Fiche n° 3 de TP

Fonctions

Objectifs : écriture et utilisation de fonctions.

Prérequis : syntaxe des fonctions ; passage des paramètres.

Travail minimum : exercices 1 à 3.

Exercice 1

Tapez sans y apporter la moindre modification le programme suivant dans le fichier `eval-poly.c`.

eval-poly.c

```c
/* eval-poly
 * 6 Septembre 2024
 * Équipe pédagogique BPI
 * Mystère
 */

#include <stdio.h>
#include <stdlib.h>
#include <assert.h>

/* Déclarations des fonctions */

/* eval_poly:
 * mystère
 * Entrée: un réel x de type double
 * Sortie: une valeur de type double
 * AE: Aucune
 * AS: eval_poly ==???
 */
double eval_poly(double x);

/* Fonction principale */
int main(void) {
  printf("Entrez une valeur réelle : ");
  double v;
  assert(scanf("%lf", &v) == 1);

  double r = eval_poly(v);
  printf("La valeur du polynome pour %lf est %lf.\n", v, r);

  return EXIT_SUCCESS;
}

/* Définitions des fonctions */

double eval_poly(double x) {
  double resultat;

  resultat = 3 * x * x - 4 * x + 1;

  return resultat;
}
```

eval-poly.c

Exécutez-le avec les valeurs 1 puis 2.
Que fait la fonction `eval_poly` ?
Complétez les commentaires dans l'entête du programme et de la fonction `eval-poly.c`

Exercice 2

1) Écrivez une fonction qui prend en paramètre deux réels et qui renvoie leur moyenne.

2) Écrivez ensuite un programme qui utilise cette fonction pour déterminer et afficher la moyenne de quatre valeurs réelles données en entrée.

Exercice 3

1) Écrivez une fonction `to_sec` qui prend en paramètre trois entiers `h`, `m` et `s` représentant une valeur horaire exprimée sous la forme « heure minute seconde » et qui retourne cette valeur horaire exprimée en secondes. Vous pourrez vous inspirer du programme que vous avez écrit pour l'exercice 2 de la fiche de TP n° 2. N'oubliez pas de préciser quelles sont les assertions d'entrées. Si vous avez le temps, utilisez la commande `assert` afin de sortir du programme si les assertions d'entrée ne sont pas satisfaites.

2) En déduire un programme `time_to_sec_fun` qui lit sur l'entrée une valeur horaire exprimée sous la forme « heure:minute:seconde » et qui affiche cette valeur exprimée en secondes :

```
$ ./time_to_sec_fun
12:34:56
45296
```

3) Écrivez une fonction `to_time` qui convertit une durée exprimée en seconde en une valeur horaire exprimée sous la forme « heure minute seconde ». Indice : cette fonction, qui ne retournera rien, pourra accepter quatre paramètres, un passé par valeur, les trois autres par adresse.

4) En déduire un programme `sec_to_time_fun` qui lit sur l'entrée une durée exprimée en seconde et qui affiche cette valeur sous la forme « heure:minute:seconde » :

```
$ ./sec_to_time_fun
45296
12:34:56
```

Exercice 4

Le but de cet exercice consiste à écrire une bibliothèque en `C`.
Cette bibliothèque contiendra les fonctions définies à la question précédente.

1) Copiez les fichiers `test_timeio.c`, `timeio.c`, `timeio.h` dans le répertoire du TP3.

2) Recopiez les déclarations des fonctions écrites dans les deux exercices précedents dans le fichier `timeio.h` et les définition dans le fichier `timeio.h`.

3) Les fichiers `timeio.h` et `timeio.c` forment une bibliothèque de fonction destinée à manipuler des durées. Ouvrez le fichier `test_timeio.c` et examinez le. La bibliothèque est appelée par la ligne `#include "timeio.h"`. Les "" signifient que la bibliothèque se trouve dans le répertoire courant. Pour faire fonctionner ce programme, il va falloir compiler séparément la bibliothèque. Pour se faire, il faut ouvrir un terminal dans le répertoire du TP3 et écrire (sur une seule ligne)

```
gcc -std=c2x -Wall -Wconversion -Werror -Wextra -Wfatal-errors -Wpedantic
    -Wwrite-strings -O2 -c timeio.c
```

Remarquez la création du fichier objet `timeio.o`.
La compilation du fichier `test_timeio.c` se fait alors de façon classique en deux étapes, à ceci près que le fichier objet `timeio.o` est utilisé dans les directives de compilations. Il faut donc écrire :

```
gcc -std=c2x -Wall -Wconversion -Werror -Wextra -Wfatal-errors -Wpedantic
    -Wwrite-strings -O2 -c test_timeio.c
```

puis

```
gcc -o test_timeio test_timeio.o timeio.o
```

Ces trois commandes se trouvent dans le fichier `directive compilation.txt`.
Testez le programme obtenu.

4) Modifier le fichier afin de tester l'appel à la fonction `to_sec`.

5) Modifier la bibliothèque afin d'ajouter une fonction `ad_1s` prenant en paramètre trois pointeurs `*ph`, `*pm` et `*ps` vers des entiers de type `int` représentant une durée sous la forme heure : minute : seconde et qui modifie ces valeurs en ajoutant 1 à la durée.
Par exemple si la durée est `12:12:12` l'ajout de une seconde donne `12:12:13`. Il ne faut pas oublier les retenues : L'ajout de une seconde à `12:59:59` donne `13:00:00`.
Testez cette fonction dans le `main` du fichier `test_timeio.c`.