**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 |  | Гечис В.Р. |
| 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».

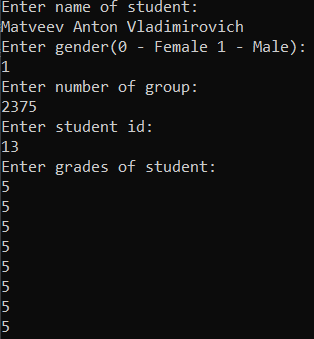
**Постановка задачи.**

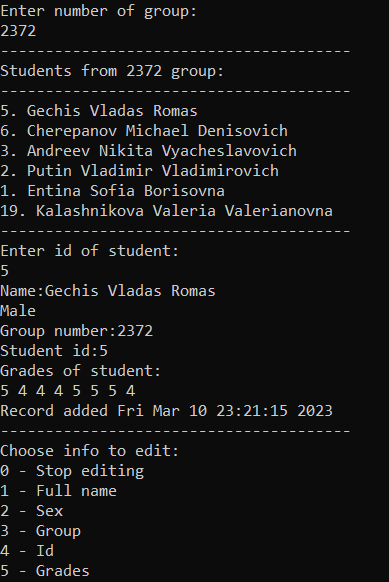
1. Создание новой записи о студенте.
2. Внесение изменений в уже имеющуюся запись.
3. Вывод всех данных о студентах.
4. Вывод информации обо всех студентах группы *N*. *N* – инициализируется пользователем.
5. Вывод топа самых успешных студентов с наивысшим по рейтингу средним баллом за прошедшую сессию.
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.**

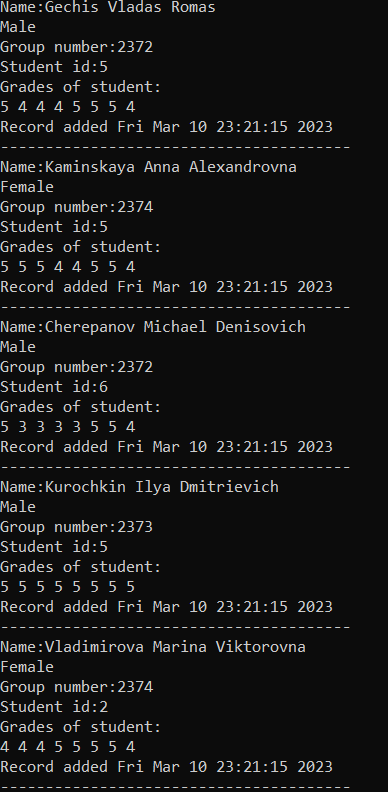
1.Создание новой записи о студенте:

Пользователь поочередно вводит данные о новом студенте

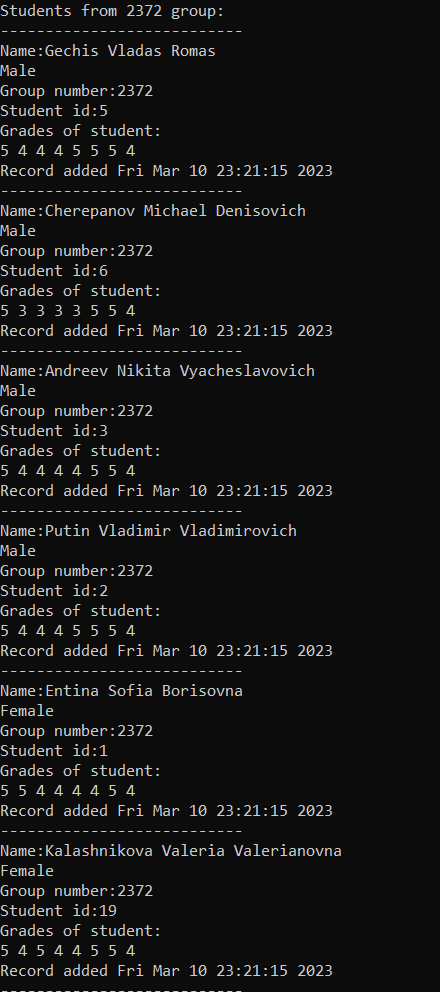


2.Внесение изменений в уже имеющуюся запись:

3. Вывод всех данных о студентах:



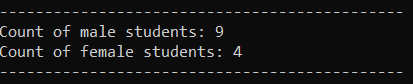
4. Вывод информации обо всех студентах группы *N*. *N* – инициализируется пользователем.



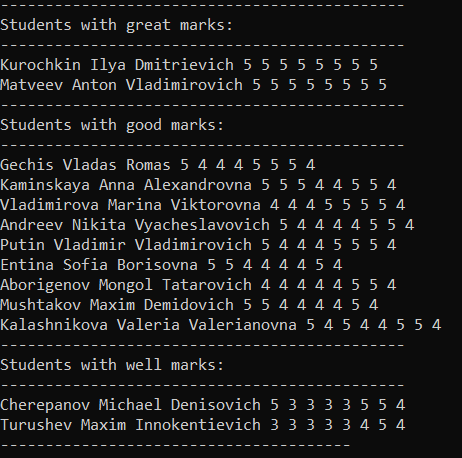
5. Вывод топа самых успешных студентов с наивысшим по рейтингу средним баллом за прошедшую сессию.



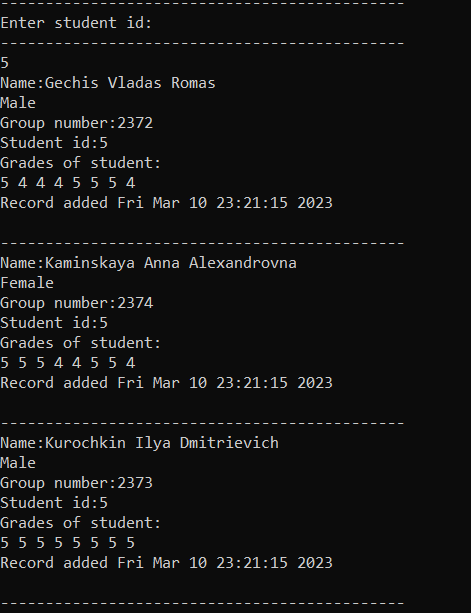
6. Вывод количества студентов мужского и женского пола.



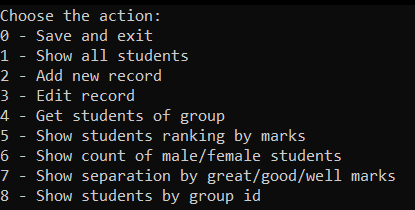
7. Вывод данных о студентах, которые не получают стипендию; учатся только на «хорошо» и «отлично»; учатся только на «отлично»;



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



9.Меню.



**Полный код программы:**

#include <iostream>  
#include <ctime>  
#include <string>  
#include <fstream>  
#include <sstream>  
  
using namespace std;  
  
struct Profile  
{  
 string full\_name;  
 bool sex;  
 int group;  
 int id;  
 int grades[8];  
 string date;  
};  
  
int countStudents(Profile students[]){  
 int count = 0;  
 for(int i = 0; i < 40; i++){  
 if ((students[i].full\_name).length()>1){  
 count +=1;  
 }  
 else{  
 return count;  
 }  
 }  
}  
void addProfile(Profile\* students){  
 Profile student;  
 cout << "Enter name of student:\n";  
 cin.get();  
 getline(cin, student.full\_name);  
// cin.get();  
 cin.clear();  
 cout << "Enter gender(0 - Female 1 - Male):\n";  
 cin >> student.sex;  
// while(student.sex!=0 || student.sex!=1){cout<<"\nerror\n";cin>>student.sex;}  
// while (!(cin >> student.sex)) {  
// cout << "error, expecting a num value\n";  
// cin.clear();  
// fflush(stdin);  
// }  
 cout << "Enter number of group:\n";  
 cin >> student.group;  
// while(!(student.group >= 2000 && student.group <= 2999)){cout<<"\nerror\n";cin>>student.group;}  
 cout << "Enter student id:\n";  
 cin >> student.id;  
// while(!(student.id >= 0 && student.id <= 29)){cout<<"\nerror\n";cin>>student.id;}  
 cout << "Enter grades of student:\n";  
 for (int i=0; i<8;i++){  
 cin >> student.grades[i];  
 if (student.grades[i] == 2){  
 cout << '\n' << "This student will be expelled. The profile will not be saved in the database.";  
 return;  
 }  
 }  
 cin.clear();  
 time\_t now = time(0);  
 char\* dt = ctime(&now);  
 student.date = dt;  
 students[countStudents(students)] = student;  
}  
void saveData(Profile student[]){  
 ofstream database; // Создаем поток записи в файл  
 database.open("C:\\Users\\vvlad\\CLionProjects\\practice2\_1\\student.txt");  
 if (!database.is\_open()) {  
 cout << '\n' << "Saving error!";  
 }  
 else  
 {  
 for(int i = 0; i< countStudents(student);i++) {  
 database << student[i].full\_name << ' ';  
 database << student[i].sex << ' ' << student[i].group << ' ' << student[i].id << ' ';  
 for (int g = 0; g < 8; g++)  
 database << student[i].grades[g];  
 database << endl;  
 }  
 cin.clear();  
 database.close();  
 }  
}  
void getStudentInfo(Profile S){  
 cout << "Name:" <<S.full\_name << endl;  
 if(S.sex){  
 cout <<"Male\n";  
 }  
 else{  
 cout <<"Female\n";  
 }  
 cout << "Group number:" <<S.group << endl;  
 cout << "Student id:" <<S.id << endl;  
 cout << "Grades of student:\n";  
 for (int i=0; i<8;i++){  
 cout << S.grades[i] << " ";  
 }  
 cout << "\nRecord added " <<S.date;  
  
}  
void readDatabase(Profile\* students){  
 int k = 0;  
 ifstream database;  
database.open("C:\\Users\\vvlad\\CLionProjects\\practice2\_1\\student.txt");  
 if (!database.is\_open()) {  
 cout << '\n' << "Error! File isn't open";}  
 else{  
 string data;  
 while (getline(database, data)) {  
 Profile student;  
 string sex, group, grades, id;  
 int i = 0;  
 int countSpace = 0;  
  
 while (countSpace<3){  
 if (data[i]== ' '){  
 student.full\_name += data[i];  
 countSpace += 1;  
 i += 1;  
  
 }  
 else {  
 student.full\_name += data[i];  
 i += 1;  
 }  
 }  
 student.full\_name.erase((student.full\_name).length()-1);  
 while (countSpace == 3){  
 if (data[i]== ' '){  
 countSpace += 1;  
 i += 1;  
  
 }  
 else {  
 sex = data[i];  
 student.sex = stoi(sex);  
 i += 1;  
 }  
 }  
 while (countSpace==4){  
 if (data[i]== ' '){  
 countSpace += 1;  
 i += 1;  
 }  
 else {  
 group += data[i];  
 i += 1;  
  
 }  
 }  
 if (group.length()==4){  
 student.group = stoi(group);  
 }  
 while (countSpace==5) {  
 if (data[i]== ' '){  
 countSpace += 1;  
 i += 1;  
 }  
 else {  
 id += data[i];  
 student.id = stoi(id);  
 i += 1;  
 }  
 }  
 while (countSpace==6 && grades.length() !=8) {  
 grades +=data[i];  
 i+=1;  
 }  
 if (grades.length()==8){  
 for(int g = 0; g < 8; g++){  
 student.grades[g] = stoi(string(1,grades[g]));  
 }  
 }  
 time\_t now = time(0);  
 char\* dt = ctime(&now);  
 student.date = dt;  
  
 students[k]=student;  
 k+=1;  
 }  
  
 }  
 cin.clear();  
 database.close();  
  
 }  
  
  
void getStudents(Profile students[]){  
 for(int i=0;i < countStudents(students);i++){  
 getStudentInfo(students[i]);  
 cout << "---------------------------------------\n";  
 }  
}  
void getStudentsFromGroup(Profile students[]){  
 int userNumberGroup;  
 cout << "Enter number of group:\n";  
 cin >> userNumberGroup;  
 cout << "Students from " << userNumberGroup << " group:\n";  
 cout << "---------------------------\n";  
 for (int i = 0; i < 40; i++){  
 if (students[i].group == userNumberGroup){  
 getStudentInfo(students[i]);  
 cout << "---------------------------\n";  
 }  
 }  
}  
void editProfile(Profile\* students){  
 int userChoice;  
 int userNumberGroup;  
 int userNumberId;  
 cout << "Enter number of group:\n";  
 cin >> userNumberGroup;  
 cout << "---------------------------------------\n";  
 cout << "Students from " << userNumberGroup << " group:\n";  
 cout << "---------------------------------------\n";  
 for (int i = 0; i < 40; i++){  
 if (students[i].group == userNumberGroup){  
 cout << students[i].id << ". " << students[i].full\_name << "\n";  
 }  
 }  
 cout << "---------------------------------------\n";  
 cout << "Enter id of student:\n";  
 cin >> userNumberId;  
 int studentGlobalId;  
 for (int i = 0; i < 40; i++){  
 if ((students[i].group == userNumberGroup) && (students[i].id==userNumberId)){  
 studentGlobalId = i;  
 break;  
 }  
 }  
 getStudentInfo(students[studentGlobalId]);  
 while(true){  
 cout << "---------------------------------------\n";  
 cout << "Choose info to edit:\n0 - Stop editing\n1 - Full name\n2 - Sex\n3 - Group\n4 - Id\n5 - Grades\n";  
 cin >> userChoice;  
 switch(userChoice){  
 case 0:  
 cout << "---------------------------------------\n";  
 getStudentInfo(students[studentGlobalId]);  
 cout << "---------------------------------------\n";  
 return;  
 case 1:  
 cout << "Enter new data:\n";  
 cin.get();  
 getline(cin, students[studentGlobalId].full\_name);  
 cin.clear();  
 break;  
 case 2:  
 cout << "Enter new data:\n";  
 cin >> students[studentGlobalId].sex;  
 break;  
 case 3:  
 cout << "Enter new data:\n";  
 cin >> students[studentGlobalId].group;  
 break;  
 case 4:  
 cout << "Enter new data:\n";  
 cin >> students[studentGlobalId].id;  
 break;  
 case 5:  
 cout << "Enter new data:\n";  
 for (int i=0; i<8;i++){  
 cin >> students[studentGlobalId].grades[i];  
 }  
 break;  
 }  
 }  
}  
float averageGrades(Profile student){  
 float summMarks = 0;  
 for (int j = 0; j < 8; j++){  
 summMarks += student.grades[j];  
 }  
 return summMarks/8;  
}  
  
void sortAvg(Profile students[]){  
 Profile studentsForSort[40];  
 for(int i = 0; i< countStudents(students);i++){  
 studentsForSort[i] = students[i];  
 }  
// memcpy(studentsForSort,students, sizeof(studentsForSort));  
// memcpy(studentsForSort,students, countStudents(students)\*sizeof(Profile));  
 for (int i = 1; i <= countStudents(studentsForSort)-1; i++){  
 for (int j = i; j > 0;j--){  
 if (averageGrades(studentsForSort[j])< averageGrades(studentsForSort[j-1])){  
 swap(studentsForSort[j-1], studentsForSort[j]);  
 } else {  
 continue;  
 }  
 }  
 }  
 cout << "\n---------------------------------------------";  
 cout << "\nBest students by average value of marks:";  
 cout << "\n---------------------------------------------\n";  
 for (int i = countStudents(studentsForSort) - 1; i >= 0; i--){  
 cout << (countStudents(studentsForSort)-i) << "." << " "<< studentsForSort[i].full\_name <<" "<< averageGrades(studentsForSort[i]);  
 cout << "\n---------------------------------------------\n";  
 }  
}  
void getStudentId(Profile students[]){  
 int userNumber;  
 cout << "\n---------------------------------------------";  
 cout << "\nEnter student id:";  
 cout << "\n---------------------------------------------\n";  
 cin >> userNumber;  
 for(int i = 0; i< countStudents(students);i++){  
 if (students[i].id==userNumber){  
 getStudentInfo(students[i]);  
 cout << "\n---------------------------------------------\n";  
 }  
 }  
}  
void countSex(Profile students[]){  
 int countMale = 0;  
 int countFemale = 0;  
 for (int i = 0; i < countStudents(students); i++){  
 if (students[i].sex) countMale+=1;  
 else countFemale +=1;  
 }  
 cout << "\n---------------------------------------------";  
 cout << "\nCount of male students: " <<countMale;  
 cout << "\nCount of female students: " <<countFemale;  
 cout << "\n---------------------------------------------\n";  
  
  
  
}  
void rateStudents(Profile students[]){  
 Profile studentsGreat[countStudents(students)];  
 Profile studentsGood[countStudents(students)];  
 Profile studentsWell[countStudents(students)];  
 int k = 0,f = 0,l = 0;  
 for (int i = 0; i< countStudents(students);i++){  
 bool flagGood = false;  
 bool flagWell = false;  
 for(int j = 0;j<8;j++){  
 if (students[i].grades[j]==3) {  
 studentsWell[k] = students[i];  
 k+=1;  
 flagWell = true;  
 break;  
 }  
 if (students[i].grades[j]==4) flagGood = true;  
 }  
 if(flagGood && !flagWell){studentsGood[f] = students[i];f++;}  
 if(!flagGood && !flagWell){studentsGreat[l] = students[i]; l++;}  
 }  
 cout << "---------------------------------------------\n";  
 cout << "Students with great marks:\n";  
 cout << "---------------------------------------------\n";  
 for (int i = 0;i<l;i++) {  
 cout << studentsGreat[i].full\_name << " ";  
  
 for (int j = 0; j < 8; j++) {  
 cout << studentsGreat[i].grades[j] << " ";  
 }  
 cout << "\n";  
  
 }  
 cout << "---------------------------------------------\n";  
 cout << "Students with good marks:\n";  
 cout << "---------------------------------------------\n";  
  
 for (int i = 0;i<f;i++) {  
 cout << studentsGood[i].full\_name << " ";  
  
 for (int j = 0; j < 8; j++) {  
 cout << studentsGood[i].grades[j] << " ";  
 }  
 cout << "\n";  
 }  
 cout << "---------------------------------------------\n";  
 cout << "Students with well marks:\n";  
 cout << "---------------------------------------------\n";  
 for (int i = 0;i<k;i++) {  
 cout << studentsWell[i].full\_name << " ";  
  
 for (int j = 0; j < 8; j++) {  
 cout << studentsWell[i].grades[j] << " ";  
 }  
 cout << "\n";  
 }  
 cout << "---------------------------------------\n";  
  
}  
int main() {  
 Profile students[40];  
 readDatabase(students);  
 int userChoice;  
 int loopMain;  
// addProfile(students);  
// cin >> loopMain;  
// while(loopMain){  
// cout <<"Choose the action:\n0 - Save and exit\n1 - Show all students\n2 - Add new record\n3 - Edit record\n4 - Get students of group"  
// "\n5 - Show students ranking by marks\n6 - Show count of male/female students\n7 - Show separation by great/good/well marks\n8 - Show students by group id\n";  
// cin >> userChoice;  
// if(userChoice==0){  
// saveData(students);  
// break;  
// }  
// if(userChoice==1){  
// getStudents(students);  
// }  
// if(userChoice==2){  
// addProfile(students);  
// }  
// }  
 while(true){  
 cout <<"Choose the action:\n0 - Save and exit\n1 - Show all students\n2 - Add new record\n3 - Edit record\n4 - Get students of group"  
 "\n5 - Show students ranking by marks\n6 - Show count of male/female students\n7 - Show separation by great/good/well marks\n8 - Show students by group id\n";  
 cin >> userChoice;  
 if(userChoice==0){  
 saveData(students);  
 break;  
 }  
 switch (userChoice) {  
 case 1:  
 getStudents(students);  
 break;  
 case 2:  
 addProfile(students);  
 break;  
 case 3:  
 editProfile(students);  
 break;  
 case 4:  
 getStudentsFromGroup(students);  
 break;  
 case 5:  
 sortAvg(students);  
 break;  
 case 6:  
 countSex(students);  
 break;  
 case 7:  
 rateStudents(students);  
 break;  
 case 8:  
 getStudentId(students);  
 break;  
 default:  
 cout <<"\nUnknown command!\n";  
 }  
 }  
 system("pause");  
}