|  |  |
| --- | --- |
| для прик эмбл | |
| МИНОБРНАУКИ РОССИИ | |
| Федеральное государственное бюджетное образовательное учреждениевысшего профессионального образования"Московский технологический университет"МИРЭА | |
|  | Факультет информационных технологий (ИТ) |
|  | Кафедра Лабораторной и прикладной информатики (ППИ) |

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ЛАБОРАТОРНЫМ РАБОТАМ** | |
| **по дисциплине** | |
| «Объектно-ориентированное программирование» | |
|  | |
| Выполнил студент группы ИНБО-04-18 | Тарасов Д.И. |
| Принял старший преподаватель | Хлебникова В.Л. |

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

Москва 2019

Содержание

[Таблица успеваемости 3](#_Toc8073375)

[Лабораторная работа №1 4](#_Toc8073376)

[Лабораторная работа №2 6](#_Toc8073377)

[Лабораторная работа №3 10](#_Toc8073378)

[Лабораторная работа №4 20](#_Toc8073379)

[Лабораторная работа №5 24](#_Toc8073380)

[Лабораторная работа №6 33](#_Toc8073381)

[Лабораторная работа №7 42](#_Toc8073382)

[Лабораторная работа №8 51](#_Toc8073383)

# Таблица успеваемости

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| №/р. | страница в отчете. | Студент выполнил:  (подпись) | Преподаватель принял:  (подпись) | Балл  (максимально возможный) | Балл (Фактический) |
| 1. |  |  |  |  |  |
| 2. |  |  |  |  |  |
| 3. |  |  |  |  |  |
| 4. |  |  |  |  |  |
| 5. |  |  |  |  |  |
| 6. |  |  |  |  |  |
| 7. |  |  |  |  |  |
| 8. |  |  |  |  |  |
|  |  |  | Итоговый балл: |  |  |

# Лабораторная работа №1

**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.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.

**Ход выполнения:**

Вариант 1

Complex.cpp

|  |  |
| --- | --- |
|  |  |
|  | #include"Complex.h" |
|  | #include<iostream>  #include<cmath>  using namespace std;  Complex::Complex(double r, double i) : real(r), img(i) {};  void Complex::define(double r, double i)  {  real = r;  img = i;  }  double Complex::module()  {  return sqrt(real\*real + img\*img);  }  Complex& Complex::operator=(const Complex& right) {  if (this == &right) {  return \*this;  }  real = right.real;  img = right.img;  return \*this;  };  Complex operator+ (Complex& left, Complex& right) {  left.real += right.real;  left.img += right.img;  return left;  };  Complex operator- (Complex& left, Complex& right) {  left.real -= right.real;  left.img -= right.img;  return left;  };  Complex operator\*(Complex& left, Complex& right) {  double r, i;  r = left.real\*right.real - left.img\*right.img;  i = left.img\*right.real + left.img\*right.real;  left.real = r;  left.img = i;  return left;  };  Complex operator/(Complex& left, Complex& right) {  Complex r, i;  i = right;  i.img = -i.img;  r = left \* i;  i = right \* i;  if (r.real != 0)  {  left.real = r.real / i.real;  left.img = 0;  }  else if (r.img != 0)  {  left.img = r.img / i.real;  left.real = 0;  }  return left;  };  ostream& operator<<(ostream& out, const Complex& c)  {  out << c.real << "+" << c.img << "i";  return out;  };  istream& operator>>(istream& in, Complex& c)  {  in >> c.real >> c.img;  return in;  }; |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
| #include"Complex.h" | |
| #include<iostream>  using namespace std;  Complex::Complex(double r, double i) : real(r), img(i) {};  Complex& Complex::operator=(const Complex& right) {  if (this == &right) {  return \*this;  }  real = right.real;  img = right.img;  return \*this;  };  Complex operator+ (Complex& left, Complex& right) {  left.real += right.real;  left.img += right.img;  return left;  };  Complex operator- (Complex& left, Complex& right) {  left.real -= right.real;  left.img -= right.img;  return left;  };  Complex operator\*(Complex& left, Complex& right) {  double r, i;  r = left.real\*right.real - left.img\*right.img;  i = left.img\*right.real + left.img\*right.real;  left.real = r;  left.img = i;  return left;  };  Complex operator/(Complex& left, Complex& right) {  Complex r, i;  i = right;  i.img = -i.img;  r = left \* i;  i = right \* i;  if (r.real != 0)  {  left.real = r.real / i.real;  left.img = 0;  }  else if (r.img != 0)  {  left.img = r.img / i.real;  left.real = 0;  }  return left;  };  ostream& operator<<(ostream& out, const Complex& c)  {  out << c.real << "+" << c.img << "i";  return out;  };  istream& operator>>(istream& in, Complex& c)  {  in >> c.real >> c.img;  return in;  }; | |

Complex.h

|  |  |
| --- | --- |
| #ifndef Complex\_H | |
| |  | | --- | | #ifndef Complex\_H | | #define Complex\_H  #include<iostream>  using namespace std;  class Complex  {  private:  double real;  double img;  public:  Complex(double r=0, double i=0);  void define(double r = 0, double i = 0);  double module();  Complex& operator=(const Complex& right);  friend Complex operator+ (Complex& left, Complex& right);  friend Complex operator- (Complex& left, Complex& right);  friend Complex operator\*(Complex& left, Complex& right);  friend Complex operator/(Complex& left, Complex& right);  friend ostream& operator<<(ostream& out, const Complex& c);  friend istream& operator>>(istream& in, Complex& c);  };  #endif | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

main.cpp

|  |  |
| --- | --- |
|  | |
| #include "Complex.h" | |
| #include <iostream>  using namespace std;  int main()  {  Complex a, b;  int sw = 1;  cout << "1 - sum, 2 - minus, 3 - multiply, 4 - divide, 0 - exit" << endl;//выбор  cin >> sw;  while (sw != 0)  {  switch (sw)  {  case 1://сложение  {  cout << "Input a,b" << endl;  cin >> a >> b;  cout << a << "+" << b << "=";  cout<< a + b << endl;  }  break;  case 2://вычитание  {  cout << "Input a,b" << endl;  cin >> a >> b;  cout << a << "-" << b << "=";  cout<< a - b << endl;  }  break;  case 3://умножение  {  cout << "Input a,b" << endl;  cin >> a >> b;  cout << a << "\*" << b << "=";  cout << a \* b << endl;  }  break;  case 4://деление  {  cout << "Input a,b" << endl;  cin >> a >> b;  cout << a << "/" << b << "=";  cout << a / b << endl;  }  break;  case 0://выход из цикла  break;  default:  cout << "Repeat input" << endl;  break;  }  cout << "1 - sum, 2 - minus, 3 - multiply, 4 - divide, 0 - exit" << endl;  cin >> sw;  }  system("pause");  return 0;  } | |
|  |  |

Вариант 2

|  |
| --- |
| #include "Complex.h" |
| #include <iostream>  #include <fstream>  #include <string>  using namespace std;  int main()  {  ifstream fin("Complex.txt");  char s1,s2;  int n,i=0;  fin >> s1;  n = s1 - '0';  Complex \*p=new Complex[n];  while (fin >> s1)  {  fin >> s2;  p[i].define(s1 - '0',s2 - '0');  i++;  }  Complex max;  for (int i=0; i < n; i++)  {  if (p[i].module() > max.module())  max = p[i];  }  cout << "Complex number: " << max << " with max module: " << max.module() << endl;  system("pause");  return 0;  } |

**Выводы:**

В ходе выполнения Лабораторной работы №1 были использованы заголовочный файл Complex.h и файлы исходные Complex.cpp, main.cpp. Были перегружены операторы: =,+,-,\*,/,<<,>>. Были использованы библиотеки iostream, fstream. Реализован метод модуля комлексного числа. Были достигнуты результаты:

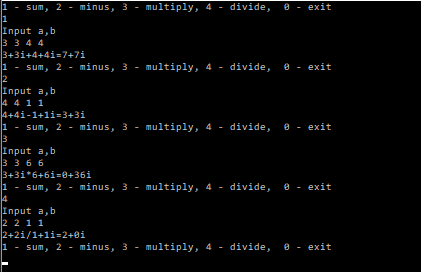


Рис. 1

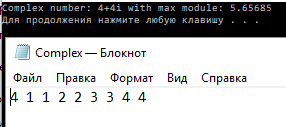


Рис. 2

<https://github.com/dneva/prac-lab/tree/master/lab1.1>

<https://github.com/dneva/prac-lab/tree/master/lab1.2>

# Лабораторная работа №2

**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

**Ход выполнения:**

Вариант 1

|  |
| --- |
| #include<iostream> |
| using namespace std;  double sum(double a, double b)  {  return a + b;  };  double min(double a, double b)  {  return a - b;  };  double multiply(double a, double b)  {  return a\*b;  };  double divide(double a, double b)  {  if (b != 0)  return a/b;  else return 0;  };  int main()  {  double a, b;  double(\*act)(double a, double b);  int s = 1;  cout << "1 - sum, 2 - minus, 3 - multiply, 4 - divide, 0 - exit" << endl;//выбор  cin >> s;  while (s != 0)  {  switch (s)  {  case 1://сложение  {  cout << "Input a,b" << endl;  cin >> a >> b;  act = sum;  cout << a << " + " << b << " = " << act(a, b) << endl;  }  break;  case 2://вычитание  {  cout << "Input a,b" << endl;  cin >> a >> b;  act = min;  cout << a << " - " << b << " = " << act(a, b) << endl;  }  break;  case 3://умножение  {  cout << "Input a,b" << endl;  cin >> a >> b;  act = multiply;  cout << a << " \* " << b << " = " << act(a, b) << endl;  }  break;  case 4://деление  {  cout << "Input a,b" << endl;  cin >> a >> b;  act = divide;  cout << a << " / " << b << " = " << act(a, b) << endl;  }  break;  case 0://выход из цикла  break;  default:  cout << "Repeat input" << endl;  break;  }  cout << "1 - sum, 2 - minus, 3 - multiply, 4 - divide, 0 - exit" << endl;  cin >> s;  }  return 0; |
| } |

Вариант 2

|  |
| --- |
| #include<iostream> |
| #include<ctime>  #include<algorithm>  using namespace std;  bool comp(int a, int b)//компанатор  {  return a > b;  }  int sum(int \*mas, unsigned size)//функция для суммы  {  int result=0;  for (int i = 0; i < size; i++) result += mas[i];  return result;  }  void f1(int \*mas, unsigned size)//функция инвертации  {  reverse(&mas[0], &mas[size]);  }  void f2(int \*mas, unsigned size)//функция отсортировать по неубыванию  {  sort(&mas[0], &mas[size]);  }  void f3(int \*mas, unsigned size)//функция отсортировать по невозрастанию  {  sort(&mas[0], &mas[size],comp);  }  void(\*switcher(int \*mas, unsigned size))(int \*mas, unsigned size)//выбор функции  {  if (sum(mas, size) == mas[0])  return f1;  else if (sum(mas, size) > mas[0])  return f2;  else  return f3;  }  int main()  {  srand(time(0));  int a[10];  cout << "Original array:" << endl;  for (int i = 0; i < 10; i++)  {  a[i] = rand() % 100-20;  cout << a[i] << " ";  }  cout << endl;  void(\*p)(int \*mas, unsigned size);  p = switcher(a,10);  p(a, 10);  cout << "Сonverted array:" << endl;  for (int i = 0; i < 10; i++) cout << a[i] << " ";  cout << endl;  system("pause");  return 0;  } |

**Выводы:**

В ходе выполнения Лабораторной работы №2 были использованы библиотеки iostream, ctime, algorithm. Реализованы функции суммы, инвертации, сортировок элементов массива и сложения, вычитания, умножения, деления чисел. Были достигнуты результаты:

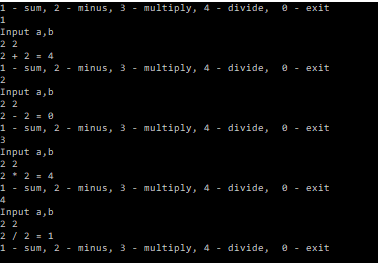


Рис. 3

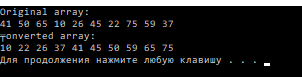


Рис. 4

[https://github.com/dneva/prac-lab/blob/master/lab2/lab2.1.cpp](https://github.com/dneva/prac-lab/blob/master/lab2/lab2.1.cpp )

<https://github.com/dneva/prac-lab/blob/master/lab2/lab2.2.cpp>

# Лабораторная работа №3

**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.

**Ход выполнения:**

Вариант 1

|  |
| --- |
| #include<iostream> |
| #include<ctime>  #include<string>  using namespace std;  class Child  {  private:  string name;//поле имени  string fam;//поле фамилии  int age;//поле возраста  public:  void define(string n, string f, int a)//метод ввода  {  name = n;  fam = f;  age = a;  }  void display()//метод вывода  {  cout << "Name: " << name << endl;  cout << "Surname: " << fam << endl;  cout << "Age: " << age << endl;  }  };  int main()  {  Child c1, c2;  string n, f;  int a;  cout << "Input iformation(name,surname,age) about first child:" << endl;  cin >> n >> f >> a;  c1.define(n,f,a);  cout << "Input iformation(name,surname,age) about second child:" << endl;  cin >> n >> f >> a;  c2.define(n, f, a);  c1.display();  cout << endl;  c2.display();  system("pause");  return 0;  } |

Вариант 2

|  |
| --- |
| #include<iostream> |
| #include<string>  using namespace std;  class Tiles  {  public:  string brand;//марка  double size\_h;//высота  double size\_w;//ширина  double price;//цена  void getData()//вывод информации  {  cout << "Brand: " << brand << endl;  cout << "Size height: " << size\_h << endl;  cout << "Size weight: " << size\_w << endl;  cout << "Price: " << price << endl;  }  };  int main()  {  Tiles t1, t2;  cout << "Input iformation(brand, size(height), size(weight), price) about the first tile" << endl;  cin >> t1.brand >> t1.size\_h >> t1.size\_w >> t1.price;  cout << "Input iformation(brand, size(height), size(weight), price) about the second tile" << endl;  cin >> t2.brand >> t2.size\_h >> t2.size\_w >> t2.price;  cout << endl;  t1.getData();  cout << endl;  t2.getData();  system("pause");  return 0;  } |

Вариант 3

|  |
| --- |
| #define \_USE\_MATH\_DEFINES |
| #include<iostream>  #include<cmath>  using namespace std;  class Complex  {  private:  double re;  double im;  public:  void define(double r, double i)//метод ввода  {  re = r;  im = i;  }  void display()//метод вывода  {  cout << re << "+" << im << "i" << endl;  }  double modul()//модуль комплексного числа  {  return sqrt(re\*re + im\*im);  }  double arg()//аргумент комплексного числа  {  if (re > 0)  return atan(im / re);  else if (re < 0 && im >= 0)  return M\_PI + atan(im / re);  else if (re < 0 && im < 0)  return -M\_PI + atan(im / re);  else if (re == 0 && im > 0)  return M\_PI / 2;  else  return -M\_PI / 2;  }  };  int main()  {  Complex c1;  double r, i;  cout << "Input complex value" << endl;  cin >> r >> i;  c1.define(r, i);  cout << endl << "Complex value:" << endl;  c1.display();  cout << "Module: " << c1.modul() << endl;  cout << "Argument: " << c1.arg() << endl;  system("pause");  return 0;  } |

Вариант 4

|  |
| --- |
| #include<iostream> |
| #include<cmath>  using namespace std;  class Vector  {  private:  double x;  double y;  double z;  public:  void define(double x1, double y1, double z1)//метод ввода  {  x = x1;  y = y1;  z = z1;  }  void display()//метод вывода  {  cout << " (" << x << ";" << y << ";" << z << ")" << endl;  }  double modul()//модуль вектора  {  return sqrt(x\*x + y\*y + z\*z);  }  void sum(Vector v)//сумма векторов  {  x += v.x;  y += v.y;  z += v.z;  }  void min(Vector v)//разница векторов  {  x -= v.x;  y -= v.y;  z -= v.z;  }  };  int main()  {  Vector v1,v2;  double x, y, z;  cout << "Input vector" << endl;  cin >> x >> y >> z;  v1.define(x,y,z);  cout << "Module: " << v1.modul() << endl;  cout << "Input vector for sum" << endl;  cin >> x >> y >> z;  v2.define(x, y, z);  cout << "Result:" << endl;  v1.sum(v2);  v1.display();  cout << "Input vector for subtraction" << endl;  cin >> x >> y >> z;  v2.define(x, y, z);  cout << "Result:" << endl;  v1.min(v2);  v1.display();  system("pause");  return 0;  } |

**Выводы:**

В ходе выполнения Лабораторной работы №3 были использованы библиотеки iostream, string, сtime, cmath. В классах реализованы методы для присвоения значений полям и вывода этих значений, методы нахождения аргумента и модуля комплексного числа, методы суммы, разности, модуля векторов. Были достигнуты результаты:

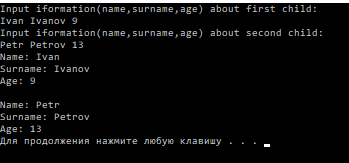


Рис. 5

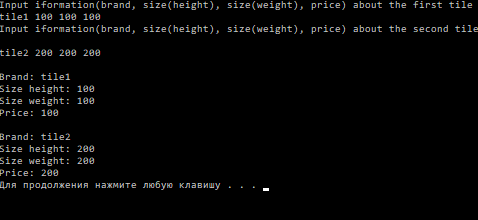


Рис. 6

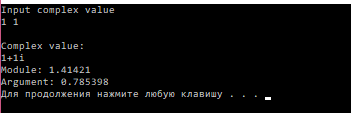


Рис. 7

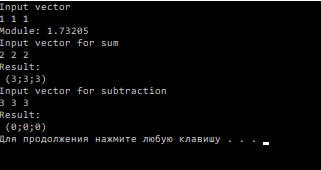


Рис. 8

<https://github.com/dneva/prac-lab/blob/master/lab3/lab3.1.cpp>

<https://github.com/dneva/prac-lab/blob/master/lab3/lab3.2.cpp>

<https://github.com/dneva/prac-lab/blob/master/lab3/lab3.3.cpp>

<https://github.com/dneva/prac-lab/blob/master/lab3/lab3.4.cpp>

# Лабораторная работа №4

**Задание:**

Из прошлой лабораторной работы дополнить всем видами

конструкторов и деструкторами классы:

1. Complex

2. Vector

3. Tiles

4. Child

**Ход выполнения:**

Вариант 1

|  |
| --- |
| #define \_USE\_MATH\_DEFINES |
| #include<iostream>  #include<cmath>  using namespace std;  class Complex  {  private:  double re;  double im;  public:  Complex(double r = 0, double i = 0) : re(r), im(i) {};//конструктор  Complex(const Complex& obj)//конструктор копирования  {  re = obj.re;  im = obj.im;  cout << "Copy constructor" << endl;  }  void define(double r, double i)//метод ввода  {  re = r;  im = i;  }  void display()//метод вывода  {  cout << re << "+" << im << "i" << endl;  }  double modul()//модуль комплексного числа  {  return sqrt(re\*re + im\*im);  }  double arg()//аргумент комплексного числа  {  if (re > 0)  return atan(im / re);  else if (re < 0 && im >= 0)  return M\_PI + atan(im / re);  else if (re < 0 && im < 0)  return -M\_PI + atan(im / re);  else if (re == 0 && im > 0)  return M\_PI / 2;  else  return -M\_PI / 2;  }  void sum(Complex &obj)//суммирование  {  re += obj.re,  im += obj.im;  }  ~Complex()//деструктор  {  cout << "Destructor" << endl;  }  };  int main()  {  Complex c1;  double r, i;  cout << "Input complex value" << endl;  cin >> r >> i;  c1.define(r, i);  cout << "Complex value:" << endl;  c1.display();  cout << "Module: " << c1.modul() << endl;  cout << "Argument: " << c1.arg() << endl;  cout << "Input complex value for sum" << endl;  Complex c2(c1);  cin >> r >> i;  c1.define(r, i);  c1.sum(c2);  c1.display();  system("pause");  return 0;  } |

Вариант 2

|  |
| --- |
| #include<iostream> |
| #include<cmath>  using namespace std;  class Vector  {  private:  double x;  double y;  double z;  public:  Vector(double x1 = 0, double y1 = 0, double z1 = 0) : x(x1), y(y1), z(z1) {};//конструктор  Vector(const Vector &obj)//конструктор копирования  {  x = obj.x;  y = obj.y;  z = obj.z;  cout << "Copy constructor" << endl;  }  void define(double x1, double y1, double z1)//метод ввода  {  x = x1;  y = y1;  z = z1;  }  void display()//метод вывода  {  cout << " (" << x << ";" << y << ";" << z << ")" << endl;  }  double modul()//модуль вектора  {  return sqrt(x\*x + y\*y + z\*z);  }  void sum(Vector v)//сумма векторов  {  x += v.x;  y += v.y;  z += v.z;  }  void min(Vector v)//разница векторов  {  x -= v.x;  y -= v.y;  z -= v.z;  }  ~Vector()//деструктор  {  cout << "Destructor" << endl;  }  };  int main()  {  Vector v1, v2;  double x, y, z;  cout << "Input vector" << endl;  cin >> x >> y >> z;  v1.define(x, y, z);  cout << "Module: " << v1.modul() << endl;  cout << "Input vector for sum" << endl;  cin >> x >> y >> z;  v2.define(x, y, z);  cout << "Result:" << endl;  v1.sum(v2);  v1.display();  cout << "Input vector for subtraction" << endl;  cin >> x >> y >> z;  v2.define(x, y, z);  cout << "Result:" << endl;  v1.min(v2);  v1.display();  system("pause");  return 0;  } |

Вариант 3

|  |
| --- |
| #include<iostream> |
| #include<string>  using namespace std;  class Tiles  {  public:  Tiles(string b = "<none>", double szh = 0, double szw = 0, double p = 0) : brand(b), size\_h(szh), size\_w(szw), price(p) {};//констуртор  Tiles(const Tiles& t)//констуртор копирования  {  brand = t.brand;  size\_h = t.size\_h;  size\_w = t.size\_w;  price = t.price;  cout << "Copy consructor" << endl;  }  string brand;//марка  double size\_h;//высота  double size\_w;//ширина  double price;//цена  void getData()//вывод информации  {  cout << "Brand: " << brand << endl;  cout << "Size height: " << size\_h << endl;  cout << "Size weight: " << size\_w << endl;  cout << "Price: " << price << endl;  }  ~Tiles()//деструктор  {  cout << "Destructor " << endl;  }  };  int main()  {  Tiles t1, t2;  cout << "Input iformation(brand, size(height), size(weight), price) about the first tile" << endl;  cin >> t1.brand >> t1.size\_h >> t1.size\_w >> t1.price;  cout << "Input iformation(brand, size(height), size(weight), price) about the second tile" << endl;  cin >> t2.brand >> t2.size\_h >> t2.size\_w >> t2.price;  t1.getData();  t2.getData();  cout << "Copy the first tile in the third one" << endl;  Tiles t3(t1);  t3.getData();  system("pause");  return 0;  } |

Вариант 4

|  |
| --- |
| #include<iostream> |
| #include<string>  using namespace std;  class Child  {  private:  string name;//поле имени  string fam;//поле фамилии  int age;//поле возраста  public:  Child(string n = "<none>", string f = "<none>", double a = 0) : name(n), fam(f), age(a) {};//констуртор  Child(const Child& c)//констуртор копирования  {  name = c.name;  fam = c.fam;  age = c.age;  cout << "Copy consructor" << endl;  }  void define(string n, string f, int a)//метод ввода  {  name = n;  fam = f;  age = a;  }  void display()//метод вывода  {  cout << "Name: " << name << endl;  cout << "Surname: " << fam << endl;  cout << "Age: " << age << endl;  }  ~Child()//деструктор  {  cout << "Destructor " << endl;  }  };  int main()  {  Child c1, c2;  string n, f;  int a;  cout << "Input iformation(name,surname,age) about first child:" << endl;  cin >> n >> f >> a;  c1.define(n, f, a);  cout << "Input iformation(name,surname,age) about second child:" << endl;  cin >> n >> f >> a;  c2.define(n, f, a);  c1.display();  c2.display();  cout << "Copy the first child in the third one" << endl;  Child c3(c1);  c3.display();  system("pause");  return 0;  } |

**Выводы:**

В ходе выполнения Лабораторной работы №3 были использованы библиотеки iostream, string, сtime, cmath. В классах реализованы методы для присвоения значений полям и вывода этих значений, методы нахождения аргумента и модуля комплексного числа, методы суммы, разности, модуля векторов. Добавлены конструкторы и деструкторы. Были достигнуты результаты:

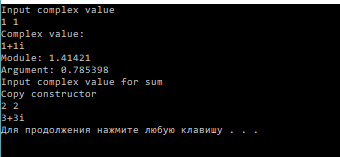


Рис. 9

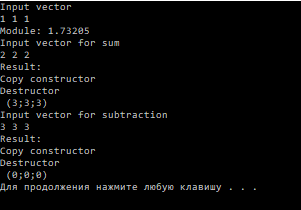


Рис. 10

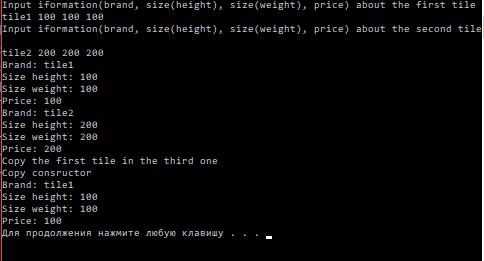


Рис. 11

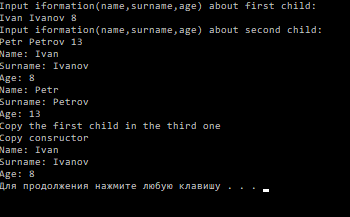


Рис. 12

<https://github.com/dneva/prac-lab/blob/master/lab4/lab4.1>

<https://github.com/dneva/prac-lab/blob/master/lab4/lab4.2>

<https://github.com/dneva/prac-lab/blob/master/lab4/lab4.3>

<https://github.com/dneva/prac-lab/blob/master/lab4/lab4.4>

# Лабораторная работа №5

**Задание:**

1. Для класса Complex перегрузить операторыприсваивания, инкремента, декремента, сравнения, ввода ивывода.

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

**Ход выполнения:**

Вариант 1

|  |
| --- |
| #include<iostream> |
| using namespace std;  class Complex  {  private:  double real;  double img;  public:  Complex(double r = 0, double i = 0);//конструктор  Complex& operator=(const Complex& right);//перегрузка оператора =  friend const Complex &operator++(Complex& left);//перегрузка оператора ++(префикс)  friend const Complex &operator--(Complex& left);//перегрузка оператора --(префикс)  friend bool operator==(const Complex &left, const Complex &right);//перегрузка оператора ==  friend ostream& operator<<(ostream& out, const Complex& c);//перегрузка оператора <<  friend istream& operator>>(istream& in, Complex& c);//перегрузка оператора >>  };  Complex::Complex(double r, double i) : real(r), img(i) {};  Complex& Complex::operator=(const Complex& right) {  if (this == &right) {  return \*this;  }  real = right.real;  img = right.img;  return \*this;  };  const Complex& operator++(Complex &left)  {  left.real++;  left.img++;  return left;  };  const Complex& operator--(Complex &left)  {  left.real--;  left.img--;  return left;  };  bool operator==(const Complex &left, const Complex &right)  {  return (left.real == right.real && left.img == right.img);  };  ostream& operator<<(ostream& out, const Complex& c)  {  out << c.real << "+" << c.img << "i";  return out;  };  istream& operator>>(istream& in, Complex& c)  {  in >> c.real >> c.img;  return in;  };  int main()  {  Complex a, b,c;  int sw = 1;  cout << "1 - inc, 2 - dec, 3 - compare, 0 - exit" << endl;//выбор  cin >> sw;  while (sw != 0)  {  switch (sw)  {  case 1://инкремент  {  cout << "Input a" << endl;  cin >> a;  c = ++a;  cout << c << endl;  }  break;  case 2://декремент  {  cout << "Input a" << endl;  cin >> a;  c=--a;  cout << c << endl;  }  break;  case 3://сравнение  {  cout << "Input a,b" << endl;  cin >> a >> b;  if (a == b)  cout << "a=b"<< endl;  else  cout << "a!=b"<<endl;  }  break;  case 0://выход из цикла  break;  default:  cout << "Repeat input" << endl;  break;  }  cout << "1 - inc, 2 - dec, 3 - compare, 0 - exit" << endl;//выбор  cin >> sw;  }  system("pause");  return 0;  } |

Вариант 2

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| #include<iostream> | |
| #include<cmath>  using namespace std;  class Vector  {  private:  double x;  double y;  double z;  public:  Vector(double x1 = 0, double y1 = 0, double z1 = 0) : x(x1), y(y1), z(z1) {};//конструктор  Vector(const Vector &obj)//конструктор копирования  {  x = obj.x;  y = obj.y;  z = obj.z;  cout << "Copy constructor" << endl;  }  Vector& operator=(const Vector& right);//перегрузка оператора =  friend bool operator==(const Vector &left, const Vector &right);//перегрузка оператора ==  friend ostream& operator<<(ostream& out, const Vector& v);//перегрузка оператора <<  friend istream& operator>>(istream& in, Vector& v);//перегрузка оператора >>  double modul()//модуль вектора  {  return sqrt(x\*x + y\*y + z\*z);  }  void sum(Vector v)//сумма векторов  {  x += v.x;  y += v.y;  z += v.z;  }  void min(Vector v)//разница векторов  {  x -= v.x;  y -= v.y;  z -= v.z;  }  ~Vector()//деструктор  {  cout << "Destructor" << endl;  }  };  Vector& Vector::operator=(const Vector& right) {  if (this == &right) {  return \*this;  }  x = right.x;  y = right.y;  z = right.z;  return \*this;  };  bool operator==(const Vector &left, const Vector &right)  {  return (left.x == right.x && left.y == right.y && left.z == right.z);  };  ostream& operator<<(ostream& out, const Vector& v)  {  out << " (" << v.x << ";" << v.y << ";" << v.z << ")";  return out;  };  istream& operator>>(istream& in, Vector& v)  {  in >> v.x >> v.y >> v.z;  return in;  };  int main()  {  Vector v1, v2;  cout << "Input vector" << endl;  cin >> v1;  cout << "Module: " << v1.modul() << endl;  cout << "Input vector for sum" << endl;  cin >> v2;  cout << "Result:" << endl;  v1.sum(v2);  cout<<v1<<endl;  cout << "Input vector for subtraction" << endl;  cin >> v2;  cout << "Result:" << endl;  v1.min(v2);  cout << v1 << endl;  cout << "Input vector for comparison" << endl;  cin >> v2;  cout << "Result:" << endl;  if (v1==v2)  cout << "v1=v2" <<endl;  else  cout << "v1!=v2" << endl;  system("pause");  return 0; | |
| } | |

**Выводы:**

В ходе выполнения Лабораторной работы №5 были использованы библиотеки iostream, cmath. Перегружены требуемые операторы. Были достигнуты результаты:

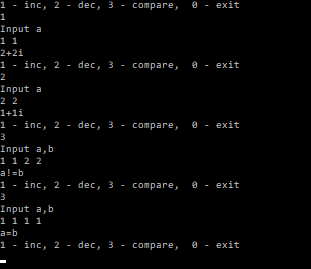


Рис. 13

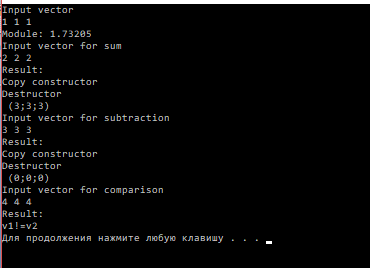


Рис. 14

<https://github.com/dneva/prac-lab/blob/master/lab5/lab5.1>

<https://github.com/dneva/prac-lab/blob/master/lab5/lab5.2>

# Лабораторная работа №6

**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.

**Ход выполнения:**

Вариант 1

|  |
| --- |
| #include<iostream> |
| #include<vector>  #include<string>  using namespace std;  class Student// класс "Студент"  {  protected:  string name;  string surname;  string group;  public:  Student(string n="", string s="", string g="") : name(n), surname(s), group(g) {};//Конструктор  void define(string n, string s, string g)  {  name = n;  surname = s;  group = g;  }  friend ostream& operator<<(ostream& out, const Student& s);//перегрузка оператора <<  };  class Grand\_Student : public Student// класс "Староста"  {  private:  vector<Student> st;//Список старосты  public:  Grand\_Student(string n = "", string s = "", string g = "") : Student(n, s, g) {};  void count(Student s)//записать студента  {  st.push\_back(s);  }  void show\_st()//показать список  {  for (Student n : st) {  cout << n << endl;  }  }  };  ostream& operator<<(ostream& out, const Student& s)  {  out << s.surname << " " << s.name << " " <<s.group;  return out;  };  int main()  {  string n,s,g;  int a;  Student stnt;  Grand\_Student gd\_stnt;  cout << "Input grand student(name,surname,group)" << endl;  cin >> n >> s >> g;  gd\_stnt.define(n, s, g);  cout << "Input number of students" << endl;  cin >> a;  for (int i=0; i < a; i++)  {  cout << "Input student(name,surname,group)" << endl;  cin >> n >> s >> g;  stnt.define(n, s, g);  gd\_stnt.count(stnt);  }  cout << "List of students:" << endl;  gd\_stnt.show\_st();  system("pause");  return 0;  } |

Вариант 2

|  |
| --- |
| #include<iostream> |
| #include<vector>  #include<string>  using namespace std;  class Alive// класс "Живое существо"  {  protected:  string name;  string condition;  public:  Alive(string n="", string c="lives") : name(n),condition(c) {};  void change\_c()  {  if (condition == "lives")  condition = "dead";  }  friend ostream& operator<<(ostream& out, const Alive& a);//перегрузка оператора <<  };  class Fish : public Alive// класс "Рыба"  {  public:  Fish(string n = "", string c = "lives") : Alive(n, c) {};  };  class Bird : public Alive// класс "Птица"  {  public:  Bird(string n = "", string c = "lives") : Alive(n,c) {};  void eat(Fish &f)//съесть рыбу  {  f.change\_c();  }  };  class Animal : public Alive// класс "Животное"  {  public:  Animal(string n = "", string c = "lives") : Alive(n, c) {};  void eat(Alive &a)//съесть живое существо  {  a.change\_c();  }  };  ostream& operator<<(ostream& out, const Alive& a)  {  out << "Name: " << a.name << " Condition: " << a.condition;  return out;  };  int main()  {  Animal lion("lion");  Bird seagull("seagull");  Fish salmon("salmon");  seagull.eat(salmon);  lion.eat(seagull);  cout << lion << endl << seagull << endl << salmon << endl;  system("pause");  return 0;  } |

Вариант 3

|  |
| --- |
| #include<iostream> |
| #include<string>  using namespace std;  class Animal// абстрактный класс "животное"  {  protected:  string name;  public:  Animal(string n) : name(n) {};  virtual void voice() = 0;//виртуальная функция "подать голос"  };  class Cat : Animal// класс "кошка"  {  public:  Cat(string n) : Animal(n) {};  void voice()  {  cout << name << " - meow!" << endl;  }  };  class Dog : Animal// класс "собака"  {  public:  Dog(string n) : Animal(n) {};  void voice()  {  cout << name << " - woof!" << endl;  }  };  int main()  {  string s;  cout << "Input cat's name" << endl;  cin >> s;  Cat c = s;  cout << "Input dog's name" << endl;  cin >> s;  Dog d = s;  cin >> s;  c.voice();  d.voice();  system("pause");  return 0;  } |

**Выводы:**

В ходе выполнения Лабораторной работы №6 были использованы библиотека iostream, vector, string. Были реализованы методы, демонстрирующие работоспоссобность классов, перегружен оператор вывода. Были достигнуты результаты:

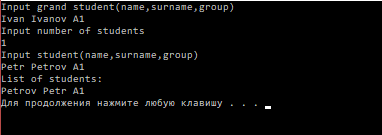


Рис. 15

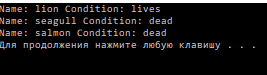


Рис. 16

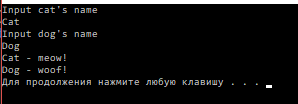


Рис. 17

<https://github.com/dneva/prac-lab/blob/master/lab6/lab6.1>

<https://github.com/dneva/prac-lab/blob/master/lab6/lab6.2>

<https://github.com/dneva/prac-lab/blob/master/lab6/lab6.3>

# Лабораторная работа №7

**Задание:**

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

**Ход выполнения:**

|  |
| --- |
| #include<iostream> |
| #include<string>  using namespace std;  class Human//Класс человек  {  protected:  string surname;  string name;  string midname;  int age;  public:  Human() {};  Human(string s, string n, string m, int a) :surname(s), name(n), midname(m), age(a) {};  virtual void print()=0;  virtual ~Human() { cout << "Human Destructor" << endl; };  };  class Student: public Human//Класс студент  {  private:  bool on\_lesson;  public:  Student(string s, string n, string m, int a, bool b) : Human(s, n, m, a), on\_lesson(b) {};  void print()  {  cout << "Surname: " << surname << endl;  cout << "Name: " << name << endl;  cout << "Midname: " << midname << endl;  cout << "Age: " << age << endl;  cout << "On lesson: " << on\_lesson;  }  };  class Boss : public Human// Класс босс  {  private:  int number\_of\_workers;  public:  Boss() {};  Boss(string s, string n, string m, int a, int now) : Human(s, n, m, a), number\_of\_workers(now) {};  void print()  {  cout << "Surname: " << surname << endl;  cout << "Name: " << name << endl;  cout << "Midname: " << midname << endl;  cout << "Age: " << age << endl;  cout << "Number of workers: " << number\_of\_workers << endl;  }  ~Boss() { cout << "Boss Destructor" << endl; }  };  int main()  {  string s, n, m;  int a, now;  bool b;  Human \*h1, \*h2;  cout << "Input student(Surname;Name;Midname;Age;On lesson)" << endl;  cin >> s >> n >> m >> a >> b;  h1 = new Student(s, n, m, a, b);  cout << "Input boss(Surname;Name;Midname;Age;Number of workers)" << endl;  cin >> s >> n >> m >> a >> now;  h2 = new Boss(s, n, m, a, now);  h1->print();  h2->print();  delete h1;  delete h2;  system("pause");  return 0; |
| } |

**Выводы:**

В ходе выполнения Лабораторной работы №7 были использованы библиотеки iostream, string. Была реализованы методы для демонстрации работоспособности производных классов. Были достигнуты результаты:

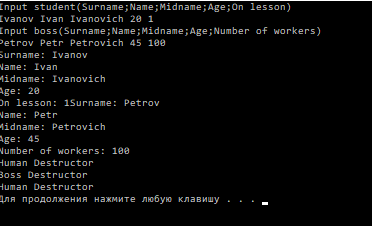


Рис. 18

<https://github.com/dneva/prac-lab/blob/master/lab7>

# Лабораторная работа №8

**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.

**Ход выполнения:**

Вариант 1

|  |
| --- |
| #include <iostream> |
| #include <string>  #include <vector>  using namespace std;  class Human//Класс человек  {  protected:  string surname;  string name;  string midname;  int age;  public:  Human() {};  Human(string s, string n, string m, int a) :surname(s), name(n), midname(m), age(a) {};  virtual void print()=0;  };  class Student: public Human//Класс студент  {  private:  bool on\_lesson;  public:  Student(string s, string n, string m, int a, bool b) : Human(s, n, m, a), on\_lesson(b) {};  void print()  {  cout << "Surname: " << surname << endl;  cout << "Name: " << name << endl;  cout << "Midname: " << midname << endl;  cout << "Age: " << age << endl;  cout << "On lesson: " << on\_lesson;  }  friend ostream& operator<<(ostream& out, const Student& s);//перегрузка оператора <<  };  ostream& operator<<(ostream& out, const Student& s)  {  out << s.surname << " " << s.name << " " <<s.midname << " " << s.age << " "<<s.on\_lesson;  return out;  };  int main()  {  vector<Student> v;//вектор  int n,a;  string s,na,m;  bool b;  cout << "Input number of students"<<endl;  cin >> n;  for(int i=0;i<n;i++)//заполнение вектора  {  cout << "Input student(Surname;Name;Midname;Age;On lesson)"<<endl;  cin >> s >> na >> m >> a >> b;  v.push\_back(Student(s, na, m, a, b));  }  cout << "Vector of Studenrs:"<< endl;  for(Student st : v)//вывод элементов вектора  {  cout << st << endl;  }  } |

Вариант 2

|  |
| --- |
| #include<iostream> |
| #include<list>  #include <iterator>  using namespace std;  class Complex  {  private:  double real;  double img;  public:  Complex(double r = 0, double i = 0);//конструктор  Complex& operator=(const Complex& right);//перегрузка оператора =  friend bool operator==(const Complex &left, const Complex &right);//перегрузка оператора ==  friend ostream& operator<<(ostream& out, const Complex& c);//перегрузка оператора <<  friend istream& operator>>(istream& in, Complex& c);//перегрузка оператора >>  };  Complex::Complex(double r, double i) : real(r), img(i) {};  Complex& Complex::operator=(const Complex& right) {  if (this == &right) {  return \*this;  }  real = right.real;  img = right.img;  return \*this;  };  bool operator==(const Complex &left, const Complex &right)  {  return (left.real == right.real && left.img == right.img);  };  ostream& operator<<(ostream& out, const Complex& c)  {  out << c.real << "+" << c.img << "i";  return out;  };  istream& operator>>(istream& in, Complex& c)  {  in >> c.real >> c.img;  return in;  };  int main()  {  int n;  double re,im;  Complex c;  list<Complex> li;//список  cout<<"Input number of complex numbers"<<endl;  cin >> n;  for(int i=0;i<n;i++)//заполнение списка  {  cout << "Input complex number" << endl;  cin >> re >> im;  li.push\_back(Complex(re,im));  }  copy(li.begin(), li.end(), ostream\_iterator<Complex>(cout," "));//вывод списка  system("pause");  return 0;  } |

Вариант 3

|  |
| --- |
| #include<iostream> |
| #include<string>  #include<queue>  #include<cmath>  using namespace std;  class Vehicle// абстрактный класс "транспортное средство"  {  protected:  double x, y, dir;  public:  Vehicle(double xx, double yy, double d) : x(xx), y(yy), dir(d) {};  virtual void move(double len) = 0;//движение транспорта  void setDir(double nd) //установить направление  {  dir = nd;  }  void returnPosition()//вывод позиции  {  cout << "x = " << x << " ; y = " << y << "; direction = " << dir << endl;  }  friend ostream& operator<<(ostream& out, const Vehicle& c);//перегрузка оператора <<  };  class Car : public Vehicle// класс "автомобиль"  {  private:  double fuel, fcons;  int passengers;  public:  Car(double xx=0, double yy=0, double d=0, double fc=0) : Vehicle(xx, yy, d), passengers(0), fuel(0), fcons(fc) {};  void move(double len)  {  if (fuel >= len / fcons)  {  x += len\*cos(dir);  y += len\*sin(dir);  fuel -= len / fcons;  }  else cout << "no fuel" << endl;  }  void refuel(double f)//заправить бензин  {  fuel += f;  cout << fuel << endl;  }  void enter(int n)//пассажиры зашли  {  if (n >= 0 && n < 4)  passengers += n;  }  void exit(int n)//пассажиры вышли  {  if (n >= 0 && n <= passengers)  passengers -= n;  }  };  ostream& operator<<(ostream& out, const Vehicle& c)  {  out << "x = " << c.x << " ; y = " << c.y << "; direction = " << c.dir;  return out;  };  int main()  {  queue<Car> q;//очередь  int n;  cout << "Input numbers of car"<<endl;  cin >> n;  for(int i=0;i<n;i++)//заполнение очереди  {  q.push(Car(i,i,i));  }  n=0;  while(!q.empty())//перебор очереди  {  n++;  cout << "Car " << n << ": " << q.front() << endl;  q.pop();  }  system("pause");  return 0;  } |

**Выводы:**

В ходе выполнения Лабораторной работы №8 были использованы библиотеки iostream,string,vector,list,iterator, queue, cmath и ранее реализованные классы. Были достигнуты результаты:

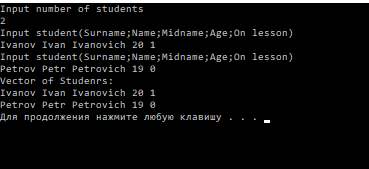


Рис. 19

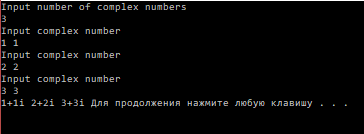


Рис. 20

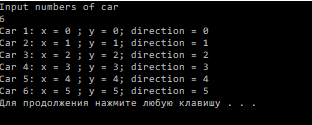


Рис. 21

<https://github.com/dneva/prac-lab/blob/master/lab8/lab8.1>

<https://github.com/dneva/prac-lab/blob/master/lab8/lab8.2>

<https://github.com/dneva/prac-lab/blob/master/lab8/lab8.3>