Fonctions

Bruno Bouzy

1er septembre 2017

Ce document est un sujet de TD TP pour les étudiants de L1 S1 de l'UFR math-info. Il rassemble des exercices de C sur les fonctions. Les exercices 1 à 5 sont du même type: pour un ensemble de nombres, appeler une fonction testant une propriété. On peut en choisir un ou deux et ne pas les faire tous. L'exercice 6 est important: utilisation et compréhension du mécanisme de passage de paramètre en sortie d'une fonction avec une adresse au lieu d'une valeur. Les exercices 7, 8, 9 sont plutôt à faire en TP.

Exercice 1

Re-écrire le programme `testcarre.c` en utilisant une fonction `carre`.

```
// testcarre.c
#include <stdio.h>
#include <math.h>
int main() {
  int n;
  printf("n ? "); scanf("%d", &n);
  if (sqrt(n)==(int)sqrt(n)) printf("%d carre.\n", n);
  else printf("%d pas carre.\n", n);
  return (0);
}
```

Exercice 2

Un nombre entier est parfait s'il est égal à la somme de ses diviseurs. Ecrire un programme `parfaits.c` affichant tous les nombres parfaits inférieurs à `10000`. On écrira une fonction, appelée `parfait`, prenant un nombre entier, et retournant 1 si le nombre est parfait, 0 sinon.

Exercice 3

Ecrire un programme appelé `unites.c` affichant le chiffre des dizaines et le chiffre des unités des nombre allant de 0 à 20. On utilisera une fonction `unite` et une fonction `dizaine`.

Exercice 4

Ecrire un programme `premiers.c` affichant les nombres premiers inférieurs à 100. On utilisera une fonction `premier` prenant un nombre entier et retournant 1 si le nombre est premier, 0 sinon.

Exercice 5

Ecrire un programme `racine.c` donnant les racines `x1` et `x2` d'une équation du second degré. On utilisera une fonction `racine` retournant `1` s'il existe au moins une racine et `0` sinon. Dans le cas où il existe au moins une racine, la fonction donnera les valeurs de ces racines en sortie de la fonction.

Exercice 6

Ecrire le programme `echange.c` du cours demandant 2 valeurs, x et y, tapées au clavier et appelant une fonction échangeant les valeurs de x et de y, passés en paramètres de la fonction. On programmera cette fonction d'échange de deux manières: avec un passage de paramètres par valeurs, puis avec un passage de paramètres par adresse. On mettra en évidence la différence entre les deux méthodes.

Exercice 7

Ecrire un programme `SOMMECUBE` affichant tous les nombres inférieurs à `MAXINT` égaux à la somme des cubes de leurs chiffres. On utilisera une fonction `UNITE`, une fonction `CHIFFRE`, une fonction `CUBE`.

Exercice 8

Ecrire un programme `operations.c` qui affiche un menu demandant à l'utilisateur de taper `A` (Addition), `S` (Soustraction), `M` (Multiplication) ou `D` (Division), qui saisit la réponse tapée au clavier (`A`, `S`, `M` ou `D`), puis qui demande deux nombres entiers tapés au clavier, et qui affiche le résultat de l'opération.

On écrira une fonction pour chacun des quatre opérateurs. On mettra ce programme dans une boucle en rajoutant la possibilité de taper `Q` pour quitter le programme.

Exercice 9

Ecrire un programme `factorielle.c` qui permette de calculer `N!` (`N!=1.2.3...(N-1).N`). On utilisera deux méthodes: récursive et itérative. Pour chaque méthode, on utilisera une fonction appelée par le programme principal. On prendra un type `int` et on vérifiera que `N < 13` avant d'appeler la fonction. Pourquoi?