Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное

образовательное учреждение высшего образования

«Комсомольский-на-Амуре государственный университет»

Факультет компьютерных технологий

Кафедра «МОП ЭВМ»

Курсовая работа

по дисциплине «?»

Вариант 11

Студент группы 7ИСб-1 И. Нозимзода

Преподаватель С.Ю. Александров

2019

Содержание

[Содержание 2](#_Toc10708911)

[Задания 4](#_Toc10708912)

[Введение 5](#_Toc10708913)

[1 Классы 6](#_Toc10708914)

[Описание программы 6](#_Toc10708915)

[Текст программы 6](#_Toc10708916)

[Тестирование программы 10](#_Toc10708917)

[2 Наследование 13](#_Toc10708918)

[Описание программы 13](#_Toc10708919)

[Текст программы 13](#_Toc10708920)

[Тестирование программы 21](#_Toc10708921)

[3 Шаблоны классов. Обработка исключительных ситуаций 25](#_Toc10708922)

[Описание программы 25](#_Toc10708923)

[Текст программы 25](#_Toc10708924)

[Тестирование программы 28](#_Toc10708925)

[4 Стандартные потоки 30](#_Toc10708926)

[Описание программы 30](#_Toc10708927)

[Текст программы 30](#_Toc10708928)

[Тестирование программы 32](#_Toc10708929)

[5 Файловые и строковые потоки. Строки класса string. 34](#_Toc10708930)

[Описание программы 34](#_Toc10708931)

[Текст программы 34](#_Toc10708932)

[Тестирование программы 35](#_Toc10708933)

[Заключение 36](#_Toc10708934)

[Список использованных источников 37](#_Toc10708935)

Задания

1. Классы.
2. Наследование.
3. Шаблоны классов. Обработка исключительных ситуаций
4. Стандартные потоки.
5. Файловые и строковые потоки. Строки класса string.

Введение

Язык С++ как средство обучения программированию обладает рядом несомненных достоинств. Он хорошо организован, строг, большинство его конструкций логичны и удобны. Развитые средства диагностики и редактирования кода делают процесс программирования приятным и эффективным.

Немаловажно, что С++ является не учебным, а профессиональным языком, предназначенным для решения широкого спектра задач, и в первую очередь - в быстро развивающейся области создания распределенных приложений.

# Классы

Классы и объекты в С++ являются основными концепциями объектно-ориентированного программирования — ООП. Объектно-ориентированное программирование — расширение структурного программирования, в котором основными концепциями являются понятия классов и объектов.

## Описание программы

Реализация системы классов для работы с одномерными массивами строк. Составить описание класса одномерных массивов строк, каждая строка задается длиной и указателем на выделенную для нее память.

Предусмотреть контроль выхода за границы массивов, возможность обращения в отдельных строк массива по индексам, выполнения операций поэлементного сцепления двух массивов с образованием нового массива, слияния двух массивов с исключением повторяющихся элементов, а также вывод на экран элемента массива и всего массива.

Написать программу, демонстрирующую работу с этим классом. Программа должна содержать меню, позволяющее осуществить проверку всех методов класса.

## Текст программы

Проект состоит из одного файла исходного кода, который приведен в листине 1.1.

Листинг . – Текст файла Lab10.cpp

|  |
| --- |
| #include <iostream>  #include <iomanip>  using namespace std;  class PointerArray  {  private:  char\*\* \_string;  int\* \_stringLength;  int \_size;  public:  void create();  void initialize();  void merge(int n1, int n2);  void mergeNotRepeatingElements(int n3, int n4);  void show();  void showByIndex(int number);  void search(int rec, int field);  };  void PointerArray::create() //Создание массива строк  {  // выделение памяти под массивы  cout << "Количество строк: "; cin >> \_size;  \_stringLength = new int[\_size];  \_string = new char\* [\_size];  // заполняем массива длины строк  for (int i = 0; i < \_size; i++)  {  cout << "Количество символов в " << i + 1 << " строке: ";  cin >> \*(\_stringLength + i);  // добавляем одну единицу к длине для символа конца строки  \*(\_stringLength + i) += 1;  }  // выделяем память под каждую строку  for (int i = 0; i < \_size; i++) {  \*(\_string + i) = new char[\_stringLength[i]];  }  cout << endl;  }  // Инициализация массивов  void PointerArray::initialize()  {  create();  cin.ignore();  for (int i = 0; i < \_size; i++)  {  cout << "Ввод " << i + 1 << " строки (до " << \*(\_stringLength + i) - 1 << " символов): ";  cin.getline(\*(\_string + i), \*(\_stringLength + i)); // char\*, int  if (!cin)  {  cin.clear();  cin.ignore();  throw exception();  }  }  }  // Ввывод всех строк  void PointerArray::show()  {  for (int i = 0; i < \_size; i++)  {  cout << \*(\_string + i);  cout << endl;  }  }  // Ввывод строк по индексу  void PointerArray::showByIndex(int number)  {  if (number < \_size)  {  for (int i = 0; i < \_size; i++)  {  if (i == number)  {  cout << \*(\_string + i) << endl;  }  }  }  else  {  cout << "Выход за пределы диапазона строк." << endl;  }  }  void PointerArray::search(int rec, int field)//Ввывод конкретного элемента конкретной строки (rec – номер строки выводимого массива, field – номер элемента в строке)  {  if (rec < \_size && field < \*(\_stringLength + rec))  {  cout << \*(\_string + rec) + field << endl;  }  else  {  cout << "Выход за пределы диапазона строк или элементов." << endl;  }  }  //Сцепление двух массивов. Значениям дополнительных переменных m1 и m2 присваиваются длины строк, которые необходимо сцепить. Их сумма есть длина нового массива.  void PointerArray::merge(int n1, int n2)  {  if (n1 < \_size && n2 < \_size)  {  cout << \*(\_string + n1) << \*(\_string + n2) << endl;  }  else  {  cout << "Выход за пределы диапазона строк." << endl;  }  }  //Слияние двух массивов  void PointerArray::mergeNotRepeatingElements(int n1, int n2)  {  if (n1 < \_size && n2 < \_size)  {  cout << \*(\_string + n1);  for (size\_t i = 0; i < \*(\_stringLength + n2); i++)  {  bool checkEntry = false;  for (size\_t j = 0; j < \*(\_stringLength + n1); j++)  {  if (\*(\*(\_string + n1) + j) == \*(\*(\_string + n2) + i))  {  checkEntry = true;  break;  }  }  if (!checkEntry)  {  cout << \*(\*(\_string + n2) + i);  }  }  }  else  {  cout << "Выход за пределы диапазона строк." << endl;  }  }  int main()  {  setlocale(LC\_ALL, "Russian");  PointerArray \_string;  int number, punkt, rec, field, n1, n2, n3, n4;  while (true)  {  cout << " Меню" << endl;  cout << "1 Создание массива строк"; cout << endl;  cout << "2 Вывод всех строк"; cout << endl;  cout << "3 Вывод строки по индексу"; cout << endl;  cout << "4 Сцепление двух массивов (поэлементное, с повторениями)"; cout << endl;  cout << "5 Слияние двух масивов (исключение повторяющихся элементов в сливаевом массиве)"; cout << endl;  cout << "6 Вывод конкретного элемента конкретной строки"; cout << endl;  cout << "7 Выход"; cout << endl;  cout << "Введите номер пункта меню: "; cin >> punkt;  switch (punkt)  {  case 1:  {  try  {  \_string.initialize();  cout << endl;  }  catch (const exception&)  {  cout << "bad alloc. Program error" << endl;  system("pause");  exit(-1);  }  }  break;  case 2:  {  \_string.show();  cout << endl;  }  break;  case 3:  {  cout << "Введите номер массива (нумерация начинаеться с 0!): "; cin >> number;  \_string.showByIndex(number);  cout << endl;  }  break;  case 4:  {  cout << "Введите номер 1-го массива: "; cin >> n1;  cout << "Введите номер 2-го массива: "; cin >> n2;  \_string.merge(n1, n2);  cout << endl;  }  break;  case 5:  {  cout << "Введите номер 1-го массива: "; cin >> n3;  cout << "Введите номер 2-го массива: "; cin >> n4;  \_string.mergeNotRepeatingElements(n3, n4);  cout << endl;  cout << endl;  }  break;  case 6:  {  cout << "Введите номер строки массива: "; cin >> rec;  cout << "Введите номер столбца: "; cin >> field;  \_string.search(rec, field);  cout << endl;  }  break;  case 7:  exit(0);  break;  }  }  } |

## Тестирование программы

Результаты работы программы приведены на рисунках 1.1 - 1.3.

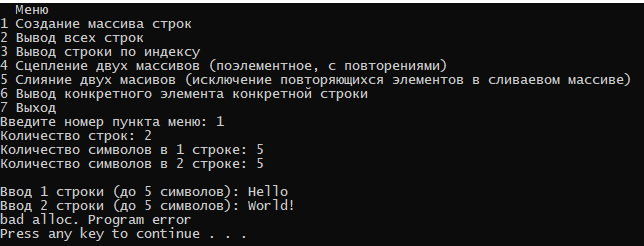


Рисунок . – Результат работы программы

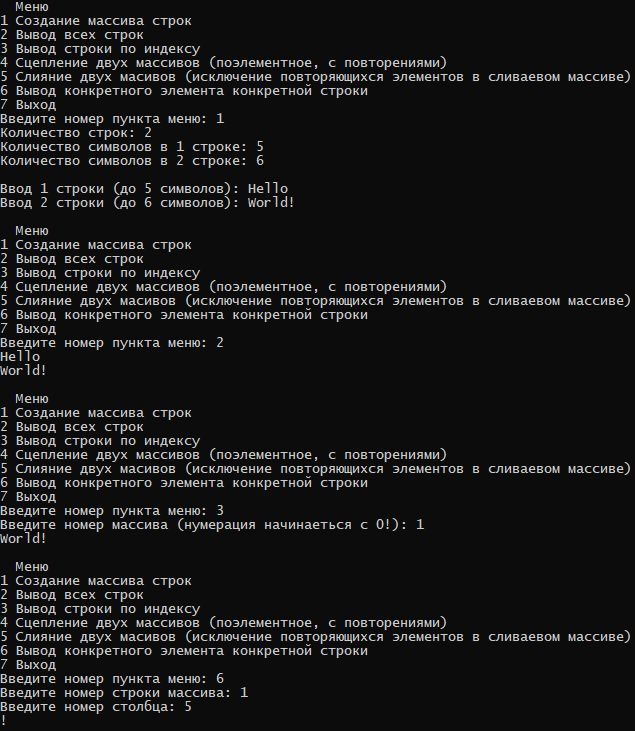


Рисунок .2 – Результат работы программы

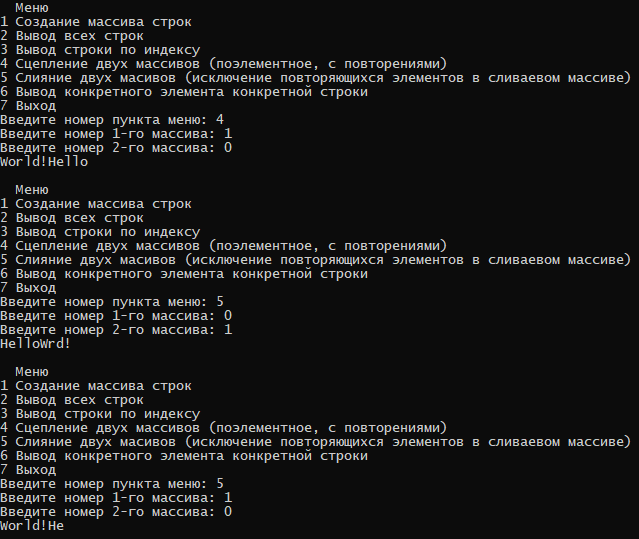


Рисунок .3 – Результат работы программы

# Наследование

Классы в объектно-ориентированных программах используются для моделирования концепций реального и программного мира. Концепции или сущности предметной области находятся в различных взаимоотношениях. Одно из таких взаимоотношений ‑ отношение наследования (именуемое также отношением родитель потомок или отношением обобщение–специализация).

## Описание программы

Написать программу, демонстрирующую работу с объектами двух типов, Quadrate и Triangle, для чего создать систему соответствующих классов. Каждый объект должен иметь идентификатор (в виде произвольной строки символов) и одно или несколько полей для хранения состояния (текущего значения) объекта. Клиенту (функции main) должны быть доступны следующие основные операции (методы): создать объект, удалить объект, показать значение объекта, Move(), Compare(T&, T&). Операции по созданию и удалению объектов инкапсулировать в классе Factory.

Предусмотреть меню, позволяющее продемонстрировать заданные операции. При необходимости в разрабатываемые классы добавляются дополнительные методы (например, конструктор копирования, операция присваивания и т.п.) для обеспечения надлежащего функционирования этих классов.

## Текст программы

Проект состоит из 5 файлов исходного кода, которые приведен в листингах 2.1-2.5.

Листинг . – Текст файла Lab11.cpp

|  |
| --- |
| #include "Factory.h"  int main()  {  setlocale(LC\_ALL, "Russian");  cout << "Демонстрация возможностей классов Quadrate(квадрат) и Triangle(треугольник)" << endl;  char Control;  while (true)  {  cout << endl << "Интерфейс программы:" << endl;  cout << "1. Объект Quadrate (квадрат)" << endl;  cout << "2. Объект Triangle (треугольник)" << endl;  cout << ">";  cin >> Control;  switch (Control)  {  case '1':  {  FactoryQuadrate factoryQuadrate;  cout << "Создание квадрата с координатами по умолчанию.." << endl;  bool breakCheck = true;  while (breakCheck)  {  cout << "\nQuadrate (квадрат)" << endl;  cout << "1. Создать квадрат с новыми координатами" << endl;  cout << "2. Удалить квадрат" << endl;  cout << "3. Показать координаты квадрата" << endl;  cout << "4. Переместить квадрат" << endl;  cout << "5. Сравнение площадей квадратов" << endl;  cout << "6. Вернуться к выбору фигуры" << endl;  cout << ">";  cin >> Control;  switch (Control)  {  case '1':  {  factoryQuadrate.addObject();  }  break;  case '2':  {  factoryQuadrate.deleteObject();  }  break;  case '3':  {  factoryQuadrate.quadrate\_.Result();  }  break;  case '4':  {  factoryQuadrate.quadrate\_.Move();  }  break;  case '5':  {  factoryQuadrate.quadrate\_.Compare();  }  break;  case '6':  breakCheck = false;  break;  default: continue;  }  }  }  break;  case '2':  {  FactoryTriangle factoryPentagon;  bool breakCheck = true;  while (breakCheck)  {  cout << "\Triangle (треугольник)" << endl;  cout << "1. Создать треугольник с новыми координатами" << endl;  cout << "2. Удалить треугольник" << endl;  cout << "3. Показать координаты треугольника" << endl;  cout << "4. Переместить треугольник" << endl;  cout << "5. Сравнение площадей треугольноков" << endl;  cout << "6. Вернуться к выбору фигуры" << endl;  cout << ">";  cin >> Control;  switch (Control)  {  case '1':  {  factoryPentagon.addObject();  }  break;  case '2':  {  factoryPentagon.deleteObject();  }  break;  case '3':  {  factoryPentagon.pentagon\_.Result();  }  break;  case '4':  {  factoryPentagon.pentagon\_.Move();  }  break;  case '5':  {  factoryPentagon.pentagon\_.Compare();  }  break;  case '6':  breakCheck = false;  break;  default: continue;  }  }  }  break;  }  }  return 0;  } |

Листинг .2– Текст файла Factory.h

|  |
| --- |
| #include "Figures.h"  class FactoryQuadrate {  public:  FactoryQuadrate() {}  void addObject();  void deleteObject();  Quadrate quadrate\_;  };  class FactoryTriangle {  public:  FactoryTriangle() {}  void addObject();  void deleteObject();  Triangle pentagon\_;  }; |

Листинг .3 – Текст файла Factory.cpp

|  |
| --- |
| #include "Factory.h"  void FactoryQuadrate::addObject() {  quadrate\_.Input();  }  void FactoryQuadrate::deleteObject() {  quadrate\_.Delete();  }  void FactoryTriangle::addObject() {  pentagon\_.Input();  }  void FactoryTriangle::deleteObject() {  pentagon\_.Delete();  } |

Листинг .4 – Текст файла Figures.h

|  |
| --- |
| #include <iostream>  #include <string>  #include <cstdlib>  using namespace std;  class Tetragon  {  protected:  double x1, x2, y1, y2;  size\_t ID\_SEQUENCE\_;  bool deleted\_;  public:  Tetragon(double a, double b, double c, double d) : x1(a), x2(b), y1(c), y2(d), ID\_SEQUENCE\_(rand()), deleted\_(false) {};  Tetragon() : x1(0), x2(5), y1(0), y2(5), ID\_SEQUENCE\_(rand()), deleted\_(false) {};  ~Tetragon() { }  virtual void Input(); // метод ввода с клавиатуры  void Delete();  void Result(); // метод вывода на экран  void Move(); // перемещение  };  class Quadrate : public Tetragon  {  public:  double wight;  void Input();  void Compare();  };  class Triangle  {  protected:  double x1, x2, x3, y1, y2, y3;  size\_t ID\_SEQUENCE\_;  bool deleted\_;  public:  Triangle() : x1(0), y1(0), x2(1), y2(1), x3(0), y3(2), ID\_SEQUENCE\_(rand()), deleted\_(false) {};  ~Triangle() { }  virtual void Input(); // метод ввода с клавиатуры  void Delete();  void Result(); // метод вывода на экран  void Move(); // перемещение  void Compare(); // сравнение площади  }; |

Листинг .5 – Текст файла Figures.cpp

|  |
| --- |
| #include "Figures.h"  #include <list>  #include <algorithm>  double SquareTriangle(double x1, double x2, double x3, double y1, double y2, double y3)  {  double r;  double a = sqrt(pow((x1 - x2), 2) + pow((y1 - y2), 2));  double b = sqrt(pow((x2 - x3), 2) + pow((y2 - y3), 2));  double c = sqrt(pow((x3 - x1), 2) + pow((y3 - y1), 2));  double p = (a + b + c) / 2;  r = sqrt(p \* (p - a) \* (p - b) \* (p - c));  return r;  }  double SquareQuadrate()  {  return 0;  }  void Tetragon::Input() {  cout << "Введите координату х левых точек" << endl; cin >> x1;  cout << "Введите координату х правых точек" << endl; cin >> x2;  cout << "Введите координату y верхних точек" << endl; cin >> y1;  cout << "Введите координату y нижних точек" << endl; cin >> y2;  ID\_SEQUENCE\_ = rand();  deleted\_ = false;  }  void Tetragon::Delete()  {  if (deleted\_)  cout << "Квадрат уже удалён!" << endl;  else  {  cout << "Квадрат удалён" << endl;  deleted\_ = true;  }  }  void Tetragon::Result() {  if (!deleted\_)  {  cout << "Идентификатор: ID" << ID\_SEQUENCE\_ << endl;  cout << "Координаты:" << endl;  cout << " A:" << "(" << x1 << "," << y1 << ")" << endl;  cout << " B:" << "(" << x2 << "," << y1 << ")" << endl;  cout << " C:" << "(" << x2 << "," << y2 << ")" << endl;  cout << " D:" << "(" << x1 << "," << y2 << ")" << endl;  }  else cout << "Сначала создайте квадрат" << endl;  }  void Tetragon::Move() {  if (!deleted\_)  {  cout << " На сколько единиц вы хотите переместить квадрат по оси X ?" << endl;  double dx;  cin >> dx;  cout << " На сколько единиц вы хотите переместить квадрат по оси Y ?" << endl;  double dy;  cin >> dy;  x1 += dx; x2 += dx;  y1 += dy; y2 += dy;  cout << "Квадрат перемещён" << endl;  cout << "Текущие координаты:" << endl;  cout << " A:" << "(" << x1 << "," << y1 << ")" << endl;  cout << " B:" << "(" << x2 << "," << y1 << ")" << endl;  cout << " C:" << "(" << x2 << "," << y2 << ")" << endl;  cout << " D:" << "(" << x1 << "," << y2 << ")" << endl;  }  else cout << "Сначала создайте квадрат" << endl;  }  void Quadrate::Input()  {  cout << "Введите координату х левой нижней точки квадрата" << endl;  cin >> x1;  cout << "Введите координату y левой нижней точки квадрата" << endl;  cin >> y1;  cout << "Введите ширину (и одновременно высоту) квадрата" << endl;  cin >> wight;  x2 = x1 + wight;  y2 = y1 + wight;  deleted\_ = false;  }  void Quadrate::Compare() {  if (!deleted\_)  {  double x3, x4, y3, y4;  cout << "Координаты второго квадрата:" << endl;  cout << "Введите координату х левой нижней точки квадрата" << endl;  cin >> x3;  cout << "Введите координату y левой нижней точки квадрата" << endl;  cin >> y3;  cout << "Введите ширину (и одновременно высоту) квадрата" << endl;  double tempW;  cin >> tempW;  x4 = x3 + tempW;  y4 = y3 + tempW;  cout << endl;  cout << "Вы ввели следующие координаты 1 квадрата:" << endl;  cout << " A:" << "(" << x1 << "," << y1 << ")" << endl;  cout << " B:" << "(" << x2 << "," << y1 << ")" << endl;  cout << " C:" << "(" << x2 << "," << y2 << ")" << endl;  cout << " D:" << "(" << x1 << "," << y2 << ")" << endl;  cout << endl;  cout << "Вы ввели следующие координаты 2 квадрата:" << endl;  cout << " A:" << "(" << x3 << "," << y3 << ")" << endl;  cout << " B:" << "(" << x4 << "," << y3 << ")" << endl;  cout << " C:" << "(" << x4 << "," << y4 << ")" << endl;  cout << " D:" << "(" << x3 << "," << y4 << ")" << endl;  cout << endl;  double square1 = wight \* wight;  double square2 = tempW \* tempW;  if (square1 > square2)  {  cout << "Площадь первого квадрата больше второго" << endl;  }  else if (square1 == square2)  {  cout << "Площади одинаковы" << endl;  }  else cout << "Площадь второго квадрата больше первого" << endl;  }  else cout << "Сначала создайте квадрат" << endl;  }  void Triangle::Input() {  cout << "Введите координату точки х1: "; cin >> x1;  cout << "Введите координату точки y1: "; cin >> y1; cout << endl;  cout << "Введите координату точки х2: "; cin >> x2;  cout << "Введите координату точки y2: "; cin >> y2; cout << endl;  cout << "Введите координату точки х3: "; cin >> x3;  cout << "Введите координату точки y3: "; cin >> y3; cout << endl;  ID\_SEQUENCE\_ = rand();  deleted\_ = false;  }  void Triangle::Delete()  {  if (deleted\_)  cout << "Треугольник уже удалён!" << endl;  else  {  cout << "Треугольник удалён" << endl;  deleted\_ = true;  }  }  void Triangle::Result() {  if (!deleted\_)  {  cout << "Идентификатор: ID" << ID\_SEQUENCE\_ << endl;  cout << "Координаты:" << endl;  cout << " A:" << "(" << x1 << "," << y1 << ")" << endl;  cout << " B:" << "(" << x2 << "," << y2 << ")" << endl;  cout << " C:" << "(" << x3 << "," << y3 << ")" << endl;  }  else cout << "Сначала создайте треугольник" << endl;  }  void Triangle::Move() {  if (!deleted\_)  {  cout << " На сколько единиц вы хотите переместить треугольник по оси X ?" << endl;  double dx;  cin >> dx;  cout << " На сколько единиц вы хотите переместить треугольник по оси Y ?" << endl;  double dy;  cin >> dy;  x1 += dx; x2 += dx; x3 += dx;  y1 += dy; y2 += dy; y3 += dy;  cout << "Треугольник перемещён" << endl;  cout << "Текущие координаты:" << endl;  cout << " A:" << "(" << x1 << "," << y1 << ")" << endl;  cout << " B:" << "(" << x2 << "," << y2 << ")" << endl;  cout << " C:" << "(" << x3 << "," << y3 << ")" << endl;  }  else cout << "Сначала создайте треугольник" << endl;  }  void Triangle::Compare() {  if (!deleted\_)  {  double x11, x22, x33, x44, x55, y11, y22, y33;  cout << "Координаты второго треугольника:" << endl;  cout << "Введите координату точки х1: "; cin >> x11;  cout << "Введите координату точки y1: "; cin >> y11; cout << endl;  cout << "Введите координату точки х2: "; cin >> x22;  cout << "Введите координату точки y2: "; cin >> y22; cout << endl;  cout << "Введите координату точки х3: "; cin >> x33;  cout << "Введите координату точки y3: "; cin >> y33; cout << endl;  cout << endl;  cout << "Вы ввели следующие координаты 1 треугольника:" << endl;  cout << " A:" << "(" << x1 << "," << y1 << ")" << endl;  cout << " B:" << "(" << x2 << "," << y2 << ")" << endl;  cout << " C:" << "(" << x3 << "," << y3 << ")" << endl;  cout << endl;  cout << "Вы ввели следующие координаты 2 треугольника:" << endl;  cout << " A:" << "(" << x11 << "," << y11 << ")" << endl;  cout << " B:" << "(" << x22 << "," << y22 << ")" << endl;  cout << " C:" << "(" << x33 << "," << y33 << ")" << endl;  cout << endl;  double square1 = SquareTriangle(x1, x2, x3, y1, y2, y3);  double square2 = SquareTriangle(x11, x22, x33, y11, y22, y33);  if (square1 > square2)  {  cout << "Площадь первого треугольника больше второго" << endl;  }  else if (square1 == square2)  {  cout << "Площади одинаковы" << endl;  }  else cout << "Площадь второго треугольника больше первого" << endl;  }  else cout << "Сначала создайте треугольник" << endl;  } |

## Тестирование программы

Результат работы программы приведен на рисунках 2.1 - 2.4.

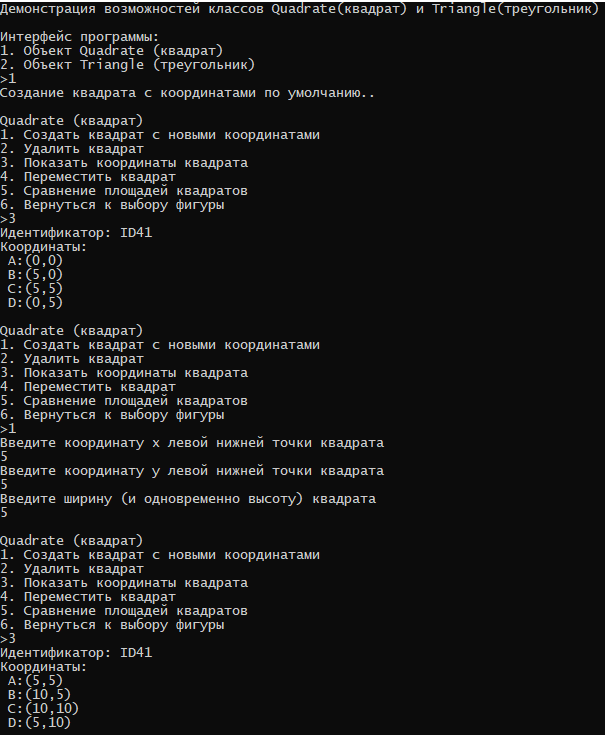


Рисунок .1 – Результат работы программы по наследованию

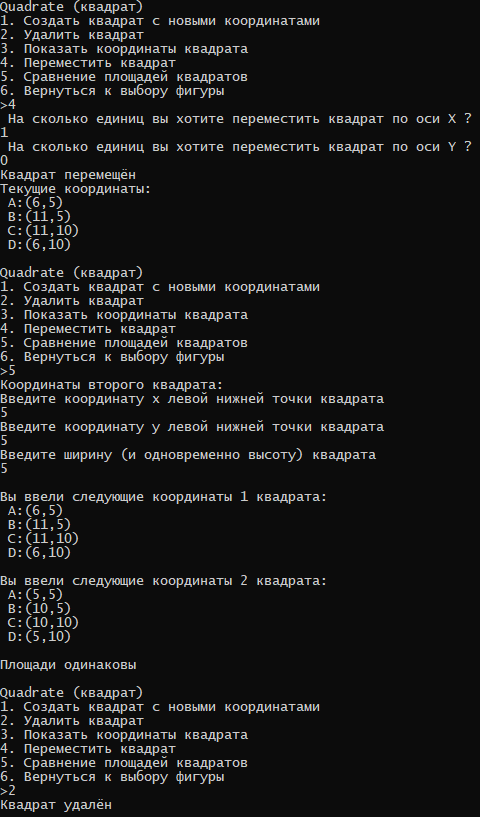


Рисунок .2 – Результат работы программы по наследованию

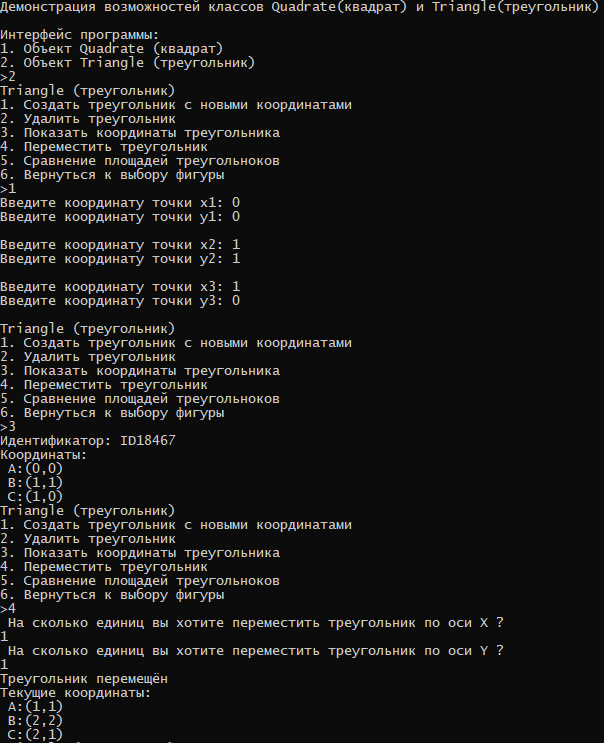


Рисунок 2.3 – Результат работы программы по наследованию

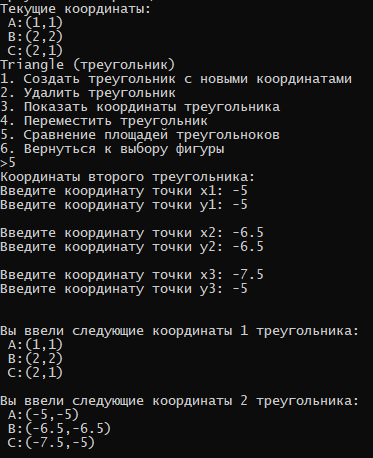


Рисунок 2.4 – Результат работы программы по наследованию

# Шаблоны классов. Обработка исключительных ситуаций

Шаблоны классов наряду с шаблонами функций поддерживают парадигму обобщенного программирования, то есть программирования с использованием типов в качестве параметров. Механизм шаблонов в C++ допускает применение абстрактного типа в качестве параметра при определении класса или функции. После того как шаблон класса определен, он может использоваться для определения конкретных классов.

## Описание программы

Требуется создать шаблон некоторого целевого класса Queue. В каждом варианте уточняются требования к реализации — указанием на применение некоторого серверного класса std::list. Это означает, что объект класса std::list используется как элемент класса Queue.

Во всех вариантах показать в клиенте main использование созданного класса, включая ситуации, приводящие к генерации исключений. Показать инстанцирование шаблона для типов int, double, std::string.

## Текст программы

Проект состоит из двух файла исходного кода, которые приведены в листингах 3.1,3.2.

Листинг 3. – Текст файла Lab12.cpp

|  |
| --- |
| #include "Queue.h"  int main()  {  setlocale(LC\_ALL, "rus");  cout << "Queue абстрактный тип данных с дисциплиной доступа к элементам «первый пришёл — первый вышел»." << endl;  cout << "Добавление элемента возможно лишь в конец очереди, выборка — только из начала очереди." << endl << endl;  cout << "Демонстрация целевого шаблонного класса Queue с разными типами данных" << endl << endl;  try  {  // для int(для всего остального делается аналогично)  Queue<int> queueInt;  cout << "Добавление в очередь значений типа int от 1 до 5" << endl;  for (int i = 1; i < 6; ++i)  queueInt.push\_back(i);  cout << "Вывод очереди: ";  queueInt.show();  cout << "Добавление в очередь значения 1" << endl;  queueInt.push\_back(1);  cout << "Первый элемент очереди: " << queueInt.front() << endl;  cout << "Последний элемент очереди: " << queueInt.back() << endl;  cout << "Освобождение очереди" << endl;  while (!queueInt.empty())  {  cout << "Освобождаемый элемент: " << queueInt.front() << endl;  queueInt.pop\_front();  }  cout << "Попытка удалить ещё один элемент из очереди.." << endl;  // исключение т.к. стек уже пуст  queueInt.pop\_front();  }  catch (std::underflow\_error& e)  {  std::cerr << e.what() << std::endl;  }  cout << endl;  try  {  Queue<double> queueDouble;  cout << "Добавление в очередь значений типа double от 1.5 до 5.5" << endl;  for (double i = 1.5; i < 6; ++i)  queueDouble.push\_back(i);  cout << "Вывод очереди: ";  queueDouble.show();  cout << "Добавление в очередь значения 1.5" << endl;  queueDouble.push\_back(1.5);  cout << "Первый элемент очереди: " << queueDouble.front() << endl;  cout << "Последний элемент очереди: " << queueDouble.back() << endl;  cout << "Освобождение очереди" << endl;  while (!queueDouble.empty())  {  cout << "Освобождаемый элемент: " << queueDouble.front() << endl;  queueDouble.pop\_front();  }  cout << "Попытка удалить ещё один элемент из очереди.." << endl;  queueDouble.pop\_front();  }  catch (std::underflow\_error& e)  {  std::cerr << e.what() << std::endl;  }  cout << endl;  try  {  Queue<string> queueString;  cout << "Добавление в очередь значений типа string от Hello1 до Hello5" << endl;  for (int i = 1; i < 6; ++i)  {  string str = "Hello" + to\_string(i);  queueString.push\_back(str);  }  cout << "Вывод очереди: ";  queueString.show();  cout << "Добавление в очередь значение Hello1" << endl;  queueString.push\_back("Hello1");  cout << "Первый элемент очереди: " << queueString.front() << endl;  cout << "Последний элемент очереди: " << queueString.back() << endl;  cout << "Освобождение очереди" << endl;  while (!queueString.empty())  {  cout << "Освобождаемый элемент: " << queueString.front() << endl;  queueString.pop\_front();  }  cout << "Попытка удалить ещё один элемент из очереди.." << endl;  queueString.pop\_front();  }  catch (std::underflow\_error& e)  {  std::cerr << e.what() << std::endl;  }  system("pause");  return 0;  } |

Листинг 3.2 – Текст файла Queue.h

|  |
| --- |
| #pragma once  #include <iostream>  #include <list>  #include <string>  #include <stdexcept>  using namespace std;  template < typename T, typename Container = std::list<T> >  class Queue  {  private:  Container container\_;  public:  explicit Queue(const Container& container = Container())  : container\_(container)  {  }  // первый в очереди  T front();  // последний в очереди  T back();  void push\_back(const T& item);  void pop\_front();  void show();  int size();  bool empty() const;  };  template<typename T, typename Container>  inline T Queue<T, Container>::front()  {  return container\_.front();  }  template<typename T, typename Container>  inline T Queue<T, Container>::back()  {  return container\_.back();  }  template<typename T, typename Container>  void Queue<T, Container>::push\_back(const T& item)  {  container\_.push\_back(item);  }  template<typename T, typename Container>  void Queue<T, Container>::pop\_front()  {  if (container\_.empty()) throw std::underflow\_error("Исключение: При удалении последнего элемента очерели произошла ошибка. Очередь пуста.");  container\_.pop\_front();  }  template<typename T, typename Container>  inline void Queue<T, Container>::show()  {  for (auto el : container\_)  {  cout << el << " ";  }  cout << endl;  }  template<typename T, typename Container>  inline int Queue<T, Container>::size()  {  return container\_.size();  }  template<typename T, typename Container>  bool Queue<T, Container>::empty() const  {  return container\_.empty();  } |

## Тестирование программы

Результат работы программы приведен на рисунке 3.1.

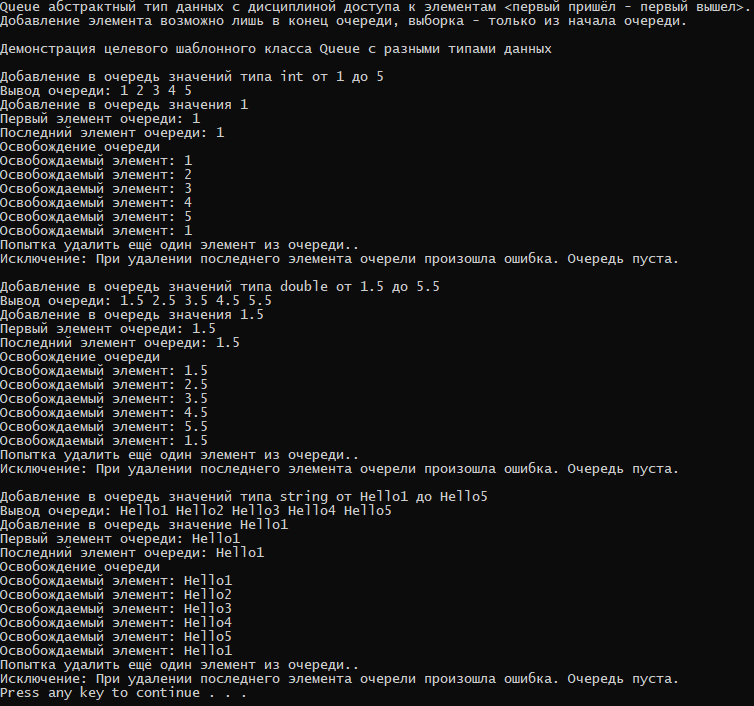


Рисунок 3.1 – Результат работы программы по шаблонам классов

# Стандартные потоки

Для поддержки потоков библиотека C++ содержит иерархию классов, построенную на основе двух базовых классов — ios и streambuf. Класс ios содержит базовые средства управления потоками, являясь родительским для других классов ввода вывода. Класс streambuf обеспечивает общие средства управления буферами потоков и их взаимодействие с физическими устройствами, являясь родительским для других буферных классов.

## Описание программы

Определить класс с именем MARSH, содержащий следующие поля: название начального пункта маршрута, название конечного пункта маршрута, номер маршрута. Определить методы доступа к этим полям и перегруженные операции извлечения и вставки для объектов типа MARSH.

Написать программу, выполняющую следующие действия: ввод с клавиатуры данных в массив, состоящий из восьми объектов типа MARSH; записи должны быть упорядочены по номерам маршрутов; вывод на экран информации о маршрутах, которые начинаются или кончаются в пункте, название которого введено с клавиатуры (если таких маршрутов нет, выдать на дисплей соответствующее сообщение).

## Текст программы

Проект состоит из одного файла исходного кода, который приведен в листинге 4.1.

Листинг 4. – Текст файла Lab13.cpp

|  |
| --- |
| #include <iostream>  #include <string>  #include <vector>  #include <Windows.h>  #include <algorithm>  using namespace std;  class Marsh  {  public:  friend istream& operator >> (istream& in, Marsh& ob);  friend ostream& operator<< (ostream& out, Marsh& ob);  friend class MarshCollection;  double getNumber()  {  return number;  }  private:  string \_startName;  string \_endName;  int number;  };  istream& operator >> (istream& in, Marsh& ob)  {  cin.ignore();  cout << "Название начального пункта маршрута: ";  getline(in,ob.\_startName);  cout << "Название конечного пункта маршрута: ";  getline(in, ob.\_endName);  cout << "Номер маршрута (целое число): "; cin >> ob.number;  return in;  }  ostream& operator<< (ostream& out, Marsh& ob)  {  out << "Название начального пункта маршрута: " << ob.\_startName << "\n";  out << "Название конечного пункта маршрута: " << ob.\_endName << "\n";  out << "Номер маршрута: " << ob.number << "\n";  return out;  }  class MarshCollection  {  private:  vector <Marsh> \_marshCollection;  public:  MarshCollection()  {  \_marshCollection.reserve(0);  }  MarshCollection(size\_t numberStudents)  {  \_marshCollection.reserve(numberStudents);  }  void sort()  {  std::sort(\_marshCollection.begin(), \_marshCollection.end(), []( Marsh & a, Marsh & b)  {  return a.getNumber() < b.getNumber();  });  }  void initInput() {  for (int i = 0; i < \_marshCollection.capacity(); i++)  {  Marsh N;  cout << "Ввод данных о " << i + 1 << " маршруте\n";  cin >> N;  cout << "\n";  \_marshCollection.push\_back(N);  }  }  void show()  {  cout << "База данных маршрутов:\n\n";  for (int i = 0; i < \_marshCollection.capacity(); i++)  {  cout << \_marshCollection[i];  cout << "\n";  }  }  void searchMarsh()  {  cin.ignore();  string str;  cout << "Поиск маршрутов по названию пунка (начала или конца): ";  getline(cin,str);  bool z = false;  for (auto marsh : \_marshCollection)  {  if (marsh.\_startName == str || marsh.\_endName == str)  {  cout << marsh;  z = true;  }  }  if (!z)  cout << "Такие названия пунктов не найдены" << endl;  }  };  int main()  {  SetConsoleCP(1251);  SetConsoleOutputCP(1251);  //system("color 0A ");  size\_t numberMarshs;  cout << "Введите количество маршрутов: ";  cin >> numberMarshs;  cout << "\n";  MarshCollection marshs(numberMarshs);  marshs.initInput();  marshs.sort();  marshs.show();  marshs.searchMarsh();  system("pause");  return 0;  } |

## Тестирование программы

Результат работы программы приведен на рисунке 4.1.

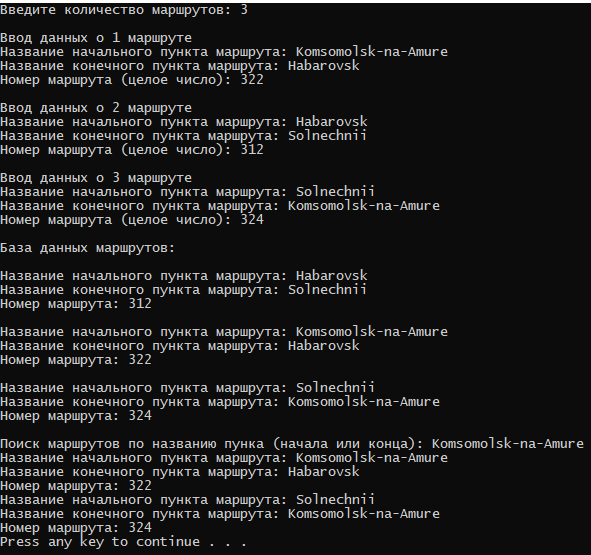


Рисунок 4.1 – Результат работы программы по стандартным потокам

# Файловые и строковые потоки. Строки класса string.

Для поддержки файлового ввода и вывода стандартная библиотека C++ содержит классы fstream, ofstream и ifstream. Как и стандартные потоки, файловые потоки обеспечивают гораздо более надежный ввод-вывод, чем старые функции библиотеки C. Для использования файловых потоков необходимо подключить заголовок <fstream>.

## Описание программы

Написать программу, которая считывает текст из файла и выводит на экран только строки, не содержащие двузначных чисел.

## Текст программы

Проект состоит из одного файла исходного кода, который приведен в листинге 5.1.

Листинг 5. – Текст файла Lab14.cpp

|  |
| --- |
| #include <iostream>  #include <string>  #include <sstream>  #include <fstream>  using namespace std;  int main()  {  setlocale(LC\_ALL, "Russian");  //system("color 0A ");  string path;  cout << "Введите полный путь к файлу ";  cin >> path;  ifstream fin(path);  if (!fin)  {  cout << "Файл не может быть открыт!" << endl << endl;  }  else  {  string line;  string word;  double found;  while (getline(fin, line))  {  stringstream ss;  ss << line;  bool entryCheck = false;  while (!ss.eof())  {  ss >> word;  if (stringstream(word) >> found)  {  double temp = found == 0 ? 1 : int(log10(found) + 1);  if (temp == 2)  {  entryCheck = true;  break;  }  }  }  if (!entryCheck)  {  cout << line << endl;  }  }  }  system("pause");  return 0;  } |

## Тестирование программы

Результат работы программы приведен на рисунках 5.1, 5.2.

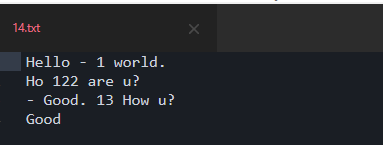


Рисунок 5.1 – Текстовый файл

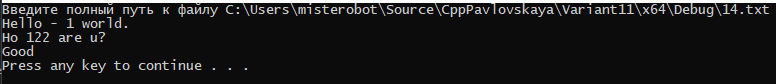


Рисунок 5.2 – Результат работы программы по файловым потокам

Заключение

В ходе изучения дисциплины «?» по изучению языка программирования C++ были рассмотрены такие темы как:

1. Классы.
2. Наследование.
3. Шаблоны классов. Обработка исключительных ситуаций
4. Стандартные потоки.
5. Файловые и строковые потоки. Строки класса string.

Полученные навыки и знания будут использоваться в дальнейших проектах.

Список использованных источников

1 Павловская Т. А., C++ Программирование на языке высокого уровня: Практикум. — СПб.: Питер, 2011. — 432 с.: ил. — (Серия «Учебное пособие»).