TD & TP 1: Fonctions

IUT CAEN – Département Informatique

Module M1103

Exercice 1

1. Écrivez une fonction `divise` qui prend en argument deux entiers et retourne un entier. La fonction retourne la valeur 1 lorsque le premier argument est un diviseur du second argument. Elle retourne la valeur 0 sinon.

2. En utilisant la fonction `divise`, écrivez une fonction `ecrire_division` qui prend en argument deux entiers et affiche un message indiquant si le second divise le premier. Par exemple :
   `ecrire_division`(4, 2) $\rightarrow$ 2 divise 4
   `ecrire_division`(5, 2) $\rightarrow$ 2 ne divise pas 5

3. Toujours en utilisant la fonction `divise`, écrivez maintenant une fonction `nb_div` qui renvoie le nombre de diviseurs d'un entier n passé en paramètre.

4. Écrivez une fonction `premier` qui renvoie vrai ou faux selon que l'entier n passé en paramètre est premier ou non.

5. Écrivez la fonction main permettant de tester l'ensemble des fonctions ci-dessus.

Exercice 2

Considrant le programme suivant:

```
#include <stdio.h>
int a, b, c;

int f(int x) {
     return a*x;
}

int g() {
        int c;
        c = a;
        a = b;
        b = c;
        return c;
}
int main() {
         int x,y;
         a = 2;
         b = 3;
         c = 4;
         x = f(a);
         y = g();
         printf("%d %d %d %d %d\n", x, y, a, b, c);
         return 0;
}
```

1. Quelles sont les variables globales ?
2. Quelles sont les variables locales à f ?
3. Quelles sont les variables locales à g ?
4. Quelles sont les variables locales à main ?
5. Quels sont les paramètres formels de f ?
6. Quels sont les paramètres formels de g ?
7. Quels sont les paramètres effectifs de f ?
8. Que vaut c à la fin de lappel à la fonction g ?
9. Que fait la fonction g ?
10. Qu'affiche le programme ?
11. Que retourne le programme ?

Exercice 3

Un nombre entier est parfait s'il est égal à la somme de ses diviseurs (sauf lui-même).
Ex : $6 = 1 + 2 + 3$ est parfait.

1. Écrivez une fonction `somme_div` qui retourne la somme des diviseurs d'un nombre passé en paramètre.
2. Écrivez une fonction `parfait` qui teste si un nombre passé en paramètre est parfait et qui retourne 1 s'il l'est et 0 sinon.
3. Écrivez un programme principal qui affiche tous les nombres parfaits inférieurs à une certaine limite.

Exercice 4

Deux nombres M et N sont appelés nombres_amis si la somme des diviseurs de M est égale à N et la somme des diviseurs de N est égale à M.

1. Écrivez une fonction `amis` qui retourne le nombre_amis (s'il existe) d'un nombre passé en parametre. Cette fonction utilise la fonction `somme_div` de lexercice 3.
2. Écrivez un programme principal qui affiche tous les nombres_amis inférieurs à une certaine limite.

Exercice 5

Écrire un programme qui construit et affiche le triangle de Pascal en calculant les coefficients binomiaux:

```
1
1     1
1     2     1
1     3     3     1
1     4     6     4     1
1     5     10    10    5     1
```

On n'utilisera pas de tableau, c.-à-d. il faudra calculer les coefficients d'après la formule: $C_p^q = \frac{p!}{q!(p-q)!}$

Exercice 6 (À faire uniquement en TP)

1. Écrivez une fonction `tasse(int n, int tab[])` qui efface toutes les occurrences du chiffre 0 dans le tableau `tab` et tasse les éléments restants (on remplira les cases vides en fin de tableau par des zéros).

2. Écrivez une fonction `inverse_tab(int n,int tab[])` qui range les éléments du tableau `tab` dans l'ordre inverse sans utiliser de tableau intermdiaire.

3. Écrivez une fonction `fusion(int n, int tabA[], int m, int tabB[], int fus[])` qui prend en argument les tableaux `tabA` (de taille n) et `tabB` (de taille m) triés par ordre croissant et remplit le tableau `fus` (de taille $n + m$) avec les éléments de `tabA` et `tabB` triés par ordre croissant.

4. Écrivez une fonction `insere(int n, int tab[], int val)` qui prend en argument un tableau `tab` de taille n, trié par ordre croissant dont la valeur de la dernière case est indéterminée et qui y insére l'entier `val` de manière à ce que le tableau reste trié.