**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационных систем**

отчет

**по практической работе №1**

**по дисциплине «Программирование»**

Тема: Структуры.

|  |  |  |
| --- | --- | --- |
| Студент гр. 2372 |  | Братко В.В. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2023

**Цель работы.**

Изучение структур, работа со структурами.

**Основные теоретические положения.**

Структуры представляют собой группы связанных между собой, как правило, разнотипных переменных, объединенных в единый объект, в отличие от массива, все элементы которого однотипны. В языке C++ структура является видом класса и обладает всеми его свойствами.  
Примечание  
Класс - в ООП представляет собой шаблон для создания объектов, обеспечивающий начальные значения состояний: инициализация полей-переменных и реализация поведения функций или методов. ООП в данном курсе не рассматривается, но иногда будут упоминаться понятия, связанные с ним.

Для определения структуры применяется ключевое слово **struct**, а сам формат определения выглядит следующим образом:

struct [имя\_типа] {

тип\_1 элемент\_1;

тип \_2 элемент\_2;

…

тип\_k элемент\_k;

} [ список\_описателей ];

Каждая входящая в структуру переменная называется членом (полем, элементом) структуры и описывается типом данных и именем. Поля структуры могут быть любого типа данных. Их количество не лимитировано.

Вся эта конструкция является инструкцией языка программирования, поэтому после нее всегда должен ставиться символ ‘;’.

struct Student

{

int age;

char name[10];

int group;

};

Любая структурная переменная занимает в памяти определенное положение, характеризующееся конкретным адресом. Для работы с адресами структурных переменных (как и для простых переменных) можно использовать указатели. Указатели на структурные переменные определяются точно так же, как и для обычных переменных. Разыменование указателя (обращение к данным по адресу, хранящемуся в указателе) осуществляется также обычным образом.

Через указатели можно работать с отдельными полями структур. Для доступа к полю структуры через указатель используется оператор ‘->’ («стрелка»), а не «точка».

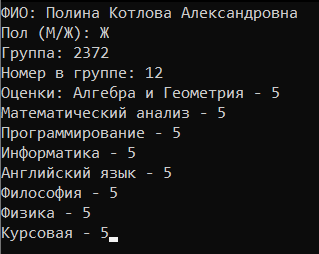
**Постановка задачи.**

1. Создание новой записи о студенте.
2. Внесение изменений в уже имеющуюся запись.
3. Вывод всех данных о студентах.
4. Вывод информации обо всех студентах группы *N*. *N* – инициализируется пользователем.
5. Вывод топа самых успешных студентов с наивысшим по рейтингу средним баллом за прошедшую сессию.
6. Вывод количества студентов мужского и женского пола.
7. Вывод данных о студентах, которые не получают стипендию; учатся только на «хорошо» и «отлично»; учатся только на «отлично»;
8. Вывод данных о студентах, имеющих номер в списке – *k*.

**Выполнение работы.**

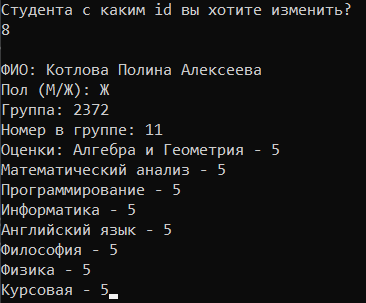
1.Создание новой записи о студенте:

Пользователь поочередно вводит данные о новом студенте

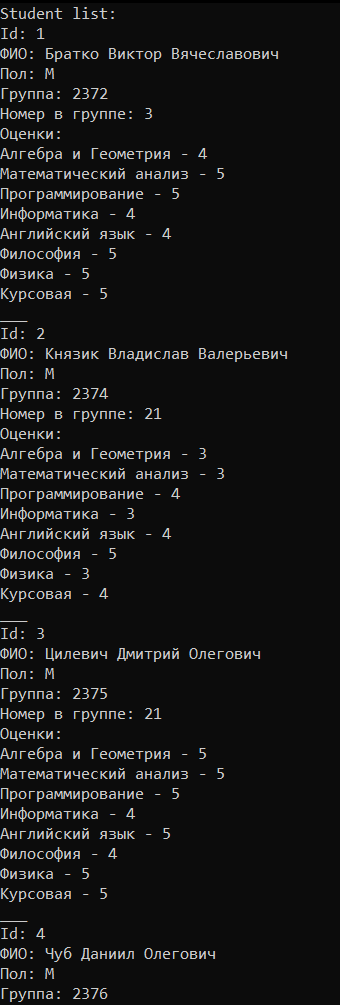


2.Внесение изменений в уже имеющуюся запись:

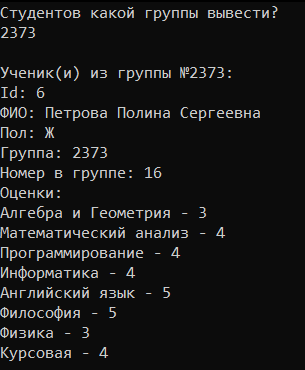
Пользователь может по id изменить информацию о конкретном пользователе



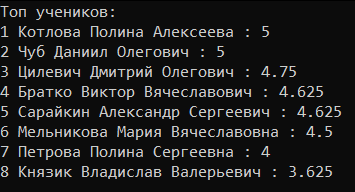
3. Вывод всех данных о студентах:



4. Вывод информации обо всех студентах группы *N*. *N* – инициализируется пользователем.



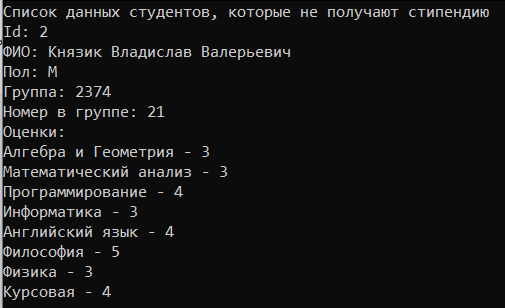
5. Вывод топа самых успешных студентов с наивысшим по рейтингу средним баллом за прошедшую сессию.

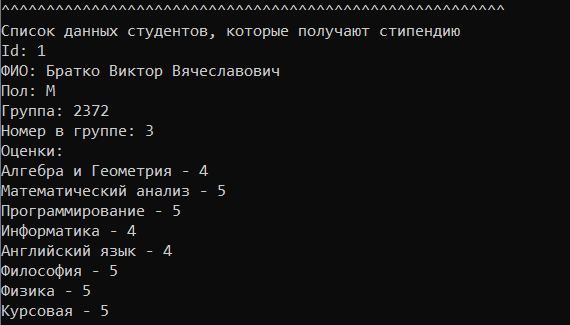


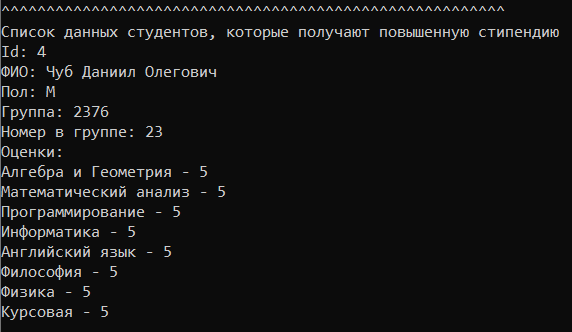
6. Вывод количества студентов мужского и женского пола.



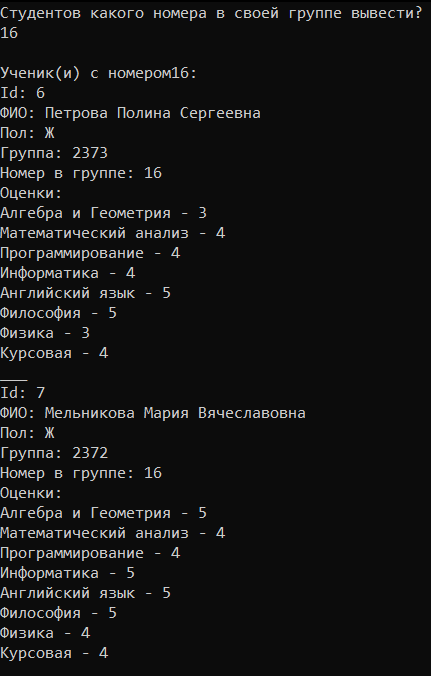
7. Вывод данных о студентах, которые не получают стипендию; учатся только на «хорошо» и «отлично»; учатся только на «отлично»;



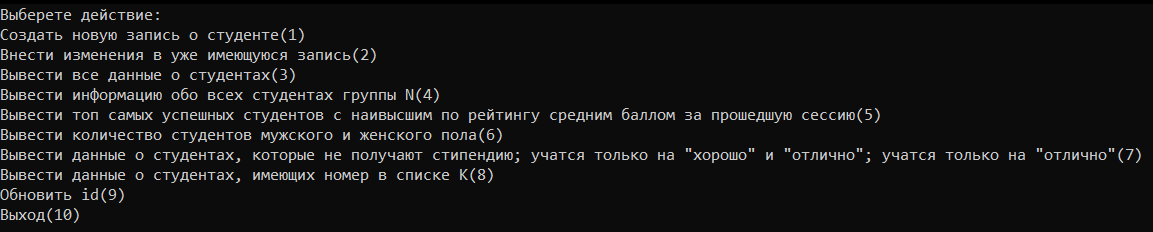




8. Вывод данных о студентах, имеющих номер в списке – *k*.



9.Меню.



10. Проверка на ошибки



**Полный код программы:**

#include <iostream>  
#include <vector>  
#include <fstream>  
#include <locale>  
#include <windows.h>  
#include <map>  
#include <algorithm>  
const std::string fileName = "C:\\FirstCursProgramm\\C++\\stepik\_lab\_2.1\\StudentList.txt";  
std::vector<std::string> listOfSubject = **{**"Алгебра и Геометрия - ", "Математический анализ - ", "Программирование - ", "Информатика - ", "Английский язык - ", "Философия - ", "Физика - ", "Курсовая - "**}**;  
using namespace std;  
  
struct Student{  
 int id;  
 string fullName;  
 string sex;  
 int group;  
 int groupId;  
 int marks[8];  
};  
  
int checkInput(){  
 int input;  
 try {  
 cin >> input;  
 if (cin.fail()) {  
 throw 1;  
 }  
 } catch (int exeption) {  
 cout << "ERROR!!!";  
 exit(0);  
 }  
 cin.sync();  
 cout << "\n";  
 return input;  
}  
  
void refreshStudentListId(vector<Student> &listOfStudent){  
 for(int i = 0; i < listOfStudent.size(); i++){  
 listOfStudent[i].id = i+1;  
 }  
}  
  
void refreshStudentList(vector<Student> &listOfStudent){  
 refreshStudentListId(listOfStudent);  
 ofstream file(fileName, ios::trunc);  
 for (int i = 0; i < listOfStudent.size(); i++){  
 string stringMarks;  
 for(int j = 0; j < 8; j++){  
 stringMarks += to\_string(listOfStudent[i].marks[j]);  
 }  
 string newLine = to\_string(listOfStudent[i].id) + "|" + listOfStudent[i].fullName + "|" + listOfStudent[i].sex + "|" + to\_string(listOfStudent[i].group) + "|" +  
 to\_string(listOfStudent[i].groupId) + "|" + stringMarks;  
 file << newLine << endl;  
 }  
 file.close();  
}  
  
void readDataBase(vector<Student> &listOfStudent){  
 ifstream inputFile(fileName);  
 if (inputFile.is\_open()) {  
 string line;  
 while (getline(inputFile, line)) {  
 Student student;  
 string stringId, stringName, stringGroup, stringGroupId;  
 int i = 0;  
 while(line[i] != '|'){  
 stringId += line[i];  
 i++;  
 }  
 i++;  
 student.id = stoi(stringId);  
 while(line[i] != '|'){  
 stringName += line[i];  
 i++;  
 }  
 i++;  
 student.fullName = stringName;  
 student.sex = line[i];  
 i+=2;  
 while(line[i] != '|'){  
 stringGroup += line[i];  
 i++;  
 }  
 i++;  
 try {  
 student.group = stoi(stringGroup);  
 }  
 catch (const std::invalid\_argument& e) {  
 cout << "ERROR" << "\n";  
 }  
 catch (const std::out\_of\_range& e) {  
 cout << "ERROR" << "\n";  
 }  
 while(line[i] != '|'){  
 stringGroupId += line[i];  
 i++;  
 }  
 i++;  
 try {  
 student.groupId = stoi(stringGroupId);  
 }  
 catch (const std::invalid\_argument& e) {  
 cout << "ERROR" << "\n";  
 }  
 catch (const std::out\_of\_range& e) {  
 cout << "ERROR" << "\n";  
 }  
 for(int j = 0; j < 8 && i<=line.length(); j++,i++){  
 student.marks[j] = stoi(std::string(1, line[i]));  
 }  
 listOfStudent.push\_back(student);  
 }  
 cin.clear();  
 inputFile.close();  
 }  
 else {  
 cout << "Невозможно открыть файл!!!" << "\n";  
 }  
}  
  
void printStudent(vector<Student> &listOfStudent, int i){  
 cout << "Id: " << listOfStudent[i].id<< "\n";  
 cout << "ФИО: " << listOfStudent[i].fullName<< "\n";  
 cout << "Пол: " << listOfStudent[i].sex<< "\n";  
 cout << "Группа: " << listOfStudent[i].group<< "\n";  
 cout << "Номер в группе: " << listOfStudent[i].groupId<< "\n";  
 cout << "Оценки:" << "\n";  
 cout << listOfSubject[0] << listOfStudent[i].marks[0]<< "\n";  
 cout << listOfSubject[1] << listOfStudent[i].marks[1]<< "\n";  
 cout << listOfSubject[2] << listOfStudent[i].marks[2]<< "\n";  
 cout << listOfSubject[3] << listOfStudent[i].marks[3]<< "\n";  
 cout << listOfSubject[4] << listOfStudent[i].marks[4]<< "\n";  
 cout << listOfSubject[5] << listOfStudent[i].marks[5]<< "\n";  
 cout << listOfSubject[6] << listOfStudent[i].marks[6]<< "\n";  
 cout << listOfSubject[7]<< listOfStudent[i].marks[7]<< "\n";  
 cout << "\_\_\_" << "\n";  
}  
  
void printDataBase(vector<Student> &listOfStudent){  
 cout << "Student list:" << "\n";  
 for (int i = 0; i < listOfStudent.size(); i++){  
 printStudent(listOfStudent, i);  
 }  
}  
  
void addStudent(vector<Student> &listOfStudent){  
 Student student;  
 student.id = listOfStudent[listOfStudent.size() - 1].id + 1;  
 cout << "ФИО: ";  
 getline(cin, student.fullName);  
 cin.clear();  
 cout << "Пол (М/Ж): ";  
 cin >> student.sex;  
 if (student.sex.size() != 1){  
 cout << "Неправильный формат ввода";  
 return;  
 }  
 cin.clear();  
 cout << "Группа: ";  
 cin >> student.group;  
 cout << "Номер в группе: ";  
 cin >> student.groupId;  
 cout << "Оценки: ";  
 string stringMarks;  
 for (int i = 0; i < 8; i++) {  
 cout << listOfSubject[i];  
 cin >> student.marks[i];  
 stringMarks += to\_string(student.marks[i]);  
 if (!(student.marks[i] >= 3 && student.marks[i] <= 5)){  
 cout << "Неправильный формат оценок";  
 return;  
 }  
 }  
 cin.clear();  
 ofstream file(fileName, std::ios::app);  
 if (file.is\_open()) {  
 string newLine = to\_string(student.id) + "|" + student.fullName + "|" + student.sex + "|" + to\_string(student.group) + "|" +  
 to\_string(student.groupId) + "|" + stringMarks;  
 file << newLine << endl;  
 file.close();  
 } else {  
 cout << "ERROR";  
 }  
 listOfStudent.push\_back(student);  
}  
  
void changeStudent(vector<Student> &listOfStudent){  
 cout << "Студента с каким id вы хотите изменить?\n";  
 int input = checkInput();  
 if (!(input>=1 && input <= listOfStudent.size())){  
 cout << "Ученика с таким id не существует!";  
 return;  
 }  
 Student student;  
 student.id = listOfStudent[input].id;  
 cout << "ФИО: ";  
 getline(cin, student.fullName);  
 cin.clear();  
 cout << "Пол (М/Ж): ";  
 cin >> student.sex;  
 if (student.sex.size() != 1){  
 cout << "Неправильный формат ввода";  
 return;  
 }  
 cin.clear();  
 cout << "Группа: ";  
 cin >> student.group;  
 cout << "Номер в группе: ";  
 cin >> student.groupId;  
 cout << "Оценки: ";  
 string stringMarks;  
 for (int i = 0; i < 8; i++) {  
 cout << listOfSubject[i];  
 cin >> student.marks[i];  
 stringMarks += to\_string(student.marks[i]);  
 if (!(student.marks[i] >= 3 && student.marks[i] <= 5)){  
 cout << "Неправильный формат оценок";  
 return;  
 }  
 }  
 cin.clear();  
 listOfStudent[input-1] = student;  
 refreshStudentList(listOfStudent);  
}  
  
void printStudentFromGroup(vector<Student> &listOfStudent, int groupNumber){  
 cout << "Ученик(и) из группы №" << groupNumber << ":\n";  
 for (int i = 0; i < listOfStudent.size(); i++){  
 if(listOfStudent[i].group == groupNumber){  
 printStudent(listOfStudent, i);  
 }  
 }  
}  
  
void printStudentFromGroupId(vector<Student> &listOfStudent, int groupIdNumber){  
 cout << "Ученик(и) с номером" << groupIdNumber << ": \n";  
 for (int i = 0; i < listOfStudent.size(); i++){  
 if(listOfStudent[i].groupId == groupIdNumber){  
 printStudent(listOfStudent, i);  
 }  
 }  
}  
  
void countMaleAndFemale(vector<Student> &listOfStudent){  
 int countOfMale = 0;  
 for(int i = 0; i < listOfStudent.size(); i++){  
 if (listOfStudent[i].sex == "М") countOfMale++;  
 }  
 cout << "Количество студентов мужского пола: " << countOfMale << "\n";  
 cout <<"Количество студентов женского пола: " << listOfStudent.size() - countOfMale;  
}  
  
void gradeOfStudent(vector<Student> &listOfStudent){  
 vector<int> badMarks;  
 vector<int> goodMarks;  
 vector<int> excellentMarks;  
 for (int i = 0; i < listOfStudent.size(); i++){  
 int count3 = 0, count4 = 0;  
 for (int j = 0; j < 8; j++){  
 if(listOfStudent[i].marks[j] == 3) count3++;  
 if(listOfStudent[i].marks[j] == 4) count4++;  
 }  
 if(count3 == 0){  
 if (count4 == 0){  
 excellentMarks.push\_back(i);  
 }else goodMarks.push\_back(i);  
 }else badMarks.push\_back(i);  
 }  
 cout << "Список данных студентов, которые не получают стипендию\n";  
 for(int i = 0; i < badMarks.size(); i++){  
 printStudent(listOfStudent, badMarks[i]);  
 }  
 cout << "^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^\n";  
 cout << "Список данных студентов, которые получают стипендию\n";  
 for(int i = 0; i < goodMarks.size(); i++){  
 printStudent(listOfStudent, goodMarks[i]);  
 }  
 cout << "^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^\n";  
 cout << "Список данных студентов, которые получают повышенную стипендию\n";  
 for(int i = 0; i < excellentMarks.size(); i++){  
 printStudent(listOfStudent, excellentMarks[i]);  
 }  
}  
  
bool compareGrades(const pair<string, float>& a, const pair<string, float>& b){  
 return a.second > b.second;  
}  
  
float calculateAverageGrade(vector<Student> &listOfStudent, float i){  
 float sum;  
 for (int j = 0; j <= 7; j++){  
 sum += listOfStudent[i].marks[j];  
 }  
 return sum/8;  
}  
  
void printTop(vector<Student> &listOfStudent){  
 map <string, float> mapStudent;  
 for(int i = 0; i < listOfStudent.size(); i++){  
 float average = calculateAverageGrade(listOfStudent, i);  
 mapStudent[listOfStudent[i].fullName] = average;  
 }  
  
 // создаем вектор пар значений (ключ-значение)  
 vector<pair<string, float>> pairs;  
 for (const auto& pair : mapStudent) {  
 pairs.push\_back(pair);  
 }  
  
 // сортируем вектор по значению  
 sort(pairs.begin(), pairs.end(), compareGrades);  
  
 // выводим отсортированные значения  
 cout << "Топ учеников:\n";  
 for (int i = 1; const auto& pair : pairs) {  
 cout << i << " " << pair.first << " : " << pair.second << endl;  
 i++;  
 }  
}  
  
int main() {  
 setlocale(LC\_ALL, "ru");  
 vector<Student> listOfStudent;  
 readDataBase(listOfStudent);  
 while(true){  
 cout << "Выберете действие:" << "\n";  
 cout << "Создать новую запись о студенте(1)"<< "\n";  
 cout << "Внести изменения в уже имеющуюся запись(2)"<< "\n";  
 cout << "Вывести все данные о студентах(3)"<< "\n";  
 cout << "Вывести информацию обо всех студентах группы N(4)"<< "\n";  
 cout << "Вывести топ самых успешных студентов с наивысшим по рейтингу средним баллом за прошедшую сессию(5)"<< "\n";  
 cout << "Вывести количество студентов мужского и женского пола(6)"<< "\n";  
 cout << "Вывести данные о студентах, которые не получают стипендию; учатся только на \"хорошо\" и \"отлично\"; учатся только на \"отлично\"(7)"<< "\n";  
 cout << "Вывести данные о студентах, имеющих номер в списке K(8)"<< "\n";  
 cout << "Обновить id(9)"<< "\n";  
 cout << "Выход(10)" << "\n";  
 int groupNumber, groupIdNumber;  
 int input = checkInput();  
 system("cls");  
 switch (input) {  
 case 1:  
 addStudent(listOfStudent);  
 break;  
 case 2:  
 changeStudent(listOfStudent);  
 break;  
 case 3:  
 printDataBase(listOfStudent);  
 break;  
 case 4:  
 cout << "Студентов какой группы вывести?\n";  
 groupNumber = checkInput();  
 printStudentFromGroup(listOfStudent, groupNumber);  
 break;  
 case 5:  
 printTop(listOfStudent);  
 break;  
 case 6:  
 countMaleAndFemale(listOfStudent);  
 break;  
 case 7:  
 gradeOfStudent(listOfStudent);  
 break;  
 case 8:  
 cout << "Студентов какого номера в своей группе вывести?\n";  
 groupIdNumber = checkInput();  
 printStudentFromGroupId(listOfStudent, groupIdNumber);  
 break;  
 case 9:  
 refreshStudentList(listOfStudent);  
 break;  
 case 10:  
 exit(1);  
 default:  
 cout << "Неправильный ввод данных!";  
 break;  
 }  
 getchar();  
 system("cls");  
 }  
 return 0;  
}

**Вывод:**

В процессе написания программы для практической работы было изучено:

1. Что такое структуры
2. Как работает со структурами
3. Как работают указатели на структуры
4. Зачем нужны структуры