Programmation Imperative

L1 2021-2022

Travaux Pratiques 4
(fonctions)

Exercice 1. *Moyenne*

Ecrire un programme principal qui appeles les deux fonctions et affiche leur moyenne. Vous devez écrire une fonction `average()` qui calcule la moyenne de quatre nombres passés en paramètres, mais avant de le faire, définissez les fonctions suivantes :

Créer une fonction `positive()` qui demande à l'utilisateur d'entrer un nombre entier positif. Si ce n'est pas positif, il doit entrer un autre nombre.

Rediger enplus une fonction $\texttt{is}_{p}ositivequirenvoietruesipositifsinonfalse$.

Exercice 2. *Impots*

Le montant de l'impôt 2022 sur les revenus de 2021 est calculé sur la base de tranches correspondant à des taux d'imposition (de 0 à 45 %). Pour une part de quotient familial :

- Jusqu'à 10225 € : taux d'imposition de 0 %
- De 10226 € à 26070 € : taux d'imposition de 11 %
- De 26071 € à 74545 € : taux d'imposition de 30 %
- De 74546 € à 160336 € : taux d'imposition de 41 %
- Plus de 160336 € : taux d'imposition de 45 %

Exemple d'un célibataire :

Un célibataire dont le revenu annuel net imposable est de 30000 €, le calcul de son impôt est le suivant :

- Jusqu'à 10225 € (tranche 1) : 0 €
- De 10 226 € à 26070 € (tranche 2) : 1742,95 €
- De 25 711 € à 30000 € (tranche 3) : 1179 €

Montant total de l'impôt : 2921,95 €, soit 9,74 % de son revenu net imposable.

Ecrire une fonction `float taxes(float income)` qui calcule le montant d'impôt en fonction du revenu annuel.

Attention : Toute fonction intermediaire pertinante est la bienvenue !

Exercice 3. *Papier, caillou, ciseaux*

On veut écrire un programme permettant de jouer au jeu « papier, caillou, ciseau ». On encodera caillou par 0, papier par 1 et ciseaux par 2. Chaque joueur propose son pari en tapant 0, 1 ou 2. Pour déterminer lequel des deux joueurs a gagné, on utilise l'algorithme suivant : Soit J1 le pari du joueur 1 et J2 celui du joueur 2.

- Si J1 et J2 sont identiques le match est nul,
- J1 gagne si J1 = (J2 + 1) mod 3,
- J2 gagne dans le cas restant.

Pour ceci :

- Ecrire une fonction `LireInf2()` qui effectue la saisie contrôlée d'un entier entre 0 et 2.
- Ecrire une fonction `arbitre()` qui recoit les paris des deux joueurs et qui renvoie 0 si le match et nul, 1 si le joueur 1 a gagné et 2 si c'est le joueur 2.
- Ecrire un programme qui arbitre 10 parties et qui affiche le score.

Recursion :

Dans les langages de programmation, si un programme vous permet d'appeler une fonction à l'intérieur de la même fonction, on parle alors d'un **appel récursif** de la fonction. Vous devez faire attention à définir une **condition d'arrêt** de la fonction, sinon elle entrera dans une boucle infinie.

Exercice 4. *Fibonacci*

Les nombres de Fibonacci sont les nombres de la séquence :

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, \ldots$$

La suite F_n des nombres de Fibonacci est définie par la relation de suivante :

$$F_n = F_{n-1} + F_{n-2}$$

avec des valeurs de départ : $F_0 = 0$ et $F_1 = 1$.

Écrire une fonction `int fib(int n)` qui renvoie F_n. Plus précisément, vous devez suivre l'algorithme suivant :

- Si n = 0, alors `fib()` doit retourner 0.
- Sinon si n = 1, alors `fib()` doit retourner 1.
- Sinon, pour n > 1, elle doit renvoyer $F_{n-1} + F_{n-2}$.

Exemple :

```
fibo(6)
»F6 = 8
```

Exercice 5. *Somme*

Ecrivez une fonction récursive `sum()` qui prend comme paramètre un entier n et retourne la somme des n premiers entiers positifs.

Exemple :

```
sum(7)
»sum = 28;
```

Ecrivez maintenant une fonction récursive `sum_pair()` qui, étant donné un nombre entier positif n, renvoie la somme des nombres de paires qui sont plus petits ou égaux que n.

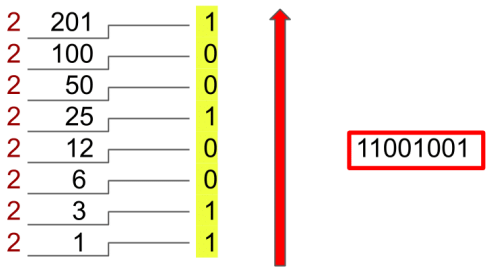
Exemple :

```
sum_pair(7)
»sum_pair = 6;
```

Exercice 6. *décimaux-binaires*

Ecrire une fonction récursive qui convertit les nombres décimaux en nombres binaires. L'image suivante est un rappel de l'algorithme que vous avez appris dans le cours d'informatique fondamentale.

Exemple :




```
dec_bin(201)
»The Binary value of the decimal number 201 is : 11001001
```

Exercice 7. *Passage par adresse et passage par référence*

— Que fait le programme suivant ?

— Quelle fonction a passage de paramètre par valeur et quelle a passage de paramètre par adresse ?

— Quelle est la sortie ? Veuillez expliquer pourquoi.

```c
#include <stdio.h>

void f1(int a, int b){
    int c;
    c = a;
    a = b;
    b = c;
}

void f2(int *a, int *b){
    int c;
    c = *a;
    *a = *b;
    *b = c;
}

int main(void) {
    int a = 4;
    int b = 5;
    int c = 6;
    f1(a,b);
    f2(&b,&c);
    printf("%d", a - c + b);
    return 0;
}
```