Programowanie

Dariusz Wardecki, wyk. XI

Powtórzenie

Znajdź błąd

```
class A{
  int x, y;
  public:
  void setValueX();
  void setValueY();
};
...
int main(){
  A obj;
  obj.x = 3;
  obj.setValueY(3);
  ...
}
```

Powtórzenie

Znajdź błąd

```
class Complex{
  double re, im;
  public:
  void setValueRe(double);
  void setValueIm(double);
};
...

int main(){
  Complex z1, z2, z3;
  z1.setValueRe(4.5);
  z2.setValueIm(3.);
  z3 = z1 + z2;
  cout << z3 << endl;
  return 0;
}
```

Konstruktor

```
class Complex{
  double re, im;
  public:
   void set_values(double, double);
   double modul();
};
double Complex::modul(){
    return (sqrt(re*re + im*im));
}
void Complex::set_values(double a, double b){
    re = a;
    im = b;
}

int main(){
  Complex z;
  z.set_values(3,4);
  cout << "Modul wynosi: " << z.modul() << endl;
  return 0;
}
```

Konstruktor

```
class Complex{
  double re, im;
  public:
   Complex(double, double);
   double modul();
};
double Complex::modul(){
    return (sqrt(re*re + im*im));
}
Complex::Complex(double a, double b){
    re = a;
    im = b;
}

int main(){
  Complex z(3,4);

  cout << "Modul wynosi: " << z.modul() << endl;
  return 0;
}
```

Konstruktor

Przeładowanie kostruktora

```
class Complex{
    ...
   Complex(double, double);
   Complex(double);
   Complex();
};
...
Complex::Complex(double a, double b){
     re = a;
     im = b;
}
Complex::Complex(double a){
     re = a;
     im = 0;
}
Complex::Complex(){
     re = 0;
     im = 0;
}

int main(){
    Complex z1(3,4);
    Complex z2(4);
    Complex z3;
...
}
```

Konstruktor

Lista inicjalizacyjna

```
class Complex{
   double re, im;
   public:
    Complex(double, double);
    Complex(double);
    Complex();
};
...
Complex::Complex(double a, double b):re(a), im(b) {}
Complex::Complex(double a):re(a), im(0) {}
Complex::Complex():re(0), im(0) {}

int main(){
    Complex z1(3,4);
    Complex z2(4);
    Complex z3; // z3() zle!
...
}
```

Destruktor

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

class CRectangle {
    int *width, *height;
  public:
    CRectangle (int,int); //konstruktor
    ~CRectangle (); // destruktor
    int area () {return (*width * *height);}
};

CRectangle::CRectangle (int a, int b) {
  width = new int;
  height = new int;
  *width = a;
  *height = b;
}

CRectangle::~CRectangle () {
  delete width;
  delete height;
}

int main () {
  CRectangle rect (3,4), rectb (5,6);
  cout << "rect area: " << rect.area() << endl;
  cout << "rectb area: " << rectb.area() << endl;
  return 0;
}
```

Przeładowanie operatorów

```
class Complex{
  double re, im;
  public:
  void setValueRe(double);
  void setValueIm(double);
};
...

int main(){
  Complex z1, z2, z3;
  z1.setValueRe(4.5);
  z2.setValueIm(3.);
  z3 = z1 + z2; //blad !!!
  cout << z3 << endl; //blad !!!
  return 0;
}
```

Przeładowanie operatorów

```
class Complex{
   double re, im;
   public:
   void setValueRe(double);
   void setValueIm(double);
   Complex add(Complex);
};
...
Complex Complex::add(Complex parm)
{
   Complex z;
   z.re = re + parm.re;
   z.im = im + parm.im;
   return (z);
}
int main(){
   Complex z1, z2, z3;
   z1.setValueRe(4.5);
   z2.setValueIm(3.);
   z3 = z1.add(z2);
   ...
   return 0;
}
```

```
class Complex{
   double re, im;
   public:
   void setValueRe(double);
   void setValueIm(double);
   Complex operator+(Complex);
};
...
Complex Complex::operator+(Complex parm)
{
   Complex z;
   z.re = re + parm.re;
   z.im = im + parm.im;
   return (z);
}
int main(){
   Complex z1, z2, z3;
   z1.setValueRe(4.5);
   z2.setValueIm(3.);
   z3 = z1 + z2;
   ...
   return 0;
}
```