Univerzitet u Sarajevu
Prirodno-matematički fakultet
Odsjek za matematiku

Dokumentacija za **Projekat 1** iz predmeta
Strukture podataka i algoritmi, na temu

Klasa Polinom

Predmetni profesor: Esmir Pilav
Predmetni asistent: Admir Beširević

Student
Adisa Bolić, 5601/M

Uvod

Polinom se čuva kao **vector<double> koeficijenti**, koji sadrži sve koeficijente polinoma od manjeg ka većem, pri čemu je to i jedini atribut klase Polinom. Također, pri svakom kreiranju polinoma se vodi računa o tome da ukoliko je zadnjih nekoliko koeficijenata 0 (tj. onih sa najvećim stepenima) oni se ne čuvaju u polinomu, već se prosto zanemare.

Pored neophodne implementacije, dodani su još konstruktor sa jednom parametrom (**vector<double>**), konstruktor bez parametara, funkcija **NapraviPolinom**, binarni operatori **operator+**, **operator-**, unarni **operator-** te funkcija **PomnoziMonomom**.

```
class Polinom{

    vector<double> koeficijenti;

public:

//---------------------------------------- KONSTRUKTORI ---------------------------------
    Polinom();
    Polinom(vector<double> k);

//---------------------------------------- OPERATORI ------------------------------------

    friend istream& operator>>(istream&, Polinom&);
    friend ostream& operator<<(ostream&, const Polinom&);
    double operator()(double);
    friend Polinom operator-(const Polinom&);
    friend Polinom operator+(const Polinom&, const Polinom&);
    friend Polinom operator-(const Polinom&, const Polinom&);
    friend Polinom operator*(const Polinom&, const Polinom&);

//---------------------------------------- DODATNE FUNKCIJE -----------------------------
    void NapraviPolinom(vector<double> k);
    void PomnoziMonomom(int m);
    double NulaPolinoma(double,double,double);
    friend void SortirajPolinome(vector<Polinom>&);

};

//---------------------------------------- DODATNE FUNKCIJE VAN KLASE-------------------
Polinom operator^(const Polinom&, int n);
void SortirajPolinome(vector<Polinom>&);
double PresjekPolinoma(const Polinom&, const Polinom&, double, double, double);
```

Figure 1 Izgled fajla "polinom.h"

Konstruktori i funkcije

1. **Polinom::Polinom(){ koeficijenti = {0};}**

Konstruktor bez parametara koji kreira nula-polinom. U ovom slučaju (i samo ovom) se dopušta da koeficijent uz najveći stepen bude 0.

2. **Polinom::Polinom(vector<double> *k*);**

Konstruktor sa jednim parametrom koji koeficijente polinoma postavlja na vrijednosti iz proslijeđenog vektora, pri čemu koeficijent na i −toj poziciji u vektoru k odgovara koeficijentu uz i-ti stepen, vodeći računa o tome da, osim u slučaju nula polinoma, ne čuva zadnje 0 kao koeficijente.

3. **friend istream& operator>>(istream &fin, Polinom &*p*);**

Operator>> služi sa unos polinoma sa tastature, u obliku zbira monoma (ne nužno sortiranih po stepenu) pri čemu se upis izvršava i ukoliko se stavlja znak "*" između koeficijenta i ako se ne stavlja, prepoznaje se ako su koeficijenti 1 i −1 nenapisani te ako stepeni 0 i 1 nisu napisani. Ukoliko je nepravilan unos baca se izuzetak sa porukom **„Neisparavan unos!“**.

4. **friend ostream& operator<<(ostream &fout, const Polinom &*o*);**

Operator<< služi za ispis polinoma na ekran, u obliku zbira monoma sortiranih po stepenu od najvećeg ka najmanjem. Pri tome se vodi računa da se monomi sa koeficijentom 0 ne ispisuju (osim u slučaju nula-polinoma), da se ne ispisuje koeficijent 1, odnosno −1 (već samo +, odnosno −), da se ne ispisuju stepen 1 te stepen 0 uz monom.

5. **double Polinom::operator()(double *x*);**

Vraća vrijednost polinoma u proslijeđenoj tački, koristeći formulu:

$$p_N x^N + p_{N-1} x^{N-1} + \ldots + p_2 x^2 + p_1 x + p_0 = (((\ldots(p_N x + p_{N-1}) x + \ldots) x + p_2) x + p_1) x + p_0$$

6. **friend Polinom operator-(const Polinom &*p*);**

Vraća polinom kojem su svi koeficijenti suprotnog predznaka u odnosu na proslijeđeni polinom.

7. **friend Polinom operator+(const Polinom &*p*, const Polinom &*q*);**

Sabira dva polinoma, pri čemu prvo kreira vektor koeficijenata polinoma zbira, izračuna njegove vrijednosti, a potom vrati taj vektor, imajući u vidu da postoji konverzija između **vector<double>** i **Polinom**. Funkcija provjerava prvo koji je polinom većeg stepena, kako bi utvrdila koji je stepen polinoma zbira i kako da sabira koeficijente.

8. friend Polinom operator-(const Polinom &p, const Polinom &q);

Oduzima dva polinoma na vrlo jednostavan način, koristeći binarni **operator+** i unarni **operator-**.

9. friend Polinom operator*(const Polinom &p , const Polinom &q);

Funkcija vraća polinom koji predstavlja proizvod polinoma p i q. Radi u vremenu $n^{\log_2 3}$, gdje je n veći od stepena p i q. To se postiže rekurzivnim *divide and conquer algoritmom,* kojeg ćemo ovdje opisat.

Naime, ako je polinom P dat kao:

$$\boldsymbol{P(x) = \sum_{i=0}^{n} a_i x^i}$$

možemo ga razbiti na zbir dva polinoma sa:

$$\boldsymbol{P(x) = \sum_{i=0}^{m-1} a_i x^i + x^m \sum_{i=0}^{n-m} a_{i+m} x^i}$$

gdje je $\boldsymbol{m = \left\lceil \frac{n}{2} \right\rceil}$. Ako označimo sa:

$$\boldsymbol{A = \sum_{i=0}^{m-1} a_i x^i \qquad i \qquad B = x^m \sum_{i=0}^{n-m} a_{i+m} x^i}$$

tada smo polimom P napisali kao $P(x) = A + x^m B$, tj. napisali smo ga preko dva polinoma A i B koji su otprilike duplo manjeg stepena od P.

Neka je sada $n = \max(p.koeficijenti.size(), q.koeficijenti.size())$, a $m = \left\lceil \frac{n}{2} \right\rceil$. Polinome p i q napišemo (preko gore opisanog postupka) u obliku $p = A + x^m B$ i $q = C + x^m D$. (dozvoljavamo da neki od A, B, C, D budu nula-polinomi, jer je potrebno da polinomi koji se množe budu istog stepena, što mi zaobilazimo upravo na način da ako je neki polinom manjeg stepena od drugog, stavljamo da mu je drugi dio nula polinom, tačnije polinom koji ima onoliko koeficijenata koji su svi 0 koliko mu treba da se dopuni do veličine drugog polinoma. Ovo je neophodno ovako uraditi, jer nije nam dozvoljeno dodavanje monoma sa koeficijentima 0 stepena većeg od stepena polinoma, na dati polinom, zbog osobine zanemarivanja takvih koeficijenata pri kreiranju polinoma).

Lagano se pokaže da vrijedi:

$$\boldsymbol{p * q = (A + x^m B) * (C + x^m D) = AC + \big((A + B)(C + D) - AC - BD\big)x^m + BDx^{2m}.}$$

Ovako smo problem množenja polinoma p i q sveli na 3 množenja polinoma $A\ i\ C, (A + B)\ i\ (C + D)$ te $B\ i\ D$, kod kojih je stepen većeg od njih duplo manji od n. Množenje monomima x^m i x^{2m} se postiže funkcijom PomnoziMonomom koja radi u linearnom vremenu. Tri množenja gore opisana vršimo rekurzivno pozivom

Polinom T1(A*C),T2(B*D),T3((A+B)*(C+D)-T1-T2),

a potom izračunamo i

T3.PomnoziMonomom(m);

Polinom T4(T2);

T4.PomnoziMonomom(2*m);

Napokon rezultat množenja polinoma p i q je dat sa:

$$\boldsymbol{T1 + T3 + T4}$$

Pošto sabiranje i oduzimanje polinoma radi u vremenu $O(n)$, gdje je n veći od stepena tih polinoma, kao i funkcija PomnoziMonomom, to dobijamo rekurzivnu relaciju:

$$\boldsymbol{T(n) = 3T\left(\frac{n}{2}\right) + O(n)}$$

a iz toga vidimo, primjenom Master teorema da je $\boldsymbol{T(n) = O(n^{\log_2 3})}$.

10. Polinom operator^(const Polinom *p* int *n*);

Stepenuje proslijeđeni polinom p na proslijeđeni stepen n. Funkcija koristi napravljeni **operator*** te se poziva rekurzivno radi efikasnosti, u cilju iskorištavanja već izračunatog dijela stepena.

11. void Polinom::NapraviPolinom(vector<double> *k*);

Pomoćna funkcija, koja radi skoro potpuno isto kao konstruktor sa jednim parametrom, osim što još vodi računa da prvo očisti postojeći vektor koeficijenata.

12. void Polinom::PomnoziMonomom(int *m*);

Pomoćna funkcija koja pomaže u implementaciji **operator*,** koja množi dati polinom sa polinomom x^m na način da pomjeri koeficijente polinoma za m mjesta „udesno“.

13. double Polinom::NulaPolinoma(double *a*,double *b*,double *e*);

Vraća eventualnu nulu polinoma, sa proslijeđenom preciznošću e, između proslijeđenih krajeva intervala a i b, pri čemu je potrebno da vrijednosti funkcije u tim tačkama budu suprotnog znaka, kako bi se osigurala postojanost nule u tom intervalu. Funkcija također radi ispravno ako su eventualno krajevi tog intervala nule polinoma. Ako vrijednosti funkcije na krajevima intervala nisu suprotnog predznaka baca se izuzetak sa porukom "Vrijednosti polinoma u tackama a i b moraju biti suprotnog znaka!". U funkciji se koristi metod bisekcija za traženje nula funkcija.

14. friend void SortirajPolinome(vector<Polinom> &*v*);

Sortira polinome u proslijeđenom vektoru po veličini, od najvećeg do najmanjeg, pri čemu je polinom p veći od polinoma q ako je $\boldsymbol{\lim_{n\to\infty} |\frac{p(x)}{q(x)}| > 1}$, što je ekvivalentno s tim da je stepen

polinoma p veći od stepena polinoma q ili da ako imaju isti stepen, apsolutna vrijednost vodećeg koeficijenta u p je veća od apsolutne vrijednost vodećeg koeficijenta u q. Funkcija koristi ugrađenu ***std*** :: ***sort*** funkciju sa odgovarajućom funkcijom poređenja.

15. **double PresjekPolinoma(const Polinom &*p*, const Polinom &*q*, double *a*, double *b*, double *e*);**

Vraća tačku eventualnog presjeka proslijeđenih polinoma, sa proslijeđenom preciznošću e, između proslijeđenih krajeva intervala a i b. Pri tome neophodno je da je $(\mathrm{p(a)} - \mathrm{p(b)})*(\mathrm{q(a)} - \mathrm{q(b)}) \leq 0$ kako bi se obezbijedilo da se polinomi sijeku na datom intervalu. Ako taj uslov nije ispunjen baca se izuzetak sa porukom:

"Vrijednost (P(a)-P(b))*(Q(a)-Q(b)) mora biti manja od 0!". Funkcija koristi već ugrađenu funkciju **NulaPolinoma**, jer je tačka presjeka polinoma p i q upravo nula polinoma $p - q$.