Programmation 2

L1 2011-2012

Travaux Pratiques 2 - Récursivité

Cette séance est consacré aux fonctions récursives. Une fonction récursive est une fonction qui contient dans sa définition un appel à elle-même. Afin de ne pas continuer à s'appeler éternellement, la fonction doit contenir une condition d'arrêt.

Exercice 1.

1. Écrire et tester la fonction suivante.

```
void recursiveCrois(int a)
{
        if(a==0)
                return;
        else
        {
                recursiveCrois(a-1);
                printf("%d ",a);
        }
}
```

2. Écrire la fonction `void recursiveDecr(int a)` qui affiche les entiers dans l'ordre décroissant de `a` à `1`. La fonction doit utiliser les mêmes instructions que `recursiveCrois`.
3. Écrire la fonction `void recursiveCpt(int a,int* cpt)` qui affiche les entiers dans l'ordre croissant de `1` à `a`. La fonction doit ajouter à l'entier pointé par `cpt` le nombre d'appel récursif lors de son exécution.

Exercice 2.

Écrire une fonction récursive `factorielle` qui calcule la factorielle d'un nombre. On rappelle que :

- $0! = 1$
- $n! = n.(n-1)!$

Exercice 3. (Plus Grand Commun Diviseur)

Écrire une fonction récursive `int pgcd(int a, int b)` qui renvoie le PGCD de deux entiers. On suppose que l'un des deux entiers est non nul.

On rappelle que si $b = 0$, PGCD$(a, b) = a$. De plus si $a = qb + r$ où r est le reste de la division euclidienne de a par b alors PGCD$(a, b) =$ PGCD(b, r).

Exercice 4.

1. Écrire la fonction récursive `int PuissanceSimple(int base,int exposant)` qui calcule la puissance d'un nombre en utilisant la formule :
   - $a^0 = 1$
   - $a^n = a.a^{n-1}$

   On suppose que `exposant` est positif.
2. Écrire la fonction récursive `int PuissanceMalin(int base,int exposant)` qui calcule la puissance d'un nombre mais en utilisant les relations suivantes :
   - $a^0 = 1$
   - $a^n = (a^{\frac{n}{2}})^2$ si n est pair
   - $a^n = a.(a^{\frac{n-1}{2}})^2$ si n est impair
3. Modifier les deux précédentes fonctions en deux nouvelles fonctions `PuissanceSimpleCpt` et `PuissanceMalinCpt` pour compter leur nombre d'appels recursifs.
4. Donnez les complexités de ces deux fonctions.

Exercice 5. (Nombres de Fibonacci)

La suite de Fibonacci $(F_i)_{i\in\mathbb{N}}$ est définie par :
- $F_0 = 0$
- $F_1 = 1$
- $F_n = F_{n-1} + F_{n-2}$ si $n > 1$.

1. Écrire une fonction C récursive, `Fibo`, qui calcule F_n .
2. Vérifier avec votre programme que le nombre d'appels à la fonction `Fibo` pour calculer F_n est exactement égal à $2F_{n+1} - 1$.
3. Écrire une fonction C itérative qui calcule le n-ième nombre de Fibonacci. Combien y a-t-il de passage dans la boucle ?
4. Que vaut PGCD$(F_{k+1}$, F_k) ? Combien d'appels à la fonction sont nécessaires pour calculer PGCD$(F_{k+1}$, F_k) ?

Exercice 6.

Écrire une fonction récursive `EcrireBase` qui prend en paramètre deux entiers `n` et `base` et qui affiche l'écriture de `n` dans la base `base`. On suppose que `base` est un entier positif inférieur à $36 = (10 + 26)$. On pourra utiliser toute les lettres de l'alphabet en majuscule. Ex : les nombres $(35)_{\text{dix}}$ et $(36)_{\text{dix}}$ s'écrivent respectivement Z et 10 en base 36.