Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное

образовательное учреждение высшего образования

«Комсомольский-на-Амуре государственный университет»

Факультет компьютерных технологий Кафедра МОП ЭВМ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

по дисциплине «Технологии программирования»

Использование стандартной библиотеки шаблонов

Вариант 2

|  |  |
| --- | --- |
| Студент группы 6ИСб-1 | Н. Давлатов |
| Руководитель работы | С.Ю. Александров |
| Нормоконтролёр | С.Ю. Александров |

**Содержание**

|  |  |  |  |
| --- | --- | --- | --- |
| Введение ....................................................................................................................... | | | 5 |
| 1 | Стандартные потоки ............................................................................................. | | 6 |
|  | 1.1 | Описание программы ..................................................................................... | 6 |
|  | 1.2 | Текст программы ............................................................................................ | 6 |
|  | 1.3 | Тестирование программы .............................................................................. | 9 |
| 2 Файловые и строковые потоки. Строки класса string ..................................... | | | 10 |
|  | 2.1 | Описание программы ................................................................................... | 10 |
|  | 2.2 | Текст программы .......................................................................................... | 10 |
|  | 2.3 | Тестирование программы ............................................................................ | 12 |
| 3 | Стандартная библиотека шаблонов .................................................................. | | 13 |
|  | 3.1 | Описание программы ................................................................................... | 13 |
|  | 3.2 | Текст программы .......................................................................................... | 13 |
|  | 3.3 | Тестирование программы ............................................................................ | 16 |
| Заключение ................................................................................................................ | | | 17 |
| Список использованных источников ...................................................................... | | | 18 |

**Введение**

В любой программе, кроме операторов языка, используются средства библиотек, включаемых в среду программирования. Различные среды предоставляют в распоряжение программиста разные наборы средств, облегчающих создание программ, — например, компиляторы Microsoft Visual C++ и Borland C++ содержат библиотеки классов для написания приложений Windows. Часть библиотек стандартизована, то есть должна поставляться с любым компилятором языка C++. Стандартную библиотеку C++ можно условно разделить на две части. К первой относятся функции, макросы, типы и константы, унаследованные из библиотеки С, а ко второй — классы и другие средства C++. Вторая часть содержит классы, шаблоны и другие средства для ввода, вывода, хранения и обработки данных как стандартных типов, так и типов, определенных пользователем.

Часть библиотеки, в которую входят контейнерные классы, алгоритмы и итераторы, называют стандартной библиотекой шаблонов (STL — Standard Template Library).

1. **Стандартные потоки**

Потоки, связанные с консольным вводом-выводом (клавиатура–экран), называются стандартными. Стандартному потоку ввода соответствует класс istream, стандартному потоку вывода — класс ostream. Оба класса являются потомками класса ios. Следующим в иерархии является класс iostream, наследующий классы istream и ostream и обеспечивающий общие средства потокового ввода-вывода [1, стр. 267].

* 1. **Описание программы**

1. *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.*

1. *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.*

**1.2** **Текст программы**

Проект состоит из 1 файла исходного кода, который приведен в листинге 1:

Листинг 1 – Текст файла University.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;  } |

**1.3** **Тестирование программы**

Результат работы программы приведён на рисунке 1.

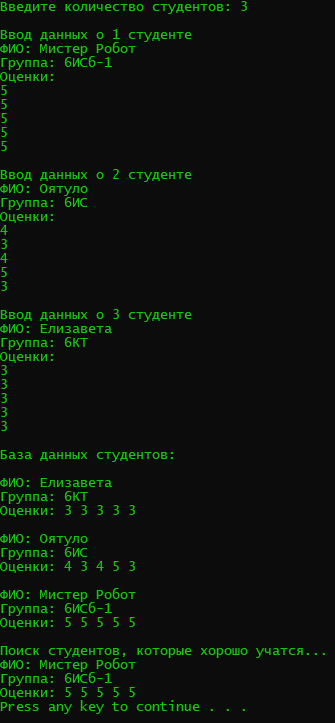


Рисунок 1 – Результат работы программы

1. **Файловые и строковые потоки. Строки класса string**

По способу доступа файлы можно разделить на последовательные, чтение и запись в которых производятся с начала байт за байтом, и файлы с произвольным доступом, допускающие чтение и запись в указанную позицию.

Строковые потоки позволяют считывать и записывать информацию из областей оперативной памяти так же, как из файла, с консоли или на дисплей.

Важнейшей особенностью класса string является управление памятью как при размещении строки, так и при ее модификациях, изменяющих длину строки. Кроме этого, строки типа string защищены от ошибочных обращений к памяти, связанных с выходом за их границы [1, стр. 280].

**2.1** **Описание программы**

*Написать программу, которая считывает текст из файла и выводит на экран только предложения, содержащие введенное с клавиатуры слово.*

**2.2** **Текст программы**

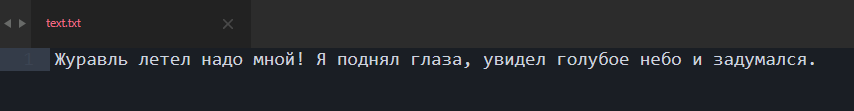
Проект состоит из одного файла исходного кода, который приведен в листинге 2:

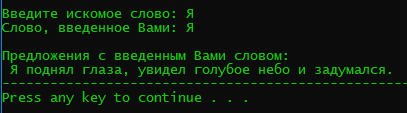
Листинг 2 – Текст файла Text.cpp

|  |
| --- |
| #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;  cout << "--------------------------------------------------------------------------------";  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;  } |

**2.3** **Тестирование программы**

Результат работы программы при считывании текста из файла \*txt (рисунок 2.1) приведён на рисунке 2.2.

Рисунок 2.1 – Файл \*txt

Рисунок 2.2 – Результат работы программы

1. **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.

Обобщенные алгоритмы реализуют большое количество процедур, применимых к контейнерам — например, поиск, сортировку, слияние и т. п., но они не являются методами контейнерных классов. Наоборот, алгоритмы представлены в STL в форме глобальных шаблонных функций [1, стр. 294].

**3.1** **Описание программы**

Написать программу, отыскивающую проход по лабиринту, с использованием контейнерного класса stack из STL. Лабиринт представляется в виде матрицы, состоящей из квадратов. Каждый квадрат либо открыт, либо закрыт. Вход в закрытый квадрат запрещен. Если квадрат. Стандартная библиотека шаблонов открыт, то вход в него возможен со стороны, но не с угла.

Программа находит проход через лабиринт, двигаясь от заданного входа. После отыскания прохода программа выводит найденный путь в виде координат квадратов.

**3.2** **Текст программы**

Проект состоит из одного файла исходного кода, который приведен в листинге 3.1:

Листинг 3.1 – Текст файла Labirint.cpp

|  |
| --- |
| #include <stack>  #include <iostream>  #include <string>  #include <algorithm>  #include <locale>  const int OPEN\_VAL = 0;  const int CLOSE\_VAL = 1;  const int START\_VAL = -1;  struct T\_cell  {  int i\_;  int j\_;  T\_cell(int i, int j) : i\_(i), j\_(j)  {}  bool operator==(const T\_cell& c)  {  return i\_ == c.i\_ && j\_ == c.j\_;  }  bool operator!=(T\_cell c)  {  return !(\*this == c);  }  };  struct T\_path\_node  {  T\_cell prev\_cell\_;  T\_cell cur\_cell\_;  T\_path\_node  (  T\_cell prev\_cell,  T\_cell cur\_cell  ) : prev\_cell\_(prev\_cell), cur\_cell\_(cur\_cell)  {}  };  std::ostream& operator<<(std::ostream& os, const T\_cell& cell)  {  os << "(" << cell.i\_ << ", " << cell.j\_ << ")";  return os;  }  typedef std::stack<T\_path\_node> T\_path\_stack;  bool fill\_stack(T\_path\_stack& path\_stack)  {  const int m = 9;  const int n = 9;  int labirint[][n] = { {1, 1, 1, 1, 1, 0, 1, 1, 1},  {1, 0, 0, 0, 1, 0, 1, 0, 1},  {1, 0, 1, 0, 0, 0, 1, 0, 1},  {1, 0, 1, 0, 1, 0, 1, 0, 1},  {1, 0, 0, 0, 0, 0, 1, 0, 1},  {1, 0, 1, 1, 1, 1, 1, 0, 1},  {1, 0, 0, 1, 0, 0, 0, 0, 1},  {1, 1, 0, 0, 0, 1, 1, 1, 1},  {1, 1, 1, 1, -1, 1, 1, 1, 1} };  //Для каждого значения в клетке от START\_VAL и меньше заполняем соседние клетки  //со значением OPEN\_VAL значением на единицу меньше, чем в текущей клетке.  //Процесс заканчивается, когда мы заполняем клетку, стоящую с краю, либо, если  //при очередном проходе мы не заполнили ни одной клетки.  for (int cur\_cell\_val = START\_VAL; ; --cur\_cell\_val)  {  //У каждой клетки со значением cur\_cell\_val заполняем все соседние клетки  //со значением OPEN\_VAL значением cur\_cell\_val - 1.  int start\_stack\_size = path\_stack.size();  for (int i = 0; i < m; ++i)  {  for (int j = 0; j < n; ++j)  {  if (labirint[i][j] == cur\_cell\_val)  {  //Обходим соседние клетки.  for (int i\_sosed = std::max(i - 1, 0);  i\_sosed <= std::min(i + 1, m - 1); ++i\_sosed)  {  for (int j\_sosed = std::max(j - 1, 0);  j\_sosed <= std::min(j + 1, n - 1); ++j\_sosed)  {  //Если столбцы или строки совпадают и клетка свободна, то:  if ((i\_sosed == i || j\_sosed == j)  && labirint[i\_sosed][j\_sosed] == OPEN\_VAL)  {  labirint[i\_sosed][j\_sosed] = cur\_cell\_val - 1;  path\_stack  .push(  T\_path\_node(  T\_cell(i, j),  T\_cell(i\_sosed, j\_sosed)  )  );  //Если клетка крайняя, то:  if (i\_sosed == 0 || i\_sosed == m - 1  || j\_sosed == 0 || j\_sosed == n - 1)  {  return true;  }  }  }  }  }  }  }  if (path\_stack.size() == start\_stack\_size) return false;  }  }  void print\_path(T\_path\_stack path\_stack)  {  std::cout << "Маршрут в лабиринте от выхода до стартовой клетки:" << std::endl;  for (;;)  {  T\_cell prev\_cell = path\_stack.top().prev\_cell\_;  std::cout << path\_stack.top().cur\_cell\_ << std::endl;  if (path\_stack.size() == 1)  {  std::cout << path\_stack.top().prev\_cell\_ << std::endl;  return;  }  while (path\_stack.top().cur\_cell\_ != prev\_cell)  {  path\_stack.pop();  }  }  }  int main()  {  std::locale::global(std::locale("rus"));  T\_path\_stack path\_stack;  if (!fill\_stack(path\_stack))  {  std::cout << "Нет выхода!"  << std::endl;  }  else  {  print\_path(path\_stack);  }  system("pause");  return 0;  } |

**3.3** **Тестирование программы**

Результат работы программы приведен на рисунке 3.1.

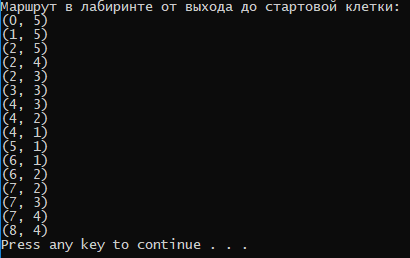


Рисунок 3.1 – Результат работы программы

**Заключение**

* ходе изучения дисциплины «Технологии программирования»

рассмотрены такие темы как:

1. Стандартные потоки ввода – вывода, перегруженные потоки
2. Файловые и строковые потоки. Строки класса string
3. Стандартная библиотека шаблонов, а именно контейнерные классы,

алгоритмы и итераторы.

Использование стандартной библиотеки шаблонов показало свою высокую эффективность в наших проектах, а так же позволило получить новый уровень абстракции для ускорения разработки. Полученные навыки мы будем использовать в дальнейших проектах.

**Список использованных источников**

1. Павловская Т. А., C/C++. Структурное и объектно-ориентированное программирование: Практикум. — СПб.: Питер, 2011. — 352 с.: ил. — (Серия

«Учебное пособие»).