- En el següent programa veurem com llegir dades des d'un fitxer i com fer ús de funcions.
  Aquest programa és un exemple per a veure com es propaguen els errors de manera diferent depenent de l'algorisme usat. Avaluarem un polinomi de dues maneres i el resultat serà diferent.

1 Volem avaluar un polinomi qualsevol de grau 7

$$p_7(x) = a_7x^7 + a_6x^6 + a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$$

en diferents punts. Per avaluar-lo en un punt z disposem dels dos algorismes:

- directament: $p_7(z) = a_7z^7 + a_6z^6 + a_5z^5 + a_4z^4 + a_3z^3 + a_2z^2 + a_1z + a_0$
- mitjançant la regla de Horner: $p_7(z) = ((((((a_7z + a_6)z + a_5)z + a_4)z + a_3)z + a_2)z + a_1)z + a_0$.

El següent programa dóna l'avaluació de $p_7(x)$ en punts equiespaiats usant els algorismes anteriors. Primer llegeix els coeficients $a_i, i = 0, \ldots, 7$, del fitxer `coeficients.dad`, que hem escrit prèviament, i els guarda en el vector `coef[8]`.

```
/* Avaluacio d'un polinomi en m punts equiespaiats a l'interval [a,b]
 * usant dos algorismes
 */
#include<stdio.h>
#include<math.h>

float poli(float, float[]);
float horner(float, float[]);

int main(void) {
   int i, m, k;
   float a, b, x, aval1, aval2, h;
   float coef[8];
   FILE *entrada;

   entrada = fopen("coeficients.dad", "r");   /* (1)  */
   if (entrada == NULL){
      printf("Error␣en␣obrir␣el␣fitxer␣%s\n", "coeficients.dad"); /* (2) */
      return 1;
   }
   for (i = 0; i < 8; i++){
       k = fscanf(entrada, "%f", &coef[i]);
   }
   fclose(entrada);

   printf("#doneu␣a,␣b,␣m␣=␣\n");
   k = scanf("␣%f␣%f␣%d", &a, &b, &m);
   h = (b-a)/m;
   printf ("#␣%8s␣%17s␣%17s␣%17s\n", "x", "poli", "horner", "diferencia");
   for (i = 0; i <= m; i++){
       x = a+i*h;
       aval1 = poli(x, coef);
       aval2 = horner(x, coef);
       printf("%15.6e␣%15.6e␣%15.6e␣%15.6e\n", x, aval1, aval2, fabs(aval1-aval2));
   }
   return 0;
}

float poli(float z, float a[]){
      int i;
      float sum;
      sum = a[0];
      for (i = 1; i <= 7; i++){
```

```
            sum = sum+a[i]*pow(z,i);
        }
        return sum;
}

float horner(float z, float a[]){
        int i;
        float sum;
        sum = a[7];
        for (i = 6; i >= 0; i--){
            sum = sum*z+a[i];
        }
        return sum;
}
```

a) Creeu el fitxer `coeficients.dad` amb les dades del polinomi

$$p(x) = x^7 - 7x^6 + 21x^5 - 35x^4 + 35x^3 - 21x^2 + 7x - 1$$

que té l'únic zero $\alpha = 1$.

Executeu el programa per $a = 0.75$, $b = 1.25$ i $m = 100, 1000$ i 10000, redireccionant la sortida a un fitxer. Mitjançant el programa `gnuplot` dibuixeu, per a diferents rangs de y, les gràfiques de les columnes 1 contra 2 i 1 contra 3; després la de les columnes 1 contra 4.

```
plot 'fitxer.res' u 1:2, 'fitxer.res' u 1:3
plot 'fitxer.res' u 1:4
```

Comenteu els resultats.

b) Tal com hem escrit aquest programa, per poder llegir d'un fitxer de dades diferent s'hauran de canviar les línies (1) i (2) tornar a compilar, muntar i executar. L'alternativa és llegir el nom del fitxer que conté les dades en cada execució. Això es pot fer substituint la línia (1) pel bloc d'instruccions:

```
char nom_fitxer[80];
printf("# Doneu el nom del fitxer d'entrada\n");
k = scanf("%s", nom_fitxer);
entrada = fopen(nom_fitxer, "r");
```

i la línia (2) per

```
printf("Error en obrir el fitxer %s\n", nom_fitxer);
```

- Amb aquesta modificació, si executem redireccionant la sortida no veurem la frase `Doneu el nom del fitxer d'entrada`. Però el programa espera que li donem el nom del fitxer.

c) Modifiqueu el programa de forma que l'escriptura es faci en un fitxer, el nom del qual el llegireu després de llegir el nom del fitxer de lectura.

d) Modifiqueu ara el programa per treballar en precisió doble. Executeu-lo i repetiu les gràfiques.

- En els programes anteriors per al vector de coeficients la dimensió estava fixada des del principi. Si volem fer un programa en què les dimensions dels vectors puguin ser diferents en cada execució cal usar assignació dinàmica de memòria. Vegem un exemple.

2 Volem calcular el producte escalar de dos vectors de dimensió $n \in \mathbb{N}$. Si $x = (x_1, x_2, \ldots, x_n)^t$ i $y = (y_1, y_2, \ldots, y_n)^t$, llavors el producte escalar val

$$x \cdot y = \sum_{i=1}^{n} x_i y_i = x_1 y_1 + x_2 y_2 + \cdots + x_n y_n$$

El següent programa llegeix la dimensió, les components dels dos vectors i calcula el seu producte escalar.

```
/* Càlcul del producte escalar de dos vectors usant memòria dinàmica*/

#include<stdio.h>

int main(void){
  int n, i;
  float *x, *y, prod;
```

```
  printf("Doneu la dimensió dels vectors, n = \n");
  scanf("%d", &n);

  x = (float *) malloc( n*sizeof(float) );
  y = (float *) malloc( n*sizeof(float) );
  if ( x == NULL || y == NULL ){
     printf("No hi ha prou memòria");
     return 1;
  }

  printf("Doneu els %d elements del vector x \n", n);
  for (i = 0; i < n; i++){
     scanf("%f", &x[i]);
  }
  printf("Doneu els %d elements del vector y \n", n);
  for (i = 0; i < n; i++){
     scanf("%f", &y[i]);
  }

  prod = 0.f;
  for (i = 0; i < n; i++){
      prod += x[i]*y[i];
  }

  printf("El producte escalar val: %16.7e \n", prod);
  free(x);
  free(y);
  return 0;
}
```

a) Executeu-lo per a diferents valors de n.

b) Modifiqueu-lo de forma que la lectura es faci des d'un fitxer i l'escriptura en un altre fitxer. Caldrà llegir en primer lloc el nom del fitxer de lectura i després el d'escriptura.

- Exercici d'autoavaluació:
  Hi ha funcions que poden ser usades en diferents programes, i per tal de no haver d'escriure en cada programa aquestes funcions les podem guardar en fitxers diferents i usar-les en cada programa, només caldrà posar la capçalera en el fitxer on s'està usant i compilar tots els fitxers **`.c`** involucrats i muntar tots els **`.o`** necessaris.

3 Donada una tolerància, *tol*, volem determinar si dos vectors, `x, y`, són o no ortogonals, és a dir, si $|x \cdot y| < tol$.

En un fitxer de nom `prod_escalar.c` escriviu una funció de prototipus

```
float prod_esc( int, float *, float *);
```

que calculi el producte escalar de dos vectors de dimensió $n \in \mathbb{N}$.

En un fitxer de nom `main_ortog.c` escriviu la funció `main`, que determini mitjançant la funció `prod_esc` si els dos vectors són o no ortogonals. Llegiu, en aquest ordre, la dimensió dels vectors, els vectors i la tolerància.

Compileu el contingut de cada fitxer, munteu els dos `objectes` obtinguts i executeu el programa amb les dades:

- $n = 3, x = (1,1,1)^t, x = (2,2,2)^t, tol = 10^{-3}$
- $n = 6, x = (1,1,1,1,1,1)^t, y = (2,-4,2,-4,2,-4)^t, tol = 10^{-3}$
- $n = 3, x = (0.9234, 0.6789, 1.1111)^t, y = (2.213, -2.9871, -0.01395)^t, tol = 10^{-3}$
- $n = 3, x = (0.9234, 0.6789, 1.1111)^t, y = (2.213, -2.9871, -0.01395)^t, tol = 10^{-6}$