www.referateok.ro – cele mai ok referate

Algebra polinoamelor incomplete de mai multe nedeterminate, implementare dinamică

Structuri de date:

Pentru implementarea unui polinom incomplet de mai mult nedeterminate am definit o clasă ***'poli'*** cu următoarele date:

Int nr_var	Numărul de nedeterminate ale polinomului
Int nr_elem	Numărul de elemente Nu se citeşte, ci este iniţializat în funcţie de elementele introduse
Float *coef	Coeficienţii polinomului (nu exista coeficienţi de valoare 0 •)
Int *gr[30]	Vector de pointeri spre întregi reprezentând pentru fiecare coeficient puterile corespunzătoare fiecărei nedeterminate Ex: Pentru polinomul $P(x,y,z)=x^2y+1$ Gr[0][0]=2, Gr[0][1]=1, Gr[0][0]=0 Gr[1][0]=0, Gr[0][1]=0, Gr[0][0]=0

Toate datele de mai sus sunt *private* pentru a nu fi permisă modificarea lor decât prin funcţiile clasei. Tot privată este şi funcţia *'pos'* pentru că nu trebuie să fie vizibilă din exterior.

Operaţiile efectuate asupra polinoamelor:

- ✓ Adunare
- ✓ Scădere
- ✓ Înmulţirea a 2 polinoame
- ✓ Calculul derivatei în raport cu o variabilă specificată prin indice
- ✓ Calculul primitivei în raport cu o variabilă specificată prin indice

```
class poli{   private:int nr_var;            int nr_elem;            float *coef;
int *gr[30];    private:int pos(int *,int);
    public:poli();
          void init();
          void af();
          void af_t();
          void add_sub(poli&,poli&,int op);
          void mul(poli&,poli&);
          void deriv(int);
          void prim(int);

          void operator=(poli&);
          ~poli();
 };
```

Prezentarea programului:

1. poli();
   - constructorul clasei

• ceea ce înţelege *float-ul* ca 'valoare 0'

- iniţializează polinomul ca polinom nul
- nu poate fi decât public

2. void init();
   - funcţie publică ce iniţializează polinomul cu valori citite de la tastatură
   - polinomul se consideră introdus corect, adică nu trebuie introduse aceleaşi secvenţe de puteri (Ex: P(x)=5x+2+3)
   - se citesc coeficienţi până când nu se mai introduce unul valid (în mod normal asta s-ar face frumos cu CTRL-Z, dar din păcate *cscanf*()-ul nu îl recunoaşte)

3. void af();
   - funcţie publică pentru afişarea caracteristicilor (polinomul efectiv, numărul de variabile, numărul de elemente)

4. void af_t();
   - funcţie publică pentru afişarea în tabelul celor 10 vectori din programul demonstrativ
   - în mod normal nu ar avea ce căuta în bibliotecă

5. void add_sub(poli&,poli&,int op);
   - funcţie publică pentru adunare/scădere; op=1 pentru adunare, op=-1 pentru scădere
   - funcţia este comună pentru micşorarea codului
   - am ales parametrii adrese, nu pentru a-i modifica ci pentru a nu lucra cu copii ale lor (nu avea nici un rost)
   - sunt permise şi apelări de genul: *p.add(p1,p)* sau chiar *p.add(p,p)*

6. void mul(poli&,poli&);
   - funcţie publică pentru înmulţire
   - aceleaşi observaţii ca la funcţia add_sub()

7. void deriv(int);
   - funcţie publică pentru derivare în raport cu o variabilă
   - indicele variabilei este minim 1 şi maxim *nr_var*

8. void prim(int);
   - funcţie publică pentru aflarea unei primitive în raport cu o variabilă
   - indicele variabilei este minim 1 şi maxim *nr_var*

9. void operator=(poli&);
   - redefinirea operatorului de atribuire (copiez varorile)
   - deşi nu-l folosesc în programul demostrativ, trebuie redefinit pentru că în interiorul clasei sunt adrese şi prin atribuirea implicită s-ar face şi atribuire de adrese…

10. ~poli();
    - destructorul clasei

Observaţie:

Datele se consideră introduse corect.

Sursa programului:

M12t2.h

```
class poli{   private:int nr_var;        //nr de variabile
           int nr_elem;       //nr de elemente
           float *coef;       //coeficientii polinomului
           int *gr[30];       //gradele corespunzatoare
   private:int pos(int *,int);//caut un anumit termen si returnez pozitia
                              //dupa conventia clasica (0..n daca a fost
                              //gasit, sau -1 daca nu)
   public:poli();                 //constructor
          void init();            //citire de la tastatura
          void af();              //afisare
          void af_t();          //afisare trunchiata
          void add_sub(poli&,poli&,int op);  //op=1 adunare, op=-1 scadere
          void mul(poli&,poli&);  //inmultire
          void deriv(int);        //derivare in raport cu o variabila
          void prim(int);         //calculeaza o primitiva in raport cu o
                                  //variabila

          void operator=(poli&);  //fac atribuire
          ~poli();                //destructor
 };
```

M12t2p14.cpp

```
#include <stdio.h>#include <conio.h>#include <math.h>#include <alloc.h>#include
<mem.h>#include "m12t2.h"#define X cprintf("\n");#define eps 1.e-6poli::poli()
//initializez un obiect nou{ coef=0; nr_var=0;nr_elem=0;
 memset(gr,0,sizeof(gr));
}

poli::~poli()
{
 delete []coef;
 for (int i=0;i<nr_elem;i++) delete[]gr[i];
 nr_elem=0;
 nr_var=0;
}

void poli::operator=(poli &p)
{
 if (coef==p.coef) return;
 this->~poli(); //il fac nul
 nr_var=p.nr_var;
 nr_elem=p.nr_elem;
 coef=new float[p.nr_elem];
 for (int i=0;i<nr_elem;i++) {
              coef[i]=p.coef[i];
              for (int j=0;j<nr_var;j++) {gr[i]=new int[p.nr_var];
                                          gr[i][j]=p.gr[i][j];
                                          }
         }
}

void poli::init()
{
 window(1,13,80,25);
 textcolor(7);
 clrscr();

 int ok;
 this->~poli(); //apelez destructorul
 cprintf("Nr de variabile:");cscanf("%d",&nr_var);X
 coef=new float[30];
 float c;
 int i=-1;
 do {
  cprintf("Coeficientul:");ok=cscanf("%f",&c);X
  if (ok)
  if (fabs(c)<eps) cprintf("Coeficientul fiind 0 nu mai trebuie sa "
                           "introduceti si gradele...\a\n\r");
```

```
               else { coef[++i]=c;
                      gr[i]=new int[nr_var];
                      for (int j=1;j<=nr_var;j++){
                        cprintf(" Gradul variabilei %d:",j);
                        cscanf("%d",&gr[i][j-1]);X
                       }
                    }
  }while (ok);
 nr_elem=i+1;
}

void poli::af()
{
 window(1,13,80,25);
 textcolor(7);
 clrscr();
 if (!nr_elem) {cprintf("Polinom nul");return;}
 for (int i=0;i<nr_elem;i++)
  {
   cprintf("%+5.2f ",coef[i]);
   for (int j=0;j<nr_var;j++)
     if (gr[i][j]!=0)
      if (gr[i][j]!=1) cprintf("x%d^%d ",j+1,gr[i][j]);
                  else cprintf("x%d ",j+1 );
  }
 cprintf("\n\r%d element%c, %d variabile...\n",nr_elem,nr_elem!=1?'e':' ',nr_var);
}

void poli::af_t()
{
 int nr_char=0;
 if (!nr_elem) {cprintf("Polinom nul");return;}
 for (int i=0;i<nr_elem;i++)
  {
   nr_char+=cprintf("%+5.2f ",coef[i]);
   for (int j=0;j<nr_var;j++)
     if (gr[i][j]!=0)
      if (gr[i][j]!=1) nr_char+=cprintf("x%d^%d ",j+1,gr[i][j]);
                  else nr_char+=cprintf("x%d ",j+1 );
   if ((55-nr_char)<nr_var*10) {cprintf("...");break;}
  }
}

int poli::pos(int *p,int dim)
{
 for (int i=0;i<nr_elem;i++)
 {
  int bad=0;
  for (int j=0;j<nr_var;j++)
   if (j<dim){ if (p[j]!=gr[i][j]) {bad=1;break;}}
        else { if (gr[i][j]) {bad=1;break;}}
  for (j=nr_var;j<dim;j++)
   if (p[j]) {bad=1;break;}

  if (!bad) return i;
 }
 return -1;
}

void poli::add_sub(poli &p1,poli &p2,int op)
{

 float *c;
 int *g[30];
 int _nr_var;          //nr de variabile va fi dat de valoarea cea mai
                       //mare
 int _nr_elem,i,j=-1;
 if (p1.nr_var>p2.nr_var) _nr_var=p1.nr_var;
                     else _nr_var=p2.nr_var;

 c=new float[30];

 if (p1.nr_elem) //daca p1 nu e vid
  if (p2.nr_elem) //daca p2 nu e vid
 {
 unsigned char *mrk=new unsigned char [p2.nr_elem]; //vector de marcaje
 memset(mrk,0,p2.nr_elem);
```

```
 for (i=0;i<p1.nr_elem;i++)
  {
   int k=p2.pos(p1.gr[i],p1.nr_var);
   if (k!=-1)
   {
    mrk[k]=1;
    float rez=p1.coef[i]+p2.coef[k]*op;
    if (!(fabs(rez)<eps)){
                          c[++j]=rez;
                          g[j]=new int[_nr_var];
                          memset(g[j],0,_nr_var*2);
                          memcpy(g[j],p1.gr[i],p1.nr_var*2);
                         }
   }
   else{
        c[++j]=p1.coef[i];
        g[j]=new int[_nr_var];
        memset(g[j],0,_nr_var*2);
        memcpy(g[j],p1.gr[i],p1.nr_var*2);
       }
 }
 for (i=0;i<p2.nr_elem;i++)
   if (!mrk[i]){
                c[++j]=p2.coef[i]*op;
                g[j]=new int[_nr_var];
                memset(g[j],0,_nr_var*2);
                memcpy(g[j],p2.gr[i],p2.nr_var*2);
               }
 }
  else { //p1 nu e vid, p2 e vid
         for (i=0;i<p1.nr_elem;i++)
               {
                c[++j]=p1.coef[i];
                g[j]=new int[_nr_var];
                memset(g[j],0,_nr_var*2);
                memcpy(g[j],p1.gr[i],p1.nr_var*2);
               }
       }
  else
  if (p2.nr_elem) //p1 vid,p2 nu
       {
         for (i=0;i<p2.nr_elem;i++)
               {
                c[++j]=p2.coef[i]*op;
                g[j]=new int[_nr_var];
                memset(g[j],0,_nr_var*2);
                memcpy(g[j],p2.gr[i],p2.nr_var*2);
               }
       }

  else { //p1 e vid, p2 e vid
         delete []c;
       }

 _nr_elem=j+1;

 this->~poli(); //distrug obiectul, creand unul nou corect,nu
                //modificandu-l pe cel vechi
 coef=c;
 for (i=0;i<_nr_elem;i++) gr[i]=g[i];
 nr_elem=_nr_elem;
 nr_var=_nr_var;
}

void poli::mul(poli &p1,poli &p2)
{

 float *c;
 int *g[30];
 int _nr_var,_nr_elem,i,j,k=-1,p,*aux;

 if (p1.nr_var>p2.nr_var) _nr_var=p1.nr_var;
                     else _nr_var=p2.nr_var;

 c=new float[30];

 if (p1.nr_elem&&p2.nr_elem) //daca ambii sunt nevizi
```

```
 {
 aux=new int[_nr_var];

 for (i=0;i<p1.nr_elem;i++)
  for (j=0;j<p2.nr_elem;j++)

  {
   float rez=p1.coef[i]*p2.coef[j];
   memset(aux,0,_nr_var*2);
   for (int l=0;l<p1.nr_var;l++) aux[l]+=p1.gr[i][l];
   for (l=0;l<p2.nr_var;l++) aux[l]+=p2.gr[j][l];

// aici caut sa vad daca a mai exista vreun termen cu aceleasi puteri

   p=-1;
   for (int k1=0;k1<k+1;k1++)
   {
    int bad=0;
    for (int k2=0;k2<_nr_var;k2++)
      if (aux[k2]!=g[k1][k2]) {bad=1;break;}
    if (!bad) {p=i;break;}
   }

   if (p==-1){ c[++k]=rez;
               g[k]=new int[_nr_var];
               memset(g[k],0,_nr_var*2);
               memcpy(g[k],aux,_nr_var*2);
             }
   else c[p]+=rez;
  }
 }
  else { delete []c;
       }

 _nr_elem=k+1;

 this->~poli();

 coef=c;
 for (i=0;i<_nr_elem;i++) gr[i]=g[i];
 nr_elem=_nr_elem;
 nr_var=_nr_var;
}


void poli::deriv(int p)
{

 float *c;
 int *g[30];
 int i,k=-1;

 c=new float[30];

 if (nr_elem) //daca e nevid
 {
 for (i=0;i<nr_elem;i++)
  {
   if (gr[i][p-1])
            { c[++k]=coef[i]*gr[i][p-1];
              g[k]=new int[nr_var];
              memset(g[k],0,nr_var*2);
              memcpy(g[k],gr[i],nr_var*2);
              g[k][p-1]-=1;
             }
  }
 }
  else { delete []c;
       }

 int _nr_var=nr_var;
 this->~poli();

 coef=c;
 nr_elem=k+1;
 for (i=0;i<nr_elem;i++) gr[i]=g[i];
 nr_var=_nr_var;
```

```
}

void poli::prim(int p)
{
 if (nr_elem) //daca e nevid
 for (int i=0;i<nr_elem;i++)
  {
   coef[i]/=gr[i][p-1]+1;
   gr[i][p-1]++;
 }
}
```

main.cpp – programul demonstrativ

```
#include <stdio.h>
#include <conio.h>
#include "m12t2.h"

poli v[10];

void fix()
{
 textcolor(7);
 cprintf("ALEGETI\n\r");
 textcolor(1);
 cprintf("1.Initializare\n\r");
 cprintf("2.Afisare\n\r");
 cprintf("3.Adunare\n\r");
 cprintf("4.Scadere\n\r");
 cprintf("5.Inmultire\n\r");
 cprintf("6.Derivare\n\r");
 cprintf("7.Primitiva\n\r");
 cprintf("8.Stergere\n\r");
 cprintf("9.Despre\n\r");
 cprintf("Q.Iesire");
 textcolor(7);
 for (int i=1;i<=12;i++){gotoxy(15,i);putchar('³');}

 gotoxy(34,1);
 cprintf("%s","...Situatia polinoamelor...");

 gotoxy(1,12);
 for (i=1;i<=79;i++) putchar('Ä');
 gotoxy(15,12);putchar('Á');

 window(16,2,18,12);
 textcolor(2);
 for (i=1;i<=10;i++) cprintf("%2d:",i);
}

void toate()
{
 window(19,2,80,11);
 textcolor(2);
 clrscr();
 for (int i=1;i<=10;i++){
   gotoxy(1,i);v[i-1].af_t();
 }
}

void in()
{
 window(1,13,80,25);
 clrscr();

 int p;
 cprintf("Initializare...\n\r");
 cprintf("Introduceti indicele polinomului\n\r");
 scanf("%d",&p);
 v[p-1].init();
 v[p-1].af();
 toate();
}

void afis()
```

```
{
 window(1,13,80,25);
 clrscr();

 int p;
 cprintf("Afisare...\n\r");
 cprintf("Introduceti indicele polinomului\n\r");
 scanf("%d",&p);
 v[p-1].af();
 toate();
}

void ad()
{
 window(1,13,80,25);
 clrscr();

 int p1,p2,p;
 cprintf("Se aduna P1 cu P2, iar rezultatul se depune in P3\n\r");
 cprintf("Introduceti cei 3 indici (separati prin SPATIU):\n\r");
 scanf("%d %d %d",&p1,&p2,&p);
 v[p-1].add_sub(v[p1-1],v[p2-1],1);
 v[p-1].af();
 toate();
}

void sc()
{
 window(1,13,80,25);
 clrscr();

 int p1,p2,p;
 cprintf("Se scade din P1 P2, iar rezultatul se depune in P3\n\r");
 cprintf("Introduceti cei 3 indici (separati prin SPATIU):\n\r");
 scanf("%d %d %d",&p1,&p2,&p);
 v[p-1].add_sub(v[p1-1],v[p2-1],-1);
 v[p-1].af();
 toate();
}

void inm()
{
 window(1,13,80,25);
 clrscr();

 int p1,p2,p;
 cprintf("Se inmulteste P1 cu P2, iar rezultatul se depune in P3\n\r");
 cprintf("Introduceti cei 3 indici (separati prin SPATIU):\n\r");
 scanf("%d %d %d",&p1,&p2,&p);
 v[p-1].mul(v[p1-1],v[p2-1]);
 v[p-1].af();
 toate();
}


void drv()
{
 window(1,13,80,25);
 clrscr();

 int p,var;
 cprintf("Derivare...\n\r");
 cprintf("Introduceti indicele polinomului:\n\r");scanf("%d",&p);
 cprintf("Introduceti indicele variabilei:\n\r");scanf("%d",&var);
 v[p-1].deriv(var);
 v[p-1].af();
 toate();
}

void prm()
{
 window(1,13,80,25);
 clrscr();

 int p,var;
 cprintf("Primitiva...\n\r");
 cprintf("Introduceti indicele polinomului:\n\r");scanf("%d",&p);
```

```
 cprintf("Introduceti indicele variabilei:\n\r");scanf("%d",&var);
 v[p-1].prim(var);
 v[p-1].af();
 toate();
}

void st()
{
 window(1,13,80,25);
 clrscr();

 int p;
 cprintf("Stergere...\n\r");
 cprintf("Introduceti indicele polinomului:\n\r");scanf("%d",&p);
 v[p-1].~poli();
 toate();
}

void about()
{
 window(1,13,80,25);
 clrscr();

 cprintf("Musaloiu-Elefteri Raluca\n\r");
 cprintf("312 CB\n\r");
 cprintf("miercuri 27.01.1999\n\r");
}


void operatii()
{
 window(1,13,80,25);
 char c=getch();
 toate();
 textcolor(7);
 while (c!='Q'){
  switch (c){
   case '1':in();break;
   case '2':afis();break;
   case '3':ad();break;
   case '4':sc();break;
   case '5':inm();break;
   case '6':drv();break;
   case '7':prm();break;
   case '8':st();break;
   case '9':about();break;
  }
 c=getch();
 }
}

void main(void)
{
 clrscr();
 fix();
 operatii();
}
```

www.referateok.ro – cele mai ok referate