|  |
| --- |
| C:\Users\Полина\Desktop\Desktop_180531_2043.jpg |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** |
|  |

|  |
| --- |
| **Институт информационных технологий (ИИТ)** |
| **Кафедра инструментального и прикладного программного обеспечения (ИиППО)** |

**Отчет по лабораторным работам**

**По дисциплине**

**«Объектное - ориентированное программирование»**

|  |  |  |
| --- | --- | --- |
| Выполнила студент  Группы ИКБО-13-18 | Тимошкин М.С. | (подпись) |
| Принял ассистент кафедры ИППО | Иванов М. Е. | (подпись) |
| Работа представлена | «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_201\_г. |  |
| Зачтено | «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_201\_г. |  |

Москва 2018

**Лабораторные занятия**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| №/р. | страница в отчете. | Студент выполнил:  (подпись) | Преподаватель принял:  (подпись) | Балл  (максимально возможный) | Балл (Фактический) |
| 1. | *4-10 с.* |  |  | 2 |  |
| 2. | *11-13 с.* |  |  | 2 |  |
| 3. | *14-18 с.* |  |  | 4 |  |
| 4. | *19-24 с.* |  |  | 6 |  |
| 5. | *25-30 с.* |  |  | 6 |  |
| 6. | *31-39 с.* |  |  | 8 |  |
| 7. | *40-42 с.* |  |  | 8 |  |
| 8. | *43-45 с.* |  |  | 8 |  |
|  |  |  | Итоговый балл: | 44 |  |

Оглавление

[Лабораторная работа №1 4](#_Toc5720141)

[Лабораторная работа №2 11](#_Toc5720142)

[Лабораторная работа №3 14](#_Toc5720143)

[Лабораторная работа №4 19](#_Toc5720144)

[Лабораторная работа №5 25](#_Toc5720145)

[Лабораторная работа №6 31](#_Toc5720146)

[Лабораторная работа №7 40](#_Toc5720147)

[Лабораторная работа №8 43](#_Toc5720148)

# Лабораторная работа №1

**Тема: «Создание многофайловых проектов»**

**Цель лабораторной работы**

Целью данной лабораторной работы освоить на практике создание многофайловых проектов на языке Си/Си++, познакомиться с директивами условной компиляции.

**Задание**

1. Напишите программу-калькулятор комплексных чисел. Для реализации необходимо разработать абстрактный тип данных ­ комплексное число. Программа должна реализовывать арифметические операции над комплексными числами. Программа должна быть представлена в виде многофайлового проекта, все прототипы функций, объявления структур должны быть вынесены в заголовочный файл с соответствующим названием. Всего должно быть три файла: файл с объявлениями, файл реализации и файл с функцией main(), демонстрирующий работу с новым типом данных. Файл с объявлениями должен называться Complex.h, файл с реализацией функций должен называться Complex.c, файл с функцией main() может называться main.c Программа должна обеспечивать удобный интерфейс пользователя для работы с ней.

2. На основе разработанного в предыдущем задании типа данных комплексное число написать программу, которая считывает информацию из файла complex.txt - количество комплексных чисел в переменную n, а сами комплексные числа в массив р. Затем происходит поиск комплексного числа с максимальным модулем в массиве р.

**Код программы**

**1 задание.**

**complex.h**

#ifndef complex\_h  
#define complex\_h  
#include <iostream>  
#include <cmath>  
using namespace std;  
struct Complex {  
private:  
 double re, im;  
public:  
 void init(double r, double i);  
 void add(Complex c1, Complex c2);  
 void subtract(Complex c1, Complex c2);  
 void multiply(Complex c1, Complex c2);  
 void divide(Complex c1, Complex c2);  
 int check();  
 void output();  
};  
#endif // complex\_h

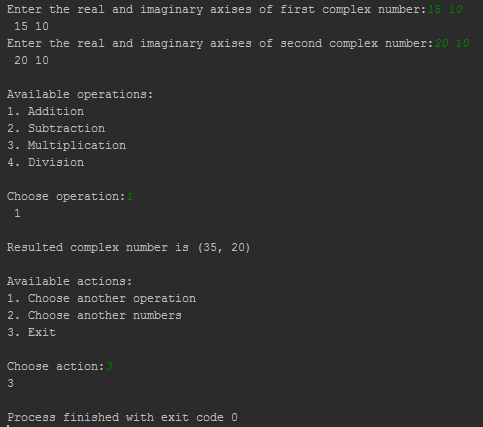
**complex.cpp**

#include "complex.h"  
void Complex::init(double r, double i) {  
 re = r;  
 im = i;  
}  
void Complex::multiply(Complex c1, Complex c2) {  
 re = c1.re \* c2.re - c1.im \* c2.im;  
 im = c2.re \* c1.im + c1.re \* c2.im;  
}  
void Complex::add(Complex c1, Complex c2) {  
 re = c1.re + c2.re;  
 im = c1.im + c2.im;  
}  
void Complex::subtract(Complex c1, Complex c2) {  
 re = c1.re - c2.re;  
 im = c1.im - c2.im;  
}  
void Complex::divide(Complex c1, Complex c2) {  
 re = (c1.re \* c2.re + c1.im \* c2.im) / (pow(c2.re, 2) + pow(c2.im, 2));  
 im = (c1.im \* c2.re - c1.re \* c2.im) / (pow(c2.re, 2) + pow(c2.im, 2));  
}  
  
int Complex::check() {  
 if ((pow(re, 2) + pow(im, 2)) == 0)  
 {  
 cout << endl << "Incorrect second complex number!" << endl;  
 return 0;  
 }  
 return 1;  
}  
void Complex::output() {  
 cout << endl << "Resulted complex number is " << "(" << re << ", " << im << ")" << endl;  
}

**main.cpp**

#include "complex.h"  
int main() {  
 bool end = true;  
 bool changeNumber = true;  
 Complex c1, c2, c3;  
 double x, y;  
 int choice;  
 while (end) {  
 if (changeNumber) {  
 cout << "Enter the real and imaginary axises of first complex number: ";  
 cin >> x >> y;  
 c1.init(x, y);  
 cout << "Enter the real and imaginary axises of second complex number: ";  
 cin >> x >> y;  
 c2.init(x, y);  
 }  
 do {  
 cout << endl << "Available operations:\n1. Addition\n2. Subtraction\n3. Multiplication\n4. Division\n\nChoose operation: ";  
 cin >> choice;  
 if (choice > 4 | choice < 1)  
 cout << endl << "Wrong number! Try again:\n";  
 } while (choice > 4 | choice < 1);  
 switch (choice) {  
 case 1:  
 c3.add(c1, c2);  
 c3.output();  
 break;  
 case 2:  
 c3.subtract(c1, c2);  
 c3.output();  
 break;  
 case 3:  
 c3.multiply(c1, c2);  
 c3.output();  
 break;  
 case 4:  
 if (c2.check())  
 {  
 c3.divide(c1, c2);  
 c3.output();  
 }  
 break;  
 }  
 do {  
 cout << endl << "Available actions:\n1. Choose another operation\n2. Choose another numbers\n3. Exit\n\nChoose action: ";  
 cin >> choice;  
 if (choice > 3 | choice < 1)  
 cout << endl << "Wrong number! Try again:\n";  
 } while (choice > 3 | choice < 1);  
 switch (choice) {  
 case 1:  
 changeNumber = false;  
 break;  
 case 2:  
 changeNumber = true;  
 break;  
 case 3:  
 end = false;  
 break;  
 }  
 }  
 return 0;  
}

**Результат работы программы**



**Рис. 1.1 - Результат работы программы**

**2 задание.**

**complex.h**

#ifndef complex\_h  
#define complex\_h  
#include <iostream>  
#include <cmath>  
using namespace std;  
struct Complex {  
private:  
 double re, im;  
public:  
 double abs;  
 void init(double r, double i);  
 void add(Complex c1, Complex c2);  
 void subtract(Complex c1, Complex c2);  
 void multiply(Complex c1, Complex c2);  
 void divide(Complex c1, Complex c2);  
 int check();  
 void output();  
};  
#endif // complex\_h

**complex.cpp**

#include "complex.h"  
void Complex::init(double r, double i) {  
 re = r;  
 im = i;  
 abs = sqrt(re\*re+im\*im);  
}  
void Complex::multiply(Complex c1, Complex c2) {  
 re = c1.re \* c2.re - c1.im \* c2.im;  
 im = c2.re \* c1.im + c1.re \* c2.im;  
}  
void Complex::add(Complex c1, Complex c2) {  
 re = c1.re + c2.re;  
 im = c1.im + c2.im;  
}  
void Complex::subtract(Complex c1, Complex c2) {  
 re = c1.re - c2.re;  
 im = c1.im - c2.im;  
}  
void Complex::divide(Complex c1, Complex c2) {  
 re = (c1.re \* c2.re + c1.im \* c2.im) / (pow(c2.re, 2) + pow(c2.im, 2));  
 im = (c1.im \* c2.re - c1.re \* c2.im) / (pow(c2.re, 2) + pow(c2.im, 2));  
}  
int Complex::check() {  
 if ((pow(re, 2) + pow(im, 2)) == 0)  
 {  
 cout << endl << "Incorrect second complex number!" << endl;  
 return 0;  
 }  
 return 1;  
}  
void Complex::output() {  
 cout << endl << "Resulted complex number is " << "(" << re << ", " << im << ")" << endl;  
}

**Complex.txt**

5 25.3-21.2i 1-69i 65.3+21i 5+2i 100-100i

**main.cpp**

#include <fstream>  
#include <iostream>  
#include <string>  
#include "complex.h"  
using namespace std;  
int main() {  
 Complex maxAbs;  
 int n;  
 double re,im;  
 fstream complex("F:\\Univer\\OOP\\laba1\\var2\\Complex.txt", ios::in);  
 complex>>n;  
 Complex p[n];  
 for(int i = 0; i < n; i++)  
 {  
 complex>>re>>im;  
 p[i].init(re,im);  
 complex.ignore(1);  
 }  
 maxAbs = p[0];  
 for(int i = 1; i< n; i++)  
 {  
 if(p[i].abs > maxAbs.abs)  
 maxAbs=p[i];  
 }  
 maxAbs.output();  
 system("pause");  
 return 0;  
}

**Результат работы программы**



**Рис. 1.2 – Результат работы программы**

**Заключение**

В ходе лабораторной работы я освоил на практике создание многофайловых проектов на языке Си/Си++, познакомился с директивами условной компиляции, а также написал программу-калькулятор комплексных чисел, программу, которая считывает информацию из файла Complex.txt

# **Лабораторная работа №2**

**Тема: «Указатели на функции»**

**Цель лабораторной работы**

Целью данной лабораторной работы освоить на практике вызов функции с использованием указателей.

**Задание**

1. Напишите программу, которая вызывает различные виды функции в зависимости от заданного условия. Можно использовать примеры выше.

2. Напишите программу с использованием некой универсальной функции, которая в качестве возвращаемого значения возвращает указатель на функцию. Эта функция в зависимости от исходного массива должна соответствующим образом преобразовать массив. Также в функции main() должен быть объявлен указатель. В теле программы указателю на функцию должно присваиваться значение. Исходные данные сгенерировать целочисленный массив из случайных чисел. Запрограммировать следующие действия в виде функций:

• если сумма элементов в массиве равна его первому элементу, то необходимо инвертировать массив

• если сумма элементов в массиве больше его первого элемента, то необходимо расположить его элементы в неубывающем порядке

• если сумма элементов массива меньше его первого элемента, то необходимо расположить его элементы в невозрастающем порядке

**Код программы**

**Задание 1.**

**main.cpp**

#include<iostream>  
using namespace std;  
int sum(int a, int b) {  
 return a + b;  
}  
  
int subtract(int a, int b) {  
 return a - b;  
}  
int main() {  
 int (\*func)(int, int);  
 int a, b, result;  
 cout << "Enter the two numbers: ";  
 cin >> a >> b;  
 if (a > b)  
 func = sum;  
 else  
 func = subtract;  
 result = func(a, b);  
 cout << "Resulter number is " << result << endl;  
 system("pause");  
 return 0;

**Результат работы программы**



**Рис 2.1 – Результат работы программы**

**Задание 2.**

#include<iostream>  
using namespace std;  
int comp1(const void \* x1, const void \* x2)  
{  
 return (\*(int\*)x1 - \*(int\*)x2);  
}  
int comp2(const void \* x1, const void \* x2)  
{  
 return (\*(int\*)x2 - \*(int\*)x1);  
}  
void func1(int \*mas)  
{  
 int temp;  
 for (int i = 0; i < 5; i++)  
 {  
 temp = mas[i];  
 mas[i] = mas[9 - i];  
 mas[9 - i] = temp;  
 }  
}  
  
void func2(int \*mas)  
{  
 qsort(mas, 10, sizeof(int), comp1);  
}  
  
void func3(int \*mas)  
{  
 qsort(mas, 10, sizeof(int), comp2);  
}  
void(\*choice(int \*mas, int sum))(int\*) {  
 void(\*func)(int\*) = NULL;  
 if (sum == mas[0]) func = func1;  
 if (sum > mas[0]) func = func2;  
 if (sum < mas[0]) func = func3;  
 return func;  
}  
int main() {  
 int sum = 0;  
 int mas[10];  
 void(\*func)(int\*);  
 for (int i = 0; i < 10; i++)  
 {  
 mas[i] = rand();  
 sum += mas[i];  
 }  
 func = choice(mas, sum);  
 func(mas);  
 for (int i = 0; i < 10; i++)  
 cout << mas[i] << " ";  
 system("pause");  
 return 0;  
  
   
}

**Результат работы программы**



**Рис 2.2 – Результат работы программы**

**Заключение**

В ходе лабораторной работы я освоил на практике вызов функций с использованием указателей.

# Лабораторная работа №3

**Тема: «Классы»**

**Цель лабораторной работы**

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

**Задание**

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.

**Код программы**

**Child.h**

#ifndef child\_h  
#define child\_h  
#include<iostream>  
#include<string>  
#include<math.h>  
using namespace std;  
class Child {  
private:  
 string name;  
 string family;  
 int age;  
public:  
 Child() {  
 cout << "Enter the name of the child: ";  
 cin >> name;  
 cout << "Enter the family of the child: ";  
 cin >> family;  
 cout << "Enter the age of the child: ";  
 cin >> age;  
 }  
 void outputData() {  
 cout << "The name of the child is " << name << endl;  
 cout << "The family of the child is " << family << endl;  
 cout << "The age of the child: " << age << endl;  
 }  
};  
#endif

**Complex.h**

#ifndef complex\_h  
#define complex\_h  
#include<iostream>  
#include<string>  
#include<math.h>  
using namespace std;  
class Complex {  
private:  
 float re, im;  
 float abs, arg;  
public:  
 Complex() {  
 cout << "Enter the real axis of number: ";  
 cin >> re;  
 cout << "Enter the imaginary axis of number: ";  
 cin >> im;  
 abs = sqrt(re\*re + im \* im);  
 arg = atan(im / re);  
 }  
 void getComplex() {  
 cout << "The module of complex number is " << abs << endl;  
 cout << "The argumebt of complex number is " << arg << endl;  
 }  
};  
#endif

**Tiles.h**

#ifndef tiles\_h  
#define tiles\_h  
#include<iostream>  
#include<string>  
#include<math.h>  
using namespace std;  
class Tiles {  
public:  
 string brand;  
 int size\_h;  
 int size\_w;  
 float price;  
 void getData() {  
 cout << "The brand of tiles is " << brand << endl;  
 cout << "The height of tile is " << size\_h << endl;  
 cout << "The width of tile is " << size\_w << endl;  
 cout << "The price of tile is " << price << endl;  
 }  
};  
#endif

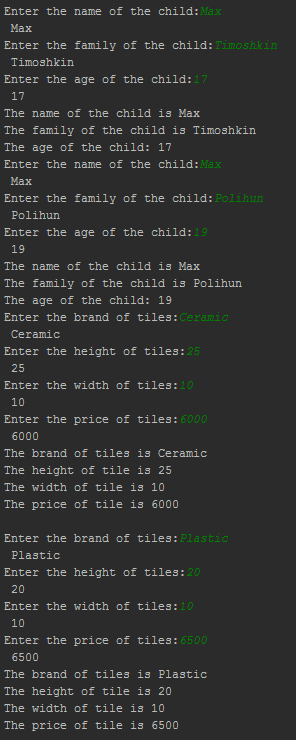
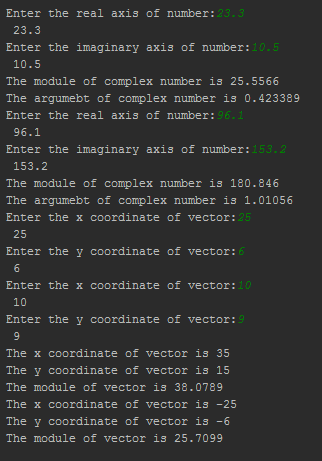
**Vector.h**

#ifndef vector\_h  
#define vector\_h  
#include<iostream>  
#include<string>  
#include<math.h>  
using namespace std;  
class Vector {  
private:  
 float x, y;  
 float abs;  
public:  
 Vector() {  
 cout << "Enter the x coordinate of vector: ";  
 cin >> x;  
 cout << "Enter the y coordinate of vector: ";  
 cin >> y;  
 }  
 void ABS() {  
 abs = sqrt(x\*x + y \* y);  
 }  
 void sum(Vector v) {  
 x += v.x;  
 y += v.y;  
 }  
  
 void sub(Vector v) {  
 x -= v.x;  
 y -= v.y;  
 }  
 void getVector() {  
 cout << "The x coordinate of vector is " << x << endl;  
 cout << "The y coordinate of vector is " << y << endl;  
 cout << "The module of vector is " << abs << endl;  
 }  
};  
#endif

**main.cpp**

#include"Child.h"  
#include"Tiles.h"  
#include"Complex.h"  
#include"Vector.h"  
int main() {  
 Child child1;  
 child1.outputData();  
 Child child2;  
 child2.outputData();  
 Tiles tiles1, tiles2;  
 cout << "Enter the brand of tiles: ";  
 cin >> tiles1.brand;  
 cout << "Enter the height of tiles: ";  
 cin >> tiles1.size\_h;  
 cout << "Enter the width of tiles: ";  
 cin >> tiles1.size\_w;  
 cout << "Enter the price of tiles: ";  
 cin >> tiles1.price;  
 tiles1.getData();  
 cout << endl;  
 cout << "Enter the brand of tiles: ";  
 cin >> tiles2.brand;  
 cout << "Enter the height of tiles: ";  
 cin >> tiles2.size\_h;  
 cout << "Enter the width of tiles: ";  
 cin >> tiles2.size\_w;  
 cout << "Enter the price of tiles: ";  
 cin >> tiles2.price;  
 tiles2.getData();  
 cout << endl;  
 Complex complex1;  
 complex1.getComplex();  
 Complex complex2;  
 complex2.getComplex();  
  
 Vector vector1;  
 Vector vector2;  
 vector1.sum(vector2);  
 vector2.sub(vector1);  
 vector1.ABS();  
 vector2.ABS();  
 vector1.getVector();  
 vector2.getVector();  
 return 0;  
}

**Результат работы программы**

**Рис. 3 – Результат работы программы**

**Заключение**

В ходе лабораторной работы я познакомился с классами в языке C++.

# Лабораторная работа №4

**Тема: «Конструкторы и деструкторы»**

**Цель лабораторной работы**

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

**Задание**

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

1. Complex

2. Vector

3. Tiles

4. Child

**Код программы**

**Child.h**

#ifndef child\_h  
#define child\_h  
#include<iostream>  
#include<string>  
#include<math.h>  
using namespace std;  
class Child {  
private:  
 string name;  
 string family;  
 int age;  
public:  
 Child() {  
 name = "Default name";  
 family = "Default family";  
 age = 0;  
 }  
 Child(string name, string family, int age)  
 {  
 this->name = name;  
 this->family = family;  
 this->age = age;  
 }  
 Child(const Child &child) :  
 name(child.name), family(child.family), age(child.age)  
 {  
 cout << "Copy constructor worked here" << endl;  
 }  
 void setData() {  
 cout << "Enter the name of the child: ";  
 cin >> name;  
 cout << "Enter the family of the child: ";  
 cin >> family;  
 cout << "Enter the age of the child: ";  
 cin >> age;  
 }  
 ~Child(){  
 cout << "Class child was destructed." << endl;  
 }  
 void outputData() {  
 cout << "The name of the child is " << name << endl;  
 cout << "The family of the child is " << family << endl;  
 cout << "The age of the child: " << age << endl;  
 }  
  
};  
  
#endif

**Complex.h**

#ifndef complex\_h  
#define complex\_h  
#include<iostream>  
#include<string>  
#include<math.h>  
using namespace std;  
class Complex {  
private:  
 float re, im;  
 float abs, arg;  
public:  
 Complex() {  
 re = 0;  
 im = 0;  
 abs = 0;  
 arg = 0;  
 }  
 Complex(float re, float im)  
 {  
 this->re = re;  
 this->im = im;  
 abs = sqrt(re\*re + im \* im);  
 arg = atan(im / re);  
 }  
 Complex(const Complex &complex) :  
 re(complex.re), im(complex.im), abs(complex.abs), arg(complex.arg)  
 {  
 cout << "Copy constructor worked here" << endl;  
 }  
 void setData() {  
 cout << "Enter the real axis of number: ";  
 cin >> re;  
 cout << "Enter the imaginary axis of number: ";  
 cin >> im;  
 abs = sqrt(re\*re + im \* im);  
 arg = atan(im / re);  
 }  
 ~Complex(){  
 cout << "Class complex was destructed." << endl;  
 }  
 void getComplex() {  
 cout << "The module of complex number is " << abs << endl;  
 cout << "The argumebt of complex number is " << arg << endl;  
 }  
};  
#endif

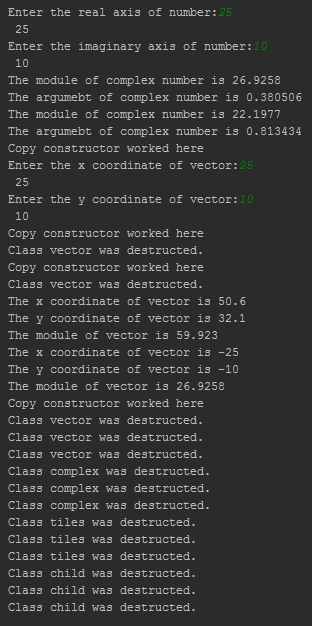
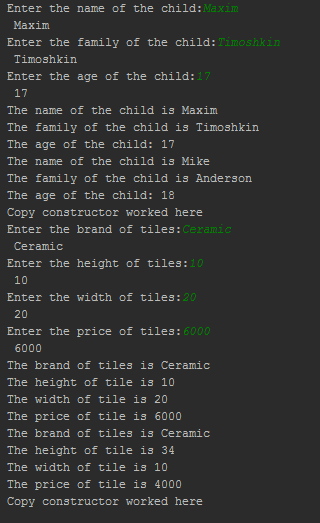
**Tileh.h**

#ifndef tiles\_h  
#define tiles\_h  
#include<iostream>  
#include<string>  
#include<math.h>  
using namespace std;  
class Tiles {  
private:  
 string brand;  
 int size\_h;  
 int size\_w;  
 float price;  
public:  
 Tiles() {  
 brand = "Default name";  
 size\_h = 0;  
 size\_w = 0;  
 price = 0;  
 }  
 Tiles(string brand, int size\_h, int size\_w, float price)  
 {  
 this->brand = brand;  
 this->size\_h = size\_h;  
 this->size\_w = size\_w;  
 this->price = price;  
 }  
 Tiles(const Tiles &tiles) :  
 brand(tiles.brand), size\_h(tiles.size\_h), size\_w(tiles.size\_w), price(tiles.price)  
 {  
 cout << "Copy constructor worked here" << endl;  
 }  
 void setData() {  
 cout << "Enter the brand of tiles: ";  
 cin >> brand;  
 cout << "Enter the height of tiles: ";  
 cin >> size\_h;  
 cout << "Enter the width of tiles: ";  
 cin >> size\_w;  
 cout << "Enter the price of tiles: ";  
 cin >> price;  
 }  
 ~Tiles(){  
 cout << "Class tiles was destructed." << endl;  
 }  
 void getData() {  
 cout << "The brand of tiles is " << brand << endl;  
 cout << "The height of tile is " << size\_h << endl;  
 cout << "The width of tile is " << size\_w << endl;  
 cout << "The price of tile is " << price << endl;  
 }  
};  
#endif

**Vector.h**

#ifndef vector\_h  
#define vector\_h  
#include<iostream>  
#include<string>  
#include<math.h>  
using namespace std;  
class Vector {  
private:  
 float x, y;  
 float abs;  
public:  
  
 Vector() {  
 x = 0;  
 y = 0;  
 abs = 0;  
 }  
 Vector(float x, float y)  
 {  
 this->x = x;  
 this->y = y;  
 abs = sqrt(x\*x + y \* y);  
 }  
 Vector(const Vector &vector) :  
 x(vector.x), y(vector.y), abs(vector.abs)  
 {  
 cout << "Copy constructor worked here" << endl;  
 }  
 void setData() {  
 cout << "Enter the x coordinate of vector: ";  
 cin >> x;  
 cout << "Enter the y coordinate of vector: ";  
 cin >> y;  
 }  
 void sum(Vector v) {  
 x += v.x;  
 y += v.y;  
 abs = sqrt(x\*x + y \* y);  
 }  
  
 void sub(Vector v) {  
 x -= v.x;  
 y -= v.y;  
 abs = sqrt(x\*x + y \* y);  
 }  
 void getVector() {  
 cout << "The x coordinate of vector is " << x << endl;  
 cout << "The y coordinate of vector is " << y << endl;  
 cout << "The module of vector is " << abs << endl;  
 }  
 ~Vector(){  
 cout << "Class vector was destructed." << endl;  
 }  
};  
#endif

**Результат программы.**



**Рис. 4 – Результат работы программы**

**Заключение**

В ходе лабораторной работы я познакомился с конструкторами и деструкторами в языке C++.

# Лабораторная работа №5

**Тема: «Перегрузка операторов в языке C++»**

**Цель лабораторной работы**

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

**Задание**

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

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

**Код программы**

**Complex.h**

#ifndef complex\_h  
#define complex\_h  
#include<iostream>  
#include<string>  
#include<math.h>  
using namespace std;  
class Complex {  
private:  
 float re, im;  
 float abs, arg;  
public:  
 Complex() {  
 re = 0;  
 im = 0;  
 abs = 0;  
 arg = 0;  
 }  
 Complex(float re, float im)  
 {  
 this->re = re;  
 this->im = im;  
 abs = sqrt(re\*re + im \* im);  
 arg = atan(im / re);  
 }  
 Complex(const Complex &complex) :  
 re(complex.re), im(complex.im), abs(complex.abs), arg(complex.arg)  
 {  
 cout << "Copy constructor worked here" << endl;  
 }  
 Complex& operator=(Complex& c1) {  
 if (this == &c1)  
 return \*this;  
 re = c1.re;  
 im = c1.im;  
 abs = c1.abs;  
 arg = c1.arg;  
 return \*this;  
 }  
 bool operator==(Complex c) {  
 return abs == c.abs;  
 }  
 bool operator !=(Complex c) {  
 return abs != c.abs;  
 }  
 bool operator >(Complex c) {  
 return abs > c.abs;  
 }  
 bool operator <(Complex c) {  
 return abs < c.abs;  
 }  
 friend ostream& operator<<(ostream& out, Complex &c) {  
 out << "The module of complex number is " << c.abs << endl;  
 out << "The argumebt of complex number is " << c.arg << endl;  
 return out;  
 }  
 friend istream& operator>>(istream& in, Complex &c) {  
 cout << "Enter the real axis of number: ";  
 in >> c.re;  
 cout << "Enter the imaginary axis of number: ";  
 in >> c.im;  
 c.abs = sqrt(c.re\*c.re + c.im \* c.im);  
 c.arg = atan(c.im / c.re);  
 return in;  
 }  
 Complex& operator++() {  
 im++;  
 re++;  
 abs = sqrt(re\*re + im \* im);  
 arg = atan(im / re);  
 return \*this;  
 }  
  
  
 Complex & operator--() {  
 im--;  
 re--;  
 abs = sqrt(re\*re + im \* im);  
 arg = atan(im / re);  
 return \*this;  
 }  
 void setData() {  
 cout << "Enter the real axis of number: ";  
 cin >> re;  
 cout << "Enter the imaginary axis of number: ";  
 cin >> im;  
 abs = sqrt(re\*re + im \* im);  
 arg = atan(im / re);  
 }  
 ~Complex(){  
 cout << "Class complex was destructed." << endl;  
 }  
 void getComplex() {  
 cout << "The module of complex number is " << abs << endl;  
 cout << "The argumebt of complex number is " << arg << endl;  
 }  
};  
  
  
#endif

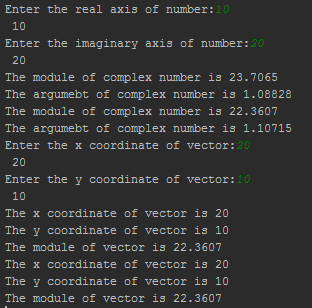
**Vector.h**

#ifndef vector\_h  
#define vector\_h  
#include<iostream>  
#include<string>  
#include<math.h>  
using namespace std;  
class Vector {  
private:  
 float x, y;  
 float abs;  
public:  
 Vector() {  
 x = 0;  
 y = 0;  
 abs = 0;  
 }  
 Vector(float x, float y)  
 {  
 this->x = x;  
 this->y = y;  
 abs = sqrt(x\*x + y \* y);  
 }  
 Vector(const Vector &vector) :  
 x(vector.x), y(vector.y), abs(vector.abs)  
 {  
 cout << "Copy constructor worked here" << endl;  
 }  
 Vector& operator=(const Vector& v1) {  
 if (this == &v1)  
 return \*this;  
 x = v1.x;  
 y = v1.y;  
 abs = v1.abs;  
 return \*this;  
 }  
 bool operator == (Vector v1) {  
 return v1.abs == abs;  
 }  
  
 bool operator!=(Vector v1) {  
 return v1.abs != abs;  
 }  
  
 bool operator >(Vector v1) {  
 return abs > v1.abs;  
 }  
 bool operator <(Vector v1) {  
 return abs < v1.abs;  
 }  
 friend ostream& operator <<(ostream &out, const Vector& i) {  
 out << "The x coordinate of vector is " << i.x << endl;  
 out << "The y coordinate of vector is " << i.y << endl;  
 out << "The module of vector is " << i.abs << endl;  
 return out;  
 }  
 friend istream& operator >>(istream &in, Vector& i) {  
 cout << "Enter the x coordinate of vector: ";  
 in >> i.x;  
 cout << "Enter the y coordinate of vector: ";  
 in >> i.y;  
 i.abs = sqrt(i.x\*i.x + i.y\*i.y);  
 return in;  
 }  
  
 void sum(Vector v) {  
 x += v.x;  
 y += v.y;  
 abs = sqrt(x\*x + y \* y);  
 }  
  
 void sub(Vector v) {  
 x -= v.x;  
 y -= v.y;  
 abs = sqrt(x\*x + y \* y);  
 }  
 ~Vector(){  
 cout << "Class vector was destructed." << endl;  
 }  
};  
#endif

**main.cpp**

#include<iostream>  
#include"Complex.h"  
#include"Vector.h"  
using namespace std;  
int main() {  
 Complex complex1;  
 Complex complex2;  
 cin >> complex1;  
 complex2 = complex1;  
 ++complex1;  
 cout << complex1;  
 cout << complex2;  
 Vector vector1;  
 Vector vector2;  
 cin >> vector1;  
 vector2 = vector1;  
 cout << vector1;  
 cout << vector2;  
 system("pause");  
 return 0;  
}

**Результат работы программы**



**Рис. 5 - Результат работы программы**

**Заключение**

В ходе лабораторной работы я изучил перегрузку операторов в языке C++ и научился использовать перегруженные операторы на практике.

# Лабораторная работа №6

**Тема: «Наследование»**

**Цель лабораторной работы**

Целью данной лабораторной работы является изучение наследованная классов в языке С++.

**Задание**

1) Создать класс «Староста», производный от класса «Студент». Новый класс должен содержать несколько дополнительных методов и полей.

2) Создать класс Alive и расширить его до Bird, Fish, Animal

3) Создать класс Animal, и расширить его до Dog, Cat.

**Код программы**

**Alive.h**

#pragma once  
#include<iostream>  
#include<string>  
using namespace std;  
class Alive{  
protected:  
 int age;  
 float speed;  
 string name;  
public:  
 void nextYear() {  
 age++;  
 }  
 void changeSpeed(float count) {  
 speed += count;  
 }  
 Alive() {  
 age = 0;  
 speed = 0;  
 name = "default";  
 }  
 friend ostream& operator<<(ostream& out, const Alive& a)  
 {  
 cout << "Age - " << a.age << endl;  
 cout << "Speed - " << a.speed << endl;  
 cout << "Name is " << a.name << endl;  
 return out;  
 }  
 friend istream& operator>>(istream& in, Alive& a)  
 {  
 cout << "Age - ";  
 in >> a.age;  
 cout << "Speed - ";  
 in >> a.speed;  
 cout << "Name is ";  
 in >> a.name;  
 return in;  
 }  
};

**Animal.h**

#pragma once  
#include<iostream>  
#include<string>  
#include"Alive.h"  
class Animal : public Alive {  
protected:  
 string kind;  
public:  
 Animal() : Alive() {  
 kind = "default";  
 }  
 friend ostream& operator<<(ostream& out, const Animal& a)  
 {  
 cout << "Age - " << a.age << endl;  
 cout << "Speed - " << a.speed << endl;  
 cout << "Name is " << a.name << endl;  
 cout << "Kind is " << a.kind << endl;  
 return out;  
 }  
 friend istream& operator>>(istream& in, Animal& a)  
 {  
 cout << "Age - ";  
 in >> a.age;  
 cout << "Speed - ";  
 in >> a.speed;  
 cout << "Name is ";  
 in >> a.name;  
 cout << "Kind is ";  
 in >> a.kind;  
 return in;  
 }  
 void changeKind(string kind) {  
 this->kind = kind;  
 }  
};

**Bird.h**

#pragma once  
#include <iostream>  
#include <string>  
#include "Alive.h"  
class Bird : public Alive   
{  
protected:  
 float flyHeight;  
public:  
 void changeFlyHeight(float height) {  
 flyHeight = height;  
 }  
 Bird() : Alive() {  
 flyHeight = 0;  
 }  
 friend ostream& operator<<(ostream& out, const Bird& b)  
 {  
 cout << "Age - " << b.age << endl;  
 cout << "Speed - " << b.speed << endl;  
 cout << "Name is " << b.name << endl;  
 cout << "Fly height is " << b.flyHeight << endl;  
 return out;  
 }  
 friend istream& operator>>(istream& in, Bird& b)  
 {  
 cout << "Age - ";  
 in >> b.age;  
 cout << "Speed - ";  
 in >> b.speed;  
 cout << "Name is ";  
 in >> b.name;  
 cout << "Fly height is ";  
 in >> b.flyHeight;  
 return in;  
 }  
};

**Cat.h**

#pragma once  
#include<iostream>  
#include<string>  
#include"Animal.h"  
class Cat : public Animal {  
protected:  
 float weight;  
public:  
 void voice() {  
 cout << "Meow" << endl;  
 }  
 Cat() : Animal() {  
 weight = 0;  
 }  
 void changeWeight(float weight) {  
 this->weight = weight;  
 }  
 friend ostream& operator<<(ostream& out, const Cat& c)  
 {  
 cout << "Age - " << c.age << endl;  
 cout << "Speed - " << c.speed << endl;  
 cout << "Name is " << c.name << endl;  
 cout << "Kind is " << c.kind << endl;  
 cout << "Weight is " << c.weight << endl;  
 return out;  
 }  
 friend istream& operator>>(istream& in, Cat& c)  
 {  
 cout << "Age - ";  
 in >> c.age;  
 cout << "Speed - ";  
 in >> c.speed;  
 cout << "Name is ";  
 in >> c.name;  
 cout << "Kind is ";  
 in >> c.kind;  
 cout << "Weight is ";  
 in >> c.weight;  
 return in;  
 }  
};

**Dog.h**

#pragma once  
#include<iostream>  
#include<string>  
#include"Animal.h"  
class Dog : public Animal {  
protected:  
 float weight;  
public:  
 void voice() {  
 cout << "Woof" << endl;  
 }  
 Dog() : Animal() {  
 weight = 0;  
 }  
 void changeWeight(float weight) {  
 this->weight = weight;  
 }  
 friend ostream& operator<<(ostream& out, const Dog& d)  
 {  
 cout << "Age - " << d.age << endl;  
 cout << "Speed - " << d.speed << endl;  
 cout << "Name is " << d.name << endl;  
 cout << "Kind is " << d.kind << endl;  
 cout << "Weight is " << d.weight << endl;  
 return out;  
 }  
 friend istream& operator>>(istream& in, Dog& d)  
 {  
 cout << "Age - ";  
 in >> d.age;  
 cout << "Speed - ";  
 in >> d.speed;  
 cout << "Name is ";  
 in >> d.name;  
 cout << "Kind is ";  
 in >> d.kind;  
 cout << "Weight is ";  
 in >> d.weight;  
 return in;  
 }  
};

**Fish.h**

#pragma once  
#include<iostream>  
#include<string>  
#include"Alive.h"  
class Fish : public Alive {  
protected:  
 float depth;  
public:  
 Fish() : Alive() {  
 depth = 0;  
 }  
 void changeDepth(float depth) {  
 this->depth = depth;  
 }  
 friend ostream& operator<<(ostream& out, const Fish& f)  
 {  
 cout << "Age - " << f.age << endl;  
 cout << "Speed - " << f.speed << endl;  
 cout << "Name is " << f.name << endl;  
 cout << "Depth is " << f.depth << endl;  
 return out;  
 }  
 friend istream& operator>>(istream& in, Fish& f)  
 {  
 cout << "Age - ";  
 in >> f.age;  
 cout << "Speed - ";  
 in >> f.speed;  
 cout << "Name is ";  
 in >> f.name;  
 cout << "Depth is ";  
 in >> f.depth;  
 return in;  
 }  
};

**Starosta.h**

#pragma once  
#include<iostream>  
#include<string>  
#include "Student.h"  
using namespace std;  
class Starosta : public Student {  
protected:  
 int duration;  
public:  
 Starosta() : Student() {  
 duration = 0;  
 }  
 void nextCourse() {  
 course++;  
 duration++;  
 }  
 friend ostream& operator<<(ostream& out, const Starosta& s)  
 {  
 cout << "Faculty is " << s.fac << endl;  
 cout << "Speciality is " << s.spec << endl;  
 cout << "Name is " << s.name << endl;  
 cout << s.course << " course" << endl;  
 cout << s.duration << " duration" << endl;  
 return out;  
 }  
 friend istream& operator>>(istream& in, Starosta& s)  
 {  
 cout << "Faculty is ";  
 in >> s.fac;  
 cout << "Speciality is ";  
 in >> s.spec;  
 cout << "Name is ";  
 in >> s.name;  
 cout << "Course - ";  
 in >> s.course;  
 cout << "Duration - ";  
 in >> s.duration;  
 return in;  
 }  
};

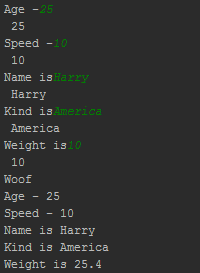
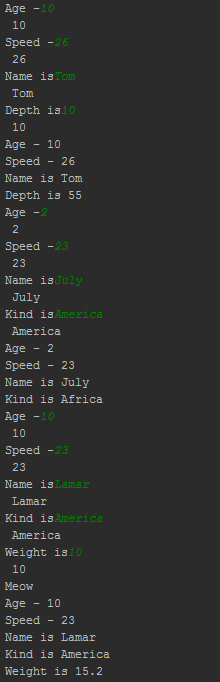
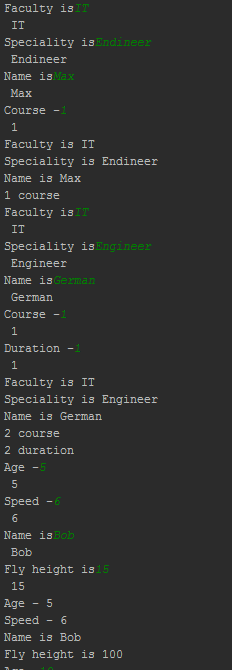
**Student.h**

#pragma once  
#include<iostream>  
#include<string>  
using namespace std;  
class Student {  
protected:  
 string fac;  
 string spec;  
 string name;  
 int course;  
public:  
 Student() {  
 fac = "default";  
 spec = "default";  
 name = "default";  
 course = 0;  
 }  
 friend ostream& operator<<(ostream& out, const Student& s)  
 {  
 cout << "Faculty is " << s.fac << endl;  
 cout << "Speciality is " << s.spec << endl;  
 cout << "Name is " << s.name << endl;  
 cout << s.course << " course" << endl;  
 return out;  
 }  
 friend istream& operator>>(istream& in, Student& s)  
 {  
 cout << "Faculty is ";  
 in >> s.fac;  
 cout << "Speciality is ";  
 in >> s.spec;  
 cout << "Name is ";  
 in >> s.name;  
 cout << "Course - ";  
 in >> s.course;  
 return in;  
 }  
 void nextCourse() {  
 course++;  
 }  
 void changeFac(string fac)  
 {  
 this->fac = fac;  
 }  
};

**main.cpp**

#include<iostream>  
#include"Student.h"  
#include"Starosta.h"  
#include"Alive.h"  
#include"Bird.h"  
#include"Fish.h"  
#include"Animal.h"  
#include"Cat.h"  
#include"Dog.h"  
int main() {  
 Student student1;  
 Starosta starosta1;  
 cin >> student1;  
 cout << student1;  
 cin >> starosta1;   
 starosta1.nextCourse();  
 cout << starosta1;  
 Bird bird1;  
 cin >> bird1;  
 bird1.changeFlyHeight(100);  
 cout << bird1;  
 Fish fish1;  
 cin >> fish1;  
 fish1.changeDepth(55);  
 cout << fish1;  
 Animal animal1;  
 cin >> animal1;  
 animal1.changeKind("Africa");  
 cout << animal1;  
 Cat cat1;  
 cin >> cat1;  
 cat1.voice();  
 cat1.changeWeight(15.2);  
 cout << cat1;  
 Dog dog1;  
 cin >> dog1;  
 dog1.voice();  
 dog1.changeWeight(25.4);  
 cout << dog1;  
 system("pause");  
 return 0;  
}

**Результат работы программы**



**Рис. 6 - Результат работы программы**

**Заключение**

В ходе лабораторной работы я изучил наследование классов в языке C++.

# Лабораторная работа №7

**Тема: «Создание абстрактных классов»**

**Цель лабораторной работы**

Целью данной лабораторной работы является изучение и создание абстрактных классов в языке С++.

**Задание**

Создать базовый абстрактных класс «Человек», имеющий нереализованную виртуальную функцию вывода информации на экран. Затем создать классы «Ученик» и «Босс», унаследованные от него.

**Код программы**

**Human.h**

#pragma once  
#include<iostream>  
#include<string>  
using namespace std;  
class Human {  
protected:  
 string surname;  
 string name;  
 string midname;  
 int age;  
public:  
 Human() {  
 surname = "Default";  
 name = "Default";  
 midname = "Default";  
 age = 0;  
 }  
 Human(string surname, string name, string midname, int age) {  
 this->surname = surname;  
 this->name = name;  
 this->midname = midname;  
 this->age = age;  
 }  
 virtual void print() {  
 cout << "I am human. My name is " << surname << " " << name << " " << midname << endl;  
 cout << "I'm " << age << " year's old";  
 }  
 ~Human() {  
 cout << "Class human was destructed here.";  
 }  
};

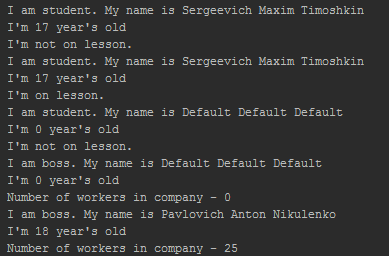
**Boss.h**

#pragma once  
#include"Human.h"  
class Boss : public Human {  
private:  
 int number\_of\_workers;  
public:  
 Boss() : Human() {  
 number\_of\_workers = 0;  
 }  
 Boss(string surname, string name, string midname, int age, int number\_of\_workers) : Human(surname, name, midname, age) {  
 this->number\_of\_workers = number\_of\_workers;  
 }  
 void set(int number) {  
 number\_of\_workers = number;  
 }  
 void print() override {  
 cout << "I am boss. My name is " << surname << " " << name << " " << midname << endl;  
 cout << "I'm " << age << " year's old" << endl;  
 cout << "Number of workers in company - " << number\_of\_workers << endl;  
 }  
 ~Boss() {  
 cout << "Class boss was destructed here." << endl;  
 }  
};

**Student.h**

#pragma once  
#include"Human.h"  
class Student : public Human {  
private:  
 bool on\_lesson;  
public:  
 Student() : Human() {  
 on\_lesson = 0;  
 }  
 Student(string surname, string name, string midname, int age, bool on\_lesson) : Human(surname,name, midname, age) {  
 this->on\_lesson = on\_lesson;  
 }  
 void change() {  
 if (on\_lesson)  
 on\_lesson = 0;  
 else  
 on\_lesson = 1;  
 }  
 void print() override {  
 cout << "I am student. My name is " << surname << " " << name << " " << midname << endl;  
 cout << "I'm " << age << " year's old" << endl;  
 if (on\_lesson)  
 {  
 cout << "I'm on lesson." << endl;  
 }  
 else  
 {  
 cout << "I'm not on lesson." << endl;  
 }  
 }  
 ~Student() {  
 cout << "Class student was destructed here." << endl;  
 }  
};

**Результат работы программы**



**Рис. 7 – Результат работы программы**

**Заключение**

В ходе лабораторной работы я изучил абстрактные классы в языке C++.

# Лабораторная работа №8

**Тема: «Библиотека STL»**

**Цель лабораторной работы**

Целью данной лабораторной работы является знакомство с библиотекой STL- стандартной библиотекой шаблонов - в языке С++, а также показать ее использование на примерах.

**Задание**

1) Используйте шаблон vector для массива данных о студентах.

2) Используйте шаблон list для двусвязного списка данных класса Complex.

3) Используйте шаблон queue для очереди авто на мойке.

**Код программы**

**Complex.h**

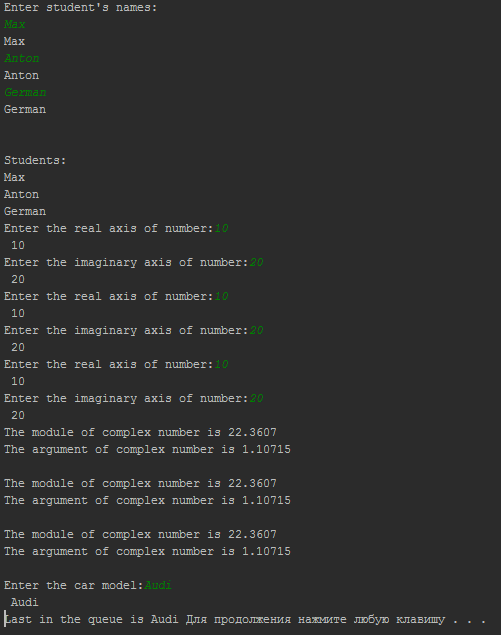
#pragma once  
#include<iostream>  
#include<string>

#include <cmath>  
using namespace std;  
class Comp {  
public:  
 float im, re, abs, arg;  
 Comp() {  
 im = 0;  
 re = 0;  
 abs = 0;  
 arg = 0;  
 }  
 friend ostream& operator<<(ostream& out, Comp &c) {  
 out << "The module of complex number is " << c.abs << endl;  
 out << "The argument of complex number is " << c.arg << endl;  
 return out;  
 }  
 friend istream& operator>>(istream& in, Comp &c) {  
 cout << "Enter the real axis of number: ";  
 in >> c.re;  
 cout << "Enter the imaginary axis of number: ";  
 in >> c.im;  
 c.abs = sqrt(c.re\*c.re + c.im \* c.im);  
 c.arg = atan(c.im / c.re);  
 return in;  
 }  
 Comp& operator=(Comp& c1) {  
 if (this == &c1)  
 return \*this;  
 re = c1.re;  
 im = c1.im;  
 abs = c1.abs;  
 arg = c1.arg;  
 return \*this;  
 }  
};

**main.cpp**

#include"Complex.h"  
#include<vector>  
#include<queue>   
#include<list>  
#include<iterator>  
int main() {  
 cout << "Enter student's names:" << endl;  
 vector<string> stud;  
 string buff = "";  
 do {  
 getline(cin, buff);  
 if (buff.size() > 0) {  
 stud.push\_back(buff);  
 }  
 } while (buff != "");  
 cout << "Students:" << endl;  
 for (unsigned int i = 0; i < stud.size(); i++)  
 cout << stud[i] << endl;  
 list<Comp>comp;  
 Comp comp1;  
 Comp comp2;  
 Comp comp3;  
 cin >> comp1;  
 cin >> comp2;  
 cin >> comp3;  
 comp.push\_back(comp1);  
 comp.push\_back(comp2);  
 comp.push\_back(comp3);  
 unsigned int size = 0;  
 size = comp.size();  
 for (unsigned int i = 0; i < size; i++) {  
 cout << comp.front() << endl;  
 comp.pop\_front();  
 }  
 queue<string>car;  
 cout << "Enter the car model: ";  
 cin.ignore();  
 getline(cin, buff);  
 car.push(buff);  
 while (!car.empty()) {  
 cout << "Last in the queue is "<< car.front() << " ";  
 car.pop();  
 }  
 system("pause");  
 return 0;  
}

**Результат работы программы**



**Рис. 8 – Результат работы программы**

**Заключение**

В ходе лабораторной работы я познакомился с библиотекой STL – стандартной библиотекой шаблонов в языке C++.