Initiation à la Programmation en C (L1 CPEI)*

TP 4

13/02/2019

§1. Fonctions

Exercice 1 (Afficher ou renvoyer?)**.** Dans les cas suivants, doit-on utiliser `printf` ou `return` ?
(1) Arrêter la fonction et donner son résultat.
(2) Afficher une information dans le terminal.
(3) À la dernière ligne de la fonction `int main()`.

Exercice 2 (Vrai ou faux ?)**.** (1) Une fonction `int f(float a)` *doit* contenir une instruction de la forme `return a;`, où `a` est de type `float`.
(2) Une fonction `int f(float a)` *doit* renvoyer une valeur entière.
(3) Une fonction `int f(float a)` *doit* afficher une valeur entière.
(4) Une fonction `void f(float a)` *doit* afficher une valeur entière.
(5) Une fonction `void f(float a)` *peut* afficher une valeur entière.
(6) Une fonction `float f(float a)` *peut* afficher une valeur entière.
(7) Une fonction `float f(float a)` *peut* afficher plusieurs valeurs entières.

§2. Récursion et itération

Exercice 3. Écrivez une fonction `f` qui prend en argument un entier n et renvoie la somme des n premiers entiers impairs. Par exemple, `f(3)` renvoie $1+3+5 = 9$. Testez cette fonction que quelques entiers. Que remarquez-vous ? Proposez un algorithme plus efficace pour `f`.

*Cours donné par prof. Roberto Amadio. Moniteur 2019 : Cédric Ho Thanh. TPs/TDs basés sur ceux des précédents moniteurs : Florien Bourse (2017), Antoine Dallon (2018). Autres contributeurs : Juliusz Chroboczek, Gabriel Radanne.

Exercice 4 (PGCD)**.** On se propose d'implémenter et de tester deux versions de la fonction `pgcd(a, b)` qui renvoie le plus grand commun diviseur de deux entiers a et b :

- *itérativement*, en essayant tous les entiers de 1 jusqu'à $\min(a, b)$, et en ne gardant que le dernier commun diviseur trouvé ;
- *récursivement*, en utilisant la relation

$$\text{pgcd}(a, b) = \text{pgcd}(b, a) = \text{pgcd}(a, b\%a).$$

Attention ! à l'ordre des arguments afin de ne pas tomber dans une boucle infinie !

D'après vous, laquelle de ces deux méthodes est la plus efficace ?

On rappelle que la suite de Fibonacci (f_n) est définie par la récurrence suivante :

$$\begin{aligned} f_0 &= 1, \\ f_1 &= 1, \\ f_n &= f_{n-2} + f_{n-1} \qquad \text{pour } n \geq 2. \end{aligned}$$

Exercice 5 (Fibonacci par récursion)**.** Nous nous intéressons ici à la méthode la plus simple pour calculer la suite de Fibonacci, et aussi la moins efficace, la *récursion*.

(1) Proposer une fonction `int fib(int n)` qui calcule le n-ième terme de la suite de Fibonacci en utilisant la définition ci-dessus.
(2) Votre algorithme termine-t-il ? Pourquoi[a] ?
(3) Combien d'appels à la fonction `fib` ont lieu lorsque l'on veut calculer `fib(5)` ?

[a] Ceci est une question qu'il faut *toujours* se poser quand on écrit un algorithme récursif.

Exercice 6 (Fibonacci par itération)**.** Comme la méthode récursive est très, *très* lente, nous allons utiliser une mémoire de 3 variables, contenant les valeurs de f_n, f_{n-1}, et f_{n-2}, quand $n \geq 2$. Aussi, on n'aura pas besoin de les recalculer quand on en aura besoin. Écrire un algorithme utilisant une boucle `for`, et qui affiche la valeur de f_n.