser de variables globales** (c'est à dire définies en dehors de toute fonction).

Chaque programme source devra avoir la structure suivante (**dans cet ordre**) :

1- Une entête
2- Directive(s) d'inclusion de fichier(s) d'entête + using namespace std;
3- Déclaration de fonction(s) (*)
4- Définition de la fonction **main()**
5- Définition de fonction(s)

(*) : les **prototypes** des différentes fonctions doivent être **soigneusement justifiés** : **rôle** des fonctions, des paramètres à transmettre et de la valeur de retour.

Exercice 1 :
Ecrire un programme qui **déclare, définit et appelle** une fonction sans paramètre et sans valeur de retour, nommée `AfficheMessage()`, qui affiche un message de bienvenue.

Exercice 2 : calcul de l'aire d'un rectangle
Ecrire un programme qui **déclare, définit et appelle** 4 fonctions :

- la fonction `SaisieLongueur()`, sans paramètre, qui affiche un message à destination de l'utilisateur (l'invitant à saisir la longeur) , réalise la saisie de la longueur et renvoie le résultat;
- la fonction `SaisieLargeur()` qui affiche un message à destination de l'utilisateur, réalise la saisie de la largeur et renvoie le résultat;
- la fonction `CalculAireRectangle()`, sans affichage et sans saisie, qui renvoie l'aire d'un rectangle dont la longueur et la largeur lui sont transmises en paramètre;
- la fonction `AfficheAire()`, qui affiche le message "Valeur de l'aire du rectangle : " suivi de la valeur de cet aire qui lui est transmise en paramètre.

Exercice 3 :
Ecrire un programme qui déclare, définit et appelle une fonction **sans affichage et sans saisie**. Cette fonction devra renvoyer le nombre de jours compris entre deux dates de la même année. Les 2 dates (jour et mois) seront **saisies dans la fonction main().**

Exercice 4 : recherche d'un nombre aléatoire

Soit le programme suivant :

```
#include <iostream>
#include <stdlib.h>          // contient les déclarations des fonctions rand() et srand()
#include <time.h>            // contient la déclaration de la fonction time() et du type time_t

using namespace std;

int main()
{
        time_t t; // déclaration d'une variable nommée t de type time_t
        int NbAleatoire;

        t = time(NULL); // appel de la fonction time()
                // cette fonction renvoie le nombre de secondes écoulées depuis le 1/01/1970 00h00
                        // ( Cet instant constitue "l'origine du temps Unix")

        srand (t);          // appel de la fonction srand() afin de modifier la "graine" utilisée par le
                            // "générateur de nombres pseudo-aléatoires" de la fonction rand()

        NbAleatoire = rand();      // appel de la fonction rand() : génère un nombre aléatoire
                                   // compris entre 0 et RAND_MAX = 2147483647
        NbAleatoire = NbAleatoire % 1000;

        cout << "Nombre aléatoire généré : " << NbAleatoire << endl;

        return 0;
}
```

1. Générer l'exécutable correspondant à ce fichier source et exécuter le plusieurs fois. Que fait ce programme ?

2. Inspirer vous du programme précédent pour créer une fonction sans paramètre, sans affichage et sans saisie, qui renvoie un nombre aléatoire compris entre 0 et 50.

3. Réaliser un programme qui :

- génère un nombre aléatoire compris entre 0 et 50 en appelant la fonction créée à la question précédente;
- demande à l'utilisateur de deviner le nombre aléatoire généré jusqu'à ce qu'il le trouve. Il faudra le guider à l'aide de messages du type "Plus grand", "Plus petit".

Exercice 5 : Manipulation des données d'un tableau

Soit le programme suivant :

```
#include <iostream>
#define TAILLE 10

using namespace std;

// Déclarations des fonctions

/* Déclaration de la fonction Menu() :
Fonction qui affiche le menu, saisit et renvoie le choix de l'utilisateur*/
int Menu();

/* Déclaration de la fonction SaisirTab() :
Fonction qui permet de saisir les éléments d'un tableau d'entiers
- 1er paramètre : adresse du 1er élément du tableau (indice 0)
- 2ème paramètre : nombre d'éléments ou taille du tableau
- Pas de valeur de retour */
void SaisirTab(int * adrPreEl, int Nb);

int main()          // Définition de la fonction main()
{
        // Déclaration des variables locales de la fonction main()
        int Tab[TAILLE]; // Déclaration d'une variable tableau contenant TAILLE éléments de type int
        int choix;          // permet de stocker le choix de l'utilisateur

        cout << "Bonjour, ce programme permet la manipulation d'un tableau de " << TAILLE << " entiers\n";
        do
        {       // Appel de la fonction Menu()
                choix = Menu();
                if (choix == 1)             // cette structure algorithmique n'est pas forcément à conserver
                                            // quand il y a plusieurs choix valides possibles
                {
                        // Appel de la fonction SaisirTab()
                        SaisirTab(Tab, TAILLE);
                }
        }while (choix != 0);
        cout << "Fin du programme\n\n";
        return 0;
}

// Définition de la fonction Menu() : A COMPLETER
int Menu()
{
        // Déclaration des variables locales de la fonction menu()
        int ch;

        cout<<"************ Menu ******************\n";
        cout<<"   Pour saisir les éléments du tableau, tapez \t1\n";
        cout<<"   Pour sortir, tapez \t\t\t\t0\n\n";
        cout<<"-->Entrez votre choix : ";
        cin >> ch;
        cout << endl;
        return ch;
}
```

```
// Définition de la fonction SaisirTab() : terminée
void SaisirTab(int * adrPreEl, int Nb)
{
        // Déclaration des variables locales de la fonction SaisirTab()
        int i;
        for (i = 0; i < Nb; i++ )
        {
                cout << "Saisir l'élément numéro " << i+1 << " : ";
                cin >> adrPreEl[i];
        }
}
```

Remarques :

- Toutes les lignes qui commencent par un # sont des **directives** interprétées par le **préprocesseur** lors de la génération de l'exécutable.
La ligne `#define TAILLE 10` est une **macrodéfinition** : le préprocesseur substituera au symbole TAILLE le "texte" 10, et cela à chaque fois que ce symbole apparaît dans le fichier source.
Intérêt : si on souhaite modifier la taille du tableau, il n'y a qu'une seule ligne de code à modifier.
- Le 1er paramètre de la fonction `SaisirTab()` est du `type int*` (adresse d'entier). Lorsque la fonction `SaisirTab()` est appelée, le système crée sur la pile la variable `adrPreEl` et l'initialise avec la valeur de l'expression `Tab` qui est bien du type `int*` et qui correspond à l'adresse de l'élément d'indice 0 du tableau `Tab` (variable locale de la fonction `main()`). Ainsi quand la fonction `SaisirTab()` "s'exécute", elle connaît les adresses des éléments du tableau `Tab` et peut donc les modifier !

1. Compléter la définition de la fonction `Menu()` afin de proposer les choix supplémentaires suivants :
   - Affichage des éléments du tableau;
   - Incrémentation de tous les éléments du tableau;
   - Décrémentation de tous les éléments du tableau;
   - Ajout à tous les éléments du tableau d'une **même valeur** (passée en paramètre); cette valeur devra être saisie par l'utilisateur en dehors de la fonction correspondante;
   - Calcul de la moyenne des éléments du tableau;

2. Pour chaque choix supplémentaire, déclarer et définir une nouvelle fonction. Les fonctions correspondant aux 4 derniers choix ne devront pas réaliser d'affichage (pas de saisie non plus).

3. Compléter la définition de la fonction `main()` et tester votre programme. Il faudra penser à afficher un message d'erreur quand le choix de l'utilisateur n'est pas valide.

Exercice 6 : Contrôle d'accès à un bâtiment

Pour être autoriser à entrer dans un bâtiment, un employé doit **passer son badge dans un lecteur** et **saisir son code à 4 chiffres** sur un pavé numérique. Chaque badge mémorise l'**identifiant à 10 chiffres** de l'employé auquel il appartient.

Dans ce programme, le **passage du badge dans le lecteur est simulé par la saisie au clavier de l'identifiant**.

Dans cette version, on gère **5 accès "employé"** :
- Identifiant et code d'accès du 1er employé : 1111111111 et 1111
- Identifiant et code d'accès du 2ème employé : 2222222222 et 2222
- Identifiant et code d'accès du 3ème employé : 3333333333 et 3333
- Identifiant et code d'accès du 4ème employé : 4444444444 et 4444
- Identifiant et code d'accès du 5ème employé : 5555555555 et 5555

Le **gardien** du bâtiment possède lui aussi un badge, d'identifiant 1234567890.
Code d'accès du gardien : 1234.

Le programme doit commencer par afficher un message d'accueil (fonction `AfficheAccueil()`):

```
Bienvenue dans la section BTS IRIS
Pour acceder aux locaux, il faut :
    - Passer votre BADGE
    - Saisir votre code d'accès (4 chiffres)
```

Ensuite, le programme doit :
- inviter l' utilisateur à passer son badge et le lire (fonction `AcquerirBadge()`);
- inviter l'utilisateur à saisir son code d'accès et le lire (fonction `AcquerirCodeAcces()`);
- vérifier si l'utilisateur est autorisé à entrer dans le bâtiment (fonction `AutoriserAcces()`);
- afficher le message "Accès autorisé" ou le message "Accès non autorisé"(fonction `AfficheAcces()`);
- recommencer jusqu'à ce que le gardien soit authentifié.

A la fin de l'exécution du programme (quand le gardien s'est authentifié), il faut afficher un message de fermeture du bâtiment (fonction `AfficheFermeture()`):

```
Fermeture du batiment
Bonne soirée à tous
```

Il faut utiliser 2 tableaux :
- un tableau qui permet le stockage de tous les identifiants (y compris celui du gardien). Je vous conseille de placer l'identifiant du gardien dans l'élément d'indice 0;
- un autre tableau qui permet le stockage de tous les codes d'accès **stockés dans le même ordre**.

La fonction `AutoriserAcces()` doit être sans saisie et sans affichage. Elle devra renvoyer une valeur permettant de distinguer :
- une authentification d'un employé "lambda";
- l'authentification du gardien;
- un échec de l'authentification.

Une évolution future du programme prévoit de pouvoir ajouter / modifier / supprimer des accès utilisateurs. De nouvelles fonctions devront être créées, elles seront appelées depuis la fonction main() et elles devront être capables de modifier les 2 tableaux cités ci-dessus. C'est la raison pour laquelle ces **2 tableaux doivent être déclarés dans la fonction `main()`**.

1. Ecrire le programme demandé en **codant et testant les différentes fonctions une par une**.

2. Modification du programme précédent :

- les fonctions AcquerirBadge() et AcquerirCodeAcces() doivent être remplacées par une seule fonction : AcquerirBadgeEtCode()

Cette nouvelle fonction doit permettre de récupérer l'identifiant et le code d'accès dans la fonction appelante (c'est à dire la fonction main()).
Etant donné qu'une fonction ne peut renvoyer qu'un seul résultat, il faut trouver une astuce !