

	ANNEE UNIVERSITAIRE 2017-2018 UE : 4TSI402U Epreuve : 4TSI402EXI Informatique modulaire Date : Lundi 7 mai 2018 Heure : 9h00 Durée : 1h30 Documents autorisés Épreuve de M. Marc
---	---

Les fichiers qui se trouvent en annexe sont ce qui reste d'un code d'un programme de manipulation de polynômes destiné à calculer des polynômes d'interpolation. Malheureusement, au cours d'une fausse manœuvre, j'ai effacé par mégarde la dernière version. Il ne me reste plus qu'une version incomplète du code source. Pouvez-vous m'aider à compléter les parties manquantes et à corriger et terminer le programme ?

Conseil : il n'est pas nécessaire de lire tout le code. Commencez par le test, le programme de test et les fichiers d'en-tête.

1 Analyse, optimisation et correction d'un programme de calcul sur les polynômes

Le code initial est constitué de 5 fichiers :

```
interpolation.c interpolation.h
polynome.c polynome.h
test_interpolation.c
```

Le résultat d'une exécution de la version complète de l'exécutable (avant perte de fichiers) se trouve en annexe 2.1. Les listings de certains fichiers sources (versions incomplètes) se trouvent en annexe 2.2. Les numéros de lignes ont été ajoutés a posteriori.

- 1) Le fichier `interpolation.h` a disparu. Écrire ce fichier.
 - 2) Quelle ligne de commande faut-il passer dans un terminal pour compiler les fichiers afin d'obtenir un exécutable nommé `test_interpolation` ? On pourra indiquer la compilation en une ligne.
 - 3) Si un `int` occupe 4 octets et un `double` en occupe 8, quel est l'espace mémoire occupé par un `Polynome` ?
 - 4) Dans la fonction `polynome_addition`, combien de fois est exécutée la ligne 64 de `polynome.c` à chaque appel de la fonction ? On donnera la réponse en fonction des caractéristiques de `P1` et `P2`.
 - 5) Dans la fonction `polynome_produit`, combien de fois est exécutée la ligne 102 de `polynome.c` à chaque appel de la fonction ?
 - 6) Écrire la définition de la fonction `polynome_evaluer` qui calcule la valeur d'un polynôme pour une valeur de la variable.
 - 7) Écrire la définition de la fonction `polynome_creer_depuis_tableau`.
- La ligne 10 du test (affichage de `P8`) donne visiblement un résultat faux.
- 8) Que devrait-on voir afficher à cette ligne ?
 - 9) Pourquoi l'erreur ne peut-elle pas venir de la fonction `polynome_afficher` ?
 - 10) De quelle fonction vient l'erreur ? Quelle est l'erreur ?
 - 11) Écrire la modification de la fonction permettant de corriger ce problème. On n'indiquera que le code modifié (et les numéros de lignes).
 - 12) Dans l'en-tête de la fonction à la ligne 9 de `interpolation.c`, que signifie `double x[]` ? Pourquoi les crochets `[]` sont-ils vides ?

- 13) Écrire le corps de la fonction `polynome_Lagrange` permettant le calcul d'un polynôme de Lagrange. L'en-tête de la fonction est donné dans le listing et son mode d'emploi est donné en commentaire. Cette fonction devra être indépendante de la représentation des polynômes. On rappelle que le j-ème polynôme de Lagrange associé aux N valeurs $x_1, x_2, \dots, x_N$ est le polynôme $P(x)$ de degré N-1 qui s'annule pour tout $x_i (i \neq j)$ et qui vaut 1 en x_j . Il peut se calculer comme un produit de polynômes de degré 1, suivi d'une normalisation.

$$P_j(x) = K \prod_{i \neq j} (x - x_i)$$

La fonction `polynome_Lagrange` est utilisée notamment dans la fonction `interpolation_Lagrange` (qui calcule un polynôme d'interpolation).

2 ANNEXES : codes

2.1 Résultat du test du programme complet

```

1 Affichage de polynomes et de leur degre
2
3 12.2 z^4 + 3.3 z^3 - 5.2 z^2 - 3          degre    4
4 - x^3      degre    3
5 - 1.2      degre    0
6 0          degre   -1
7 1          degre    0
8 4.2 y^4 - 3.2 y^2 - 2 y + 2.4      degre    4
9 - 8.4 y^4 + 6.4 y^2 + 4 y - 4.8      degre    4
10          degre    4
11
12 Polynome d'interpolation
13
14 - 0.5 x^3 + 3.5 x^2 - 7 x + 4
15 P1(1)=0
16 P1(2)=0
17 P1(3)=1
18 P1(4)=0
19
20 Interpolation polynomiale
21
22 P(z) = 0.35 z^3 - 3.8 z^2 + 11.15 z - 6.7
23 Xi      Yi      P(Xi)
24 1        1        1
25 2        3.2      3.2
26 3        2        2
27 4       -0.5     -0.5

```

2.2 Codes sources

```

1 /* test_interpolation.c
2  Test des polynomes et de l'interpolation polynomiale
3 */
4 #include <stdio.h>
5 #include "polynome.h"
6 #include "interpolation.h"
7
8 /* Programme de Test */
9 #define PLUS polynome_addition
10 #define MOINS polynome_soustraction
11
12

```

```

13 int main(void)
14 {
15     Polynome P1,P2,P3,P4,P5,P6,P7,P8,PInterp;
16     double X[4]={1.0,2.0,3.0,4.0};
17     double Y[4]={1.0,3.2,2.0,-0.5};
18     double tabP[5]={2.4,-2,-3.2,0,4.2};
19     int i,j;
20
21     printf("Affichage de polynomes et de leur degre\n\n");
22     P1=PLUS(monome(12.2,4),PLUS(MOINS(monome(3.3,3),monome(5.2,2)),monome(-3.0,0)));
23     P2=monome(-1.0,3);
24     P3=monome(-1.2,0);
25     P4=monome(0,3);
26     P5=monome(1,0);
27     P6=polynome_creer_depuis_tableau(tabP,4);
28     P7=polynome_produit_par_constant(-2.0,P6);
29     P8=polynome_produit_par_constant(0.0,P6);
30     polynome_afficher(P1,"z"); printf("\tdegre    %d \n",polynome_degre(P1));
31     polynome_afficher(P2,"x"); printf("\tdegre    %d \n",polynome_degre(P2));
32     polynome_afficher(P3,"z"); printf("\tdegre    %d \n",polynome_degre(P3));
33     polynome_afficher(P4,"z"); printf("\tdegre    %d \n",polynome_degre(P4));
34     polynome_afficher(P5,"x"); printf("\tdegre    %d \n",polynome_degre(P5));
35     polynome_afficher(P6,"y"); printf("\tdegre    %d \n",polynome_degre(P6));
36     polynome_afficher(P7,"y"); printf("\tdegre    %d \n",polynome_degre(P7));
37     polynome_afficher(P8,"y"); printf("\tdegre    %d \n",polynome_degre(P8));
38
39     printf("\n Polynome d'interpolation\n\n");
40     P1=polynome_Lagrange(4, X, 2);
41     polynome_afficher(P1,"x");
42     printf("\n");
43     for(j=0;j<4;j++)
44         printf("P1(%lg)=%lg\n",X[j],polynome_evaluer(P1,X[j]));
45
46
47     printf("\n Interpolation polynomiale\n\n");
48     PInterp=interpolation_Lagrange(4,X,Y);
49     printf("P(z) = ");
50     polynome_afficher(PInterp,"z"); printf("\n");
51     printf("Xi\tYi\tP(Xi)    \n");
52     for(i=0;i<4;i++)
53         printf("%lg\t%lg\t%lg\n",X[i],Y[i],polynome_evaluer(PInterp,X[i]));
54     printf("\n");
55
56
57     return 0;
58 }

```

```

1  /* interpolation.c
2  Interpolation polynomiale
3  */
4  #include <stdio.h>
5  #include <stdlib.h>
6  #include "interpolation.h"
7
8  /* (j+1)-eme polynome de Lagrange pour les valeurs x[0],x[1],...x[NbPts-1] */
9  Polynome polynome_Lagrange(int NbPts, double x[], int j)
10 {
11     /* A COMPLETER */
12 }
13
14 /* Interpolation des points (x[i],y[i]) (i=0,1,...,NbPts-1) */

```

```

15 /* par la methode des polynomes de Lagrange */
16 Polynome interpolation_Lagrange(int NbPts, double x[], double y[])
17 {
18     Polynome PolyInter;
19     int j;
20
21     PolyInter=polynome_nul();
22     for (j=0;j<NbPts;j++)
23         PolyInter=polynome_addition(PolyInter,
24                                     polynome_produit_par_constant( y[j],
25                                                                     polynome_Lagrange(NbPts,x,j)));
26
27     return PolyInter;
28 }

```

```

1 /* polynome.h
2  Gestion de polynomes et interpolation polynomiale
3 */
4 #ifndef POLYNOME_H
5 #define POLYNOME_H
6
7 #define DEGREMAX 10
8     /* type Polynome
9      - seuls les (degre+1) premiers coefficients sont significatifs
10     - coeffs[d] est le coefficient de degre d
11     - coeffs[degre] est toujours non nul
12     - pour le polynome nul, degre vaut -1
13 */
14 typedef struct {
15     int degre;
16     double coeffs[DEGREMAX+1];
17     } Polynome;
18
19 int polynome_degre(Polynome P);
20
21 Polynome polynome_nul(void);
22 Polynome monome(double coefficient, int degre);
23 Polynome polynome_creer_depuis_tableau(double coefficients[],int degre);
24
25 void polynome_afficher(Polynome poly,char nom_variable[]);
26 double polynome_evaluer(Polynome P, double x);
27
28 Polynome polynome_addition(Polynome P1, Polynome P2);
29 Polynome polynome_soustraction(Polynome P1, Polynome P2);
30
31 Polynome polynome_produit(Polynome P1, Polynome P2);
32 Polynome polynome_produit_par_constant(double a, Polynome P1);
33
34 #endif

```

```

1 /* polynome.c
2  Gestion de polynomes et interpolation polynomiale
3 */
4 #include <stdio.h>
5 #include <math.h>
6 #include "polynome.h"
7
8 int polynome_degre(Polynome P)
9 {
10     return P.degre;
11 }

```

```

12
13 Polynome polynome_nul(void)
14 {
15     Polynome poly_retour;
16     poly_retour.degre= -1;
17
18     return poly_retour;
19 }
20
21 Polynome monome(double coefficient, int degre)
22 {
23     Polynome poly_retour;
24     int d;
25     if (coefficient==0.0)
26         return polynome_nul();
27     poly_retour.degre=degre;
28     poly_retour.coeffs[degre]=coefficient;
29     for (d=degre-1;d>=0;d--)
30         poly_retour.coeffs[d]=0.0;
31     return poly_retour;
32 }
33
34 void polynome_afficher(Polynome poly, char nom_variable[])
35 {
36     int deg;
37
38     if (poly.degre==-1)
39         printf("0 ");
40
41     for (deg=poly.degre;deg>=0;deg--)
42         if (poly.coeffs[deg]!=0.0)
43             {
44                 if (poly.coeffs[deg]<0.0)
45                     printf("- ");
46                 else if (poly.degre!=deg)
47                     printf("+ ");
48                 if (deg==0 || fabs(poly.coeffs[deg])!=1.0)
49                     printf("%g ", fabs(poly.coeffs[deg]));
50                 if (deg==1)
51                     printf("%s ", nom_variable);
52                 else if (deg>1)
53                     printf("%s^%d ", nom_variable, deg);
54             }
55 }
56
57 Polynome polynome_addition(Polynome P1, Polynome P2)
58 {
59     Polynome P;
60     int deg;
61
62     for (deg=0;deg<=P1.degre || deg<=P2.degre;deg++)
63     {
64         if (deg>P1.degre)
65             P.coeffs[deg]=P2.coeffs[deg];
66         else if (deg>P2.degre)
67             P.coeffs[deg]=P1.coeffs[deg];
68         else
69             P.coeffs[deg]=
70                 P1.coeffs[deg]+P2.coeffs[deg];
71     }
72     P.degre=deg-1;

```

```

73 while (P.degree>=0 && P.coeffs[P.degree]==0)
74     P.degree --;
75
76 return P;
77 }
78
79 Polynome polynome_soustraction(Polynome P1, Polynome P2)
80 {
81     Polynome P2M;
82     P2M=polynome_produit_par_constante(-1.0,P2);
83
84     return polynome_addition(P1,P2M);
85 }
86
87 Polynome polynome_produit(Polynome P1, Polynome P2)
88 {
89     Polynome P;
90     int degreP,degreP1,degreP2;
91
92     P.degree=P1.degree+P2.degree;
93
94     for (degreP=0;degreP<=P.degree;degreP++)
95     {
96         P.coeffs[degreP]=0.0;
97     }
98
99     for (degreP1=0; degreP1<=P1.degree; degreP1++)
100         for (degreP2=0; degreP2<=P2.degree; degreP2++)
101             {
102                 P.coeffs[degreP1+degreP2] += P1.coeffs[degreP1] * P2.coeffs[degreP2];
103             }
104     return P;
105 }
106
107 Polynome polynome_produit_par_constante(double a, Polynome P1)
108 {
109     Polynome P;
110     int deg;
111
112     P.degree=P1.degree;
113     for (deg=0;deg<=P.degree;deg++)
114     {
115         P.coeffs[deg]= a* P1.coeffs[deg];
116     }
117     return P;
118 }

```

FIN