TP1

Exercice 0

Choisissez un environnement de développement C++ et utilisez un tutoriel du web (site du zéro pour Visual (Windows), code::blocks (multi OS), XCode (mac), Qt) pour apprendre à créer un projet vide pour une application console.

Créez un fichier d'entête (`.h`). Créez deux fichiers sources (`.cpp`). Incluez (avec `#include`) le fichier d'entête dans chacun des deux fichiers sources.

Dans le fichier d'entête (`.h`), définir une fonction `int g();` qui renvoie un `int` de valeur `2`.

Dans un des deux fichiers sources, définir la fonction `int main();` qui fait appelle à la fonction `g`.

Que se passe t-il quand vous compilez ? Pourquoi ? Corrigez les erreurs.

Dans les 4 exercices suivants, différentes fonctions vous sont demandées. Ecrire les prototypes des fonctions dans un fichier d'entête (`.h`) et leur définitions dans un fichier "`.cpp`". Ecrire une fonction principale dans un troisième fichier utilisant ces fonctions. Utiliser les instructions d'inclusion conditionnelles du préprocesseurs (`#if !defined(X)`, `#define X` et `#endif`). A la fin des exercices, votre compilateur ne devrait trouver ni erreur, ni warning.

Exercice 1

Réécrire le programme suivant, en ne faisant appel qu'aux nouvelles possibilités d'entrées-sorties de C++, donc en évitant les appels à `printf` et `scanf`. Utiliser une constante plutôt que l'instruction `#define` du préprocesseur.

```
#include<stdio.h>
#define PI 3.14159
void exercice1(){
     int r; double p, s;
     printf("donnez le rayon entier d'un cercle:");
     scanf("%d",&r);
     p=2*PI*r;
     s=PI*r*r;
     printf("le cercle de rayon %d ",r);
     printf("a un perimetre de %f et une surface de %f\n",p,s);
}
```

Exercice 2

Le programme suivant est correct en C. Corriger les erreurs qui sont détectées par un compilateur C++ et utiliser les nouvelles possibilités d'E/S du C++.

```
exercice2(){
     int x1=10, x2=15, y1;
     y1=f(x1);
     printf("valeur de f(x1)=%d," , y1);
     printf(" valeur de f(x2)=%d\n", f(x2));
}

f(int x){ return x*x-3; }
```

Exercice 3

Le programme suivant est correct en C. Réécrire ce programme en utilisant au plus les nouvelles possibilités du C++. Réécrire la fonction `sum` sans utiliser l'opérateur `[]`.

```
#define MAX 10
typedef struct Duree {
     unsigned int h;
     unsigned int m;
} Duree;

Duree sum(Duree* x, int n){
     int i;
     unsigned int somme_h=0, somme_m=0;
     Duree d;
     for(i=0; i<n; i++) {
          somme_h+=x[i].h;
          somme_m+=x[i].m;
     }
     somme_h+=somme_m/60;
     somme_m%=60;
     d.h=somme_h;
     d.m=somme_m;
     return d;
}

void exercice3(){
     Duree d;
     Duree tab[MAX];
     d=sum(tab,5);
}
```

Exercice 4

Soit la structure :

```
struct personne {
     char nom[30];
     unsigned int age;
};
```

a) Ecrire une fonction `raz` qui permet pour une variable de type `personne` donnée d'initialiser son champ `nom` avec une chaîne de caractères vide et son champ `age` avec la valeur `0`.
b) Ecrire une fonction `affiche_struct` qui permet d'afficher les attributs d'une personne dont l'adresse (de type `const personne*`) est passée en argument.
c) Ecrire une fonction `affiche_tab` qui permet d'afficher les attributs d'un tableau de personne dont l'adresse est passée en argument.
d) Ecrire une fonction `init_struct` qui d'initialiser une structure dont l'adresse est passée en argument avec une chaîne de caractères et un entier passés en arguments.
e) Ecrire une fonction `copy_struct` qui permet de copier les différents champs d'une variable de type `personne` donnée dans une autre variable de type `personne`.
f) Ecrire une fonction `copy_tab` qui permet de copier un tableau de variables de type `personne` dans un autre tableau du même type.
g) Ecrire une fonction `exercice4` qui utilise l'ensemble de ces fonctions.