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 7

Студент группы 6ИСб-1 И. Нозимзода

Преподаватель С.Ю. Александров

2018

Содержание

[Содержание 2](#_Toc9287451)

[Задания 4](#_Toc9287452)

[Введение 5](#_Toc9287453)

[1 Классы 6](#_Toc9287454)

[1.1 Описание программы 6](#_Toc9287455)

[1.2 Текст программы 6](#_Toc9287456)

[1.3 Тестирование программы 10](#_Toc9287457)

[2 Наследование 13](#_Toc9287458)

[2.1 Описание программы 13](#_Toc9287459)

[2.2 Текст программы 13](#_Toc9287460)

[2.3 Тестирование программы 19](#_Toc9287461)

[3 Шаблоны классов. Обработка исключительных ситуаций 21](#_Toc9287462)

[3.1 Описание программы 21](#_Toc9287463)

[3.2 Текст программы 21](#_Toc9287464)

[3.3 Тестирование программы 24](#_Toc9287465)

[4 Стандартные потоки 25](#_Toc9287466)

[4.1 Описание программы 25](#_Toc9287467)

[4.2 Текст программы 25](#_Toc9287468)

[4.3 Тестирование программы 27](#_Toc9287469)

[5 Файловые и строковые потоки. Строки класса string. 29](#_Toc9287470)

[5.1 Описание программы 29](#_Toc9287471)

[5.2 Текст программы 29](#_Toc9287472)

[5.3 Тестирование программы 30](#_Toc9287473)

[6 Стандартная библиотека шаблонов 31](#_Toc9287474)

[6.1 Описание программы 31](#_Toc9287475)

[6.2 Текст программы 31](#_Toc9287476)

[6.3 Тестирование программы 39](#_Toc9287477)

[Заключение 42](#_Toc9287478)

[Список использованных источников 43](#_Toc9287479)

Задания

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

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

Составить описание класса прямоугольников со сторонами, параллельными осям координат.

Реализовать:

- перемещения прямоугольников на плоскости,

- изменение размеров,

- построение наименьшего прямоугольника, содержащего два заданных прямоугольника, и прямоугольника, являющегося общей частью (пересечением) двух прямоугольников.

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

## Текст программы

Проект состоит из трёх файлов исходного кода, которые приведен в листинах 1.1-1.3.

Листинг . – Текст файла Lab10.cpp

|  |
| --- |
| #include "Rectangle.h"  #include <iostream>  using namespace std;  int main()  {  setlocale(LC\_ALL, "Russian");  cout << "Создание прямоугольника без параметров (использование конструктора по умолчанию)" << endl;  Rectangle ObjectA; // Создаем объект без параметров, то есть будет использован конструктор по умолчанию  char Control;  while (true)  {  cout << endl;  cout << "Интерфейс программы:" << endl;  cout << "1. Ввод данных" << endl;  cout << "2. Вывод" << endl;  cout << "3. Перемещение" << endl;  cout << "4. Изменение размера" << endl;  cout << "5. Минимальный прямоугольник содержащий 2 заданных" << endl;  cout << "6. Прямоугольник получившийся на пересечении 2 заданных" << endl;  cout << ">";  cin >> Control;  switch (Control)  {  case '1':  ObjectA.Input();  break;  case '2':  ObjectA.Result();  break;  case '3':  ObjectA.Peremeshenie();  break;  case '4':  ObjectA.Size();  break;  case '5':  ObjectA.Minimal();  break;  case '6':  ObjectA.Сross();  break;  default: break;  }  }  system("pause");  return 0;  } |

Листинг .2 – Текст файла Rectangle.h

|  |
| --- |
| class Rectangle {  private:  double x1, x2, y1, y2, dx, dy, r;  public:  // Конструктор с 4 параметрами  Rectangle(double a, double b, double c, double d) : x1(a), x2(b), y1(c), y2(d) {};  // Конструктор по умолчанию  Rectangle() : x1(2), x2(5), y1(2), y2(0) {};  ~Rectangle() { } // Деструктор  void Input(); // метод ввода с клавиатуры  void Result(); // метод вывода на экран  void Peremeshenie(); // перемещение  void Size(); // изменение размера  void Minimal(); // построение минимального  void Сross(); // пересечение 2 прямоугольников  }; |

Листинг .3 – Текст файла Rectangle.cpp

|  |
| --- |
| #include "Rectangle.h"  #include <iostream>  using namespace std;  void Rectangle::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;  }  void Rectangle::Result() {  cout << "Вы ввели следующие координаты прямоугольника:" << endl;  cout << " A:" << "(" << x1 << "," << y1 << ")" << endl;  cout << " B:" << "(" << x2 << "," << y1 << ")" << endl;  cout << " C:" << "(" << x2 << "," << y2 << ")" << endl;  cout << " D:" << "(" << x1 << "," << y2 << ")" << endl;  }  void Rectangle::Peremeshenie() {  cout << " На сколько единиц вы хотите переместить прямоугольник по оси X ?" << endl;  cin >> dx;  cout << " На сколько единиц вы хотите переместить прямоугольник по оси Y ?" << endl;  cin >> dy;  cout << "Получились следующие координаты прямоугольника:" << endl;  cout << " A:" << "(" << x1 + dx << "," << y1 + dy << ")" << endl;  cout << " B:" << "(" << x2 + dx << "," << y1 + dy << ")" << endl;  cout << " C:" << "(" << x2 + dx << "," << y2 + dy << ")" << endl;  cout << " D:" << "(" << x1 + dx << "," << y2 + dy << ")" << endl;  }  void Rectangle::Size() {  cout << " Во сколько раз изменяем размер прямоугольника ?" << endl;  cin >> r;  cout << "Получились следующие координаты прямоугольника:" << endl;  cout << " A:" << "(" << x1 << "," << y1 << ")" << endl;  cout << " B:" << "(" << x2 \* r << "," << y1 << ")" << endl;  cout << " C:" << "(" << x2 \* r << "," << y2 << ")" << endl;  cout << " D:" << "(" << x1 << "," << y2 << ")" << endl;  }  void Rectangle::Minimal() {  double x3, x4, y3, y4, x\_min = x1, x\_max = x1, y\_min = y1, y\_max = y1;  cout << " 2 прямоугольник:" << endl;  cout << "Введите координату х левых точек" << endl; cin >> x3;  cout << "Введите координату y верхних точек" << endl; cin >> y3;  cout << "Введите координату х правых точек" << endl; cin >> x4;  cout << "Введите координату y нижних точек" << endl; cin >> y4;  cout << endl << "Вы ввели следующие координаты 1 прямоугольника:" << 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 прямоугольника:" << endl;  cout << " A:" << "(" << x3 << "," << y3 << ")" << endl;  cout << " B:" << "(" << x4 << "," << y3 << ")" << endl;  cout << " C:" << "(" << x4 << "," << y4 << ")" << endl;  cout << " D:" << "(" << x3 << "," << y4 << ")" << endl;  cout << endl;  double A[4] = { x1,x2,x3,x4 }, B[4] = { y1,y2,y3,y4 };  for (int i = 0; i < 4; i++)  {  if (x\_max < A[i])  {  x\_max = A[i];  }  if (y\_max < B[i])  {  y\_max = B[i];  }  if (x\_min > A[i])  {  x\_min = A[i];  }  if (y\_min > B[i])  {  y\_min = B[i];  }  }  cout << " Наименьший прямоугольник содержащий оба предыдущих имеет следующие координаты: " << endl;  cout << " A:" << "(" << x\_min << "," << y\_max << ")" << endl;  cout << " B:" << "(" << x\_max << "," << y\_max << ")" << endl;  cout << " C:" << "(" << x\_max << "," << y\_min << ")" << endl;  cout << " D:" << "(" << x\_min << "," << y\_min << ")" << endl;  }  void Rectangle::Сross() {  double x3, x4, y3, y4, x\_min = x1, x\_max = x1, y\_min = y1, y\_max = y1, x1\_sred, x2\_sred, y1\_sred, y2\_sred;  cout << " 2 прямоугольник:" << endl;  cout << "Введите координату х левых точек" << endl; cin >> x3;  cout << "Введите координату y верхних точек" << endl; cin >> y3;  cout << "Введите координату х правых точек" << endl; cin >> x4;  cout << "Введите координату y нижних точек" << endl; cin >> y4;  cout << endl;  cout << "Вы ввели следующие координаты 1 прямоугольника:" << 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 прямоугольника:" << endl;  cout << " A:" << "(" << x3 << "," << y3 << ")" << endl;  cout << " B:" << "(" << x4 << "," << y3 << ")" << endl;  cout << " C:" << "(" << x4 << "," << y4 << ")" << endl;  cout << " D:" << "(" << x3 << "," << y4 << ")" << endl;  cout << endl;  double A[4] = { x1,x2,x3,x4 }, B[4] = { y1,y2,y3,y4 };  for (int i = 0; i < 4; i++)  {  if (x\_max < A[i])  {  x\_max = A[i];  }  if (y\_max < B[i])  {  y\_max = B[i];  }  if (x\_min > A[i])  {  x\_min = A[i];  }  if (y\_min > B[i])  {  y\_min = B[i];  }  }  for (int i = 0; i < 4; i++)  {  if ((A[i] != x\_max) && (A[i] != x\_min))  {  x1\_sred = A[i];  break;  }  }  for (int i = 0; i < 4; i++)  {  if ((A[i] != x\_max) && (A[i] != x\_min) && (A[i] != x1\_sred))  {  x2\_sred = A[i];  break;  }  }  for (int i = 0; i < 4; i++)  {  if ((B[i] != y\_max) && (B[i] != y\_min))  {  y1\_sred = B[i];  break;  }  }  for (int i = 0; i < 4; i++)  {  if ((B[i] != y\_max) && (B[i] != y\_min) && (B[i] != y1\_sred))  {  y2\_sred = B[i];  break;  }  }  cout << " Прямоугольник на пересечении двух предыдущих имеет следущие координаты: " << endl;  cout << " A:" << "(" << x1\_sred << "," << y2\_sred << ")" << endl;  cout << " B:" << "(" << x2\_sred << "," << y2\_sred << ")" << endl;  cout << " C:" << "(" << x2\_sred << "," << y1\_sred << ")" << endl;  cout << " D:" << "(" << x1\_sred << "," << y1\_sred << ")" << endl;  } |

## Тестирование программы

Результаты работы программы приведены на рисунках 1.1,1.2.

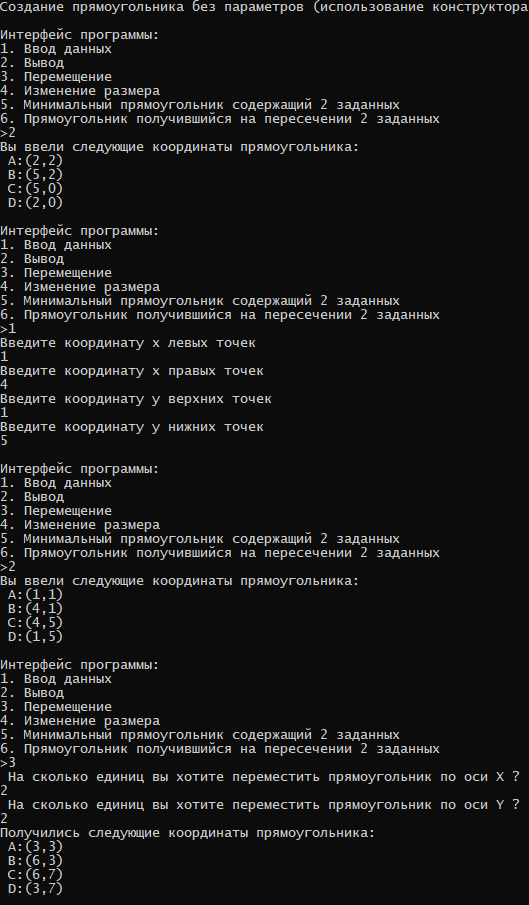
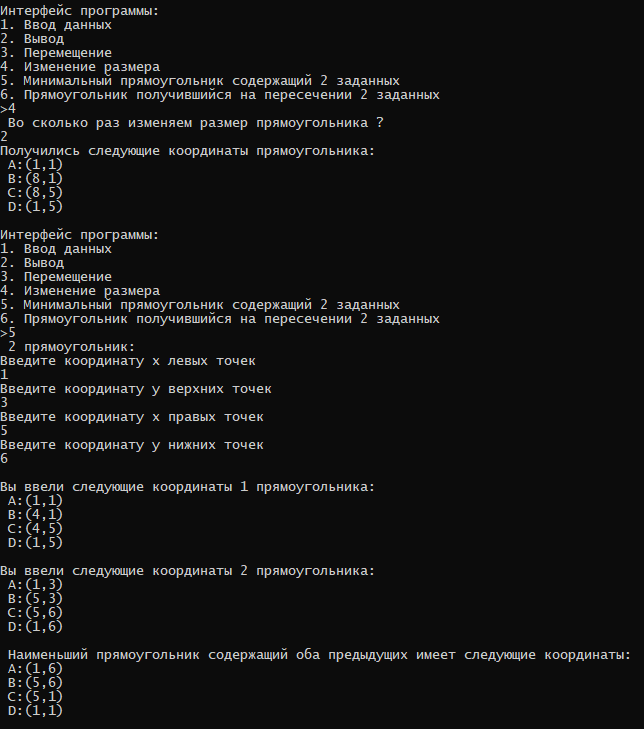
  
Рисунок . – Результат работы программы

Рисунок .2 – 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, SymbString и DecString, 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, operator +(SymbString, SymbString), operator +(DecString, DecString). 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 7 файлов исходного кода, которые приведен в листингах 2.1-2.7.

Листинг . – Текст файла Lab11.cpp

|  |
| --- |
| #include "Factory.h"  int main()  {  setlocale(LC\_ALL, "rus");  cout << "Демонстрация возможностей классов SymbString и DecString" << endl;  char Control;  while (true)  {  cout << endl << "Интерфейс программы:" << endl;  cout << "1. Объект SymbString (символьная строка)" << endl;  cout << "2. Объект DecString (десятичная строка)" << endl;  cout << ">";  cin >> Control;  switch (Control)  {  case '1':  {  FactorySymbString factorySymbString;  bool breakCheck = true;  while (breakCheck)  {  cout << "\nSymbString (символьная строка)" << endl;  cout << "1. Добавить объект" << endl;  cout << "2. Удалить объект" << endl;  cout << "3. Показать созданные объекты" << endl;  cout << "4. Сложить объекты" << endl;  cout << "5. Вернуться к выбору типа объекта" << endl;  cout << ">";  cin >> Control;  switch (Control)  {  case '1':  {  cout << "Введите строку: ";  cin.ignore();  string symbString;  getline(cin, symbString);  factorySymbString.addObject(symbString);  break;  }  case '2':  factorySymbString.deleteObject();  break;  case '3':  factorySymbString.printContainer();  break;  case '4':  {  cout << "Первый объект" << endl;  cout << "Введите строку: ";  cin.ignore();  string str;  getline(cin, str);  SymbString symbString1(str);  cout << "Второй объект" << endl;  cout << "Введите строку: ";  cin.ignore();  getline(cin, str);  SymbString symbString2(str);  SymbString symbString = symbString1 + symbString2;  cout << "Результат" << endl;  symbString.printObject();  cout << "Резултат добавлен в базу объектов" << endl;  factorySymbString.addObject(symbString);  break;  }  case '5':  breakCheck = false;  break;  default: continue;  }  }  }  break;  case '2':  {  FactoryDecString factoryDecString;  bool breakCheck = true;  while (breakCheck)  {  cout << "\nDecString (десятичная строка)" << endl;  cout << "1. Добавить объект" << endl;  cout << "2. Удалить объект" << endl;  cout << "3. Показать созданные объекты" << endl;  cout << "4. Сложить объекты" << endl;  cout << "5. Вернуться к выбору типа объекта" << endl;  cout << ">";  cin >> Control;  switch (Control)  {  case '1':  {  cout << "Введите строку: ";  cin.ignore();  string str;  getline(cin, str);  if ((str.find\_first\_not\_of("1234567890.-") != string::npos) && str.find(' ') != string::npos)  {  cout << "В строке могут присутствовать символы только от 0 до 9 без пробелов";  break;  };  factoryDecString.addObject(str);  break;  }  case '2':  factoryDecString.deleteObject();  break;  case '3':  factoryDecString.printContainer();  break;  case '4':  {  cout << "Первый объект" << endl;  cout << "Введите строку: ";  cin.ignore();  string str;  getline(cin, str);  if ((str.find\_first\_not\_of("1234567890.-") != string::npos) && str.find(' ') != string::npos)  {  cout << "В строке могут присутствовать символы только от 0 до 9 без пробелов";  break;  };  DecString decString1(str);  cout << "Второй объект" << endl;  cout << "Введите строку: ";  getline(cin, str);  if ((str.find\_first\_not\_of("1234567890.-") != string::npos) && str.find(' ') != string::npos)  {  cout << "В строке могут присутствовать символы только от 0 до 9 без пробелов";  break;  };  DecString decString2(str);  DecString decString = decString1 + decString2;  cout << "Результат" << endl;  decString.printObject();  cout << "Резултат добавлен в базу объектов" << endl;  factoryDecString.addObject(decString);  break;  }  case '5':  breakCheck = false;  break;  default: continue;  }  }  break;  }  default: continue;  }  }  return 0;  } |

Листинг .2– Текст файла Factory.h

|  |
| --- |
| #include <vector>  #include "SymbString.h"  #include "DecString.h"  class FactorySymbString {  public:  FactorySymbString() {}  void addObject(string);  void addObject(SymbString);  void deleteObject();  void printContainer();  private:  vector <SymbString> symbStringContainers\_;  };  class FactoryDecString {  public:  FactoryDecString() {}  void addObject(string);  void addObject(DecString);  void deleteObject();  void printContainer();  private:  vector <DecString> decStringContainers\_;  }; |

Листинг .3 – Текст файла Factory.cpp

|  |
| --- |
| #include "Factory.h"  void FactorySymbString::addObject(string str) {  SymbString symbString(str);  symbStringContainers\_.push\_back(symbString);  }  void FactorySymbString::addObject(SymbString symbString)  {  symbStringContainers\_.push\_back(symbString);  }  void FactorySymbString::deleteObject() {  if (!symbStringContainers\_.empty())  {  symbStringContainers\_.erase(symbStringContainers\_.end());  cout << "Последний созданный объект удалён" << endl;  }  else cout << "Объекты не найдены" << endl;  }  void FactorySymbString::printContainer()  {  if (!symbStringContainers\_.empty())  {  for (SymbString var : symbStringContainers\_)  {  var.printObject();  cout << endl;  }  }  else cout << "Список объектов пуст" << endl;  }  void FactoryDecString::addObject(string str) {  DecString symbString(str);  decStringContainers\_.push\_back(symbString);  }  void FactoryDecString::addObject(DecString symbString)  {  decStringContainers\_.push\_back(symbString);  }  void FactoryDecString::deleteObject() {  if (!decStringContainers\_.empty())  {  decStringContainers\_.erase(decStringContainers\_.end());  cout << "Последний созданный объект удалён" << endl;  }  else cout << "Объекты не найдены" << endl;  }  void FactoryDecString::printContainer()  {  if (!decStringContainers\_.empty())  {  for (DecString var : decStringContainers\_)  {  var.printObject();  cout << endl;  }  }  else cout << "Список объектов пуст" << endl;  } |

Листинг .4 – Текст файла DecString.h

|  |
| --- |
| #include <iostream>  #include <string>  using namespace std;  static int DecString\_ID\_SEQUENCE = 0;  class DecString  {  friend class FactoryDecString;  public:  DecString() { id\_ = DecString\_ID\_SEQUENCE++; }  DecString(string str) { id\_ = DecString\_ID\_SEQUENCE++; string\_ = str; }  void printObject();  friend const DecString operator+(const DecString& left, const DecString& right);  private:  int id\_;  string string\_;  }; |

Листинг .5 – Текст файла DecString.cpp

|  |
| --- |
| #include "DecString.h"  void DecString::printObject() {  cout << "Идентификатор объекта: ID" << id\_ << endl;  cout << "Значение объекта: " << string\_ << endl;  }  const DecString operator+(const DecString& left, const DecString& right)  {  DecString decString;  double d1 = atof(left.string\_.c\_str());  double d2 = atof(right.string\_.c\_str());  double d = d1 + d2;  decString.string\_ = to\_string(d);  return decString;  } |

Листинг .6 – Текст файла SymbString.h

|  |
| --- |
| #include <iostream>  #include <string>  using namespace std;  static int SymbString\_ID\_SEQUENCE = 0;  class SymbString  {  friend class FactorySymbString;  public:  SymbString() { id\_ = SymbString\_ID\_SEQUENCE++; }  SymbString(string str) { id\_ = SymbString\_ID\_SEQUENCE++; string\_ = str; }  void printObject();  friend const SymbString operator+(const SymbString& left, const SymbString& right);  private:  int id\_;  string string\_;  }; |

Листинг .7 – Текст файла SymbString.cpp

|  |
| --- |
| #include "SymbString.h"  void SymbString::printObject() {  cout << "Идентификатор объекта: ID" << id\_ << endl;  cout << "Значение объекта: " << string\_ << endl;  }  const SymbString operator+(const SymbString& left, const SymbString& right)  {  SymbString symbString;  symbString.string\_ = left.string\_ + right.string\_;  return symbString;  } |

## Тестирование программы

Результат работы программы приведен на рисунке 2.1,2.2.

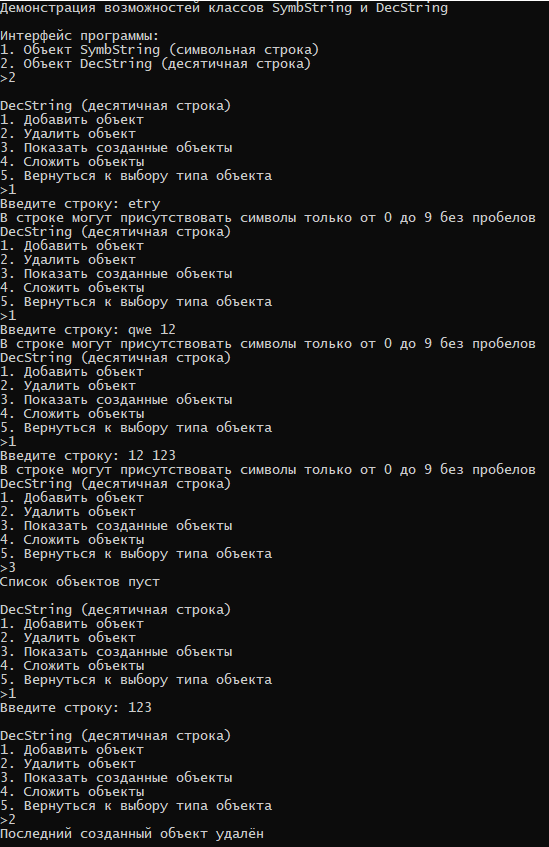


Рисунок .1 – Результат работы программы по наследованию

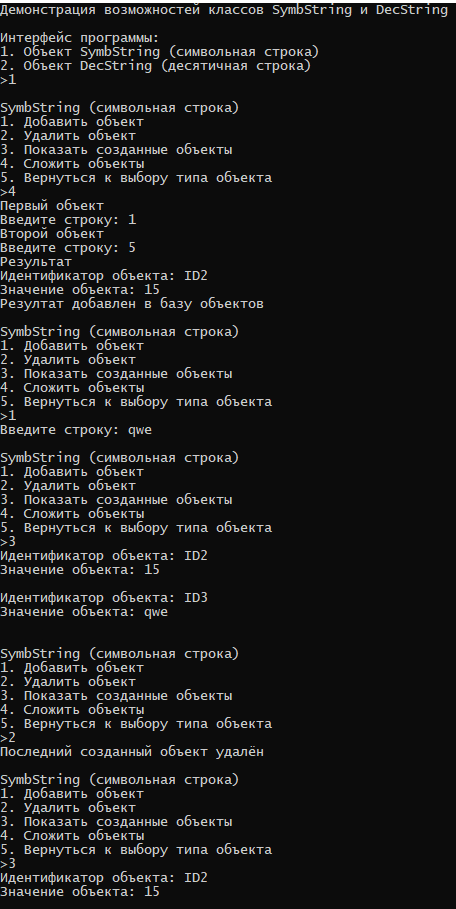


Рисунок .2 – Результат работы программы по наследованию

# 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

*Вычислить и вывести на экран значения функции, заданной* графически*, на итервале от до с шагом dx.* Требуется создать шаблон некоторого целевого класса Stack. Серверный класс std::list. Это означает, что объект класса std::list используется как элемент класса Stack. 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 файлов исходного кода, которые приведены в листинах 3.1,3.2.

Листинг 3. – Текст файла Lab12.cpp

|  |
| --- |
| #include "Stack.h"  int main()  {  setlocale(LC\_ALL, "rus");  cout << "Демонстрация целевого шаблонного класса Stack с типом int" << endl;  try  {  // для int(для всего остального делается аналогично)  Stack<int> stackInt;  cout << "Добавление в стек значений типа int от 0 до 9" << endl;  for (int i = 0; i < 10; ++i)  stackInt.push(i);  cout << "Вывод стека" << endl;  while (!stackInt.empty())  {  std::cout << stackInt.top() << ' ';  stackInt.pop();  }  cout << "\nОчистка стека...Стек пуст" << endl;  cout << "Попытка удалить ещё один элемент из стека.." << endl;  // исключение т.к. стек уже пуст  stackInt.pop();  }  catch (std::underflow\_error& e)  {  std::cerr << e.what() << std::endl;  }  cout << endl;  try  {  Stack<double> stackDouble;  cout << "Добавление в стек значений типа double от 0.5 до 9.5" << endl;  for (double i = 0.5; i < 10; ++i)  stackDouble.push(i);  cout << "Вывод стека" << endl;  while (!stackDouble.empty())  {  std::cout << stackDouble.top() << ' ';  stackDouble.pop();  }  cout << "\nОчистка стека...Стек пуст" << endl;  cout << "Попытка удалить ещё один элемент из стека.." << endl;  // исключение т.к. стек уже пуст  stackDouble.pop();  }  catch (std::underflow\_error& e)  {  std::cerr << e.what() << std::endl;  }  cout << endl;  try  {  Stack<string> stackString;  cout << "Добавление в стек значений типа string от '0' до '9'" << endl;  for (int i = 0; i < 10; ++i)  {  string s = "'" + to\_string(i) + "'";  stackString.push(s);  }  cout << "Вывод стека" << endl;  while (!stackString.empty())  {  std::cout << stackString.top() << ' ';  stackString.pop();  }  cout << "\nОчистка стека...Стек пуст" << endl;  cout << "Попытка удалить ещё один элемент из стека.." << endl;  // исключение т.к. стек уже пуст  stackString.pop();  }  catch (std::underflow\_error& e)  {  std::cerr << e.what() << std::endl;  }  system("pause");  return 0;  } |

Листинг 3.2 – Текст файла Stack.h

|  |
| --- |
| #pragma once  #include <iostream>  #include <list>  #include <string>  #include <stdexcept>  using namespace std;  template < typename T, typename Container = std::list<T> >  class Stack  {  private:  Container container\_;  public:  explicit Stack(const Container& container = Container())  : container\_(container)  {  }  void push(const T& item);  void pop();  T& top();  const T& top() const;  bool empty() const;  };  template<typename T, typename Container>  void Stack<T, Container>::push(const T& item)  {  container\_.push\_back(item);  }  template<typename T, typename Container>  void Stack<T, Container>::pop()  {  if (container\_.empty()) throw std::underflow\_error("Исключение: При удалении последнего элемента стека произошла ошибка. Стек пуст.");  container\_.pop\_back();  }  template<typename T, typename Container>  T& Stack<T, Container>::top()  {  if (container\_.empty()) throw std::underflow\_error("Исключение: При установки ссылки на последний элемента стека произошла ошибка. Стек пуст.");  return container\_.back();  }  template<typename T, typename Container>  inline const T& Stack<T, Container>::top() const  {  if (container\_.empty()) throw std::underflow\_error("Исключение: При установки ссылки на последний элемента стека произошла ошибка. Стек пуст.");  return container\_.back();  }  template<typename T, typename Container>  bool Stack<T, Container>::empty() const  {  return container\_.empty();  } |

## Тестирование программы

Результат работы программы приведен на рисунке 3.1.

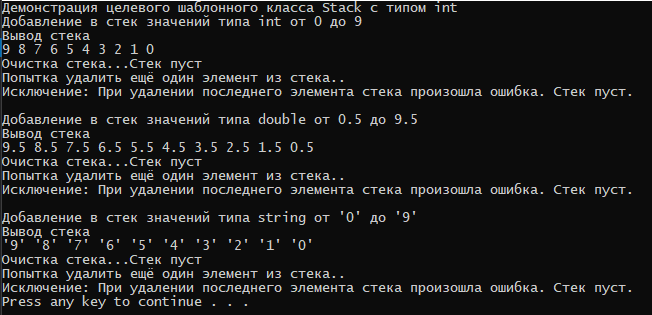


Рисунок 3.1 – Результат работы программы по шаблонам классов

# 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 TRAIN, содержащий следующие поля:*

* *- название пункта назначения;*
* *- номер поезда;*
* *- время отправления.*

*Определить методы доступа к этим полям и перегруженные операции извлечения и вставки для объектов типа TRAIN.*

*Написать программу, выполняющую следующие действия:*

* *- ввод с клавиатуры данных в массив, состоящий из восьми объектов типа TRAIN; записи должны быть размещены в алфавитном порядке по названиям пунктов назначения;*
* *- вывод на экран информации о поездах, отправляющихся после введенного с клавиатуры времени, если таких поездов 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<vector>  #include<string>  #include <algorithm>  using namespace std;  class Train {  public:  Train() {}  Train(string pn, int num, string time) {  pn\_ = pn;  num\_ = num;  time\_ = time;  }  string getPn() { return pn\_; };  string getTime() { return time\_; };  friend ostream& operator<<(ostream& out, const Train& rv);  friend istream& operator>>(istream& in, Train& rv);  private:  string pn\_;  int num\_;  string time\_;  };  ostream& operator<<(ostream& out, const Train& rv) {  cout << "Пункт назначения: " << rv.pn\_ << endl;  cout << "Номер поезда: " << rv.num\_ << endl;  cout << "Время отправления: " << rv.time\_ << endl;  return out;  }  istream& operator>>(istream& in, Train& rv) {  cout << "Введите пункт назначения: ";  getline(cin, rv.pn\_);  while (true)  {  cout << "Введите номер поезда: ";  int num;  cin >> num;  cin.ignore(256, '\n');  if (cin.fail()) // если предыдущее извлечение оказалось неудачным,  {  cin.clear(); // то возвращаем cin в 'обычный' режим работы  cin.ignore(32767, '\n'); // и удаляем значения предыдущего ввода из входного буфера  cout << "Вводите только целые числа!" << endl;  }  else  {  rv.num\_ = num;  break;  }  }  cout << "Введите время отправления: ";  getline(cin, rv.time\_);  return in;  }  class TrainCollection {  public:  TrainCollection(int count) {  for (size\_t i = 0; i < count; i++)  addTrain();  }  void addTrain() {  Train train;  cin >> train;  cout << endl;  trainCollections\_.push\_back(train);  }  void printTrainCollections() {  for (auto train : trainCollections\_)  {  cout << train;  cout << endl;  }  }  void sortTrainCollections() {  sort(  trainCollections\_.begin(),  trainCollections\_.end(),  [](Train& a, Train& b) {  return a.getPn() < b.getPn();  }  );  }  void searchAndPrintTrainByTime(string time) {  bool inp = false;  for (auto train : trainCollections\_)  {  if (train.getTime() == time)  {  cout << train;  inp = true;  }  }  if (inp != true) cout << "Поезда не найдены" << endl;  }  private:  vector <Train> trainCollections\_;  };  int main(void)  {  setlocale(LC\_ALL, "rus");  cout << "Введите начальное количество поездов: ";  size\_t count;  cin >> count;  cin.ignore(256, '\n');  TrainCollection trainCollection(count);  cout << "База поездов" << endl;  trainCollection.printTrainCollections();  cout << "База поездов после сортировки" << endl;  trainCollection.sortTrainCollections();  trainCollection.printTrainCollections();  cout << "Введите время отправления для поиска поездов: ";  string time;  getline(cin, time);  trainCollection.searchAndPrintTrainByTime(time);  system("pause");  return 0;  } |

## Тестирование программы

Результат работы программы приведен на рисунке 4.1.

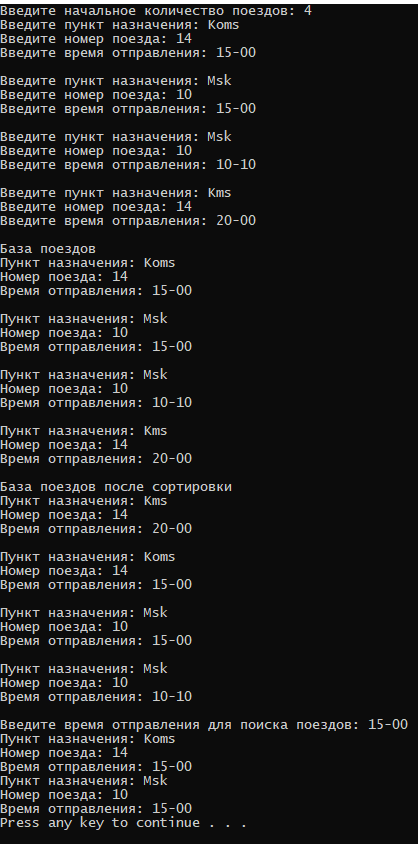


Рисунок 4.1 – Результат работы программы по стандартным потокам

# 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

*Используя файловые потоки, написать программу, которая считывает текст из файла и определяет, сколько в нем слов, состоящих из не более чем четырех буквами.*

## Текст программы

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

Листинг 5. – Текст файла Lab14.cpp

|  |
| --- |
| #include <fstream>  #include <iostream>  #include <string>  using namespace std;  int main()  {  setlocale(LC\_ALL, "rus");  cout << "Путь к файлу: ";  string path;  getline(cin, path);  ifstream input(path);  if (input.fail() == true)  {  cout << "Не удаётся открыть файл" << endl;  system("pause");  return 0;  }  string word;  cout << "Поиск слов, состоящих из не более чем 4 букв.." << endl;  bool inp = false;  while (input >> word)  {  if (word.length() < 5)  {  cout << word << endl;  inp = true;  }  }  if (inp == false) cout << "Слова, состоящие из не более чем 4 букв, не найдены" << endl;  input.close();  system("pause");  return 0;  } |

## Тестирование программы

Результат работы программы приведен на рисунке 5.1.

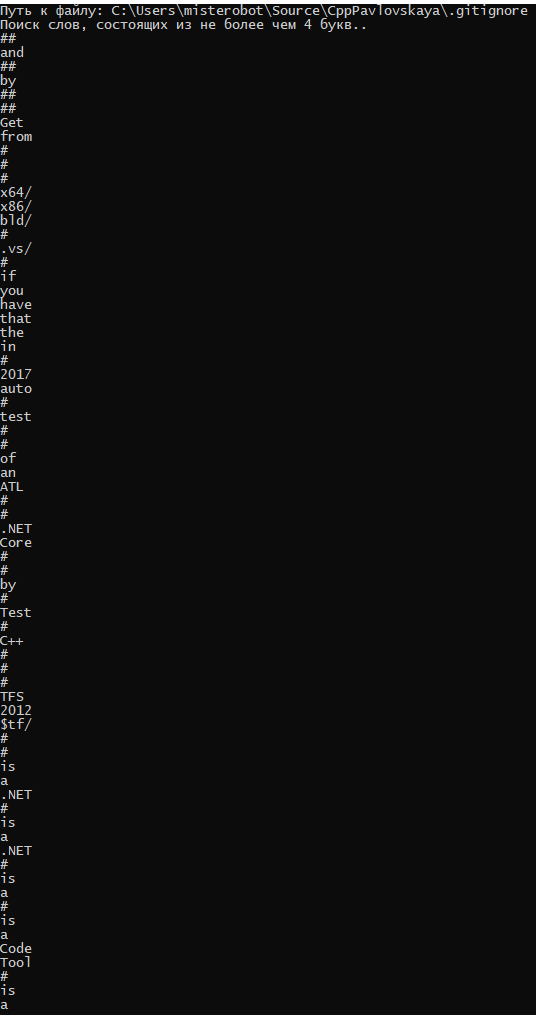


Рисунок 5.1 – Результат работы программы по файловым потокам

# Стандартная библиотека шаблонов

Стандартная библиотека шаблонов (Standard Template Library, STL) состоит из двух основных частей: набора контейнерных классов и набора обобщенных алгоритмов.

Контейнеры — это объекты, содержащие другие однотипные объекты. Контейнерные классы являются шаблонными, поэтому хранимые в них объекты могут быть как встроенных, так и пользовательских типов. Эти объекты должны допускать копирование и присваивание. Встроенные типы этим требованиям удовлетворяют; то же самое относится к классам, если конструктор копирования или операция присваивания не объявлены в них закрытыми или защищенными. В контейнерных классах реализованы такие типовые структуры данных, как стек, список, очередь и т. д.

## Описание программы

*Написать программу «Моя записная книжка».*

*Предусмотреть возможность работы с произвольным числом записей, поиска записи по какому-либо признаку (например, по фамилии, дате рождения или номеру телефона), добавления и удаления записей, сортировки по разным полям.*

*Хранение данных организовать с помощью класса map или multimap.*

## Текст программы

Проект состоит из 5 файлов исходного кода, которые приведены в листингах 6.1-6.5.

Листинг 6. – Текст файла Lab15.cpp

|  |
| --- |
| #include "Notebook.h"  int main()  {  setlocale(LC\_ALL, "Russian");  cout << "Моя записная книжка" << endl;  int num;  while (true)  {  cout << "Введите cтартовое количество записей: ";  cin >> num;  cin.ignore(256, '\n');  if (cin.fail()) // если предыдущее извлечение оказалось неудачным,  {  cin.clear(); // то возвращаем cin в 'обычный' режим работы  cin.ignore(32767, '\n'); // и удаляем значения предыдущего ввода из входного буфера  cout << "Вводите только целые числа!" << endl;  }  else break;  }  Notebook notebook(num);  system("cls");  while (true)  {  size\_t selectedMenuItem;  while (true)  {  cout << "\nМоя записная книжка (текущее количество записей " << notebook.getSize() << ")" << endl;  cout << "1.Вывод всех записей" << endl;  cout << "2.Добавление записи" << endl;  cout << "3.Удаление записи" << endl;  cout << "4.Поиск записей" << endl;  cout << "5.Сортировка записей " << endl;  cout << "6.Очистить консоль" << endl;  cout << "> ";  cin >> selectedMenuItem;  cin.ignore(256, '\n');  if (cin.fail()) // если предыдущее извлечение оказалось неудачным,  {  cin.clear(); // то возвращаем cin в 'обычный' режим работы  cin.ignore(32767, '\n'); // и удаляем значения предыдущего ввода из входного буфера  cout << "Вводите только целые числа!" << endl;  }  else break;  }  switch (selectedMenuItem)  {  case 1:  {  notebook.printNotebook();  }  break;  case 2:  {  notebook.addNote();  }  break;  case 3:  {  cout << "Номер удаляемой записи: ";  size\_t number;  cin >> number;  notebook.eraseNote(number);  }  break;  case 4:  {  bool exit = false;  while (!exit)  {  cout << "Критерии поиска записи:" << endl;  cout << "1.Поиск по номеру" << endl;  cout << "2.Поиск по фамилии" << endl;  cout << "3.Поиск по дате рождения" << endl;  cout << "4.Поиск по номеру телефона" << endl;  cout << "5.Назад в гл. меню" << endl;  cout << ">";  size\_t t;  cin >> t;  switch (t)  {  case 1:  {  cout << "Введите номер: ";  size\_t recNumber;  cin >> recNumber;  notebook.searchAndPrintNoteByNumber(recNumber);  }  break;  case 2:  {  cout << "Введите фамилию: ";  string recNumber;  cin >> recNumber;  notebook.searchAndPrintNoteByLastName(recNumber);  }  break;  case 3:  {  cout << "Введите дату рождения: ";  string recNumber;  cin >> recNumber;  notebook.searchAndPrintNoteByLastName(recNumber);  }  break;  case 4:  {  cout << "Введите номер телефона: ";  string recNumber;  cin >> recNumber;  notebook.searchAndPrintNoteByLastName(recNumber);  }  break;  case 5:  exit = true;  break;  }  }  }  break;  case 5:  {  bool exit = false;  while (!exit)  {  cout << "Критерии сортировки записей:" << endl;  cout << "1.По фамилиям" << endl;  cout << "2.По дате рождения" << endl;  cout << "3.Назад в гл. меню" << endl;  cout << ">";  size\_t t;  cin >> t;  switch (t)  {  case 1:  {  notebook.sortAndPrintNotebookByLastName();  }  break;  case 2:  {  notebook.sortAndPrintNotebookByDateOfBirth();  }  break;  case 3:  exit = true;  break;  }  }  }  break;  case 6:  {  system("cls");  }  break;  default: system("cls");  }  }  return 0;  } |

Листинг 6.2 – Текст файла Note.h

|  |
| --- |
| #include <iostream>  using namespace std;  class Note  {  public:  Note();  Note(string, string, string);  ~Note();  string getLastName();  string getDateOfBirth();  string getPhoneNumber();  friend ostream& operator<<(ostream&, const Note&);  friend istream& operator>>(istream&, Note&);  private:  string lastName\_;  string dateOfBirth\_;  string phoneNumber\_;  }; |

Листинг 6.3 – Текст файла Note.cpp

|  |
| --- |
| } #include "Note.h"  Note::Note()  {  lastName\_ = "";  dateOfBirth\_ = "";  phoneNumber\_ = "";  }  Note::Note(string SN, string D, string NM)  {  lastName\_ = SN;  dateOfBirth\_ = D;  phoneNumber\_ = NM;  }  Note::~Note()  {  }  string Note::getLastName() {  return lastName\_;  }  string Note::getDateOfBirth()  {  return dateOfBirth\_;  }  string Note::getPhoneNumber()  {  return phoneNumber\_;  }  ostream& operator<<(ostream& out, const Note& rv) {  out << "Фамилия: " << rv.lastName\_ << endl;  out << "Дата рождения: " << rv.dateOfBirth\_ << endl;  out << "Номер телефона: " << rv.phoneNumber\_ << endl;  return out;  }  istream& operator>>(istream& in, Note& rv) {  cout << "Введите фамилию: ";  in >> rv.lastName\_;  cout << "Введите дату рождения: ";  in >> rv.dateOfBirth\_;  cout << "Введите номер телефона: ";  in >> rv.phoneNumber\_;  return in;  } |

Листинг 6.4 – Текст файла Notebook.h

|  |
| --- |
| } #include <map>  #include <vector>  #include "Note.h"  #include <algorithm>  #include <iterator>  class Notebook  {  public:  Notebook();  Notebook(size\_t);  ~Notebook();  void addNote();  void eraseNote(size\_t);  void printNotebook();  void sortAndPrintNotebookByLastName();  void sortAndPrintNotebookByDateOfBirth();  void searchAndPrintNoteByNumber(size\_t);  void searchAndPrintNoteByLastName(string);  void searchAndPrintNoteByDateOfBirth(string);  void searchAndPrintNoteByPhoneNumber(string);  size\_t getSize();  private:  map<size\_t, Note> notebook\_;  };  struct NoteTempStruct  {  size\_t number\_;  string lastName\_;  string dateOfBirth\_;  string phoneNumber\_;  };  class NoteTempVectorCollection  {  public:  vector<NoteTempStruct>noteTempVector;  }; |

Листинг 6.5 – Текст файла Notebook.cpp

|  |
| --- |
| #include "Notebook.h"  size\_t recordNumber = 0;  Notebook::Notebook()  {  }  Notebook::Notebook(size\_t count) {  for (size\_t i = 0; i < count; i++)  addNote();  }  Notebook::~Notebook() {  }  void Notebook::addNote() {  Note note;  cin >> note;  notebook\_.insert(pair<size\_t, Note>(++recordNumber, note));  cout << endl;  }  void Notebook::eraseNote(size\_t number)  {  bool inp = false;  for (auto pos = notebook\_.begin(); pos != notebook\_.end();)  {  if (pos->first == number)  {  pos = notebook\_.erase(pos);  inp = true;  }  else ++pos;  }  if (inp != true) cout << "Запись не найдена" << endl;  }  void Notebook::printNotebook() {  if (notebook\_.empty())  {  cout << "Записная книжка пуста" << endl;  return;  }  for (auto note : notebook\_)  {  cout << "Запись номер " << note.first << endl;  cout << note.second;  cout << endl;  }  }  void Notebook::sortAndPrintNotebookByLastName() {  if (getSize() != 0)  {  NoteTempVectorCollection noteTempVectorCollection;  for (auto note : notebook\_)  {  NoteTempStruct noteTempVecto;  noteTempVecto.number\_ = note.first;  noteTempVecto.lastName\_ = note.second.getLastName();  noteTempVecto.dateOfBirth\_ = note.second.getDateOfBirth();  noteTempVecto.phoneNumber\_ = note.second.getPhoneNumber();  noteTempVectorCollection.noteTempVector.push\_back(noteTempVecto);  }  sort(  noteTempVectorCollection.noteTempVector.begin(),  noteTempVectorCollection.noteTempVector.end(),  [](NoteTempStruct& a, NoteTempStruct& b) {  return a.lastName\_ < b.lastName\_;  }  );  for (auto note : noteTempVectorCollection.noteTempVector)  {  cout << endl;  cout << "Запись номер " << note.number\_ << endl;  cout << "Фамилия: " << note.lastName\_ << endl;  cout << "Дата рождения: " << note.dateOfBirth\_ << endl;  cout << "Номер телефона: " << note.phoneNumber\_ << endl;  }  }  else cout << "Записная книжка пуста" << endl;  }  void Notebook::sortAndPrintNotebookByDateOfBirth()  {  if (getSize() != 0)  {  NoteTempVectorCollection noteTempVectorCollection;  for (auto note : notebook\_)  {  NoteTempStruct noteTempVecto;  noteTempVecto.number\_ = note.first;  noteTempVecto.lastName\_ = note.second.getLastName();  noteTempVecto.dateOfBirth\_ = note.second.getDateOfBirth();  noteTempVecto.phoneNumber\_ = note.second.getPhoneNumber();  noteTempVectorCollection.noteTempVector.push\_back(noteTempVecto);  }  sort(  noteTempVectorCollection.noteTempVector.begin(),  noteTempVectorCollection.noteTempVector.end(),  [](NoteTempStruct& a, NoteTempStruct& b) {  return a.dateOfBirth\_ < b.dateOfBirth\_;  }  );  for (auto note : noteTempVectorCollection.noteTempVector)  {  cout << endl;  cout << "Запись номер " << note.number\_ << endl;  cout << "Фамилия: " << note.lastName\_ << endl;  cout << "Дата рождения: " << note.dateOfBirth\_ << endl;  cout << "Номер телефона: " << note.phoneNumber\_ << endl;  }  }  else cout << "Записная книжка пуста" << endl;  }  void Notebook::searchAndPrintNoteByNumber(size\_t number)  {  bool inp = false;  for (auto note : notebook\_)  {  if (note.first == number)  {  cout << note.second;  inp = true;  }  }  if (inp != true) cout << "Записи не найдены" << endl;  }  void Notebook::searchAndPrintNoteByLastName(string lastName)  {  bool inp = false;  for (auto note : notebook\_)  {  if (note.second.getLastName() == lastName)  {  cout << "Запись номер " << note.first << endl;  cout << note.second;  inp = true;  }  }  if (inp != true) cout << "Записи не найдены" << endl;  }  void Notebook::searchAndPrintNoteByDateOfBirth(string dateOfBirth)  {  bool inp = false;  for (auto note : notebook\_)  {  if (note.second.getDateOfBirth() == dateOfBirth)  {  cout << "Запись номер " << note.first << endl;  cout << note.second;  inp = true;  }  }  if (inp != true) cout << "Записи не найдены" << endl;  }  void Notebook::searchAndPrintNoteByPhoneNumber(string phoneNumber)  {  bool inp = false;  for (auto note : notebook\_)  {  if (note.second.getPhoneNumber() == phoneNumber)  {  cout << "Запись номер " << note.first << endl;  cout << note.second;  inp = true;  }  }  if (inp != true) cout << "Записи не найдены" << endl;  }  size\_t Notebook::getSize()  {  return notebook\_.size();  } |

## Тестирование программы

Результат работы программы приведен на рисунках 6.1-6.3.

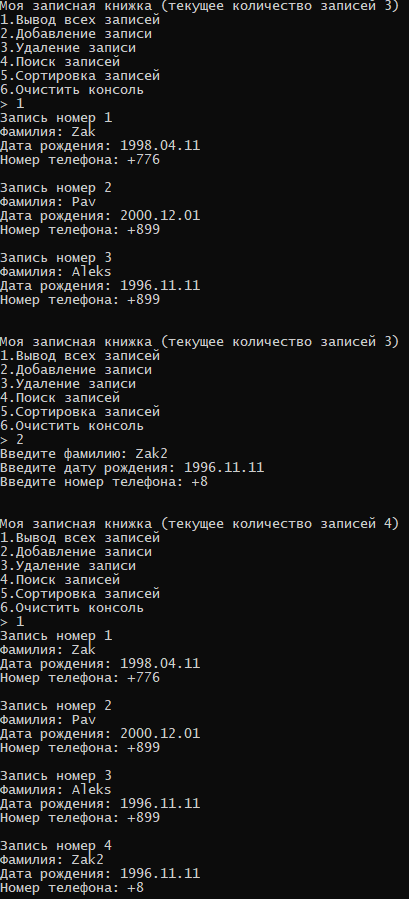


Рисунок 6.1 – Результат работы программы по библиотеке STL

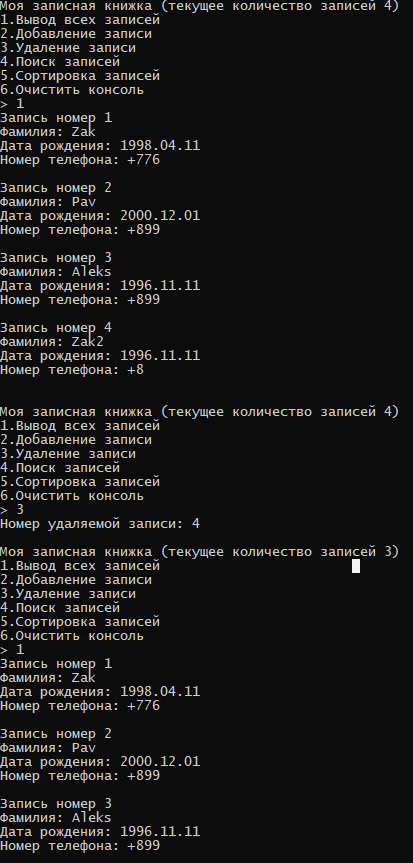


Рисунок 6.2 – Результат работы программы по библиотеке STL

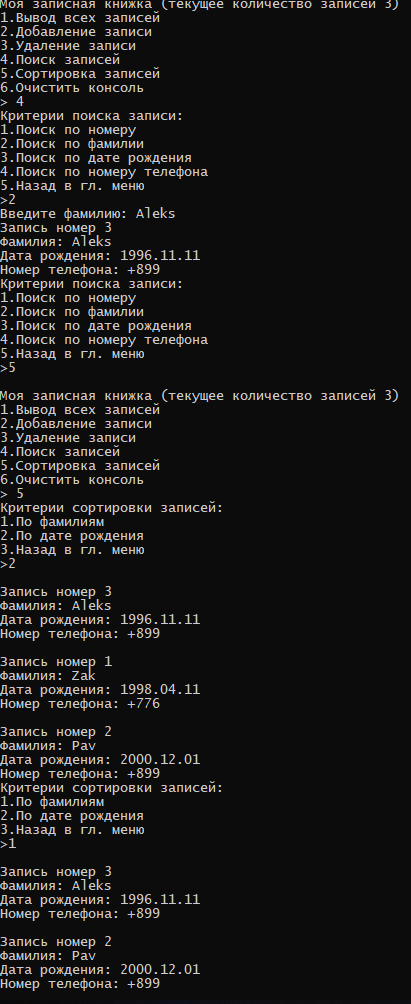


Рисунок 6.3 – Результат работы программы по библиотеке STL

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.
6. Стандартная библиотека шаблонов.

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»).