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 9

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

Составить описание класса для определения одномерных массивов строк фиксированной длины. Предусмотреть возможность обращения к отдельным строкам массива по индексам, контроль выхода за пределы массива, выполнения операций поэлементного сцепления двух массивов с образованием нового массива, слияния двух массивов с исключением повторяющихся элементов, вывода на экран элемента массива по заданному индексу 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);  };  //Создание массива строк  void PointerArray::create()  {  // выделение памяти под массивы  cout << endl;  cout << "Количество строк: "; cin >> \_size;  \_string = new char\* [\_size];  // определение длины строк  cout << "Строки фиксированной длины." << endl;  cout << "Введите количество символов для строк: ";  cin >> \_stringLength; \_stringLength += 1;  // выделяем память под каждую строку  for (int i = 0; i < \_size; i++) {    \*(\_string + i) = new char[\_stringLength];  }  cout << endl;  }  // Инициализация массивов  void PointerArray::initialize()  {  create();  cin.ignore();  cout << "Ввод строк фиксированной длины. Максимальное количество символов: " << \_stringLength - 1 << endl;  for (int i = 0; i < \_size; i++)  {  cout << "Ввод " << i + 1 << " строки: ";  cin.getline(\*(\_string + i), \_stringLength); // 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)  {  cout << \*(\_string + rec) + field << endl;  }  else  {  cout << "Выход за пределы диапазона строк фиксированной длины 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; i++)  {  bool checkEntry = false;  for (size\_t j = 0; j < \_stringLength; 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;  while (true)  {  cout << " Меню" << endl;  cout << "1 Создание массива строк фикс. длины"; 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 строки (нумерация начинаеться с 0!): "; cin >> number;  \_string.showByIndex(number);  cout << endl;  }  break;  case 4:  {  cout << "Примечание: нумерация строк начинаеться с 0!" << endl;  cout << "Введите номер 1-ой строки: "; cin >> n1;  cout << "Введите номер 2-ой строки: "; cin >> n2;  \_string.merge(n1, n2);  cout << endl;  }  break;  case 5:  {  cout << "Примечание: нумерация строк начинаеться с 0!" << endl;  cout << "Введите номер 1-ой строки: "; cin >> n1;  cout << "Введите номер 2-ой строки: "; cin >> n2;  \_string.mergeNotRepeatingElements(n1, n2);  cout << endl;  cout << endl;  }  break;  case 6:  {  cout << "Примечание: нумерация строк и стобцов начинаеться с 0!" << endl;  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.

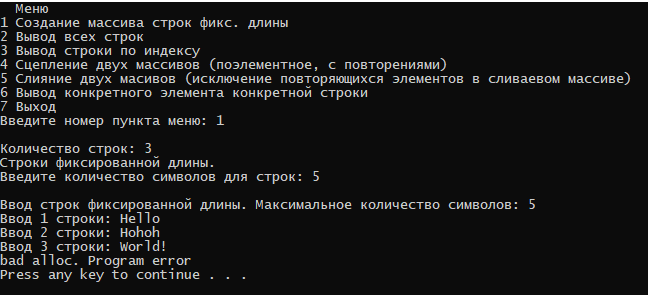


Рисунок . – Результат работы программы

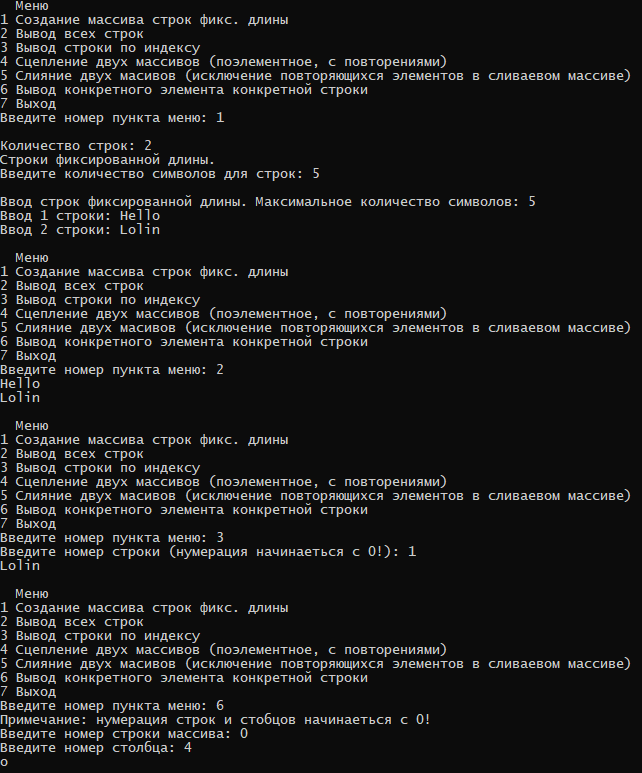


Рисунок .2 – Результат работы программы

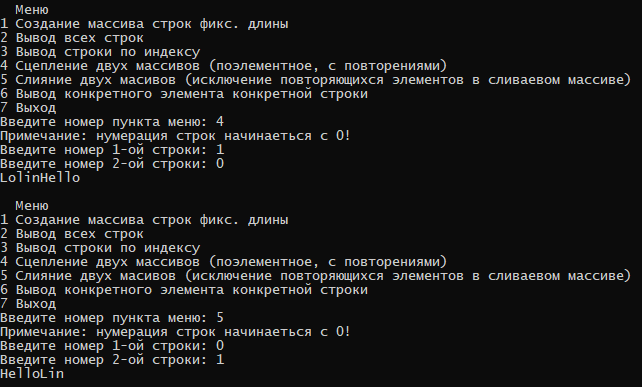


Рисунок .3 – Результат работы программы

# 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, SymbString и HexString, 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 +(HexString, HexString). 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.)

## 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 HexString" << endl;  char Control;  while (true)  {  cout << endl << "Интерфейс программы:" << endl;  cout << "1. Объект SymbString (символьная строка)" << endl;  cout << "2. Объект HexString (шестнадцатеричная 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: ";  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':  {  FactoryHexString factoryHexString;  bool breakCheck = true;  while (breakCheck)  {  cout << "\nHexString (шестнадцатеричная 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 << "Примечание: в строке могут присутствовать цифры от 0 до 9 и буквы A,B,C,D,E,F без пробелов" << endl;  cout << "Пример: 5CD8" << endl << endl;  cout << "Введите строку: ";  cin.ignore();  string str;  getline(cin, str);  if ((str.find\_first\_not\_of("0123456789ABCDEF") != string::npos))  {  cout << "Ошибка ввода." << endl;  break;  };  factoryHexString.addObject(str);  break;  }  case '2':  factoryHexString.deleteObject();  break;  case '3':  factoryHexString.printContainer();  break;  case '4':  {  cout << "Примечание: в строке могут присутствовать цифры от 0 до 9 и буквы A,B,C,D,E,F без пробелов" << endl;  cout << "Пример: 5CD8" << endl << endl;  cout << "Первый объект" << endl;  cout << "Введите строку: ";  cin.ignore();  string str;  getline(cin, str);  if ((str.find\_first\_not\_of("0123456789ABCDEF") != string::npos))  {  cout << "Ошибка ввода." << endl;  break;  };  HexString decString1(str);  cout << "Второй объект" << endl;  cout << "Введите строку: ";  getline(cin, str);  if ((str.find\_first\_not\_of("0123456789ABCDEF") != string::npos))  {  cout << "Ошибка ввода." << endl;  break;  };  HexString decString2(str);  HexString decString = decString1 + decString2;  cout << "Результат" << endl;  decString.printObject();  cout << "Резултат добавлен в базу объектов" << endl;  factoryHexString.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 "HexString.h"  class FactorySymbString {  public:  FactorySymbString() {}  void addObject(string);  void addObject(SymbString);  void deleteObject();  void printContainer();  private:  vector <SymbString> symbStringContainers\_;  };  class FactoryHexString {  public:  FactoryHexString() {}  void addObject(string);  void addObject(HexString);  void deleteObject();  void printContainer();  private:  vector <HexString> hexStringContainers\_;  }; |

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())  {  auto a = symbStringContainers\_.size() - 1;  symbStringContainers\_.erase(symbStringContainers\_.begin() + a);  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 FactoryHexString::addObject(string str) {  HexString symbString(str);  hexStringContainers\_.push\_back(symbString);  }  void FactoryHexString::addObject(HexString symbString)  {  hexStringContainers\_.push\_back(symbString);  }  void FactoryHexString::deleteObject() {  if (!hexStringContainers\_.empty())  {  auto a = hexStringContainers\_.size() - 1;  hexStringContainers\_.erase(hexStringContainers\_.begin() + a);  cout << "Последний созданный объект удалён" << endl;  }  else cout << "Объекты не найдены" << endl;  }  void FactoryHexString::printContainer()  {  if (!hexStringContainers\_.empty())  {  for (HexString var : hexStringContainers\_)  {  var.printObject();  cout << endl;  }  }  else cout << "Список объектов пуст" << endl;  } |

Листинг .4 – Текст файла HexString.h

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

Листинг .5 – Текст файла HexString.cpp

|  |
| --- |
| #include "HexString.h"  void HexString::printObject() {  cout << "Идентификатор объекта: ID" << id\_ << endl;  cout << "Значение объекта: " << string\_ << endl;  }  const HexString operator+(const HexString& left, const HexString& right)  {  HexString hexString;  int num1 = stoi(left.string\_, 0, 16);  int num2 = stoi(right.string\_, 0, 16);  string hexBase = "0123456789ABCDEF";  int resDec = num1 + num2;  string resHex;  while (resDec > 0)  {  resHex = hexBase[resDec % 16] + resHex;  resDec /= 16;  }  hexString.string\_ = resHex;  return hexString;  }  int HexString::HexString\_ID\_SEQUENCE = 0; |

Листинг 2.6 – Текст файла SymbString.h

|  |
| --- |
| #include <iostream>  #include <string>  using namespace std;  class MyString  {  protected:  int id\_;  string string\_;  };  class SymbString : public MyString  {  friend class FactorySymbString;  public:  static int SymbString\_ID\_SEQUENCE;  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);  }; |

Листинг 2.7 – 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;  }  int SymbString::SymbString\_ID\_SEQUENCE = 0; |

## Тестирование программы

Результат работы программы приведен на рисунках 2.1 - 2.2.

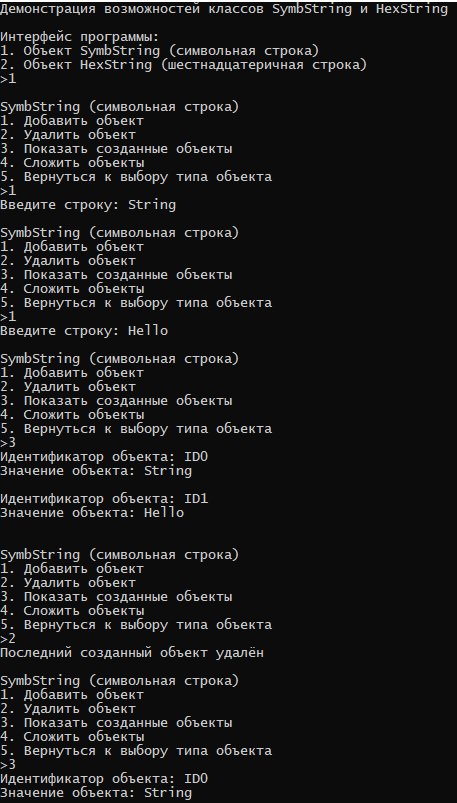


Рисунок .1 – Результат работы программы по наследованию

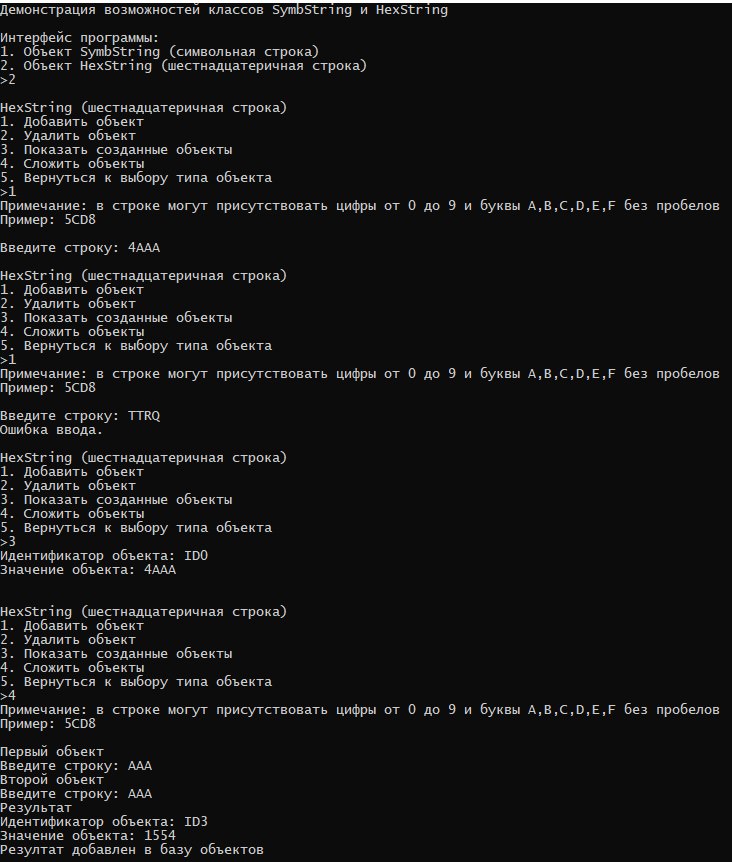


Рисунок .2 – Результат работы программы по наследованию

# 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

Требуется создать шаблон некоторого целевого класса Queue. В каждом варианте уточняются требования к реализации — указанием на применение некоторого серверного класса Vect. Это означает, что объект класса Vect 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 трёх файлов исходного кода, которые приведены в листингах 3.1-3.3.

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.begin() << endl;  cout << "Последний элемент очереди: " << \*(queueInt.end()-1) << endl;  cout << "Освобождение очереди..." << endl;  while (!queueInt.empty())  {  cout << "Освобождаемый элемент: " << \*queueInt.begin() << endl;  queueInt.pop\_begin();  }  cout << "Попытка удалить ещё один элемент из очереди.." << endl;  // исключение т.к. очередь уже пуста  queueInt.pop\_begin();  }  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.begin() << endl;  cout << "Последний элемент очереди: " << \*(queueDouble.end()-1) << endl;  cout << "Освобождение очереди..." << endl;  while (!queueDouble.empty())  {  cout << "Освобождаемый элемент: " << \*queueDouble.begin() << endl;  queueDouble.pop\_begin();  }  cout << "Попытка удалить ещё один элемент из очереди.." << endl;  queueDouble.pop\_begin();  }  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.begin() << endl;  cout << "Последний элемент очереди: " << \*(queueString.end()-1) << endl;  cout << "Освобождение очереди..." << endl;  while (!queueString.empty())  {  cout << "Освобождаемый элемент: " << \*queueString.begin() << endl;  queueString.pop\_begin();  }  cout << "Попытка удалить ещё один элемент из очереди.." << endl;  queueString.pop\_begin();  }  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 "Vect.h"  template < typename T, typename Container = Vect<T> >  class Queue : public Vect <T>  {  private:  Container container\_;  public:  explicit Queue(const Container& container = Container())  : container\_(container)  {  }  bool empty() const;  void push\_begin(T\* \_P, const T& \_x);  void pop\_begin(); // удаление элемента из конца вектора  void show();  };  template<typename T, typename Container>  bool Queue<T, Container>::empty() const  {  return this->size() == 0 ? true : false;  }  template<typename T, typename Container>  void Queue<T, Container>::push\_begin(T\* \_P, const T& \_x)  {  Queue<T, Container>::insert(\_P, \_x);  }  template<typename T, typename Container>  void Queue<T, Container>::pop\_begin()  {  if (this->last == this->first)  throw underflow\_error("Исключение: При удалении элемента очереди произошла ошибка. Очередь пуста.");;  size\_t n = this->size() - 1; // новый размер  T\* new\_first = new T[n \* sizeof(T)];  T\* new\_last = new\_first + n;  for (size\_t i = 0; i < n; ++i)\*  (new\_first + i) = \*(this->first + i + 1);  this->Destroy();  this->first = new\_first;  this->last = new\_last;  }  template<typename T, typename Container>  inline void Queue<T, Container>::show()  {  if (this->size() > 0)  {  size\_t n = this->size();  for (size\_t i = 0; i < n; ++i)  cout << \*(this->first + i) << " ";  }  else  {  cout << "void";  }  cout << endl;  } |

Листинг 3.3 – Текст файла Vect.h

|  |
| --- |
| #pragma once  #include <iostream>  #include <stdexcept>  #include <string>  using namespace std;  class VectError {  public:  VectError() {}  virtual void ErrMsg() const { std::cerr << "Error with Vect object.\n"; }  void Continue() const {  #ifdef DEBUG  std::cerr << "Debug: program is being continued.\n";  #else  throw;  #endif  }  };  class VectRangeErr : public VectError {  public:  VectRangeErr(int \_min, int \_max, int \_actual) :  min(\_min), max(\_max), actual(\_actual) {}  void ErrMsg() const {  std::cerr << "Error of index: ";  std::cerr << "possible range: " << min << " - " << max << ", ";  std::cerr << "actual index: " << actual << std::endl;  Continue();  }  private:  int min, max;  int actual;  };  class VectPopErr : public VectError {  public:  void ErrMsg() const { std::cerr << "Error of pop\n"; Continue(); }  };  template<class T> class Vect { // ------------------------ Template class Vect  public:  explicit Vect() : first(0), last(0), markName("class Deque") {}  explicit Vect(size\_t \_n, const T& \_v = T()) {  Allocate(\_n);  for (size\_t i = 0; i < \_n; ++i)\* (first + i) = \_v;  }  Vect(const Vect&); // конструктор копирования  Vect& operator =(const Vect&); // операция присваивания  ~Vect() {  cout << "Destructor of " << markName << endl;  Destroy();  first = 0, last = 0;  }  void mark(string& name) { markName = name; } // установить отладочное имя  string mark() const { return markName; } // получить отладочное имя  size\_t size() const; // получить размер вектора  T\* begin() const { return first; } // получить указатель на 1-й элемент  T\* end() const { return last; } // получить указатель на элемент,  // следующий за последним  T& operator[](size\_t i); // операция индексирования  void insert(T\* \_P, const T& \_x); // вставка элемента в позицию \_P  void push\_back(const T& \_x); // вставка элемента в конец вектора  void pop\_back(); // удаление элемента из конца вектора  void show() const; // вывод в cout содержимого вектора  protected:  void Allocate(size\_t \_n) {  first = new T[\_n \* sizeof(T)];  last = first + \_n;  }  void Destroy() {  for (T\* p = first; p != last; ++p) p->~T();  delete[] first;  }  T\* first; // указатель на 1-й элемент  T\* last; // указатель на элемент, следующий за последним  string markName;  };  template<class T>  Vect<T>::Vect(const Vect& other) { // ------------------- Конструктор копирования  size\_t n = other.size();  Allocate(n);  for (size\_t i = 0; i < n; ++i)\* (first + i) = \*(other.first + i);  markName = string("Copy of ") + other.markName;  cout << "Copy constructor: " << markName << endl;  }  template<class T>  Vect<T>& Vect<T>::operator =(const Vect& other) { // -------- Операция присваивания  if (this == &other) return \*this;  Destroy();  size\_t n = other.size();  Allocate(n);  for (size\_t i = 0; i < n; ++i)\* (first + i) = \*(other.first + i);  return \*this;  }  template<class T>  size\_t Vect<T>::size() const { // ---------------------- Получение размера вектора  if (first > last) throw VectError();  return (0 == first ? 0 : last - first);  }  template<class T>  T& Vect<T>::operator[](size\_t i) { // ---------------- Операция доступа по индексу  if (i < 0 || i >(size() - 1))  throw VectRangeErr(0, last - first - 1, i);  return (\*(first + i));  }  template<class T> // ------------- Вставка элемента со значением \_x в позицию \_P  void Vect<T>::insert(T\* \_P, const T& \_x) {  size\_t n = size() + 1; // новый размер  T\* new\_first = new T[n \* sizeof(T)];  T\* new\_last = new\_first + n;  size\_t offset = \_P - first;  for (size\_t i = 0; i < offset; ++i)\* (new\_first + i) = \*(first + i);  \*(new\_first + offset) = \_x;  for (size\_t i = offset; i < n; ++i)\* (new\_first + i + 1) = \*(first + i);  Destroy();  first = new\_first;  last = new\_last;  }  template<class T>  void Vect<T>::push\_back(const T& \_x) { // ------- Вставка элемента в конец вектора  if (!size()) { Allocate(1); \*first = \_x; }  else insert(end(), \_x);  }  template<class T>  void Vect<T>::pop\_back() { // ----------------- Удаление элемента из конца вектора  if (last == first) throw VectPopErr();  T\* p = end() - 1;  p->~T();  last--;  }  template<class T>  void Vect<T>::show() const { // ---------------- Вывод в cout содержимого вектора  if (size() > 0)  {  cout << "Элементы вектора: ";  size\_t n = size();  for (size\_t i = 0; i < n; ++i)  cout << \*(first + i) << " ";  cout << endl;  cout << endl;  }  else  {  cout << "Вектор пуст";  cout << endl;  cout << endl;  }  } |

## Тестирование программы

Результат работы программы приведен на рисунке 3.1.

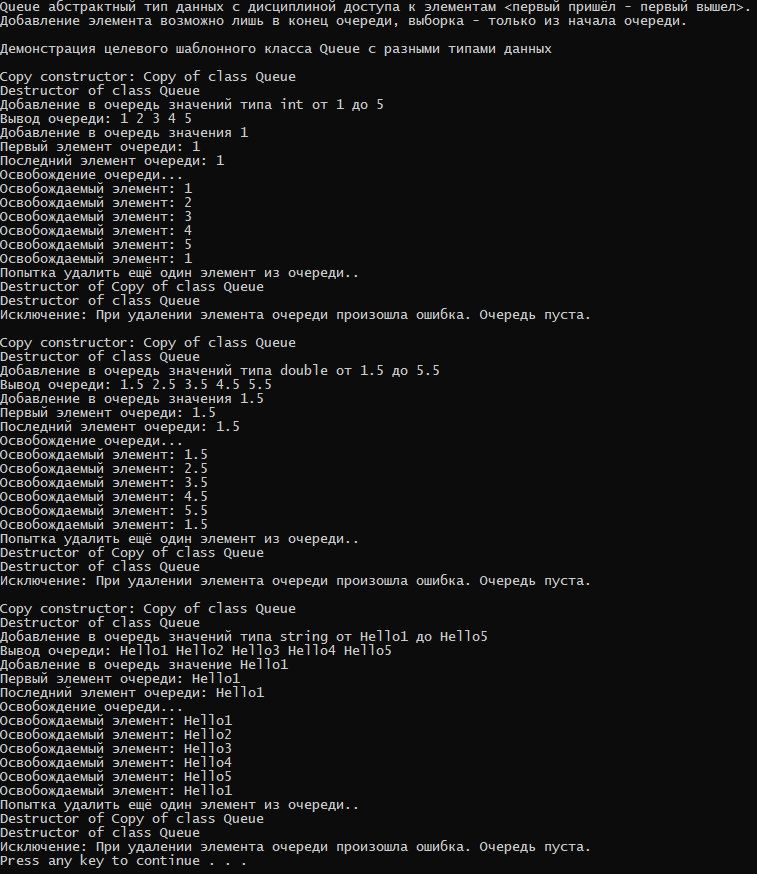


Рисунок 3.1 – Результат работы программы по шаблонам классов

# 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 упорядочены по номерам поездов; вывод на экран информации о поезде, номер которого введен с клавиатуры (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 <string>  #include <vector>  #include <Windows.h>  #include <algorithm>  using namespace std;  class Train  {  public:  friend istream& operator >> (istream& in, Train& ob);  friend ostream& operator<< (ostream& out, Train& ob);  friend class TrainCollection;  double getNumber()  {  return \_number;  }  private:  string \_destination;  string \_departureTime;  int \_number;  };  istream& operator >> (istream& in, Train& ob)  {  cin.ignore();  cout << "Название пункта назначения: ";  getline(in,ob.\_destination);  cout << "Время отправления: ";  getline(in, ob.\_departureTime);  cout << "Номер поезда (целое число): "; cin >> ob.\_number;  return in;  }  ostream& operator<< (ostream& out, Train& ob)  {  out << "Название пункта назначения: " << ob.\_destination << "\n";  out << "Время отправления: " << ob.\_departureTime << "\n";  out << "Номер поезда: " << ob.\_number << "\n";  return out;  }  class TrainCollection  {  private:  vector <Train> \_trainCollection;  public:  TrainCollection()  {  \_trainCollection.reserve(0);  }  TrainCollection(size\_t numberTrain)  {  \_trainCollection.reserve(numberTrain);  }  void sort()  {  std::sort(\_trainCollection.begin(), \_trainCollection.end(), []( Train & a, Train & b)  {  return a.getNumber() < b.getNumber();  });  }  void initInput() {  for (int i = 0; i < \_trainCollection.capacity(); i++)  {  Train N;  cout << "Ввод данных о " << i + 1 << " поезде\n";  cin >> N;  cout << "\n";  \_trainCollection.push\_back(N);  }  }  void show()  {  cout << "База данных поездов:\n\n";  for (int i = 0; i < \_trainCollection.capacity(); i++)  {  cout << \_trainCollection[i];  cout << "\n";  }  }  void searchTrain()  {  cin.ignore();  int number;  cout << "Поиск поездов по номеру: "; cin >> number;  bool z = false;  for (auto train : \_trainCollection)  {  if (train.\_number == str)  {  cout << train;  z = true;  }  }  if (!z)  cout << "Такие поездов нет!" << endl;  }  };  int main()  {  SetConsoleCP(1251);  SetConsoleOutputCP(1251);  size\_t numberTrain;  cout << "Введите количество поездов: ";  cin >> numberTrain;  cout << "\n";  TrainCollection trainCollection(numberTrain);  trainCollection.initInput();  trainCollection.sort();  trainCollection.show();  trainCollection.searchTrain();  system("pause");  return 0;  } |

## Тестирование программы

Результат работы программы приведен на рисунке 4.1.

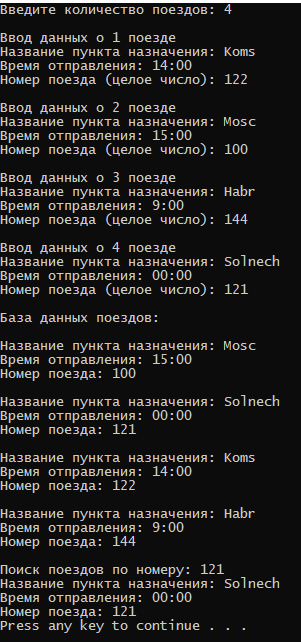


Рисунок 4.1 – Результат работы программы по стандартным потокам

# 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 предложения, состоящие из заданного количества слов.

## 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;  system("pause");  return -1;  }  int count;  cout << "Введите количество слов в предложении (предложения разделяются символами .!?): "; cin >> count;  string text;  while (getline(fin, text))  {  int pos1 = 0;  int pos2 = -1;  while (pos2 != 0)  {  stringstream sentence;  string sentenceTemp;  // нахождение позиции конца предложения (начало предложения pos1)  pos2 = text.find\_first\_of(".?!", pos1) + 1;  // запись найденного предложения в string поток  if (pos2 == 0)  {  sentence << text.substr(pos1);  }  else  {  sentence << text.substr(pos1, pos2);  }  // определение количества слов в предложении и вывод  bool entryCheck = false;  int countTemp = 0;  while (!sentence.eof())  {  sentence >> sentenceTemp;  countTemp++;  }  if (count == countTemp)  {  cout << sentence.str() << endl;  }  // смещение указателя на одно предложение  pos1 = pos2;  }  }  cout << endl;  fin.close();  system("pause");  return 0;  } |

## Тестирование программы

Результат работы программы приведен на рисунках 5.1, 5.2.

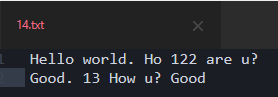


Рисунок 5.1 – Текстовый файл

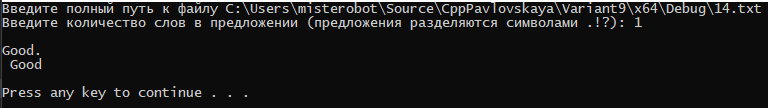


Рисунок 5.2 – Результат работы программы по файловым потокам

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»).