PROGRAMMATION C
TD5

LICENCE MATHS-INFO
13 FÉVRIER 2012

Exercice 1 - allocation dynamique

Un polynôme $P(X) = \sum_{k=0}^{K} a_k X^k$ d'ordre K sera stocké dans un tableau p tel que, pour $0 \leq k \leq K$, $p[k] = a_k$. Ecrivez une fonction

```
double * derive_polynome(double * p, int K)
```

qui prend un polynôme et son degré en arguments et renvoie sa dérivée sous la forme d'un nouveau polynôme.

Exercice 2 - allocation dynamique et désallocation

Dans la continuité de l'exercice précédent, écrivez une fonction

```
double * derive_n_polynome(double * p, int K, int n)
```

calculant la dérivée n-ième d'un polynôme, en veillant à la bonne gestion de la mémoire.

Exercice 3 - amélioration avec un type polynôme

Nous adoptons une autre approche en introduisant le type `polynome` :

```
typedef struct {double * a; int K;} polynome;
```

Créez les fonctions suivantes, en veillant à une bonne gestion de la mémoire :

- `polynome * creePolynome(int K)` : crée un polynôme d'ordre K initialisé à 0.
- `void supprimePolynome(polynome * p)` : supprime un polynôme en libérant l'espace mémoire correspondant.
- `void modifieCoefficientPolynome(polynome * p, int k, double c)` : donne la valeur `c` au coefficient d'ordre `k`.
- `double valeurPolynome(polynome * p, double x)` : calcule la valeur de `p` en `x`.
- `polynome * derive_polynome(polynome * p)` : dérive un polynôme.
- `polynome * derive_n_polynome(polynome * p, int n)` : dérivée n-ième.

Exemple de test :

```
polynome p,q;
p = creePolynome(3);
modifieCoefficientPolynome(p,0,-1.5);
modifieCoefficientPolynome(p,3,2.3);
q = derive_n_polynome(p,2);
printf("p''(0)=%g\n",valeurPolynome(q,0));
supprimePolynome(p);
supprimePolynome(q);
```

Exercice 4

Que fait (finalement) le programme ci-dessous (édition 2001 du International Obfuscated C Code Contest — www.ioccc.org) ?

```
m(char*s,char*t) {
 return *t-42?*s?63==*t|*s==*t&&m(s+1,t+1):!*t:m(s,t+1)||*s&&m(s+1,t);
}
main(int c,char **v) { return!m(v[1],v[2]); }
```

Exercice 5 : zéro par dichotomie

Un zéro d'une fonction f est une valeur z_0 telle que $f(z_0) = 0$. Une valeur approchée à ϵ près de z_0 est une valeur z telle que $|z - z_0| < \epsilon$. Etant donnés un intervalle $[a_0, b_0]$ et une fonction f continue sur $[a_0, b_0]$ telle que $f(a_0)f(b_0) < 0$, écrivez une fonction qui calcule une valeur approchée à ϵ près d'un zéro de f. Vous coderez cette méthode sous la forme d'une fonction dont les arguments sont a_0, b_0, ϵ et f.

La méthode proposée, dite par dichotomie, consiste à maintenir un intervalle (initialement $[a_0, b_0]$ dans lequel la fonction change de signe, et dont la longueur diminue de moitié à chaque itération, jusqu'à obtenir un intervalle dont n'importe quel point est une valeur approchée à ϵ près de z_0.