Objektno programiranje (C++)

Četvrta domaća zadaća (ak. god. 2022./2023.)

Datum objave na Merlinu: petak, 12. svibnja 2023. od 20:00h	Rok za predaju preko Merlina: ponedjeljak, 22. svibnja 2023. do 23:00h
Ukupan broj zadataka: **1 zadatak**	Ukupno moguće ostvariti bodova: **10 bodova**

Zadatak 1. *(10 bodova)* Podsjetnik: **Polinom n-tog stupnja** (nad $\mathbb{R}$) je funkcija $f : \mathbb{R} \to \mathbb{R}$ dana s $f(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$ gdje su $n \in \mathbb{N}_0$, $a_0, a_1, \ldots, a_n \in \mathbb{R}$, $a_n \neq 0$. Brojeve $a_0, a_1, \ldots, a_n$ zovemo koeficijenti polinoma, a_n vodeći koeficijent, a a_0 slobodni koeficijent. Ako je $f \neq 0$, broj n zovemo **stupanj polinoma**. Ako je $f(x) = 0$ za sve $x \in \mathbb{R}$, onda polinom f zovemo **nul-polinom**. Svaki član $a_i x^i$ nazivamo **monom**.

Svaki polinom možemo predstaviti preslikavanjem koje nenegativnim cijelim brojevima pridružuje koeficijent uz odgovarajuću potenciju (`map<unsigned,double>`). Primjerice, za polinom $p(x) = 7x^0 - 4.01x^2 + 15.2x^{10}$ imamo preslikavanje: $0 \mapsto 7$, $2 \mapsto -4.01$, $10 \mapsto 15.2$. Potrebno je implementirati klasu `Polinom` koja omogućuje standardne operacije s polinomima. S obzirom da želimo imati klasu koja se ponaša poput pokazivača (da ne bi bilo bespotrebnog kopiranja moguće velikog resursa), ali se ne želimo baviti implementacijom kontrole kopiranje, klasa `Polinom` ima jedan pametan pokazivač (`shared_ptr`) na preslikavanje koje predstavlja taj polinom.

Vaš zadatak je dopuniti datoteku `Polinom.h` u kojoj je potrebno implementirati sve tražene funkcije (konstruktori/operatori/funkcije članice) ili dopuniti datoteku `Polinom-template.h` s parametriziranim verzijama tih funkcija (pritom za tip `T` (ili `U`) kojim se parametrizira možete pretpostaviti da je *floating-point* tip poput `double`, `float`, `long double`). Ispravno dopunjena datoteka `Polinom.h` nosi maksimalno 8 bodova, a ispravno dopunjena datoteka `Polinom-template.h` nosi maksimalno 10 bodova. Potrebno je predati samo jednu od navedenih datoteka (ako više puta predate zadaću i pritom imate obje datoteke, tada će se ocijeniti samo dopunjena `Polinom-template.h`). Datoteka `main.cpp`, kao i prikaz izlaza za unos `-2x^3+3x^0-2.1x^3+2x^5`, može se vidjeti na kraju ovog dokumenta. Pritom je prvi prikazani polinom na tom prikazu izlaza slučajno generiran.

Slijedi opis funkcija članica/konstruktora/operatora:

1. Konstruktor bez argumenata stvara nul-polinom, a konstruktor koji prima jedan nenegativan broj n stvara polinom stupnja n (pripazite na ovaj uvjet!) čiji su koeficijenti a_i, za $i \in \{0, 1, \ldots, n\}$, slučajno generirani brojevi tipa `double` (ili u parametriziranoj verziji tipa `T`) pri čemu je pri generiranju koeficijenata korištena normalna distribucija s očekivanjem 0 i standardnom devijacijom 1.

2. Funkcija `stupanj` vraća stupanj polinoma (za nul-polinom vraća -1).

3. Operator konverzije u `bool` vraća `true` ako se ne radi o nul-polinomu (inače vraća `false`).

4. Operator za ispis ispisuje 0 ako se radi o nul-polinomu, a inače ispisuje polinom u jednom retku kao niz monoma (bez razmaka između) pri čemu se mora ispisati koeficijent (obavezno s predznakom + ili -), znak `'x'`, znak `'^'`, potencija (pri čemu su potencije u uzlaznom poretku te se ne ispisuju monomi s koeficijentom 0). Primjerice, polinom $p(x) = 4x^5 - 3.2 + 5x$ ispiše se kao `-3.2x^0+5x^1+4x^5`.

5. Operator za unos učitava jedan redak - ako se radi o praznom retku, tada se radi o nul-polinomu, a inače se redak sastoji od monoma (bez razmaka između) u obliku koeficijent s predznakom, znak `'x'`, znak `'^'`, potencija. Primjerice, unos `-2x^3+3x^0-2.1x^3+2x^5` predstavlja polinom $p(x) = 2x^5 - 4.1x^3 + 3$. Pokušajte ne spremati podatke za monome čiji su koeficijenti jednaki 0.

6. Operatori +, - i * rade standardno zbrajanje, oduzimanje i množenje polinoma. Unarni operator - za polinom $p(x)$ vraća polinom u kojem svi koeficijenti imaju suprotan predznak od odgovarajućih koeficijenata u polinomu $p(x)$.

7. Operator / radi dijeljenje polinoma, a operator % za polinome $p(x)$ i $q(x)$ vraća ostatak pri dijeljenju polinoma $p(x)$ polinomom $q(x)$. Oba moraju baciti `runtime_error` iznimku pri pokušaju dijeljenja nulom (vidi `main.cpp` za testiranje i pripadni primjer izlaza).

8. Binarni operatori iz prethodnih točaka 6. i 7. imaju pripadne operatore +=, -=, *=, /=, %=.

9. Operator pridruživanja s argumentom koji je inicijalizacijska lista: ta lista predstavlja redom koeficijente $a_0, a_1, \ldots, a_n$. Primjerice, `p = {2,3.1,0,0,5};` će promijeniti polinom p u polinom $p(x) = 5x^4 + 3.1x + 2$.

10. Operator indeksiranja `[]` (preopterećen po `const`) za broj n vraća koeficijent a_n (pri čemu će to biti neki broj različit od nula ili nula - primjerice, za $p(x) = x^2$, `p[100]` je 0).

11. Operator zagrade `()` za broj x vraća vrijednost $p(x)$. Vraćena vrijednost je tipa `double` (čak i u parametriziranoj verziji).

12. Operatori pred/postdekrementiranja i pred/postinkrementiranja smanjuju ili povećavaju slobodni koeficijent polinoma za 1 - primjerice, za $p(x) = x^2 + x^3$, `++p` daje polinom $p(x) = 1 + x^2 + x^3$. Pazite što vraća pred, a što post inkrementiranje/dekrementiranje.

Datoteka **Polinom.h** (potrebno nadopuniti ako se rješava neparametrizirana verzija):

```
#ifndef POLINOM_H
#define POLINOM_H

#include <iostream>
#include <map>
#include <memory>
#include <random>
#include <sstream>
#include <string>
#include <stdexcept>
#include <initializer_list>

using namespace std;

class Polinom {
    friend ostream& operator<<(ostream&, const Polinom&);
    friend istream& operator>>(istream&, Polinom&);
    friend Polinom operator+(const Polinom&, const Polinom&);
    friend Polinom operator-(const Polinom&, const Polinom&);
    friend Polinom operator*(const Polinom&, const Polinom&);
    friend Polinom operator/(const Polinom&, const Polinom&);
    friend Polinom operator%(const Polinom&, const Polinom&);
  public:
    Polinom();
    Polinom(unsigned);
    operator bool() const;
    Polinom& operator=(initializer_list<double>);
    Polinom& operator+=(const Polinom&);
    Polinom& operator-=(const Polinom&);
    Polinom& operator*=(const Polinom&);
    Polinom& operator/=(const Polinom&);
    Polinom& operator%=(const Polinom&);
    Polinom& operator++();
    Polinom& operator--();
    Polinom operator++(int);
    Polinom operator--(int);
    Polinom operator-() const;
    double& operator[](size_t);
    double operator[](size_t) const;
    double operator()(double) const;
    int stupanj() const; //-1 za nul-polinom
  private:
    shared_ptr<map<unsigned,double>> koeficijenti;
};
```

```
/* ISPOD NAPISATI SVE POTREBNE DEFINICIJE */

/* ...  */

#endif
```

Datoteka **Polinom-template.h** (potrebno nadopuniti ako se rješava parametrizirana verzija):

```
#ifndef POLINOM_H
#define POLINOM_H

#include <iostream>
#include <map>
#include <memory>
#include <random>
#include <sstream>
#include <string>
#include <stdexcept>
#include <initializer_list>

using namespace std;

template<typename T>
class Polinom;

template<typename T>
ostream& operator<<(ostream&, const Polinom<T>&);
template<typename T>
istream& operator>>(istream&, Polinom<T>&);
template<typename T>
Polinom<T> operator+(const Polinom<T>&, const Polinom<T>&);
template<typename T>
Polinom<T> operator-(const Polinom<T>&, const Polinom<T>&);
template<typename T>
Polinom<T> operator*(const Polinom<T>&, const Polinom<T>&);
template<typename T>
Polinom<T> operator/(const Polinom<T>&, const Polinom<T>&);
template<typename T>
Polinom<T> operator%(const Polinom<T>&, const Polinom<T>&);

template<typename T>
class Polinom {
  friend ostream& operator<<<T>(ostream&, const Polinom<T>&);
  friend istream& operator>><T>(istream&, Polinom<T>&);
  template<typename U>
  friend Polinom<U> operator+(const Polinom<U>&, const Polinom<U>&);
```

```
  template<typename U>
  friend Polinom<U> operator-(const Polinom<U>&, const Polinom<U>&);
  template<typename U>
  friend Polinom<U> operator*(const Polinom<U>&, const Polinom<U>&);
  template<typename U>
  friend Polinom<U> operator/(const Polinom<U>&, const Polinom<U>&);
  template<typename U>
  friend Polinom<U> operator%(const Polinom<U>&, const Polinom<U>&);
  public:
    Polinom();
    Polinom(unsigned);
    operator bool() const;
    Polinom& operator=(initializer_list<T>);
    Polinom& operator+=(const Polinom&);
    Polinom& operator-=(const Polinom&);
    Polinom& operator*=(const Polinom&);
    Polinom& operator/=(const Polinom&);
    Polinom& operator%=(const Polinom&);
    Polinom& operator++();
    Polinom& operator--();
    Polinom operator++(int);
    Polinom operator--(int);
    Polinom operator-() const;
    T& operator[](size_t);
    T operator[](size_t) const;
    double operator()(double) const;
    int stupanj() const; //-1 za nul-polinom
  private:
    shared_ptr<map<unsigned,T>> koeficijenti;
};

/* ISPOD NAPISATI SVE POTREBNE DEFINICIJE */

/* Primjer1:
template<typename T>
Polinom<T>::Polinom() {
    ...
} */

/* Primjer2:
template<typename T>
Polinom<T> operator+(const Polinom<T> &lp,
        const Polinom<T> &dp) {
    ...
} */
```

```
/* Primjer3:
template<typename T>
Polinom<T> Polinom<T>::operator--(int) {
    ...
} */

#endif
```

Datoteka **main.cpp** za testiranje:

```
#include <iostream>
#include <initializer_list>
#include "Polinom.h"
/* gornju liniju zamijenite donjom ako
testirate parametriziranu verziju */
//#include "Polinom-template.h"
using namespace std;

int main() {
    Polinom p1(5), p, q;
    /* gornju liniju zamijenite donjom ako
       testirate parametriziranu verziju */
    //Polinom<float> p1(5), p, q;
    cout << "Slucajno generirani polinom:\n"
         << p1 << endl;
    cin >> p1;
    cout << "Ucitani polinom:  " << p1
         << " (stupanj polinoma:  " << p1.stupanj()
         << ")" << endl;

    p = {2,3,0,0,5};
    p[6] = 1;
    q = {0,0,2,-1,4};

    cout << "p(x) = " << p << endl;
    cout << "q(x) = " << q << endl;
    cout << "(-q)(x) = " << -q << endl;
    cout << "p(2.3) = " << p(2.3) << endl;

    cout << "(p++)(x) = " << p++ << endl;
    cout << "(++p)(x) = " << ++p << endl;
    cout << "(q--)(x) = " << q-- << endl;
    cout << "(--q)(x) = " << --q << endl;
    cout << "(p+q)(x) = " << p + q << endl;
```

```
    cout << "(p-q)(x) = " << p - q << endl;
    cout << "(p*q)(x) = " << p * q << endl;
    cout << "(p/q)(x) = " << p / q << endl;
    cout << "(p%q)(x) = " << p % q << endl;

    cout << "p += q => p(x) = " << (p += q) << endl;
    cout << "p -= q => p(x) = " << (p -= q) << endl;
    cout << "p *= q => p(x) = " << (p *= q) << endl;
    cout << "p /= q => p(x) = " << (p /= q) << endl;
    cout << "p %= q => p(x) = " << (p %= q) << endl;

    q = {0};
    cout << "q(x) = " << q << endl;
    if(q)
        cout << "q nije nul-polinom" << endl;
    else
        cout << "q je nul-polinom" << endl;
    cout << "(p/q)(x) = ";
    try {
        cout << p / q << endl;
    } catch (runtime_error &greska) {
        cout << greska.what() << endl;
    }
    cout << "(p%q)(x) = ";
    try {
        cout << p % q << endl;
    } catch (runtime_error &greska) {
        cout << greska.what() << endl;
    }

    return 0;
}
```

Napomena: Gore prikazane datoteke bilo je moguće preuzeti preko Merlina.

Prikaz izlaza za primjer naveden u tekstu zadatka može se vidjeti na sljedećoj stranici.

```
sehorva@DESKTOP-S92RB8H: ~/Polinom
sehorva@DESKTOP-S92RB8H:~/Polinom$ ./prog
Slucajno generirani polinom:
-1.85679x^0+0.818444x^1+0.200739x^2-1.57404x^3-0.132637x^4-0.0487521x^5
-2x^3+3x^0-2.1x^3+2x^5
Ucitani polinom: +3x^0-4.1x^3+2x^5 (stupanj polinoma: 5)
p(x) = +2x^0+3x^1+5x^4+1x^6
q(x) = +2x^2-1x^3+4x^4
(-q)(x) = -2x^2+1x^3-4x^4
p(2.3) = 296.856
(p++)(x) = +2x^0+3x^1+5x^4+1x^6
(++p)(x) = +4x^0+3x^1+5x^4+1x^6
(q--)(x) = +2x^2-1x^3+4x^4
(--q)(x) = -2x^0+2x^2-1x^3+4x^4
(p+q)(x) = +2x^0+3x^1+2x^2-1x^3+9x^4+1x^6
(p-q)(x) = +6x^0+3x^1-2x^2+1x^3+1x^4+1x^6
(p*q)(x) = -8x^0-6x^1+8x^2+2x^3+3x^4+12x^5+8x^6-5x^7+22x^8-1x^9+4x^10
(p/q)(x) = +1.14062x^0+0.0625x^1+0.25x^2
(p%q)(x) = +6.28125x^0+3.125x^1-1.78125x^2+1.01562x^3
p += q => p(x) = +2x^0+3x^1+2x^2-1x^3+9x^4+1x^6
p -= q => p(x) = +4x^0+3x^1+5x^4+1x^6
p *= q => p(x) = -8x^0-6x^1+8x^2+2x^3+3x^4+12x^5+8x^6-5x^7+22x^8-1x^9+4x^10
p /= q => p(x) = +4x^0+3x^1+5x^4+1x^6
p %= q => p(x) = +6.28125x^0+3.125x^1-1.78125x^2+1.01562x^3
q(x) = 0
q je nul-polinom
(p/q)(x) = Dijeljenje nul-polinomom!
(p%q)(x) = Dijeljenje nul-polinomom!
```

□