**Оглавление**

[Лабораторная работа №1 4](#_Toc515396385)

[Цель лабораторной работы: 4](#_Toc515396386)

[Текст задания: 4](#_Toc515396387)

[UML-диаграмма: 4](#_Toc515396388)

[Код программы: 5](#_Toc515396389)

[Файл main.cpp: 5](#_Toc515396390)

[Файл Complex.h: 6](#_Toc515396391)

[Файл Complex.c: 6](#_Toc515396392)

[Скриншоты выполнения программы: 7](#_Toc515396393)

[Выводы: 7](#_Toc515396394)

[Лабораторная работа №2 8](#_Toc515396395)

[Цель лабораторной работы: 8](#_Toc515396396)

[Текст задания: 8](#_Toc515396397)

[UML-диаграмма: 8](#_Toc515396398)

[Код программы: 9](#_Toc515396399)

[Файл main.cpp: 9](#_Toc515396400)

[Скриншоты выполнения программы: 10](#_Toc515396401)

[Выводы: 10](#_Toc515396402)

[Лабораторные работы №3-5 11](#_Toc515396403)

[Цели лабораторных работ: 11](#_Toc515396404)

[Работа №3: 11](#_Toc515396405)

[Работа №4: 11](#_Toc515396406)

[Работа №5: 11](#_Toc515396407)

[Тексты заданий: 11](#_Toc515396408)

[Работа №3: 11](#_Toc515396409)

[Работа №4: 12](#_Toc515396410)

[Работа №5: 12](#_Toc515396411)

[Код программы: 14](#_Toc515396412)

[Файл main.cpp: 14](#_Toc515396413)

[Файл Child.h: 15](#_Toc515396414)

[Файл Child.cpp: 15](#_Toc515396415)

[Файл Tiles.h: 16](#_Toc515396416)

[Файл Tiles.cpp: 16](#_Toc515396417)

[Файл Complex.h: 17](#_Toc515396418)

[Файл Complex.cpp: 18](#_Toc515396419)

[Файл Vector.h: 19](#_Toc515396420)

[Файл Vector.cpp: 20](#_Toc515396421)

[Скриншоты выполнения программы: 21](#_Toc515396422)

[Выводы: 21](#_Toc515396423)

[Лабораторная работа №6 22](#_Toc515396424)

[Цель лабораторной работы: 22](#_Toc515396425)

[Текст задания: 22](#_Toc515396426)

[UML-диаграммы: 22](#_Toc515396427)

[Код программы: 24](#_Toc515396428)

[Файл main.cpp: 24](#_Toc515396429)

[Скриншоты выполнения программы: 25](#_Toc515396430)

[Выводы: 25](#_Toc515396431)

[Лабораторная работа №7 26](#_Toc515396432)

[Цель лабораторной работы: 26](#_Toc515396433)

[Текст задания: 26](#_Toc515396434)

[UML-диаграммы: 26](#_Toc515396435)

[Код программы: 27](#_Toc515396436)

[Файл main.cpp: 27](#_Toc515396437)

[Файл Human.h: 27](#_Toc515396438)

[Файл Human.cpp: 27](#_Toc515396439)

[Файл Student.h: 27](#_Toc515396440)

[Файл Student.cpp: 28](#_Toc515396441)

[Файл Boss.h: 28](#_Toc515396442)

[Файл Boss.cpp: 29](#_Toc515396443)

[Скриншоты выполнения программы: 29](#_Toc515396444)

[Выводы: 29](#_Toc515396445)

[Лабораторная работа №8 30](#_Toc515396446)

[Цель лабораторной работы: 30](#_Toc515396447)

[Текст задания: 30](#_Toc515396448)

[UML-диаграмма: 30](#_Toc515396449)

[Код программы: 30](#_Toc515396450)

[Файл main.cpp: 30](#_Toc515396451)

[Скриншоты выполнения программы: 32](#_Toc515396452)

[Выводы: 32](#_Toc515396453)

# Лабораторная работа №1

Цель лабораторной работы:

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

Текст задания:

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.с, файл с функцией main() может называться main.с 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 p. Затем происходит поиск комплексного числа с максимальным модулем в массиве p.

UML-диаграмма:

|  |
| --- |
| Complex |
| +re: double  +im: double |
| +double abs\_complex(Complex \*comp);  +Complex \*plus (Complex \*comp1, Complex \*comp2);  +Complex \*minus (Complex \*comp1, Com plex \*comp2);  +Complex \*multiply (Complex \*comp1, Complex \*comp2);  +Complex \*divide (Complex \*comp1, Complex \*comp2);  +void print\_complex(Complex \*complex);  +int read (FILE \*file, Complex \*complex);  +int calculate (FILE \*input\_file); |

Код программы:

### Файл main.cpp:

#include <stdio.h>  
#include <malloc.h>  
#include "Complex.h"  
  
int main(int argc, const char \*argv[]) {  
 FILE \*input\_file = stdin;  
 if (argc == 1) {  
 printf("=>Input your complexes. Example: (2+3i)/(-1-25i) or (2 - 2i) + (2 - 2i)\n");  
 return calculate((FILE \*) input\_file);  
 }  
 if ((input\_file = fopen(argv[1], "r")) == NULL) {  
 printf("=>>[ERROR] Cant open file %s\n", argv[1]);  
 return -1;  
 }  
 printf("=>>[SUCCESS] File %s is opened!\n", argv[1]);  
 printf("=>Count start:\n");  
  
 int size = 10;  
 Complex \*array = (Complex \*) malloc(sizeof(Complex) \* size);  
 int n = 0;  
  
 while (read(input\_file, array + n) != -1) {  
 if (n - 1 >= size)  
 array = realloc(array, sizeof(Complex) \* size \* 2);  
  
 printf("%d: ", n);  
 print\_complex(array + n);  
 n++;  
 fscanf(input\_file, " ");  
 }  
 printf("=>Count finished: %d\n", n);  
  
 double max = abs\_complex(&array[0]);  
 Complex max\_comp = array[0];  
 for (int i = 1; i < n; ++i) {  
 if(max < abs\_complex(&array[i])){  
 max = abs\_complex(&array[i]);  
 max\_comp = array[i];  
 }  
 }  
 printf("Max: |%lf|\n", max);  
 print\_complex(&max\_comp);  
 free(array);  
 fclose(input\_file);

return 0;  
}

### Файл Complex.h:

#ifndef LAB1\_C\_COMPLEX\_H  
#define LAB1\_C\_COMPLEX\_H  
  
typedef struct complex{  
 double re, im;  
} Complex;  
  
double abs\_complex(Complex \*comp);  
Complex \*plus(Complex \*comp1, Complex \*comp2);  
Complex \*minus(Complex \*comp1, Complex \*comp2);  
Complex \*multiply (Complex \*comp1,Complex \*comp2);  
Complex \*divide (Complex \*comp1, Complex \*comp2);  
void print\_complex(Complex \*complex);  
int read(FILE \*file, Complex \*complex);  
int calculate(FILE \*input\_file);  
#endif //LAB1\_C\_COMPLEX\_H

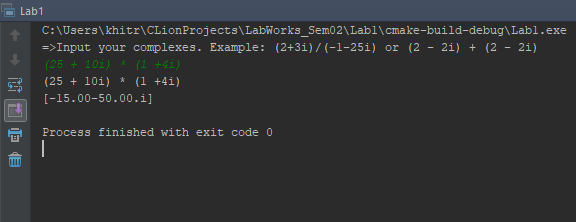
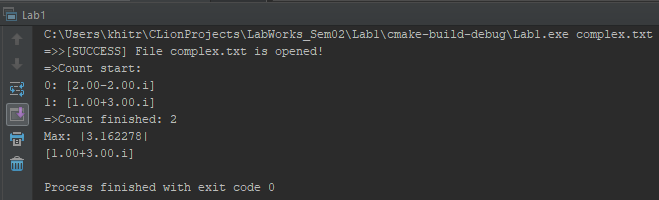
### Файл Complex.c:

#include <stdio.h>  
#include <math.h>  
#include "Complex.h"  
  
double abs\_complex(Complex \*comp) {  
 return sqrt(comp->re \* comp->re + comp->im \* comp->im);  
}  
  
Complex \*plus(Complex \*comp1, Complex \*comp2) {  
 comp1->re += comp2->re;  
 comp1->im += comp2->im;  
 return comp1;  
}  
  
Complex \*minus(Complex \*comp1, Complex \*comp2) {  
 comp1->re -= comp2->re;  
 comp1->im -= comp2->im;  
 return comp1;  
}  
  
Complex \*multiply (Complex \*comp1,Complex \*comp2) {  
 comp1->re = comp1->re\*comp2->re - comp1->im\*comp2->im;  
 comp1->im = comp1->re\*comp2->im + comp1->im\*comp2->re;  
 return comp1;  
}  
  
Complex \*divide (Complex \*comp1, Complex \*comp2) {  
 double r = comp2->re \* comp2->re + comp2->im \* comp2->im;  
 comp1->re = (comp1->re \* comp2->re + comp1->im \* comp2->im) / r;  
 comp1->im = (comp1->im \* comp2->re - comp1->re \* comp2->im) / r;  
 return comp1;  
}  
  
void print\_complex(Complex \*complex){  
 printf("[%.2lf%+.2lf.i]\n", complex->re, complex->im);  
}  
  
int read(FILE \*file, Complex \*complex) {  
 char op;  
 int res;

if (

(res =fscanf(file, "(%lf %c %lfi)", &complex->re, &op, &complex->im) )

!= EOF && res == 3) {  
 complex->im \*= op == '-' ? -1 : 1;  
 return 1;  
 }  
 return -1;  
}  
  
int calculate(FILE \*input\_file){  
 Complex complex1;  
 Complex complex2;  
 char operator;  
 char op1, op2;  
 if ( fscanf(input\_file, "(%lf %c %lfi) %c (%lf %c %lfi)", &complex1.re, &op1, &complex1.im, &operator, &complex2.re, &op2, &complex2.im) < 7)  
 operator = 'e';  
 complex1.im \*= op1 == '-' ? -1 : 1;  
 complex2.im \*= op2 == '-' ? -1 : 1;  
 switch (operator) {  
 case '+':  
 plus(&complex1, &complex2);  
 break;  
 case '-':  
 minus(&complex1, &complex2);  
 break;  
 case '\*':  
 multiply(&complex1, &complex2);  
 break;  
 case '/':  
 divide(&complex1, &complex2);  
 break;  
 default:  
 printf("=>>Input error.\n");  
 return -1;  
 }  
 print\_complex(&complex1);  
 fclose(input\_file);  
}

Скриншоты выполнения программы:

Выводы:

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

# Лабораторная работа №2

Цель лабораторной работы:

Целью данной лабораторной работы освоить на практике вызов

функции с использованием указателей.

Текст задания:

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

UML-диаграмма:

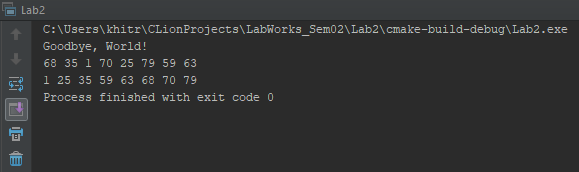
|  |
| --- |
| Main.cpp |
| +int size;  +int\* array;  +void(\*)(int\*, int) ptr\_array; |
| +void f1();  +void f2();  +void reverse(int\*, int);  +void Sort(int\*, int);  +reverseSort(int\*, int); |

Код программы:

### Файл main.cpp:

#include <stdio.h>  
#include <stdlib.h>  
  
void f1(){  
 printf("Hello, World!\n");  
}  
void f2(){  
 printf("Goodbye, World!\n");  
}  
  
void reverse(int \*a, int size) {  
 int temp;  
 for (int i = 0; i < size/2; ++i) {  
 temp = a[size - i - 1];  
 a[size - i - 1] = a[i];  
 a[i] = temp;  
 }  
}  
void Sort(int \*a, int size) {  
 int temp;  
 for (int i = 0; i < size - 1; i++) {  
 for (int j = 0; j < size - i - 1; j++) {  
 if (a[j] > a[j + 1]) {  
 temp = a[j];  
 a[j] = a[j + 1];  
 a[j + 1] = temp;  
 }  
 }  
 }  
}  
void reverseSort(int \*a, int size) {  
 Sort(a, size);  
 reverse(a, size);  
}  
int size = 8;  
int \*array;  
void (\*ptr\_array)(int \*, int);  
  
void (\*switcher())(int\* a, int size) {  
 int sum = 0;  
 for (int i = 1; i < size; ++i) {  
 sum += array[i];  
 }  
 if(sum == array[0])  
 return reverse;  
 else if(sum > array[0])  
 return Sort;  
 else return reverseSort;  
}  
  
int main() {  
 //#1  
 void (\*ptr\_f)();  
 if (rand() % 10+1 > 5)  
 ptr\_f = f1;  
 else ptr\_f = f2;  
 ptr\_f();  
  
 //#2  
 array = (int \*) malloc(sizeof(int)\*size);  
 for (int i = 0; i < size; ++i) {  
 array[i] = rand() % 100 + 1;  
 printf("%d ",array[i]);  
 }  
 printf("\n");  
  
 ptr\_array = switcher();  
 ptr\_array(array, size);  
  
 for (int i = 0; i < size; ++i) {  
 printf("%d ",array[i]);  
 }  
  
 free(array);  
 return 0;  
}

Скриншоты выполнения программы:



Выводы:

Освоили на практике вызов функции с использованием указателей.

# Лабораторные работы №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.

Тексты заданий:

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-диаграммы:

|  |
| --- |
| 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:

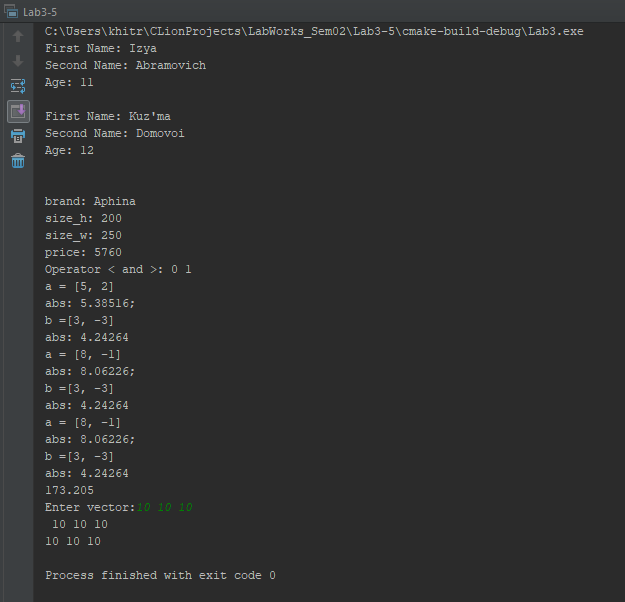
#include <cmath>  
#include "Complex.h"  
  
Complex::Complex(double r, double i) {  
 re = r;  
 im = i;  
}  
Complex::Complex(){  
 re = 0;  
 im = 0;  
}  
Complex::~Complex() = default;  
  
double const Complex::absComplex() {  
 return sqrt(re \* re + im \* im);  
}  
  
Complex &Complex::operator=(const 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.

# Лабораторная работа №6

Цель лабораторной работы:

Целью данной лабораторной работы является изучение

наследованная классов в языке С++.

Текст задания:

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.

UML-диаграммы:

|  |
| --- |
| Student |
| +string name;  +string group; |
| +Student(string name, string group);  +string getName();  +void changeGroup(string newGroup); |

|  |
| --- |
| Chief: public Student |
| +bool hasJournal; |
| +Chief(bool hasJournal,string nameC, string groupC)  :Student(nameC, groupC)  +bool isJournal(); |

|  |
| --- |
| Alive |
| -isAlive; |
| +Alive(bool isAlive); |

|  |
| --- |
| Bird: public Alive |
| -bool canFly; |
| +Bird (bool isAlive, bool canFly); |

|  |
| --- |
| Fish: public Alive |
| -bool canSwim; |
| +Fish(bool isAlive, bool canSwim); |

|  |
| --- |
| Animal: public Alive |
| -int age; |
| +Animal(bool isAlive, int age);  +virtual void say() =0; |

|  |
| --- |
| Dog: public Animal |
|  |
| +void say() override; |

|  |
| --- |
| Cat: public Animal |
|  |
| +void say() override; |

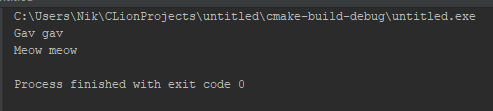
Код программы:

### Файл main.cpp:

#include <iostream>  
#include <utility>  
  
using namespace std;  
//#1  
class Student {  
public:  
 string name;  
 string group;  
  
 Student(string name, string group) {  
 this->name = std::move(name);  
 this->group = std::move(group);  
 }  
  
 string getName() {  
 return name;  
 }  
  
 void changeGroup(string newGroup){  
 group = std::move(newGroup);  
 }  
};  
class Chief: public Student{  
private:  
 bool hasJournal;  
public:  
 Chief(bool hasJournal,string nameC, string groupC):Student(nameC,groupC){  
 this->hasJournal = hasJournal;  
 }  
  
 bool isJournal(){  
 return hasJournal;  
 }  
};  
//end #1  
  
//#2  
class Alive{  
private:  
 bool isAlive;  
public:  
 explicit Alive(bool isAlive){  
 this->isAlive = isAlive;  
 }  
};  
  
class Bird: public Alive {  
private:  
 bool canFly;  
public:  
 Bird(bool isAlive, bool canFly) : Alive{isAlive} {  
 this->canFly = canFly;  
 }  
};  
  
class Fish:public Alive {  
private:  
 bool canSwim;  
public:  
 Fish(bool isAlive, bool canSwim) : Alive{isAlive} {  
 this->canSwim = canSwim;  
 }  
};  
  
class Animal: public Alive{  
private:  
 int age;  
public:  
 explicit Animal(bool isAlive, int age):Alive{isAlive}{  
 this->age=age;  
 };  
  
 virtual void say() =0;  
};  
//end #2

//#3  
class Dog: public Animal { ;  
public:  
 explicit Dog(bool isAlive, int age) : Animal{isAlive, age}{}  
  
 void say() override {  
 cout << "Gav gav\n";  
 }  
};  
class Cat: public Animal {  
public:  
 explicit Cat(bool isAlive, int age) : Animal{isAlive, age} {};  
  
 void say() override {  
 cout << "Meow meow\n";  
 }  
};  
//end #3  
  
int main() {  
 Dog dog(true, 5);  
 dog.say();  
 Cat cat(true, 7);  
 cat.say();  
 return 0;  
}

Скриншоты выполнения программы:



Выводы:

Изучили наследование классов в языке С++.

# Лабораторная работа №7

Цель лабораторной работы:

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

Текст задания:

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

UML-диаграммы:

|  |
| --- |
| Human |
| -string surname;  -string name;  -string midname;  -int age; |
| +Human();  +Human(string surname, string name, string midname, int age);  +virtual ~Human()= default;  +virtual void print()= 0; |

|  |
| --- |
| Student: public Human |
| -bool on\_lesson; |
| Student(string surname, string name, string midname, int age, bool on\_lesson);  +~Student() = default;  +void print() override; |

|  |
| --- |
| Boss: public Human |
| -int number\_of\_workers; |
| +Boss();  +Boss(string surname, string name, string midname, int age, int numw);  +~Boss() = default;  +void print() override; |

Код программы:

### Файл main.cpp:

#include "Student.h"  
#include "Boss.h"  
  
int main() {  
 Student student("SURNAME", "NAME", "MIDNAME", 20, true);  
 student.print();  
 Boss boss("SURNAME", "NAME", "MIDNAME", 55, 241);  
 boss.print();  
 return 0;  
}

### Файл Human.h:

#ifndef LAB7\_HUMAN\_H  
#define LAB7\_HUMAN\_H  
  
#include <iostream>  
  
using namespace std;  
  
class Human {  
protected:  
 string surname;  
 string name;  
 string midname;  
 int age;  
  
public:  
 Human();  
  
 Human(string surname, string name, string midname, int age);  
  
 virtual ~Human()= default;  
  
 virtual void print()= 0;  
};  
  
#endif //LAB7\_HUMAN\_H

### Файл Human.cpp:

#include "Human.h"  
  
Human::Human() {  
 surname = "";  
 name = "";  
 midname = "";  
 age = 0;  
}  
  
Human::Human(string surname, string name, string midname, int age) {  
 this->surname = surname;  
 this->name = name;  
 this->midname = midname;  
 this->age = age;  
}

### Файл Student.h:

#ifndef LAB7\_STUDENT\_H  
#define LAB7\_STUDENT\_H  
  
#include "Human.h"  
  
class Student: public Human {  
private:  
 bool on\_lesson;  
public:  
 Student(string surname, string name, string midname, int age, bool on\_lesson);  
  
 ~Student() = default;  
  
 void print() override;  
};  
  
#endif //LAB7\_STUDENT\_H

### Файл Student.cpp:

#include "Student.h"  
  
Student::Student(  
 string surname, string name, string midname, int age, bool on\_lesson)  
 :Human(surname, name, midname, age) {  
 this->on\_lesson = on\_lesson;  
}  
  
void Student::print() {  
 cout << "\nSurname: " << surname << "\nName: " << name  
 << "\nMidname: " << midname << "\nAge: " << age  
 << "\nOn lesson: " << on\_lesson;  
}

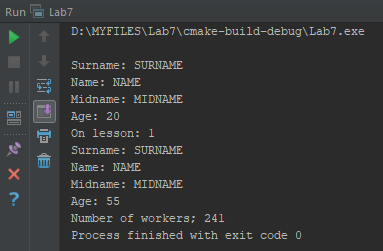
### Файл Boss.h:

#ifndef LAB7\_BOSS\_H  
#define LAB7\_BOSS\_H  
  
#include "Human.h"  
  
class Boss: public Human {  
private:  
 int number\_of\_workers;  
public:  
 Boss();  
  
 Boss(string surname, string name, string midname, int age, int numw);  
  
 ~Boss() = default;  
  
 void print() override;  
};  
  
#endif //LAB7\_BOSS\_H

### Файл Boss.cpp:

#include "Boss.h"  
  
Boss::Boss(string surname, string name, string midname, int age, int numw):Human( surname, name, midname, age) {  
 number\_of\_workers = numw;  
}  
  
void Boss::print() {  
 cout << "\nSurname: " << surname << "\nName: " << name  
 << "\nMidname: " << midname << "\nAge: " << age  
 << "\nNumber of workers; " << number\_of\_workers;  
}

Скриншоты выполнения программы:



Выводы:

Изучили наследование классов и виртуальные функции в языке С++.

# Лабораторная работа №8

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.

Текст задания:

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

UML-диаграмма:

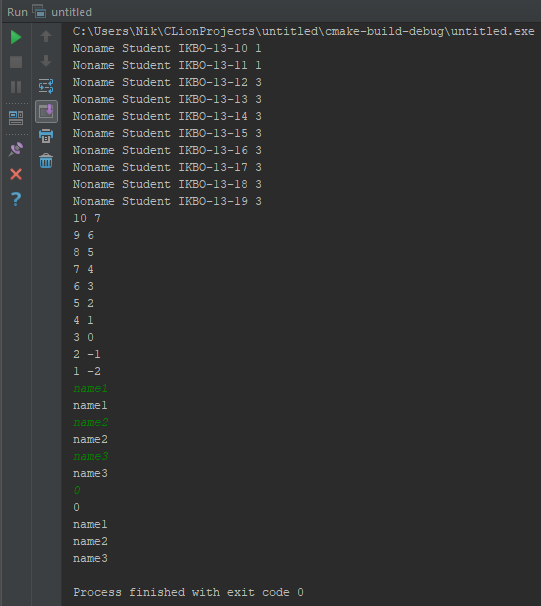
|  |
| --- |
| Student |
| + string name;  +string group;  +int course; |
| +Student(string name, string group, int course);  + void print(); |

Код программы:

### Файл main.cpp:

#include <iostream>  
#include <vector>  
#include <complex>  
#include <list>  
#include <queue>  
  
using namespace std;  
  
class Student{  
public:  
 string name;  
 string group;  
 int course;

Student(string name, string group, int course){  
 this->name = name;  
 this->group = group;  
 this->course = course;  
 }  
 void print(){  
 cout<< name << " " << group << " " << course<<endl;  
 }  
};  
  
int main() {  
 vector<Student> vector;  
 for (int i = 0; i < 10; ++i) {  
 Student student{"Noname Student", "IKBO-13-1" + to\_string(i), 2 % (i + 1) + 1};  
 vector.push\_back(student);  
 }  
 for (int j = 0; j < 10; ++j) {  
 vector[j].print();  
 }  
  
 list<complex<double>> list\_comp;  
 for (int k = 0; k < 10; ++k) {  
 list\_comp.push\_back(complex<double>(k + 1, k - 2));  
 }  
  
  
 for (int l = 0; l < 10; ++l) {  
 complex<double> comp = list\_comp.back();  
 cout << comp.real() << " " << comp.imag() << endl;  
 list\_comp.pop\_back();  
 }  
  
 queue<string> carQueue;  
 string buffer = "1";  
 while (buffer != "0") {  
 cin >> buffer;  
 carQueue.push(buffer);  
 }  
  
 for (int m = 0; m <= carQueue.size(); ++m) {  
 cout << carQueue.front() << endl;  
 carQueue.pop();  
 }  
  
}

Скриншоты выполнения программы:

Выводы:

Познакомились с библиотекой STL – стандартной библиотекой шаблонов - в языке С++, а также показали ее использование на примерах.