TP1 – Les fonctions

Projet de programmation M1

30 Septembre 2014

Exercice 1. [Copie] Exécutez le programme suivant :

```
#include <stdio.h>

void f(int y) {
  y = 0;
}

int main() {
  int x = 1;
  f(x);
  printf("%d", x);
  return 0;
}
```

1. Quelle est la valeur de x à la fin du programme ? Pourquoi ? Si vous trouvez cela choquant, appelez `f` sur `3*x`.
2. Remplacez `x` par `y` dans la fonction `main`. Qu'affiche ce programme ? Si vous trouvez cela choquant, dites-vous que c'est la même notion de variables libres/liées qu'en mathématiques.

Exercice 2. [La machine à café] En C, on peut lire une entrée clavier avec la fonction `scanf` et la stocker dans une variable. Pour lire un entier, on peut par exemple faire :

```
    scanf("%d", &x); // x est une variable
```

1. Écrire une fonction `somme` qui demande à l'utilisateur un entier jusqu'à ce qu'il entre l'entier 0. La fonction retourne alors la somme de tous les entiers entrés jusque-là.
2. Écrire une fonction `machine_a_cafe` qui prend en argument un entier prix. La fonction demande ensuite à l'utilisateur de rentrer des pièces (des entiers) jusqu'à ce que la valeur prix soit dépassée. La fonction retourne finalement la monnaie due.

Exercice 3. [Le digicode] On rappelle qu'en C, si `a` et `b` sont entiers, on a `a/b` et `a%b` qui valent respectivement le quotient et le reste de la division euclidienne de `a` par `b`.

1. Écrivez une fonction `unite` qui étant donné un entier, renvoie le chiffre de ses unités (en base 10).
2. Écrivez une fonction `digicode_idiot` qui prend un entier `code` en argument. Cet entier, écrit en base 10, représente un code si on le lit de droite à gauche. Par exemple 1664 représente le code $4, 6, 6, 1$. Votre fonction doit demander successivement les chiffres du code à l'utilisateur. Dès que celui-ci se trompe de chiffre, la fonction doit afficher une erreur et inviter l'utilisateur à recommencer depuis le début.
3. Si vous ne connaissez pas la valeur de `code` dans la fonction `digicode_idiot`, en combien de coups (en fonction de la longueur du code) pouvez-vous la trouver ? Expliquez alors le nom de cette fonction.
4. Écrire une fonction `digicode` qui prend un entier `code` en argument dont tous les chiffres sont distincts et accepte dès que le code apparaît dans la suite de chiffres entrés par l'utilisateur. Votre fonction marche-t-elle encore si `code = 421321` (donc le digicode est 1,2,3,1,2,4) ? Pourquoi ?
5. (difficile, à faire quand vous avez fini le reste du TP. Appelez-moi avant) Écrire une fonction `digicode` qui marche dans tous les cas.

Exercice 4. [Récurrez !] Une fonction est dite récursive si elle s'appelle elle-même. Vous pouvez rapprocher cela des suites définies par récurrence. Bien sûr, ces fonctions ont toujours des "cas de base" pour lesquels elles ne s'appellent pas elles-même, sinon le calcul ne se terminerait jamais. Pour résoudre des problèmes avec une fonction récursive, on fait souvent un raisonnement proche de celui qu'on fait en raisonnant par récurrence en mathématiques : si je sais faire ça pour tous les cas plus petits, comment puis-je résoudre ce cas-là ?

1. En remarquant que $0! = 1$ et $n! = n \times (n-1)!$ pour $n \geq 1$, écrivez une fonction `factorielle` qui calcule $n!$ sans utiliser de boucles. Faites de même pour `pgcd`.
2. Écrire `fibonacci` en récursif aussi et exécutez votre fonction pour $u_0 = u_1 = 1$ et $n = 40$. Que remarquez-vous par rapport à la version impérative ? À votre avis, pourquoi ? Écrivez `fibonacci` avec un seul appel récursif (indice : vous pouvez le faire en 3 lignes, pensez aux autres arguments !). Qu'en est-il des performances de cette nouvelle version ?
3. Écrire une fonction récursive qui étant donné n et b, écrit à l'écran $a_0 \dots a_k$ où $n = (a_k \dots a_0)_b$ est l'écriture de n en base b, c'est-à-dire $a_k \neq 0$, $0 \leq a_i < b$ et $\sum_{i=0}^{k} a_i b^i = n$.
4. Les tours d'Hanoi. On se propose de jouer au jeu suivant : on a 3 piquets et sur le piquet le plus à gauche, on a empilé n anneaux de diamètres différents, par ordre croissant. On a le droit de déplacer n'importe quel anneau qui est en haut de la pile d'un piquet vers un autre piquet, mais on ne doit jamais poser un anneau sur un anneau plus petit. Le but du jeu est de déplacer les anneaux du piquet de gauche au piquet de droite. Écrivez une fonction qui, étant donné n, écrit une suite d'opérations à effectuer pour résoudre le problème. Combien d'opérations votre solution nécessite-t-elle ? Est-ce optimal ?