|  |  |
| --- | --- |
|  | |
| МИНОБРНАУКИ РОССИИ | |
| Федеральное государственное бюджетное образовательное учреждениевысшего образования"Московский технологический университет"МИРЭА | |
| Институт информационных технологий (ИТ) | |
| Кафедра инструментального и прикладного программного обеспечения (ИППО) | |

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ЛАБОРАТОРНЫМ РАБОТАМ** | |
| **по дисциплине** | |
| **«Объектно-ориентированное программирование»** | |
|  | |
| Выполнил студент группы ИКБО-13-17 | *Хитров Н.О.* |
| Принял ассистент кафедры ИиППО | *Синицын А.В.* |

|  |  |  |
| --- | --- | --- |
| Лабораторные работы выполнены | «\_\_»\_\_\_\_\_\_\_201\_\_ г. |  |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_201\_\_ г. |  |

Москва 2018

**Оглавление**

[Лабораторная работа №3-5 3](#_Toc510360725)

[Цели лабораторных работ: 3](#_Toc510360726)

[Работа №3: 3](#_Toc510360727)

[Работа №4: 3](#_Toc510360728)

[Работа №5: 3](#_Toc510360729)

[Текст задания: 3](#_Toc510360730)

[Работа №3: 3](#_Toc510360731)

[Работа №4: 3](#_Toc510360732)

[Работа №5: 4](#_Toc510360733)

[UML-диаграмма: 4](#_Toc510360734)

[Код программы: 5](#_Toc510360735)

[Файл main.cpp: 6](#_Toc510360736)

[Файл Child.h: 6](#_Toc510360737)

[Файл Child.cpp: 7](#_Toc510360738)

[Файл Tiles.h: 7](#_Toc510360739)

[Файл Tiles.cpp: 8](#_Toc510360740)

[Файл Complex.h: 8](#_Toc510360741)

[Файл Complex.cpp: 9](#_Toc510360742)

[Файл Vector.h: 11](#_Toc510360743)

[Файл Vector.cpp: 11](#_Toc510360744)

[Скриншоты выполнения программы: 13](#_Toc510360745)

[Выводы: 13](#_Toc510360746)

# Лабораторная работа №3-5

Цели лабораторных работ:

Работа №3:

1. Целью данной лабораторной работы является знакомство с классами в языке С++.

Работа №4:

1. Целью данной лабораторной работы является знакомство с конструкторами и деструкторами в языке С++.

Работа №5:

1. Целью данной лабораторной работы является изучение перегрузки операторов в языке С++ и использование перегруженных операторов на практике.

Текст задания:

Работа №3:

1. Определить класс Child, который содержит такие поля (члены класса): закрытые — имя ребенка, фамилию и возраст, публичные — методы ввода данных и отображения их на экран. Объявить два объекта класса, внести данные и показать их.

2. Создать класс Tiles (кафель), который будет содержать поля с открытым доступом: brand, size\_h, size\_w, price и метод класса getData(). В главной функции объявить пару объектов класса и внести данные в поля. Затем отобразить их, вызвав метод getData().

3. Создать класс Complex, в котором реализовано комплексное число. В данном классе должны присутствовать методы, позволяющие рассчитать и вывеси модуль и аргументы данного числа.

4. Реализовать класс Vector, позволяющий хранить в себе математический вектор. В классе должно присутствовать метод позволяющей получить модуль вектора и методы, позволяющие складывать и вычитать разные векторы.

Работа №4:

Из прошлой лабораторной работы дополнить всем видами конструкторов и деструкторами классы:

1. Complex

2. Vector

3. Tiles

4. Child

Работа №5:

1. Для класса Complex перегрузить операторы присваивания, инкремента, декремента, сравнения, ввода и вывода.

2. Для класса Vector перегрузить операторы присваивания, сравнения, ввода и вывода.

UML-диаграмма:

|  |
| --- |
| Child |
| -string firstName;  -string secondName;  -int age; |
| +Child(string firstName, string secondName, int age);  +Child();  +~Child();  +void setFirstName(string name);  +void setSecondName(string name);  +void setAge(int age);  +void getInfo(); |

|  |
| --- |
| Tiles |
| +string brand;  +int size\_h;  +int size\_w;  +int price; |
| +Tiles(string brand, int size\_h, int size\_w, int price);  +Tiles();  +~Tiles();  +void getData(); |

|  |
| --- |
| Vector |
| +double \*values;  +int size; |
| +Vector(const double \*values, int size);  +Vector();  +~Vector();  +Vector &operator=(const Vector &v);  +Vector operator+(const Vector &vector);  +Vector operator-(const Vector &vector);  +double absVector();  +friend bool operator>(Vector& vl, Vector& vr);  +friend bool operator<(Vector& vl, Vector& vr);  +friend ostream &operator<<(ostream &out, Vector &v);  +friend istream &operator>>(istream &in, Vector &v); |

|  |
| --- |
| Complex |
| -double re;  -double im; |
| +Complex(double r, double i);  +Complex();  +~Complex();  +double const absComplex();  +Complex &operator=(const Complex &c);  +Complex &operator+=(Complex &c);  +Complex operator+(const Complex &c);  +Complex operator-(const Complex &c);  +Complex operator\*(const Complex &c);  +Complex operator/(const Complex &c);  +friend Complex operator++(Complex &c);  +friend Complex operator++(Complex &c, int);  +friend const Complex operator--(Complex &c, int);  +friend const Complex operator--(Complex &c);  +friend bool operator>(Complex& cl, Complex& cr);  +friend bool operator<(Complex& cl, Complex& cr);  +friend ostream &operator<<(ostream &, Complex &);  +friend istream &operator>>(istream &, Complex &);  +friend Complex operator++(Complex &c);  +friend Complex operator++(Complex &c, int);  +friend const Complex operator--(Complex &c, int);  +friend const Complex operator--(Complex &c);  +friend bool operator>(Complex& cl, Complex& cr);  +friend bool operator<(Complex& cl, Complex& cr);  +friend ostream &operator<<(ostream &, Complex &);  +friend istream &operator>>(istream &, Complex &); |

Код программы:

### Файл main.cpp:

#include "Child.h"  
#include "Tiles.h"  
#include "Complex.h"  
#include "Vector.h"  
  
**int** main() {  
 //#1  
 Child child\_1("Izya", "Abramovich", 11), child\_2;  
 child\_2.setFirstName("Kuz'ma");  
 child\_2.setSecondName("Domovoi");  
 child\_2.setAge(12);  
  
 child\_1.getInfo();  
 cout << endl;  
 child\_2.getInfo();  
 cout << endl;  
 //end #1  
  
 //#2  
 Tiles tiles1("Aurora", 100, 120, 2534);  
 Tiles tiles2("Aphina", 200, 250, 5760);  
 cout << endl;  
 tiles2.getData();  
 //end #2  
  
 //#3  
 Complex a(5, 2);  
 Complex b(3, -3);  
 cout << "Operator < and >: " << (a < b) << " " << (a > b) << endl;  
 cout << "a = " << a << ";\nb =" << b << endl;  
 a += b;  
 cout << "a = " << a << ";\nb =" << b << endl;  
 a++, b++, --a, --b;  
 cout << "a = " << a << ";\nb =" << b << endl;  
 //end #3  
  
 //#4  
 **double** temp[] = {100, 100, 100};  
 Vector vector(temp, 3);  
 cout << vector.absVector() << endl;

cout<<"Enter vector: ";  
 cin>>vector;  
 Vector vector2(temp, 3);  
 vector2=vector;  
 cout<<vector2;  
 **return** 0;  
}

### Файл Child.h:

#ifndef LAB3\_CHILD\_H  
#define LAB3\_CHILD\_H  
  
#include <iostream>  
**using namespace** std;  
  
**class** Child {  
**private**:  
 string firstName;  
 string secondName;  
 **int** age;  
**public**:  
 Child(string firstName, string secondName, **int** age);  
  
 Child();  
  
 ~Child();  
  
 **void** setFirstName(string name);  
  
 **void** setSecondName(string name);  
  
 **void** setAge(**int** age);  
  
 **void** getInfo();  
};  
  
  
#endif //LAB3\_CHILD\_H

### Файл Child.cpp:

#include "Child.h"

Child::Child(string firstName, string secondName, **int** age) {  
 **this**->firstName = firstName;  
 **this**->secondName = secondName;  
 **this**->age = age;  
}  
  
Child::Child() {  
 **this**->firstName = "";  
 **this**->secondName = "";  
 **this**->age = 0;  
}  
  
Child::~Child() = **default**;  
  
**void** Child::setFirstName(string name) {  
 firstName = name;  
}  
  
**void** Child::setSecondName(string name) {  
 secondName = name;  
}  
  
**void** Child::setAge(**int** age) {  
 **this**->age = age > 0 ? age : 0;  
}  
  
**void** Child::getInfo() {  
 cout << "First Name: " << firstName << endl <<  
 "Second Name: " << secondName << endl <<  
 "Age: " << age << endl;  
}

### Файл Tiles.h:

#ifndef LAB3\_TILES\_H  
#define LAB3\_TILES\_H  
  
#include <iostream>  
**using namespace** std;  
  
**class** Tiles {  
**public**:  
 string brand;  
 **int** size\_h;  
 **int** size\_w;  
 **int** price;  
  
 Tiles(string brand, **int** size\_h, **int** size\_w, **int** price);  
  
 Tiles();  
  
 ~Tiles();  
  
 **void** getData();  
};  
  
  
#endif //LAB3\_TILES\_H

### Файл Tiles.cpp:

#include "Tiles.h"  
  
Tiles::Tiles(string brand, **int** size\_h, **int** size\_w, **int** price) {  
 **this**->brand = brand;  
 **this**->size\_h = size\_h;  
 **this**->size\_w = size\_w;  
 **this**->price = price;  
}  
Tiles::Tiles(){  
 **this**->brand = "";  
 **this**->size\_h = 0;  
 **this**->size\_w = 0;  
 **this**->price = 0;  
}  
Tiles::~Tiles() = **default**;  
  
**void** Tiles::getData() {  
 cout <<  
 "brand: " << brand << endl <<  
 "size\_h: " << size\_h << endl <<  
 "size\_w: " << size\_w << endl <<  
 "price: " << price << endl;  
}

### Файл Complex.h:

#ifndef LAB3\_COMPLEX\_H  
#define LAB3\_COMPLEX\_H  
  
#include <iostream>  
  
**using namespace** std;  
  
**class** Complex {  
**private**:  
 **double** re, im;  
  
**public**:  
 Complex(**double** r, **double** i);  
  
 Complex();  
  
 ~Complex();  
  
 **double const** absComplex();  
  
 Complex &**operator**=(**const** Complex &c);  
  
 Complex &**operator**+=(Complex &c);  
  
 Complex **operator**+(**const** Complex &c);  
  
 Complex **operator**-(**const** Complex &c);  
  
 Complex **operator**\*(**const** Complex &c);  
  
 Complex **operator**/(**const** Complex &c);  
  
 **friend** Complex **operator**++(Complex &c);  
  
 **friend** Complex **operator**++(Complex &c, **int**);  
  
 **friend const** Complex **operator**--(Complex &c, **int**);  
  
 **friend const** Complex **operator**--(Complex &c);  
  
 **friend bool operator**>(Complex& cl, Complex& cr);  
  
 **friend bool operator**<(Complex& cl, Complex& cr);  
  
 **friend** ostream &**operator**<<(ostream &, Complex &);  
  
 **friend** istream &**operator**>>(istream &, Complex &);  
};  
  
#endif //LAB3\_COMPLEX\_H

### Файл Complex.cpp:

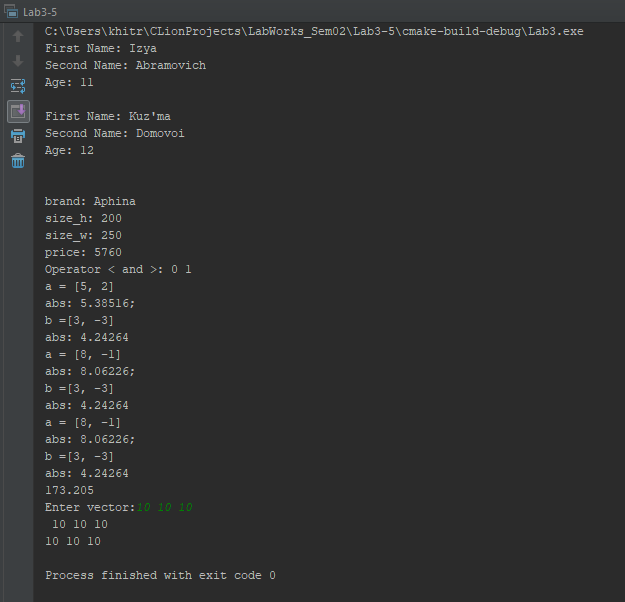
#include <cmath>  
#include "Complex.h"  
  
Complex::Complex(**double** r, **double** i) {  
 re = r;  
 im = i;  
}  
Complex::Complex(){  
 re = 0;  
 im = 0;  
}  
Complex::~Complex() = **default**;  
  
**double const** Complex::absComplex() {  
 **return** sqrt(re \* re + im \* im);  
}  
  
Complex &Complex::**operator**=(**const** Complex &c) {  
 re = c.re;  
 im = c.im;  
 **return** \***this**;  
}  
  
Complex &Complex::**operator**+=(Complex &c) {  
 re += c.re;  
 im += c.im;  
 **return** \***this**;  
}  
  
Complex Complex::**operator**+(**const** Complex &c) {  
 **return** Complex(re + c.re, im + c.im);  
}  
  
Complex Complex::**operator**-(**const** Complex &c) {  
 **return** Complex(re - c.re, im - c.im);  
}  
  
Complex Complex::**operator**\*(**const** Complex &c) {  
 **return** Complex(re \* c.re - im \* c.im, re \* c.im + im \* c.re);  
}  
  
Complex Complex::**operator**/(**const** Complex &c) {  
 Complex temp;  
  
 **double** r = c.re \* c.re + c.im \* c.im;  
 temp.re = (re \* c.re + im \* c.im) / r;  
 temp.im = (im \* c.re - re \* c.im) / r;  
  
 **return** temp;  
}  
  
Complex **operator**++(Complex& c) {  
 c.re++;  
 **return** c;  
}  
  
Complex **operator**++(Complex& c, **int**) {  
 Complex old = c;  
 c.re++;  
 **return** old;  
}  
**const** Complex **operator**--(Complex& c, **int**){  
 Complex old = c;  
 c.re--;  
 **return** old;  
}  
  
**const** Complex **operator**--(Complex& c){  
 c.re--;  
 **return** c;  
}  
  
**bool operator**>(Complex& cl, Complex& cr){  
 **return** cl.absComplex() > cr.absComplex();  
}  
  
**bool operator**<(Complex& cl, Complex& cr){  
 **return** cl.absComplex() < cr.absComplex();  
}  
  
ostream &**operator**<<(ostream &out, Complex &c) {  
 out << "[" << c.re << ", " << c.im << "]" << endl << "abs: " << c.absComplex();  
 **return** out;  
}  
  
istream &**operator**>>(istream &in, Complex &c) {  
 in >> c.re >> c.im;  
 **return** in;  
}

### Файл Vector.h:

#ifndef LAB3\_VECTOR\_H  
#define LAB3\_VECTOR\_H  
  
#include <iostream>  
**using namespace** std;  
  
**class** Vector {  
**public**:  
 **double** \*values;  
 **int** size;  
  
 Vector(**const double** \*values, **int** size);  
  
 Vector();  
  
 ~Vector();  
  
 Vector &**operator**=(**const** Vector &v);  
  
 Vector **operator**+(**const** Vector &vector);  
  
 Vector **operator**-(**const** Vector &vector);  
  
 **double** absVector();  
  
 **friend bool operator**>(Vector& vl, Vector& vr);  
  
 **friend bool operator**<(Vector& vl, Vector& vr);  
  
 **friend** ostream &**operator**<<(ostream &out, Vector &v);  
  
 **friend** istream &**operator**>>(istream &in, Vector &v);  
};  
  
#endif //LAB3\_VECTOR\_H

### Файл Vector.cpp:

#include "Vector.h"  
#include <cmath>  
  
Vector::Vector(**const double** \*values, **int** size) {  
 **this**->size = size;  
 **this**->values = **new double**[size];  
  
 **for** (**int** i = 0; i < **this**->size; ++i)  
 **this**->values[i] = values[i];  
}  
Vector::Vector(){  
 size = 0;  
 values = **nullptr**;  
}  
Vector::~Vector() {  
 **delete**[] values;  
}  
Vector &Vector::**operator**=(**const** Vector &v){  
 **for** (**int** i = 0; i < size; ++i) {  
 values[i] = v.values[i];  
 }  
 **return** \***this**;  
}  
Vector Vector::**operator**+(**const** Vector &vector) {  
 Vector temp;  
 temp.size = size;  
 temp.values = **new double**[size];  
  
 **for** (**int** i = 0; i < size; ++i) {  
 temp.values[i] = values[i] + vector.values[i];  
 }  
 **return** temp;  
}  
  
Vector Vector::**operator**-(**const** Vector &vector) {  
 Vector temp;  
 temp.size = size;  
 temp.values = **new double**[size];  
  
 **for** (**int** i = 0; i < size; ++i) {  
 temp.values[i] = values[i] - vector.values[i];  
 }  
 **return** temp;  
}  
  
**double** Vector::absVector(){  
 **double** sum=0;  
 **for** (**int** i = 0; i < size; ++i) {  
 sum+= pow(values[i], 2);  
 }  
 **return** sqrt(sum);  
}  
  
**bool operator**>(Vector& vl, Vector& vr){  
 **return** vl.absVector() > vr.absVector();  
}  
  
**bool operator**<(Vector& vl, Vector& vr){  
 **return** vl.absVector() < vr.absVector();  
}  
  
ostream &**operator**<<(ostream &out, Vector &v) {  
 **for** (**int** i = 0; i < v.size; ++i) {  
 cout<< v.values[i]<<" ";  
 }  
 cout<<endl;  
 **return** out;  
}  
  
istream &**operator**>>(istream &in, Vector &v) {  
 **for** (**int** i = 0; i < v.size; ++i) {  
 in >> v.values[i];  
 }  
 **return** in;  
}

Скриншоты выполнения программы:

Выводы:

1. Познакомились с классами в языке С++.
2. Познакомились с конструкторами и деструкторами в языке С++.
3. Изучили перегрузки операторов в языке С++ и научились использовать перегруженные операторы на практике.