PREKLAPANJE OPERATORA

1. Realizovati apstraktni tip podataka koji predstavljaju kompleksne brojeve. Klasu obezbediti konstruktorima i destruktorima. Izvršiti preklapanje operatora osnovnih aritmetičkih operacija, inkrementiranja, post i preinkrementiranja, jednakosti i nejednakosti, kao i operatora za ispis na standardni izlaz (<<).

```
#pragma once
#include <iostream>
using namespace std;

class Complex
{
   double real, imag;
public:
   Complex();
   Complex(const Complex&);
   Complex(double);
   Complex(double, double);
   //Complex cSab(Complex);
   //Complex cOduz(Complex);
   double cRe() const { return real; };
   double cIm() const { return imag; };

   void pisi() const { cout << "(" << real << "," << imag << ")" << endl; };

   //friend Complex operator+(Complex, Complex);
   //friend Complex operator-(Complex, Complex);
   //friend Complex operator*(Complex, Complex);
   //friend Complex operator/(Complex, Complex);
   friend Complex operator+(const Complex&, const Complex&);
   friend Complex operator-(const Complex&, const Complex&);
   friend Complex operator*(const Complex&, const Complex&);
   friend Complex operator/(const Complex&, const Complex&);
   Complex& operator+=(const Complex&);
   Complex& operator++();
   Complex operator++(int);

   //void operator@(Complex k) {}

   ~Complex();
   friend ostream& operator<<(ostream&, const Complex&);
   friend bool operator==(const Complex&, const Complex&);
   friend bool operator!=(const Complex&, const Complex&);
};
```

Complex/Complex.h

```
#include "Complex.h"

Complex::Complex()
{
   //cout << "Pozvan je podrazumevani konstruktor" << endl;
}

Complex::Complex(double a, double b) : real(a), imag(b)
{
   //cout << "Pozvan je konstruktor sa parametrima" << endl;
}

Complex::Complex(double a) : real(a), imag(0)
{
   cout << "Pozvan je konstruktor konverzije" << endl;
}

Complex::Complex(const Complex& c) : real(c.real), imag(c.imag)
{
   //cout << "Pozvan je konstruktor kopije" << endl;
}

/*
Complex Complex::cSab(Complex a)
{
   Complex priv;
   priv.real = real + a.real;
   priv.imag = imag + a.imag;
   return priv;
}

Complex Complex::cOduz(Complex a) {
   Complex priv;
   priv.real = real - a.real;
   priv.imag = imag - a.imag;
   return priv;
}
*/

Complex::~Complex()
{
   //cout << "Pozvan je destruktor" << endl;
}


Complex operator+(const Complex& c1, const Complex& c2)
//Complex operator+(Complex c1, Complex c2)
{
   //cout << "c1 = (" << c1.real << ", " << c1.imag << ")" << endl;
   //cout << "c2 = (" << c2.real << ", " << c2.imag << ")" << endl;
   return Complex(c1.real + c2.real, c1.imag + c2.imag);
}

Complex operator-(const Complex& c1, const Complex& c2)
//Complex operator-(Complex c1, Complex c2)
{
   return Complex(c1.real - c2.real, c1.imag - c2.imag);
}

Complex operator*(const Complex& c1, const Complex& c2)
```

```
61 //Complex operator*(Complex c1, Complex c2)
62 {
63    return Complex(c1.real * c2.real - c1.imag * c2.imag, c1.imag * c2.real + c1.↩
          real * c2.imag);
64 }
65
66 Complex operator/(const Complex& c1, const Complex& c2)
67 //Complex operator/(Complex c1, Complex c2)
68 {
69    double m = c2.real * c2.real + c2.imag * c2.imag;
70    return Complex((c1.real * c2.real + c1.imag * c2.imag) / m, (c1.real * c2.imag -↩
          c1.imag * c2.real) / m);
71 }
72
73 Complex& Complex::operator+=(const Complex& c)
74 {
75    //cout << "param = (" << c.real << ", " << c.imag << ")" << endl;
76    this->real = this->real + c.real;
77    this->imag = this->imag + c.imag;
78    return *this;
79 }
80
81 Complex& Complex::operator++()
82 {
83    return (*this) += 1;
84 }
85
86 Complex Complex::operator++(int)
87 {
88    Complex tmp(*this);
89    (*this) += 1;
90    return tmp;
91 }
92
93 ostream& operator<<(ostream& out, const Complex& c)
94 {
95    return out << "(" << c.real << ',' << c.imag << ")" << endl;
96 }
97
98 bool operator==(const Complex& c1, const Complex& c2)
99 {
100    return (c1.real == c2.real) && (c1.imag == c2.imag);
101 }
102
103 bool operator!=(const Complex& c1, const Complex& c2)
104 {
105    return !(c1 == c2);
106 }
```

Complex/Complex.cpp

```
1 #include "Complex.h"
2
3 int main()
4 {
5
6    double a, b;
7    cout << "Unesite vrednost za realni i imaginarni deo c1" << '\n';
8    cin >> a >> b;
```

```
9     Complex c1(a, b);
10
11    cout << c1;
12    cout << c1++;
13    //c1++.pisi();
14
15    cout << ++c1;
16    //(++c1).pisi();
17
18    cout << "Unesite vrednost za realni i imaginarni deo c2" << '\n';
19    cin >> a >> b;
20    Complex c2(a, b);
21    Complex c3;
22    //c3 = c1.cSab(c2);
23    c3 = c1 + c2;
24    //c3 = c1 + 1;
25    cout << "Zbir dva data kompleksna broja je " << "(" << c3.cRe() << "," << c3.cIm↩
          () << ")" << endl;
26
27    //c3 = c1.cOduz(c2);
28    c3 = c1 - c2;
29    cout << "Razlika dva data kompleksna broja je " << "(" << c3.cRe() << "," << c3.↩
          cIm() << ")" << endl;
30
31    c3 += 2;
32    c3++;
33    cout << "(" << c3.cRe() << "," << c3.cIm() << ")" << endl;
34
35    cout << endl << endl;
36
37    if (c1 != c3)
38    {
39       cout << "Nisu isti :)" << endl;
40    }
41
42
43    /*
44    Complex *c22 = new Complex(3, 4);
45    cout << "(" << c22->cRe() << "," << c22->cIm() << ")" << endl;
46    delete c22;
47    */
48
49    return 0;
50 }
```

Complex/Source.cpp

2. Realizovati klasu student koja će imati podatke o imenu i prezimenu studenta, godini studija i godini upisa, kao i statičku promenljivu koja računa ukupan broj studenata. Klasa ima odgovarajuće konstruktore i destruktor, preklopljene operatore dodele i postfiksnog i prefiksnog inkrementiranja (inkrementiranje povećava godinu studija za jedan), kao i prijateljsku funkciju koja za dati skup studenata računa, na svakoj godini, koji student je najduže na studijama i ispisuje njegovo ime, datum upisa kao i godinu na kojoj je.

```
1 #pragma once
2 #include <iostream>
3 #include <string.h>
```

```
4  using namespace std;
5
6  class Student{
7  private:
8     char *ime;
9     char *prezime;
10    int gd_upisa;
11    int gd_studija;
12 public:
13    static int ukupno;
14    Student();
15    Student(char*, char*, int, int);
16    Student(const Student &);
17    ~Student();
18    Student& operator=(const Student &);
19    Student& operator++();
20    Student operator++(int);
21    friend void Pretrazi(Student *, int);
22    void citaj() { cout << ime << " " << prezime << " " << gd_upisa << " " << ↩
          gd_studija << endl; }
23 };
```

Student/Student.h

```
1  #include "Student.h"
2
3  int Student::ukupno=0;
4
5  Student::Student() : ime(0), prezime(0)
6  {
7     cout << "[Student] default konstruktor" << endl;
8     ukupno++;
9  }
10
11 Student::Student(char *ime1, char *prezime1, int a, int b) : gd_upisa(a), ↩
       gd_studija(b)
12 {
13    cout << "[Student] konstruktor sa 4 parametra" << endl;
14    int n1 = strlen(ime1);
15    ime = new char[n1 + 1];
16    strcpy_s(ime, n1 + 1 , ime1);
17    int n2 = strlen(prezime1);
18    prezime = new char[n2 + 1];
19    strcpy_s(prezime, n2 + 1, prezime1);
20    ukupno++;
21 }
22
23 Student::Student(const Student &stari) : gd_studija(stari.gd_studija), gd_upisa(↩
       stari.gd_upisa)
24 {
25    cout << "[Student] konstruktor kopije" << endl;
26    int n1 = strlen(stari.ime);
27    ime = new char[n1 + 1];
28    strcpy_s(ime, n1 + 1, stari.ime);
29    int n2 = strlen(stari.prezime);
30    prezime = new char[n2 + 1];
31    strcpy_s(prezime, n2 + 1, stari.prezime);
32    ukupno = ukupno++;
33 }
```

```
34
35 Student &Student::operator=(const Student &stari)
36 {
37    cout << "[Student] operator dodele" << endl;
38    if(&stari != this)
39    {
40       delete[] ime;
41       delete[] prezime;
42       gd_studija = stari.gd_studija;
43       gd_upisa = stari.gd_upisa;
44       int n1 = strlen(stari.ime);
45       ime = new char[n1 + 1];
46       strcpy_s(ime, n1 + 1, stari.ime);
47       int n2 = strlen(stari.prezime);
48       prezime = new char[n2 + 1];
49       strcpy_s(prezime, n2 + 1, stari.prezime);
50    }
51    return *this;
52 }
53
54 Student::~Student()
55 {
56    cout << "[Student] destuktor" << endl;
57    delete[] ime; ime = 0;
58    delete[] prezime; prezime = 0;
59    ukupno--;
60 }
61
62 Student &Student::operator++()
63 {
64    gd_studija += 1;
65    return *this;
66 }
67
68 Student Student::operator++(int)
69 {
70    Student temp(*this);
71    temp.gd_studija += 1;
72    return temp;
73 }
74
75 void Pretrazi(Student *grupa, int broj)
76 {
77    Student temp[5];
78    for(int i = 0; i < 5; i++)
79    {
80       temp[i] = Student("Nema", "studenata", 2003, i+1);
81    }
82    for(int j = 0; j < broj; j++)
83    {
84       if(grupa[j].gd_studija == 1)
85       {
86          if(grupa[j].gd_upisa < temp[0].gd_upisa) temp[0] = grupa[j];
87       }
88       else if(grupa[j].gd_studija == 2)
89       {
90          if(grupa[j].gd_upisa < temp[1].gd_upisa) temp[1] = grupa[j];
91       }
92       else if(grupa[j].gd_studija == 3)
93       {
```

```
94            if(grupa[j].gd_upisa < temp[2].gd_upisa) temp[2] = grupa[j];
95         }
96         else if(grupa[j].gd_studija == 4)
97         {
98            if(grupa[j].gd_upisa < temp[3].gd_upisa) temp[3] = grupa[j];
99         }
100        else
101        {
102           if(grupa[j].gd_upisa < temp[4].gd_upisa) temp[4] = grupa[j];
103        }
104     }
105     for(int i = 0; i < 5; i++)
106     {
107        cout << "Student koji najduze studira na " << temp[i].gd_studija << " -oj ↩
               godini se zove";
108        cout << temp[i].ime << " " << temp[i].prezime;
109        if(temp[i].gd_upisa != 2003)
110        {
111           cout << " i upisao je " << temp[i].gd_upisa << endl;
112        }
113     }
114  }
```

Student/Student.cpp

```
1  #include "Student.h"
2
3  int main()
4  {
5     Student ps[5];
6     int upis;
7     int gd;
8     char pom1[20], pom2[20];
9     ps[0] = Student("Slavica", "Petranovic", 2001, 1);
10    cout << "Unesite podatke za 5 studenata: godinu upisa, godinu na kojoj je";
11    cout << " ime i prezime" << endl;
12    for (int i = 1; i < 5; i++)
13    {
14       cin >> upis >> gd >> pom1 >> pom2;
15       ps[i] = Student(pom1, pom2, upis, gd);
16    }
17    Student a("Janko", "Marjanovic", 122, 12);
18    Student b;
19    b = a;
20    b.citaj();
21    Pretrazi(ps, 5);
22    for(int i = 0; i < 5; i++)
23    {
24       ps[i].citaj();
25    }
26
27    return 0;
28 }
```

Student/Source.cpp