ГУАП

КАФЕДРА № 14

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Канд.тех.наук |  |  |  | К.А. Курицын |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №2 |
|  |
| по дисциплине:ТЕХНОЛОГИЯ ПРОГРАММИРОВАНИЯ |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 1944 |  |  |  | Н.М. Швец |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2021

**1. Постановка задачи**

1. Стандартные потоки

Определить класс с именем STUDENT, содержащий следующие поля:

* фамилия и инициалы;
* номер группы;
* успеваемость (массив из нескольких элементов).

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

Заранее число объектов не известно. Написать программу, выполняющую следующие действия:

- записи о студентах 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..

2. Файловые и строковые потоки

С использованием файловых и строковых потоков написать программу, которая считывает текст из файла и выводит на экран только предложения, содержащие заданное с клавиатуры слово.

1. **Формализация задачи**

Был создан класс Student, содержащий фамилию студента, его инициалы, номер группы и массив из трех элементов структур Education. Структура содержит два поля – предмет и оценку.

В классе реализованы конструкторы, деструктор, методы get и set, с помощью дружественных функций перегружены операторы извлечения и вставки.

Функция main содержит реализацию программы. В ней определен массив объектов Student и реализована работа с ним. Использованы функции сортировки массива и поиска по нему согласно заданию.

Реализована работа с файлами согласно второму заданию.

1. **Исходный код**

Далее приведен исходный код программы из всех файлов.

Таблица 1-Main.cpp

|  |
| --- |
| #include <iostream>  #include <string>  #include <Windows.h>  #include <fstream>  #include "Student.h"  using namespace std;  Student\* students;  int n, index = 0;  void sort();  void exc();  int main()  {  setlocale(LC\_ALL, "rus");  SetConsoleCP(1251);  SetConsoleOutputCP(1251);  // Задание 1  char choice = '0';  do {  cout << "Введите количество студентов: " << endl;  cin >> n;  } while (n <= 0);  students = new Student[n];  while (choice != '5')  {  try  {  cout << "1 - Ввести элемент \n2 - Вывод всех элементов \n3 - Редактировать элемент \n4 - Удалить элемент \n5 - Выход \nПункт меню: ";  cin >> choice;  if (choice == '1')  {  if (index == n)  throw exception("Массив уже заполнен");  cin >> students[index];  index++;  sort();  }  else if (choice == '2')  {  if (index == 0)  throw exception("Массив пуст");  for (int i = 0; i < index; i++)  cout << i + 1 << ": " << students[i];  }  else if (choice == '3')  {  int i;  cout << "Введите номер элемента: ";  cin >> i;  i--;  if (i < 0 || i >= index)  throw exception("Неверный номер");  cout << "Редактируемое поле: \n\t1 - Фамилия \n\t2 - Инициалы \n\t3 - Группа \n\t4 - Предмет и оценку\n\tПункт меню: ";  cin >> choice;  switch (choice)  {  case '1':  {  string buf;  cout << "Новая фамилия: ";  cin >> buf;  students[i].setSurname(buf);  }  break;  case '2':  {  string buf;  cout << "Новые инициалы: ";  cin >> buf;  students[i].setInitials(buf);  }  break;  case '3':  {  int buf;  cout << "Новая группа: ";  cin >> buf;  students[i].setGroup(buf);  }  break;  case '4':  {  string bufStr1, bufStr2, bufStr3;  int bufInt1, bufInt2, bufInt3;  char buf;  cout << "Введите номер предмета от 1 до 3: ";  cin >> buf;  students[i].getEducation(bufStr1, bufInt1, bufStr2, bufInt2, bufStr3, bufInt3);  if (buf == '1')  {  cout << "Введите новый предмет: ";  cin >> bufStr1;  cout << "Введите новую оценку: ";  cin >> bufInt1;  }  else if (buf == '2')  {  cout << "Введите новый предмет: ";  cin >> bufStr2;  cout << "Введите новую оценку: ";  cin >> bufInt2;  }  else if (buf == '3')  {  cout << "Введите новый предмет: ";  cin >> bufStr3;  cout << "Введите новую оценку: ";  cin >> bufInt3;  }  else  throw exception("Неверный номер предмета");  students[i].setEducation(bufStr1, bufInt1, bufStr2, bufInt2, bufStr3, bufInt3);  }  break;  default:  throw exception("Неверный пункт меню");  }  choice = '3';  }  else if (choice == '4')  {  int i;  cout << "Введите номер элемента: ";  cin >> i;  i--;  if (i < 0 || i >= index)  throw exception("Неверный номер");  for (int j = i; j < index; j++)  students[j] = students[j + 1];  index--;  }  else if (choice == '5')  {  break;  }  else  {  throw exception("Неверный пункт меню");  }  }  catch (const exception& exp)  {  cout << "Исключение: " << exp.what() << endl;  }  system("pause");  system("cls");  }  exc();  delete[]students;  // Задание 2  system("pause");  system("cls");  ifstream fin;  string bufStr;  string word;  int cnt = 0;  int j = 0;  string word\_now;  fin.open("file.txt");  try  {  if (!fin.is\_open())  throw exception("Файл отсутствует");  std::cout <<"Введите слово для поиска" << std::endl;  cin>> word;  while (getline(fin,bufStr ))  {  for (int i = 0; i < bufStr.size(); i++)  {  while (bufStr[i + 1] != ' ' && bufStr[i + 1] + 1 != ',' && bufStr[i + 1] + 1 != '.' && bufStr[i + 1] + 1 != ';' && bufStr[i + 1] + 1 != '-' && bufStr[i + 1] != '\0')  {    word\_now.push\_back(bufStr[i]);  if (word == word\_now)  {  cout << bufStr << endl;  cnt++;  break;  }  i++;    }  word\_now.clear();  i++;  }  }  if (cnt == 0)  cout << "Таких строк нет" << endl;  }  catch (exception& exp)  {  cout << "Исключение: " << exp.what() << endl << endl;  }  fin.close();  return 0;  }  void sort() {  string bufStr1, bufStr2, bufStr3;  int bufInt1, bufInt2, bufInt3;  float sr1, sr2;  Student bufStudent;  for (int i = 0; i < index - 1; i++)  {  for (int j = 0; j < index - i - 1; j++)  {  students[j].getEducation(bufStr1, bufInt1, bufStr2, bufInt2, bufStr3, bufInt3);  sr1 = (float)(bufInt1 + bufInt2 + bufInt3) / 3;  students[j + 1].getEducation(bufStr1, bufInt1, bufStr2, bufInt2, bufStr3, bufInt3);  sr2 = (float)(bufInt1 + bufInt2 + bufInt3) / 3;  if (sr1 > sr2)  {  bufStudent = students[j];  students[j] = students[j + 1];  students[j + 1] = bufStudent;  }  }  }  }  void exc() {  string bufStr1, bufStr2, bufStr3;  int bufInt1, bufInt2, bufInt3;  int cnt = 0;  cout << "Студенты, имеющие оценки 4,5 " << endl;  for (int i = 0; i < index; i++)  {  students[i].getEducation(bufStr1, bufInt1, bufStr2, bufInt2, bufStr3, bufInt3);  if (bufInt1 >3 || bufInt2 >3 || bufInt3 >3 )  {  cout << "\t" << cnt + 1 << ": " << students[i].getSurname() << ", " << students[i].getGroup() << endl;  cnt++;  }  }  if (cnt == 0)  cout << "\tТаких студентов нет" << endl;  } |

Таблица 2-Student.h

|  |
| --- |
| #pragma once  #include <string>  #include <iostream>  using namespace std;  struct Education {  string item;  int grade;  };  class Student  {  string surname;  string initials;  int group;  Education education[3];    public:  Student();  Student(Student & obj);  ~Student();  void setSurname(string buf);  string getSurname();  void setInitials(string buf);  string getInitials();  void setGroup(int buf);  int getGroup();  void setEducation(string bufStr1, int bufInt1, string bufStr2, int bufInt2, string bufStr3, int bufInt3);  void getEducation(string & bufStr1, int & bufInt1, string & bufStr2, int & bufInt2, string & bufStr3, int & bufInt3);  friend istream & operator>> (istream & in, Student & object);  friend ostream & operator<< (ostream & out, Student & object);  }; |

Таблица 3-Student.cpp

|  |
| --- |
| #include "Student.h"  Student::Student() {  cout << "Конструктор без параметров" << endl;  }  Student::Student(Student & obj) {  cout << "Конструктор копирования" << endl;  string bufStr1, bufStr2, bufStr3;  int bufInt1, bufInt2, bufInt3;  setSurname(obj.getSurname());  setInitials(obj.getInitials());  setGroup(obj.getGroup());  obj.getEducation(bufStr1, bufInt1, bufStr2, bufInt2, bufStr3, bufInt3);  setEducation(bufStr1, bufInt1, bufStr2, bufInt2, bufStr3, bufInt3);  }  Student::~Student() {  cout << "Деструктор" << endl;  }  void Student::setSurname(string buf) {  if (buf == "")  throw exception("Неверная фамилия");  surname = buf;  }  string Student::getSurname() { return surname; }  void Student::setInitials(string buf) {  if (buf == "" || buf.size() != 4 || buf.find('.') == -1)  throw exception("Неверные инициалы");  initials = buf;  }  string Student::getInitials() { return initials; }  void Student::setGroup(int buf) {  if (buf < 1000 || buf > 10000)  throw exception("Неверный номер группы");  group = buf;  }  int Student::getGroup() { return group; }  void Student::setEducation(string bufStr1, int bufInt1, string bufStr2, int bufInt2, string bufStr3, int bufInt3) {  if (bufStr1 == "" || bufStr2 == "" || bufStr3 == "")  throw exception("Неверный предмет");  if (bufInt1 < 2 || bufInt1 > 5 || bufInt2 < 2 || bufInt2 > 5 || bufInt3 < 2 || bufInt3 > 5)  throw exception("Неверная оценка");  education[0].item = bufStr1;  education[0].grade = bufInt1;  education[1].item = bufStr2;  education[1].grade = bufInt2;  education[2].item = bufStr3;  education[2].grade = bufInt3;  }  void Student::getEducation(string & bufStr1, int & bufInt1, string & bufStr2, int & bufInt2, string & bufStr3, int & bufInt3) {  bufStr1 = education[0].item;  bufInt1 = education[0].grade;  bufStr2 = education[1].item;  bufInt2 = education[1].grade;  bufStr3 = education[2].item;  bufInt3 = education[2].grade;  }  istream & operator>> (istream & in, Student & object) {  string bufStr1, bufStr2, bufStr3;  int bufInt1, bufInt2, bufInt3;  cout << "Введите фамилию: ";  in >> bufStr1;  object.setSurname(bufStr1);  cout << "Введите инициалы (И.О.): ";  in >> bufStr1;  object.setInitials(bufStr1);  cout << "Введите номер группы: ";  in >> bufInt1;  object.setGroup(bufInt1);  cout << "Введите\n\tпредмет1: ";  in >> bufStr1;  cout << "\tоценка1: ";  in >> bufInt1;  cout << "\tпредмет2: ";  in >> bufStr2;  cout << "\tоценка2: ";  in >> bufInt2;  cout << "\tпредмет3: ";  in >> bufStr3;  cout << "\tоценка3: ";  in >> bufInt3;  object.setEducation(bufStr1, bufInt1, bufStr2, bufInt2, bufStr3, bufInt3);  cout << "Студент внесен в список" << endl;  return in;  }  ostream & operator<< (ostream & out, Student & object) {  string bufStr1, bufStr2, bufStr3;  int bufInt1, bufInt2, bufInt3;  out << "Студент: " << object.getSurname() << " " << object.getInitials() << endl;  out << "Группа: " << object.getGroup() << endl;  object.getEducation(bufStr1, bufInt1, bufStr2, bufInt2, bufStr3, bufInt3);  out << "Предметы: " << "\n\t1 - " << bufStr1 << "\t" << bufInt1 <<  "\n\t2 - " << bufStr2 << "\t" << bufInt2 <<  "\n\t3 - " << bufStr3 << "\t" << bufInt3 << endl;  return out;  } |

1. **Результаты работы программы**

Результат работы программы приведен на рисунках 1-11.

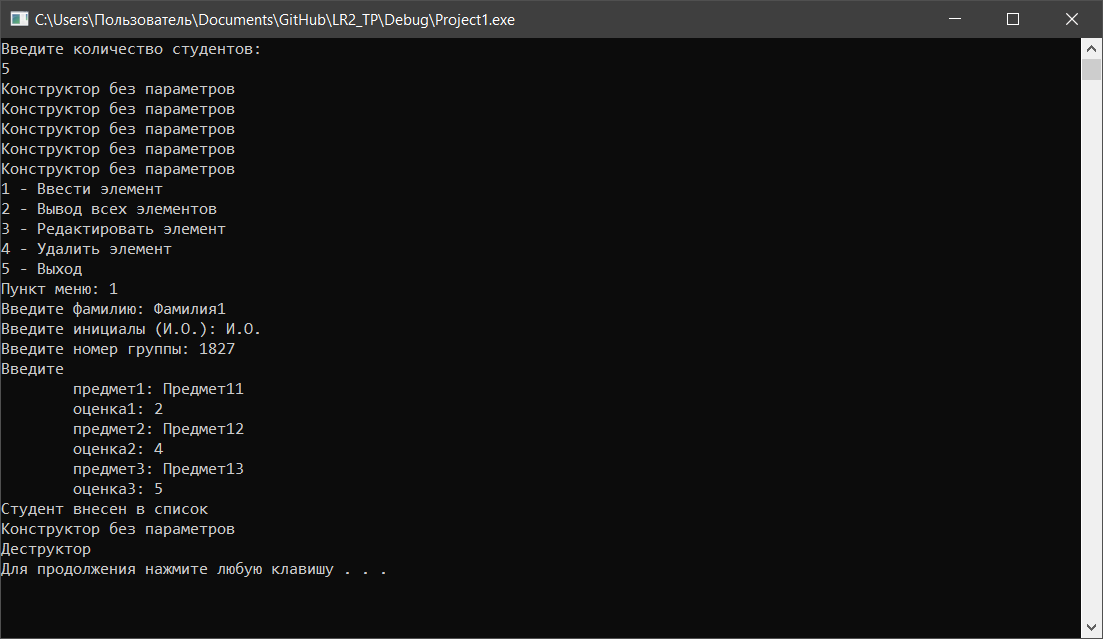


Рисунок 2 – Создание массива, добавление элемента

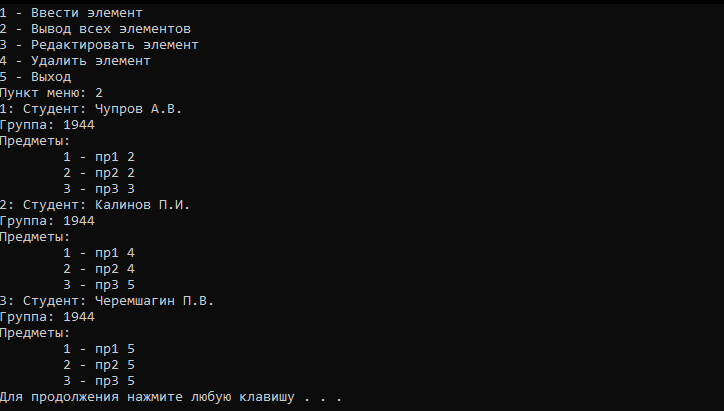


Рисунок 3 – Вывод элементов с сортировкой

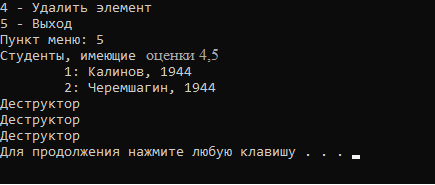


Рисунок 4 – Выполнение задания

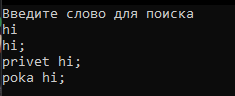


Рисунок 7 – Работа с файлом

1. **Выводы**

Таким образом, была написана программы для выполнения задания, описанного в пункте 1. При проведении тестирования результаты получились совпадающие с предполагаемыми, следовательно, программа написана верно.