TD 2 Programmation – DUT 1 – Définition et tests de fonctions

P.Courtieu

Septembre 2017

Résumé

On reprend en partie le TP 1 en remplaçant les procédures par des fonctions. On teste systématiquement les fonctions dans une procédure dédié.

1 Rappels

La bibliothèque d'entrée-sortie : inout.h et inout.c et mettez les dans un répertoire `td2`. Pour cela vous pouvez faire les commandes suivantes (une seule fois) dans un terminal (menu pricipal : `terminal/Konsole`) :

```
mkdir td2
cd td2
wget "http://deptinfo.cnam.fr/~courtiep/inout/inout.h"
wget "http://deptinfo.cnam.fr/~courtiep/inout/inout.c"
```

Programmez et tester ces fonctions *une par une* (testez une fonction dès que vous pensez qu'elle est finie). Pour tester une fonction `f` on programme un ou plusieurs appels à cette procédure dans une procédure `test_f` de la manière vue en cours : on test le résultat obtenu par rapport au résultat attendu et on affiche un message d'erreur si ils ne sont pas égaux.

2 Fonctions retournant une valeur entière

1. `int SommeTroisInt(int x, int y, int z)` qui retourne la somme des trois entiers passés en paramètres.
2. `int max(int x, int y)` qui retourne le plus grand des deux entiers passés en paramètres (si ils sont égaux, on en retourne un).
3. `int max3(int x, int y, int z)` qui retourne le plus grand des trois entiers passés en paramètres (si 2 ou plus sont égaux, on retourne l'un des plus grand).
4. `int max4(int x, int y, int z, int t)` qui retourne le plus grand des quatre entiers passés en paramètres (si 2 ou plus sont égaux, on retourne l'un des plus grand).
5. `void PlusSommeProd(int z, int t)` qui retourne le plus grand entier entre la somme et le produit des deux arguments.

3 Fonction retournant une valeur booléenne (test)

Une fonction de test doit retourner une valeur utilisable comme condition dans un `if (condition) ...`. Elle doit donc retourner l'entier 0 pour signifier que le test est faux, et un entier non nul (1 de préférence) si le test est vrai.

Pour plus de clarté on utilisera les synonymes suivants :

```
#define BOOL int
#define TRUE 1
#define FALSE 0
```

1. `BOOL testPlusGrandEq(int x, int y)` qui teste le premier paramètre est plus grand ou égal au deuxième.
   Pour tester la fonction faites :

```
if (testPlusGrandEq(10,8)) {
  ecrireString("test testPlusGrandEq réussi: 10 >= 8");
} else {
  ecrireString("test testPlusGrandEq échoué: 10 < 8");
}

if (testPlusGrandEq(8,8)) {
  ecrireString("test testPlusGrandEq réussi: 8 >= 8");
} else {
  ecrireString("test testPlusGrandEq échoué: 8 < 8");
}
etc
```

2. `BOOL testPlusPetit(int x, int y)` qui se comporte comme `testPlusGrandEq` mais en inversant le test (premier argument strictement plus petit).
3. `BOOL ecritTrie3(int x, int y, int z)` qui teste si les trois arguments sont ordonnés par ordre croissant. On tolère les arguments consécutifs égaux.
4. `BOOL testDeuxEgaux(int x, int y, int z, int t)` qui teste si parmi les 4 arguments au moins deux sont égaux.

3.1 Menu

Écrivez un programme dans la fonction `main` qui propose les différentes fonctionnalités des procédures de la section précédente.

Le déroulement du programme doit être le suivant :

1. Affichage des opérations disponibles avec un numéro pour chaque.
2. l'utilisateur tape un entier (+ « entrée »)
3. invitation à taper le premier argument de la procédure
4. invitation à taper le deuxième argument de la procédure
5. etc
6. lancement de la procédure
7. Saut de ligne et fin de programme.

Si vous avez le temps imaginez un moyen que le programme recommence à l'étape 1 après l'étape 7 plutôt que de s'arrêter.

Remarque : si vous aviez déjà écrit ce menu au TP1, vous devriez pouvoir reprogrammer les procédures du TP1 facilement en utilisant les fonctions d'aujourd'hui et réutiliser le code du menu du TP1.