

Examen de Info C

Durée : 1h
Aucun document autorisé
Mobiles interdits

- 1. Écrivez la procédure `identité` qui initialise une matrice carrée $n \times n$ passée en paramètre à la matrice identité (i.e. des 1 sur la première diagonale, et des 0 ailleurs). Important : votre procédure **ne doit pas utiliser** de fonctions de la bibliothèque C, ni d'énoncés ou expressions conditionnels (i.e. `if` ou `?:`), et doit faire exactement n^2 affectations.

```
/*
 * antécédent : n>0
 * conséquent : m est la matrice identité
 */
void identité(const int n, int m[][n]) {
    // algo : on parcourt la demi-matrice inférieure
    // et on met les 0 dans le 2 demi-matrices
    // puis le 1 sur la diagonale
    // écrire le 1er 1
    m[0][0] = 1;
    for (int i=1; i<n; i++) {
        // mettre les 0 de part et d'autre de la diagonale
        for (int j=0; j<i; j++)
            m[i][j] = m[j][i] = 0;
        // mettre 1 sur la diagonale à la ligne i
        m[i][i] = 1;
    }
}
```

Maintenant, nous considérerons des **polynômes**, combinaison linéaire des produits de puissances à une inconnue. Un polynôme p de degré n (positif) à une inconnue x possède la forme suivante :

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_i x^i + \dots + a_1 x + a_0$$

Les coefficients a_i d'un polynôme p seront mémorisés dans un tableau de réels **double** tel que `p[i]` est le coefficient du degré i . Le degré maximum d'un polynôme est donné par la constante `DEGRÉMAX`.

- 2. À l'aide de **typedef**, déclarez le type `polynôme` pour représenter un polynôme comme défini ci-dessus.

```
typedef double polynôme[DEGRÉMAX+1];
```

- 3. Écrivez la procédure `init0` qui initialise à 0 un polynôme p passé en paramètre.

```
/*
 * Rôle : initialise le polynôme p à 0
 */
void init0(polynôme p) {
    for (int i=0; i<=DEGRÉMAX; i++)
        p[i] = 0.0;
}
```

- 4. Écrivez la fonction `degré` qui renvoie le degré (le plus élevé $\leq$ `DEGRÉMAX`) d'un polynôme p passé en paramètre. Par exemple, le degré du polynôme $p = 1.2x^7 + x + 3.1$ est égale à 7.

```
/*
 * Antécédent : p polynôme initialisé (valide)
 * Rôle : renvoie le degré du polynôme p
 */
int degré(const polynôme p) {
    for (int i = DEGRÉMAX; i>=0; i--)
        if (p[i]!=0.0) return i;
    // p = polynôme de degré 0
    return 0;
}
```

- 5. Écrivez la procédure `add` qui calcule la somme $p_3 = p_1 + p_2$. Les polynômes p_1 , p_2 et p_3 sont les paramètres de la procédure. Vous devrez nécessairement utiliser la fonction `degré` précédente.

```
/*
 * Antécédent : p1 et p2 deux polynômes initialisés (valides)
 * Conséquent : p3 = p1 + p2
 */
void add(const polynôme p1, const polynôme p2, polynôme p3) {
    // déterminer le degré du polynôme le plus élevé
    int d1 = degré(p1);
    int d2 = degré(p2);
    int dmax = d1>d2 ? d1 : d2;
    // mettre à 0 le coefficient des degrés inexistantes
    for (int i=DEGRÉMAX; i>dmax; i--) p3[i]=0.0;
    // calculer la somme
    for (int i=dmax; i>=0; i--) p3[i] = p1[i] + p2[i];
}
```

- 6. On veut évaluer un polynôme p pour une valeur de x . Attention : on ne veut pas calculer les évaluations à la puissance successives qui sont coûteuses. Pour cela, vous utiliserez le schéma de HORNER. Le polynôme $p(x) = p[0] \times x^0 + p[1] \times x^1 + p[2] \times x^2 + \dots + p[n] \times x^n$ s'interprète

$p(x) = (((\dots(p[n] \times x + p[n-1]) \times x + \dots + p[1]) \times x + p[0])$. Par exemple, le polynôme $3x^3 - 2x^2 + 4x + 1$ s'interprète $((3x - 2)x + 4)x + 1$.

Écrivez la fonction `éval` qui évalue un polynôme `p` selon le schéma de HORNER, donc sans évaluation à la puissance. Le polynôme `p` et la valeur `x` sont les deux paramètres de la fonction. Vous devrez nécessairement utiliser la fonction `degré` écrite précédemment.

```
/*
 * Rôle : renvoie l'évaluation de p(x)
 * Note : Algorithme de Horner
 */
double eval(const polynôme p, const double x) {
    int n = degré(p);
    double valeur = p[n];
    // valeur = \sum_{k=n}^n p[k]x^{k-n} = p[n]
    for (int i=n-1; i>=0; i--)
        // valeur = \sum_{k=n}^i p[k]x^{k-i}
        valeur = valeur*x + p[i];
    // valeur = \sum_{k=n}^0 p[k]x^k
    return valeur;
}
```

.....

- 7. Écrivez le programme qui :
- déclare et initialise deux polynômes `p1` et `p2` respectivement aux valeurs $3x^3 - 2x^2 + 4x + 1$ et $x^5 + 2x^2$;
 - calcule leur somme dans un troisième polynôme `p3`;
 - écrit sur la sortie standard l'évaluation de `p3` avec comme valeur d'inconnue 2.

```
int main(void) {
    // déclaration et initialisation des polynômes
    polynôme p1 = {1, 4, 2, 3}; // 3x^3 - 2x^2 + 4x + 1
    polynôme p2 = {0, 0, 2, 0, 0, 1}; // x^5 + 2x^2
    // calculer leur somme
    polynôme p3;
    add(p1, p2, p3); // p3 = x^5 + 3x^3 + 4x + 1
    //
    // écrire sur la S/S l'évaluation de p3 pour x=2
    printf("éval(p3) = %f\n", eval(p3, 2)); // 65
    //
    return EXIT_SUCCESS;
}
```

.....