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 2

Студент группы 7ИСб-1 И. Нозимзода

Преподаватель С.Ю. Александров

2019

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 12](#_Toc9287458)

[2.1 Описание программы 12](#_Toc9287459)

[2.2 Текст программы 12](#_Toc9287460)

[2.3 Тестирование программы 21](#_Toc9287461)

[3 Шаблоны классов. Обработка исключительных ситуаций 23](#_Toc9287462)

[3.1 Описание программы 23](#_Toc9287463)

[3.2 Текст программы 23](#_Toc9287464)

[3.3 Тестирование программы 25](#_Toc9287465)

[4 Стандартные потоки 27](#_Toc9287466)

[4.1 Описание программы 27](#_Toc9287467)

[4.2 Текст программы 27](#_Toc9287468)

[4.3 Тестирование программы 29](#_Toc9287469)

[5 Файловые и строковые потоки. Строки класса string. 31](#_Toc9287470)

[5.1 Описание программы 31](#_Toc9287471)

[5.2 Текст программы 31](#_Toc9287472)

[5.3 Тестирование программы 32](#_Toc9287473)

[6 Стандартная библиотека шаблонов 33](#_Toc9287474)

[6.1 Описание программы 33](#_Toc9287475)

[6.2 Текст программы 33](#_Toc9287476)

[6.3 Тестирование программы 37](#_Toc9287477)

[Заключение 39](#_Toc9287478)

[Список использованных источников 40](#_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

Описать класс, реализующий бинарное дерево, обладающее возможностью добавления новых элементов, удаления существующих, поиска элемента по ключу, а также последовательного доступа ко всем элементам.

Написать программу, использующую этот класс для представления англо-русского словаря. Программа должна содержать меню, позволяющее выполнить проверку всех методов класса. Предусмотреть возможность создания словаря из файла и с клавиатуры.

## Текст программы

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

Листинг . – Текст файла Lab10.cpp

|  |
| --- |
| #include <iostream>  #include <string>  #include <fstream>  #include <windows.h>  using namespace std;  struct node  {  string Key;  string Count;  node\* Left;  node\* Right;  };  class TREE  {  private:  node\* Tree; //Указатель на корень дерева.  void Search(string, string, node\*\*);  public:  TREE()  {  Tree = NULL;  }  node\*\* GetTree()  {  return &Tree;  } //Получение вершины дерева.  node\* GetTreee()  {  return Tree;  } //Получение вершины дерева.  void BuildTree();  node\* pairRemove(string, string, node\*);  void searchPair(string, node\*\*);  void VyvodPair(node\*\*, int);  void to\_file(node\*, ofstream&);  void from\_file();  };  void TREE::BuildTree()  {  string Key;  string Count;  cout << "Английское слово: "; cin >> Key;  cout << "Русское слово: "; cin >> Count;  Search(Key, Count, &Tree);  }  node\* TREE::pairRemove(string key, string count, node\* Node)  {  if (Node == NULL)  return Node;  if (key == Node->Key) {  node\* tmp;  if (Node->Right == NULL)  {  tmp = Node->Left;  }  else {  node\* ptr = Node->Right;  if (ptr->Left == NULL) {  ptr->Left = Node->Left;  tmp = ptr;  }  else {  node\* pmin = ptr->Left;  while (pmin->Left != NULL) {  ptr = pmin;  pmin = ptr->Left;  }  ptr->Left = pmin->Right;  pmin->Left = Node->Left;  pmin->Right = Node->Right;  tmp = pmin;  }  }  if (Tree == Node)  {  Tree = tmp;  }  delete Node;  return tmp;  }  else if (key < Node->Key)  Node->Left = pairRemove(key, count, Node->Left);  else  Node->Right = pairRemove(key, count, Node->Right);  return Node;  }  void TREE::Search(string key, string count, node\*\* p)  {  if (\*p == NULL)  {  \*p = new(node);  (\*\*p).Key = key; (\*\*p).Count = count;  (\*\*p).Left = NULL; (\*\*p).Right = NULL;  }  else  if (key < (\*\*p).Key) Search(key, count, &((\*\*p).Left));  else  if (key > (\*\*p).Key) Search(key, count, &((\*\*p).Right));  else if ((\*\*p).Key == key) Search(key, count, &((\*\*p).Right));  }  void TREE::searchPair(string key, node\*\* w)  {  if (\*w != NULL)  {  if ((\*\*w).Key == key)  {  cout << (\*\*w).Key << "-" << (\*\*w).Count << endl;  }  searchPair(key, &(\*\*w).Right);  searchPair(key, &(\*\*w).Left);  }  }  void TREE::VyvodPair(node\*\* w, int l)  {  int i;  if (\*w != NULL)  {  VyvodPair(&((\*\*w).Right), l + 1);  for (i = 1; i <= l; i++) cout << " ";  cout << (\*\*w).Key << "-" << (\*\*w).Count << endl;  VyvodPair(&((\*\*w).Left), l + 1);  }  }  void TREE::to\_file(node\* w, ofstream &out)  {  if (w != NULL)  {  out << w->Key;  out << endl;  out << w->Count;  out << endl;  to\_file((\*w).Right, out);  to\_file((\*w).Left, out);  }  }  void TREE::from\_file()  {  ifstream in("database.txt");  string eng;  string rus;  while (getline(in, eng) && getline(in, rus))  {  Search(eng, rus, &Tree);  };  cout << "Данные успешно загружены из файла" << endl;  }  int main()  {  SetConsoleCP(1251); // Ввод с консоли в кодировке 1251  SetConsoleOutputCP(1251); // Вывод на консоль в кодировке 1251.  TREE A;  string eng;  string rus;  char key;  while (true)  {  cout << endl;  cout << "Англо-русский словарь" << endl  << "Выберите действие:" << endl  << "1. Добавить пару." << endl  << "2. Удалить пару." << endl  << "3. Найти пары по ключу." << endl  << "4. Вывод словаря {англ-рус}." << endl  << "5. Сохранить словать в файл." << endl  << "6. Загрузить словарь из файла." << endl  << "7. Выход." << endl  << ">";  cin >> key;  switch (key)  {  case '1':  A.BuildTree();  break;  case '2':  cout << "Введите английское слово: "; cin >> eng;  cout << "Введите русское слово: "; cin >> rus;  A.pairRemove(eng, rus, A.GetTreee());  break;  case '3':  cout << "Введите английское слово: "; cin >> eng;  A.searchPair(eng, A.GetTree());  break;  case '4':  cout << "\nВывод бинарного дерева:\n";  A.VyvodPair(A.GetTree(), 0);  break;  case '5':  {  ofstream out("database.txt", ios\_base::trunc);  A.to\_file(A.GetTreee(), out);  cout << "Данные о работниках сохранены в файл database.txt (найти его можно в папке проекта)" << endl;  cout << "Открыть файл с сотрудниками?\n1. Да\n2. Нет" << endl;  char a;  cin >> a;  if (a == '1')  {  ShellExecute(0, 0, "notepad.exe", "database.txt", 0, SW\_SHOW);  }  }  break;  case '6':  A.from\_file();  break;  case '7':  exit(1);  break;  }  }  return 0;} |

## Тестирование программы

Результаты работы программы приведены на рисунках 1.1,1.2.

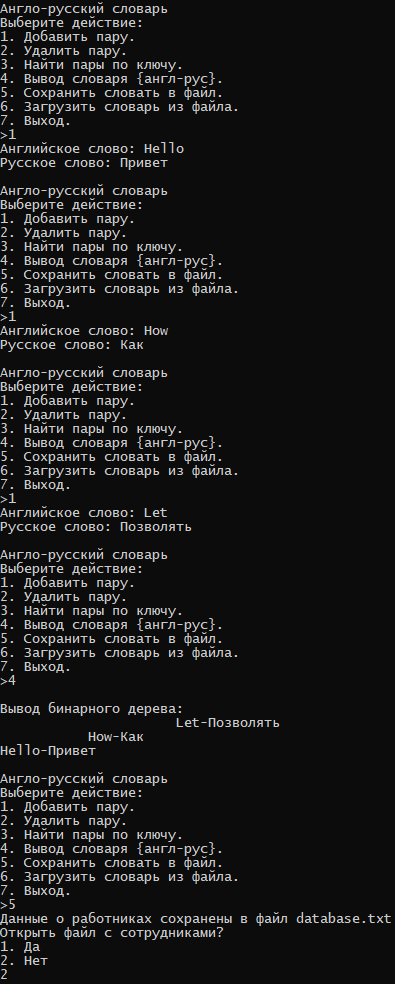


Рисунок . – Результат работы программы

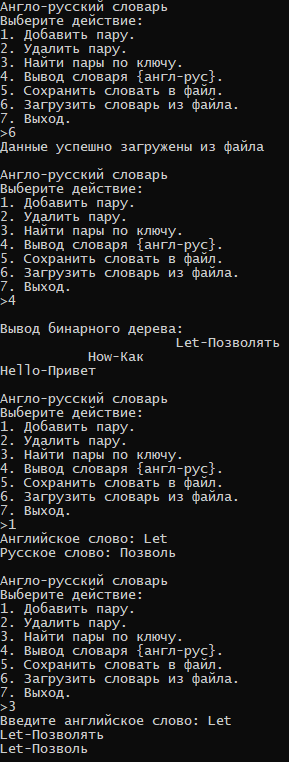


Рисунок .2 – Результат работы программы

# 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 BinString, 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 +(BinString, BinString). 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 BinString" << endl;  char Control;  while (true)  {  cout << endl << "Интерфейс программы:" << endl;  cout << "1. Объект SymbString (символьная строка)" << endl;  cout << "2. Объект BinString (двоичная 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':  {  FactoryBinString factoryBinString;  bool breakCheck = true;  while (breakCheck)  {  cout << "\nBinString (двоичная 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("01") != string::npos))  {  cout << "В строке могут присутствовать символы 0 и 1 без пробелов";  break;  };  factoryBinString.addObject(str);  break;  }  case '2':  factoryBinString.deleteObject();  break;  case '3':  factoryBinString.printContainer();  break;  case '4':  {  cout << "Первый объект" << endl;  cout << "Введите строку: ";  cin.ignore();  string str;  getline(cin, str);  if ((str.find\_first\_not\_of("01") != string::npos))  {  cout << "В строке могут присутствовать символы 0 и 1 без пробелов";  break;  };  BinString decString1(str);  cout << "Второй объект" << endl;  cout << "Введите строку: ";  getline(cin, str);  if ((str.find\_first\_not\_of("01") != string::npos))  {  cout << "В строке могут присутствовать символы 0 и 1 без пробелов";  break;  };  BinString decString2(str);  BinString decString = decString1 + decString2;  cout << "Результат" << endl;  decString.printObject();  cout << "Резултат добавлен в базу объектов" << endl;  factoryBinString.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 "BinString.h"  class String {  protected:  int id\_;  string string\_;  };  class FactorySymbString {  public:  FactorySymbString() {}  void addObject(string);  void addObject(SymbString);  void deleteObject();  void printContainer();  private:  vector <SymbString> symbStringContainers\_;  };  class FactoryBinString {  public:  FactoryBinString() {}  void addObject(string);  void addObject(BinString);  void deleteObject();  void printContainer();  private:  vector <BinString> 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())  {  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 FactoryBinString::addObject(string str) {  BinString symbString(str);  decStringContainers\_.push\_back(symbString);  }  void FactoryBinString::addObject(BinString symbString)  {  decStringContainers\_.push\_back(symbString);  }  void FactoryBinString::deleteObject() {  if (!decStringContainers\_.empty())  {  auto a = decStringContainers\_.size() - 1;  decStringContainers\_.erase(decStringContainers\_.begin() + a);  cout << "Последний созданный объект удалён" << endl;  }  else cout << "Объекты не найдены" << endl;  }  void FactoryBinString::printContainer()  {  if (!decStringContainers\_.empty())  {  for (BinString var : decStringContainers\_)  {  var.printObject();  cout << endl;  }  }  else cout << "Список объектов пуст" << endl;  } |

Листинг .4 – Текст файла BinString.h

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

Листинг .5 – Текст файла BinString.cpp

|  |
| --- |
| #include "BinString.h"  void BinString::printObject() {  cout << "Идентификатор объекта: ID" << id\_ << endl;  cout << "Значение объекта: " << string\_ << endl;  }  const BinString operator+(const BinString& left, const BinString& right)  {  BinString decString;  int len1 = left.string\_.size();  int len2 = right.string\_.size();  // Выровняем длины строк  string ls1 = len1 < len2 ? left.string\_ : right.string\_;  string ls2 = len1 < len2 ? right.string\_ : left.string\_;  for (int i = ls1.size(); i < ls2.size(); ++i) ls1 = '0' + ls1;  string result;  int carry = 0;  for (int i = ls2.size() - 1; i >= 0; --i) {  int bit1 = ls1.at(i) - '0'; // '0' => 0, '1' => 1  int bit2 = ls2.at(i) - '0';  char sum = (bit1 ^ bit2 ^ carry) + '0';  result = sum + result;  carry = (bit1 & carry) | (bit2 & carry) | (bit1 & bit2);  }  if (carry) result = '1' + result;  decString.string\_ = result;  return decString;  }  int BinString::DecString\_ID\_SEQUENCE = 0; |

Листинг 2.6 – Текст файла SymbString.h

|  |
| --- |
| #include <iostream>  #include <string>  using namespace std;  class SymbString : String  {  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.

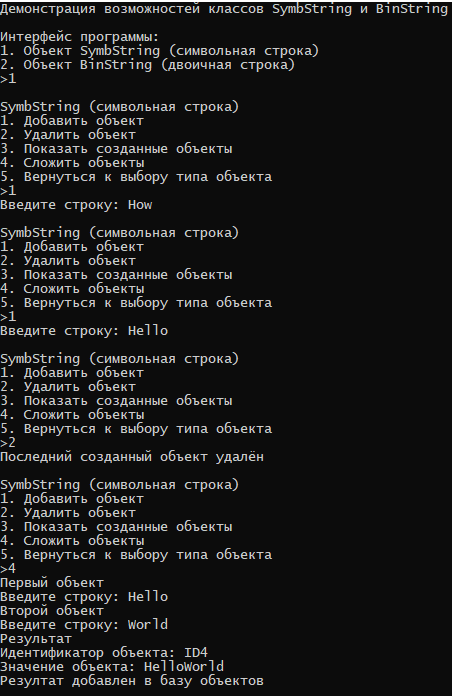


Рисунок .1 – Результат работы программы по наследованию

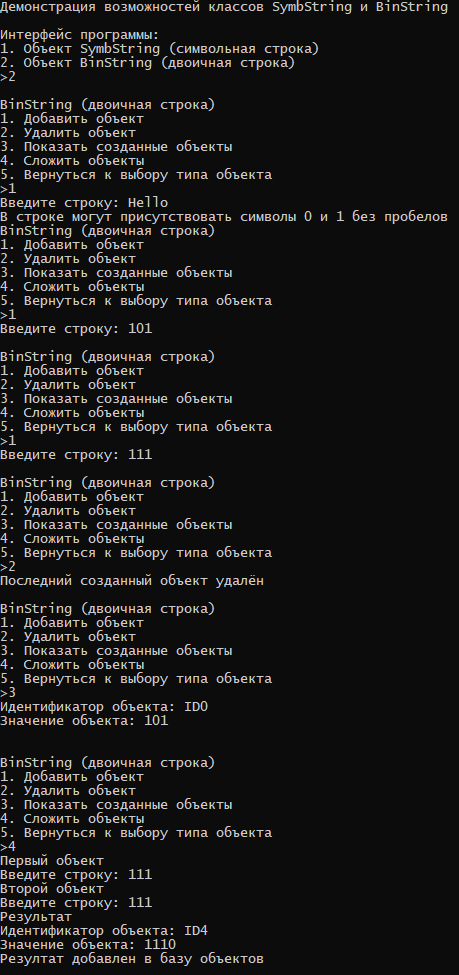


Рисунок .2 – Результат работы программы по наследованию

# 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 List.

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 "List.h"  int main()  {  setlocale(LC\_ALL, "rus");  cout << "List представляет собой контейнер, который поддерживает быструю вставку" << endl  << "и удаление элементов из любой позиции в контейнере." << endl  << "Быстрый произвольный доступ не поддерживается (т.к элементы расположены в памяти в хаотичном порядке)." << endl  << "List реализован в виде двусвязного списка." << endl << endl;  cout << "Демонстрация целевого шаблонного класса List с типом int" << endl;  try  {  // для int(для всего остального делается аналогично)  List<int> listInt(0);  cout << "Добавление в список значений типа int от 0 до 9" << endl;  for (int i = 0; i < 10; ++i)  listInt.push\_back(i);  cout << "Вывод списка" << endl;  listInt.output();  cout << "Введите элемент для удаления ";  int el; cin >> el;  listInt.remove(el);  cout << "Получившийся список" << endl;  listInt.output();  cout << "Введите элемент для вставки в начало ";  cin >> el;  listInt.push\_front(el);  cout << "Получившийся список" << endl;  listInt.output();  while (listInt.size() > 0)  {  listInt.pop\_back();  }  cout << "\nОчистка списка...Список пуст" << endl;  cout << "Попытка удалить ещё один элемент из списка.." << endl;  // исключение т.к. стек уже пуст  listInt.pop\_back();  }  catch (std::underflow\_error& e)  {  std::cerr << e.what() << std::endl;  }    cout << endl;  cout << "Демонстрация целевого шаблонного класса List с типом double" << endl;  try  {  // для int(для всего остального делается аналогично)  List<double> listInt(0);  cout << "Добавление в список значений типа double от 0.5 до 9.5" << endl;  for (double i = 0.5; i < 10; ++i)  listInt.push\_back(i);  cout << "Вывод списка" << endl;  listInt.output();  cout << "Введите элемент для удаления ";  double el; cin >> el;  listInt.remove(el);  cout << "Получившийся список" << endl;  listInt.output();  cout << "Введите элемент для вставки в начало ";  cin >> el;  listInt.push\_front(el);  cout << "Получившийся список" << endl;  listInt.output();  while (listInt.size() > 0)  {  listInt.pop\_back();  }  cout << "\nОчистка списка...Список пуст" << endl;  cout << "Попытка удалить ещё один элемент из списка.." << endl;  // исключение т.к. стек уже пуст  listInt.pop\_back();  }  catch (std::underflow\_error& e)  {  std::cerr << e.what() << std::endl;  }  cout << endl;  cout << "Демонстрация целевого шаблонного класса List с типом string" << endl;  try  {  List<string> listInt(0);  cout << "Добавление в список значений типа string от '0' до '9'" << endl;  for (int i = 0; i < 10; ++i)  {  string s = "'" + to\_string(i) + "'";  listInt.push\_back(s);  }  cout << "Вывод списка" << endl;  listInt.output();  cout << "Введите элемент для удаления ";  string el; cin >> el;  listInt.remove(el);  cout << "Получившийся список" << endl;  listInt.output();  cout << "Введите элемент для вставки в начало";  cin >> el;  listInt.push\_front(el);  cout << "Получившийся список" << endl;  listInt.output();  while (listInt.size() > 0)  {  listInt.pop\_back();  }  cout << "\nОчистка списка...Список пуст" << endl;  cout << "Попытка удалить ещё один элемент из списка.." << endl;  // исключение т.к. стек уже пуст  listInt.pop\_back();  }  catch (std::underflow\_error& e)  {  std::cerr << e.what() << std::endl;  }  cout << endl;  system("pause");  return 0;  } |

Листинг 3. – Текст файла List.h

|  |
| --- |
| #pragma once  #include <iostream>  #include <string>  using namespace std;  template<typename T>  class ListNode  {  public:  T value;  ListNode<T>\* next;  ListNode<T>\* prev;  public:  ListNode(T data)  {  value = data;  next = 0;  prev = 0;  }  };  template <typename T>  class List  {  private:  ListNode<T>\* \_pBegin;  ListNode<T>\* \_pEnd;  unsigned int \_size;  static const int defaultValue = 0;  public:  List();  List(int size);  List(int size, T data);  ListNode<T>\* back(); //указатель на конец  ListNode<T>\* begin(); // указатель на начало  bool empty(); // проверка на пустоту  ListNode<T>\* erase(ListNode<T>\* pWhere); // удалить узел  ListNode<T>\* find(T data); // найти первый узел, содержащий data  ListNode<T>\* insertBefore(ListNode<T>\* Where, T data); // вставка до  ListNode<T>\* insertАfter(ListNode<T>\* Where, T data); // вставка после  void assign(T data);  void output(); // вывод  bool pop\_back(); // удаление с конца  bool pop\_front(); // удаление с начала  void push\_back(T data); // добавление в конец  void push\_front(T data); // добавление в начало  void remove(T data); // удалить все узлы, содержащие такую data  void resize(unsigned int newLen); // изменить размер  unsigned int size(); // размер списка  private:  inline void Next(ListNode<T>\*\* pCurNode)  {  \*pCurNode = (\*pCurNode)->next;  }  inline void Prev(ListNode<T>\*\* pCurNode)  {  \*pCurNode = (\*pCurNode)->prev;  }  };  template<typename T>  List<T> ::List() :\_pBegin(NULL),  \_pEnd(NULL),  \_size(0)  {  }  template<typename T>  List<T>::List(int size) : \_pBegin(NULL),  \_pEnd(NULL),  \_size(0)  {  resize(size);  }  template<typename T>  List<T>::List(int size, T data) :\_pBegin(NULL),  \_pEnd(NULL),  \_size(0)  {  resize(size);  assign(data);  }  template<typename T>  ListNode<T>\* List<T>::back()  {  return \_pEnd;  }  template<typename T>  ListNode<T>\* List<T>::begin()  {  return \_pBegin;  }  template<typename T>  bool List<T>::empty()  {  return \_size == 0;  }  template<typename T>  ListNode<T>\* List<T>::erase(ListNode<T>\* pWhere)  {  if (pWhere == NULL)  return false;  if (pWhere == \_pBegin)  Next(&\_pBegin);  else if (pWhere == \_pEnd)  Prev(&\_pEnd);  ListNode<T>\* pPrevNode = pWhere->prev;  ListNode<T>\* pNextNode = pWhere->next;  if (pNextNode)  pNextNode->prev = pPrevNode;  if (pPrevNode)  pPrevNode->next = pNextNode;  delete pWhere;  \_size--;  return pNextNode;  }  template<typename T>  ListNode<T>\* List<T>::find(T data)  {  ListNode<T>\* cur = NULL;  for (cur = \_pBegin; cur; Next(&cur))  if (cur->value == data)  break;  return cur;  }  template<typename T>  ListNode<T>\* List<T>::insertBefore(ListNode<T>\* pWhere, T data)  {  if (pWhere == NULL && \_size != 0) // некорретный указатель  return NULL;  ListNode<T>\* pNewNode = new ListNode<T>(data);  if (pWhere == NULL && \_size == 0) // список пуст  {  \_pBegin = pNewNode;  \_pEnd = pNewNode;  }  else  {  // Есть предыдущий узел  if (pWhere->prev)  {  ListNode<T>\* pPrevNode = pWhere->prev;  pPrevNode->next = pNewNode;  pNewNode->prev = pPrevNode;  pNewNode->next = pWhere;  pWhere->prev = pNewNode;  }  // Where = \_pBegin  else  {  \_pBegin->prev = pNewNode;  pNewNode->next = \_pBegin;  \_pBegin = pNewNode;  }  }  \_size++;  return pNewNode;  }  template<typename T>  ListNode<T>\* List<T>::insertАfter(ListNode<T>\* Where, T data)  {  if (Where == NULL && \_size != 0) // некорректный указатель  return NULL;  ListNode<T>\* pNewNode = new ListNode<T>(data);  if (Where == NULL && \_size == 0)  {  \_pBegin = pNewNode;  \_pEnd = pNewNode;  }  else  {  // Есть следующий элемент  if (Where->next)  {  ListNode<T>\* pNextNode = Where->next;  Where->next = pNewNode;  pNewNode->prev = Where;  pNewNode->next = pNextNode;  pNextNode->prev = pNewNode;  }  // Where = \_pEnd  else  {  \_pEnd->next = pNewNode;  pNewNode->prev = \_pEnd;  \_pEnd = pNewNode;  }  }  \_size++;  return pNewNode;  }  template<typename T>  void List<T>::assign(T data)  {  for (ListNode<T>\* cur = \_pBegin; cur; Next(&cur))  cur->value = data;  }  template<typename T>  void List<T>::output()  {  for (ListNode<T>\* cur = \_pBegin; cur; Next(&cur))  cout << cur->value << ' ';  cout << endl;  }  template<typename T>  bool List<T>::pop\_back()  {  if(empty()) throw underflow\_error("Исключение: При удалении последнего элемента списка произошла ошибка. Список пуст.");  return erase(\_pEnd);  }  template<typename T>  bool List<T>::pop\_front()  {  return erase(\_pBegin);  }  template<typename T>  void List<T>::push\_back(T data)  {  insertАfter(\_pEnd, data);  }  template<typename T>  void List<T>::push\_front(T data)  {  insertBefore(\_pBegin, data);  }  template<typename T>  void List<T>::remove(T data)  {  bool nodesExist = false;;  for (ListNode<T>\* cur = \_pBegin; cur;)  if (cur->value == data)  {  cur = erase(cur);  nodesExist = true;  }  else  Next(&cur);  if(!nodesExist) throw underflow\_error("Исключение: Удаляемый элемент отсутствует.");  }  template<typename T>  void List<T>::resize(unsigned int newLen)  {  if (\_size > newLen)  while (\_size != newLen)  pop\_back();  else if (\_size < newLen)  while (\_size != newLen)  push\_back(defaultValue);  }  template<typename T>  unsigned int List<T>::size()  {  return \_size;  } |

## Тестирование программы

Результат работы программы приведен на рисунке 3.1.

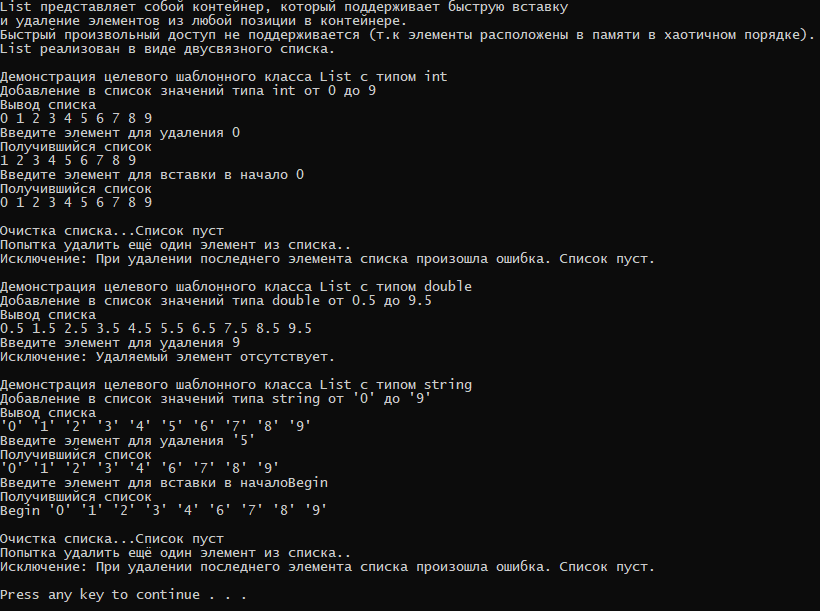


Рисунок 3.1 – Результат работы программы по шаблонам классов

# 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 STUDENT, содержащий следующие поля: фамилия и инициалы; номер группы; успеваемость (массив из пяти элементов). Определить методы доступа к этим полям и перегруженные операции извлечения и вставки для объектов типа STUDENT.

Написать программу, выполняющую следующие действия: ввод с клавиатуры данных в массив, состоящий из десяти объектов типа STUDENT; записи должны быть упорядочены по возрастанию среднего балла; вывод на дисплей фамилий и номеров групп для всех студентов, имеющих оценки 4 и 5; если таких студентов нет, вывести 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 Student  {  public:  friend istream& operator >> (istream& in, Student& ob);  friend ostream& operator<< (ostream& out, Student& ob);    double getGPA()  {  double sum = 0;  for (size\_t i = 0; i < 5; i++)  sum += ses[i];  return sum / 5;  }  bool statusGoodStudent()  {  int x = 0;  for (int j = 0; j < 5; j++)  if (ses[j] == 4 || ses[j] == 5)  x++;  if (x == 5)  return true;  else return false;  }  private:  string name;  string group;  int ses[5];  };  istream& operator >> (istream& in, Student& ob)  {  cin.get();  cout << "ФИО: ";  getline(in,ob.name);  cout << "Группа: ";  in >> ob.group;  cout << "Оценки:\n";  for (size\_t i = 0; i < 5; i++)  in >> ob.ses[i];  return in;  }  ostream& operator<< (ostream& out, Student& ob)  {  out << "ФИО: " << ob.name << "\n";  out << "Группа: " << ob.group << "\n";  out << "Оценки: ";  for (size\_t i = 0; i < 5; i++)  out << ob.ses[i] << " ";  out << "\n";  return out;  }  class University  {  private:  vector <Student> students;  public:  University()  {  students.reserve(0);  }  University(size\_t numberStudents)  {  students.reserve(numberStudents);  }  void sort()  {  std::sort(students.begin(), students.end(), []( Student & a, Student & b)  {  return a.getGPA() < b.getGPA();  });  }  void initInput() {  for (int i = 0; i < students.capacity(); i++)  {  Student N;  cout << "Ввод данных о " << i + 1 << " студенте\n";  cin >> N;  cout << "\n";  students.push\_back(N);  }  }  void show()  {  cout << "База данных студентов:\n\n";  for (int i = 0; i < students.capacity(); i++)  {  cout << students[i];  cout << "\n";  }  }  void showStars()  {  cout << "Поиск студентов, которые хорошо учатся..." << endl;  bool z = false;  for (int i = 0; i < students.size(); i++)  {  if (students[i].statusGoodStudent())  {  cout << students[i];  z = true;  }  }  if (!z)  cout << "Среди студентов нет хорошистов" << endl;  }  };  int main()  {  SetConsoleCP(1251);  SetConsoleOutputCP(1251);  //system("color 0A ");  size\_t numberStudents;  cout << "Введите количество студентов: ";  cin >> numberStudents;  cout << "\n";  University KnASU(numberStudents);  KnASU.initInput();  KnASU.sort();  KnASU.show();  KnASU.showStars();  system("pause");  return 0;  } |

## Тестирование программы

Результат работы программы приведен на рисунке 4.1.

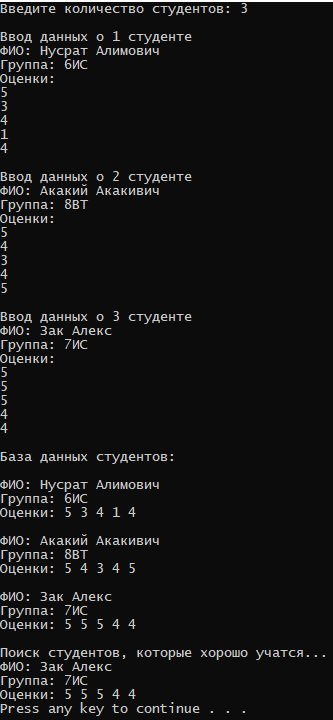


Рисунок 4.1 – Результат работы программы по стандартным потокам

# 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, содержащие введенное с клавиатуры слово.

## Текст программы

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

Листинг 5. – Текст файла Lab14.cpp

|  |
| --- |
| #define \_CRT\_SECURE\_NO\_WARNINGS  #include <iostream>  #include <iomanip>  #include <Windows.h>  #include <fstream>  using namespace std;  int main()  {  setlocale(LC\_ALL, "Russian");  //system("color 0A ");  ifstream fin("text.txt");  if (!fin)  {  cout << "Файл не может быть открыт!" << endl << endl;  }  else  {  fin.seekg(0, ios::end);  long len = fin.tellg();  char \*str1 = new char[len + 1];  fin.seekg(0, ios::beg);  fin.read(str1, len);  str1[len] = '\0';  int k = 0, i = 0, j = 0; // k - начало нового предложения, i - счетчик циклов, j - счетчик найденных слов  char \*token;  char deliteli[] = ",.!?\|/<>)(\*;:' ";  const int bukvy = 15;  char slovo[bukvy];  const int max = 10000;  char predlojeniya[max], rezerv[max];  cout << "Здравствуйте, Вы находитесь в программе, которая ищет в установленном текстовом файле предложения c введенным Вами словом.\n\nВведите искомое слово: ";  cin >> setw(bukvy) >> slovo;  OemToCharA(slovo, slovo);  cout << "Слово, введенное Вами: " << slovo << endl;  // считаем кол-во точек/вопр/воскл в исходном тексте, ведь в тексте может вообще не быть предложений  int n = 0;  for (i = 0; i < strlen(str1); i++)  if (str1[i] == '.' || str1[i] == '!' || str1[i] == '?') n++;  if (n == 0) cout << endl << "Текст в файле вообще не содержит предложений" << endl << endl;  // основная часть  else  {  cout << endl << "Предложения с введенным Вами словом:" << endl;  for (i = 0; i < strlen(str1); i++)  {  if (str1[i] == '.' || str1[i] == '!' || str1[i] == '?')  {  strncpy(predlojeniya, str1 + k, i - k + 1);  predlojeniya[i - k + 1] = '\0';  strcpy(rezerv, predlojeniya); // функция strtok вычленяет токены, разделяемые любым из символов ",.!?\|/<>)(\*;:' ",  token = strtok(predlojeniya, deliteli); // из исходной строки (при этом разрушая ее!!!). Поэтому я сделал её дубликат (rezerv)  while (token != NULL)  {  if (!strcmp(token, slovo)) { cout << rezerv; j++; }  token = strtok(NULL, deliteli);  }  k = i + 1;  }  }  if (j == 0) cout << endl << "К сожалению, текст не содержит предложений со словом <" << slovo << ">" << endl;  cout << endl << "--------------------------------------------------------------------------------" << endl;  }  }  system("pause");  return 0;  } |

## Тестирование программы

Результат работы программы приведен на рисунках 5.1, 5.2.

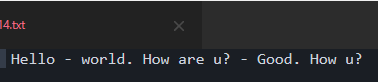


Рисунок 5.1 – Текстовый файл

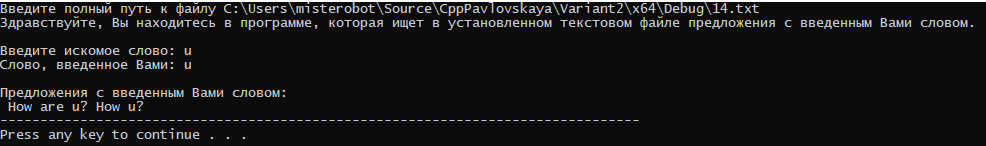


Рисунок 5.2 – Результат работы программы по файловым потокам

# 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, решающую игру-головоломку «Игра в 15». Начальное размещение номеров — случайное. В программе использовать контейнерные классы STL.*

## Текст программы

Проект состоит из 3 файлов исходного кода, которые приведены в листингах 6.1-6.3.

Листинг 6. – Текст файла Lab15.cpp

|  |
| --- |
| #include "Game15Puzzle.h"  int main()  {  setlocale(LC\_ALL, "rus");  Game15Puzzle game15Puzzle;  game15Puzzle.mainMenu();  return 0;  } |

Листинг 6.2 – Текст файла Game15Puzzle.h

|  |
| --- |
| #pragma once  #include <iostream>  #include <iomanip>  #include <vector>  using namespace std;  #define size 4  class Game15Puzzle  {  public:  Game15Puzzle();  void mainMenu();  private:  vector<vector<int>> board;  void cleanBoard();  bool checkForSolution();  bool checkBoard();  void showBoard();  void changeBoard();  void fillBoard();  void startGame();  }; |

Листинг 6.3 – Текст файла Game15Puzzle.cpp

|  |
| --- |
| #include "Game15Puzzle.h"  Game15Puzzle::Game15Puzzle()  {  board.resize(size);  for (size\_t i = 0; i < size; ++i)  {  board[i].resize(size);  for (size\_t j = 0; j < size; ++j)  board[i][j] = 0;  }  }  void Game15Puzzle::mainMenu()  {  system("cls");  std::cout << "\nИгра в 15"  << std::endl;  std::cout << "1.Новая игра\n"  << "2.Правила игры\n"  << "3.Выход\n";  bool y = 0;  int choice;  while (!y)  {  std::cout << ">";  std::cin >> choice;  if (choice < 1 || choice > 3)  std::cout << "Некорректный ввод! Попробуйте снова." << std::endl;  else  y = 1;  }  switch (choice)  {  case 1:  system("cls");  std::cout << "Подсказка: в текущей версии игры пустая клетка представлена нулём." << std::endl;  std::cout << "Подсказка: для перемещения пустой клетки (нуля) вверх, вниз, влево или вправо\n"  << "требуется ввести значение соседнего элемента, который поменяется с пустой клеткой." << std::endl;  system("pause");  startGame();  break;  case 2:  system("cls");  std::cout << endl  << "Игра в 15 это популярная головоломка, придуманная в 1878 году Ноем Чепмэном.\n"  << "Игра представляет собой поле для игры размером 4х4 на котором случайным образом расположены числа от 1 до 15.\n"  << "Одна клетка остается пустой. Перемещая соседние числа, упорядочите их расположение (слева – направо и сверху – вниз).\n"  << "Например, первый ряд – 1, 2, 3, 4, второй – 5, 6, 7, 8 и т.д. «Вытаскивать» числа и перескакивать через числа не разрешается."  << std::endl;  system("pause");  mainMenu();  break;  case 3:  exit(0);  }  }  void Game15Puzzle::startGame()  {  do  {  cleanBoard();  fillBoard();  } while (!checkForSolution());  int turns = 0;  while (!checkBoard())  {  showBoard();  changeBoard();  turns++;  }  std::cout << "Вы победили за "  << turns << " хода!"  << std::endl;  system("pause");  }  void Game15Puzzle::cleanBoard()  {  for (int i = 0; i < size; i++)  for (int j = 0; j < size; j++)  board[i][j] = 0;  }  bool Game15Puzzle::checkForSolution()  {  int solution[size\*size] = { 0 };  int index = 0;  for (int i = 0; i < size; i++)  for (int j = 0; j < size; j++, index++)  solution[index] = board[i][j];  int inv = 0;  for (int i = 0; i < size \* size; i++)  if (solution[i])  for (int j = 0; j < i; j++)  if (solution[j] > solution[i])  ++inv;  for (int i = 0; i < size \* size; i++)  if (solution[i] == 0)  inv += 1 + i / 4;  return (inv & 1) ? 0 : 1;  }  bool Game15Puzzle::checkBoard()  {  int count = 1;  for (int i = 0; i < size; i++)  for (int j = 0; j < size; j++, count++)  {  if (i == size - 1 && j == size - 1 && board[i][j] == 0) break;  if (board[i][j] != count)  return 0;  }  return 1;  }  void Game15Puzzle::showBoard()  {  system("cls");  std::cout << "Игра в 15"  << std::endl;  for (int i = 0; i < size; i++)  {  std::cout << "-------------" << std::endl;  std::cout << "|";  for (int j = 0; j < size; j++)  {  std::cout << std::left  << std::setw(2)  << board[i][j]  << "|";  }  std::cout << std::endl;  }  std::cout << "-------------" << std::endl;  }  void Game15Puzzle::changeBoard()  {  bool y = 0;  while (!y)  {  int turn = 0;  do  {  std::cout << ">";  std::cin >> turn;  } while (turn < 0 || turn > 15);  if (turn == 0)  {  mainMenu();  return;  }  int getx = 0;  int gety = 0;  for (int i = 0; i < size; i++)  for (int j = 0; j < size; j++)  if (board[i][j] == turn)  {  getx = i;  gety = j;  }  int dx[] = { 0, 0, 1, -1 };  int dy[] = { 1, -1, 0, 0 };  for (int i = 0; i < 4; i++)  {  int newx = getx + dx[i];  int newy = gety + dy[i];  if (newx < 0 || newx > size - 1 ||  newy < 0 || newy > size - 1)  continue;  if (board[newx][newy] == 0)  {  board[newx][newy] = turn;  board[getx][gety] = 0;  y = 1;  break;  }  }  if (!y)  std::cout << "Введите числа находящиеся слева, справа, сверху или снизу (обращайте внимание на стены)!" << std::endl;  }  }  void Game15Puzzle::fillBoard()  {  bool already[15] = { 0 };  int count = 1;  for (int i = 0; i < size; i++)  for (int j = 0; j < size; j++, count++)  {  if (i == size - 1 && j == size - 1)  break;  while (true)  {  int randNum = rand() % 15;  if (!already[randNum])  {  already[randNum] = 1;  board[i][j] = randNum + 1;  break;  }  }  }  } |

## Тестирование программы

Результат работы программы приведен на рисунках 6.1-6.5.

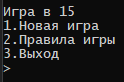


Рисунок 6.1 – Главное меню

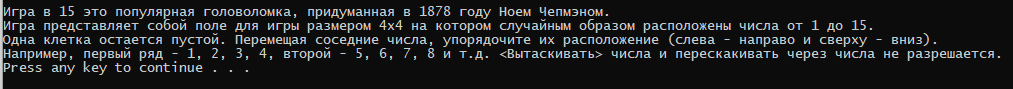


Рисунок 6.2 – Правила игры

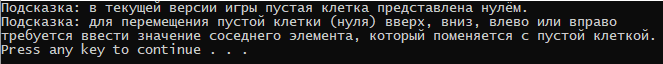


Рисунок 6.3 – Подсказки перед игрой

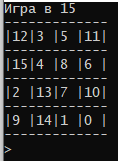


Рисунок 6.4 – Игровой стол

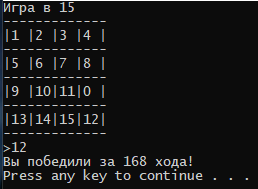


Рисунок 6.4 – Победа

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»).