**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. Объединения.**

|  |  |  |
| --- | --- | --- |
| Студент гр. 4372 |  | Непряхин И.А. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2025

**Цель работы.**

Изучение и организация структур; получение практических навыков работы со структурами; определение преимуществ и недостатков использования структур.

**Основные теоретические положения.**

**Структура** — это пользовательский тип данных в языке C, который позволяет объединить переменные разных типов в одну логическую единицу. Структуры удобно использовать для представления сложных объектов, состоящих из различных характеристик.

**1. Объявление структур.**

Структуры объявляются следующим образом:  
struct Item {

int value;

char name[256];

};

**2. Инициализация двумерных массивов**

Структуры не могут инициализированы при объявления(кроме разве что конструкторов класса). Можно создать экземпляр структуры. Например:

struct Item item;

item.value = 10;

strcpy(item.name, “Spoon”);

Через точку происходит обращение к полям структур.

**3. Указатели на структуры.**

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».

Например:  
 struct Item \*p = &item;

p->value = 10;

**4. Перебор элементов двумерного массива**

На структуры во многом похожи объединения. Объединения также хранят набор элементов, но в отличие от структуры все элементы объединения имеют нулевое смещение. А это значит, что разные элементы занимают в памяти один и тот же участок.

Для определения объединений применяется ключевое слово union и следующий формальный синтаксис:

union название\_объединения {

определение элементов;

};

**Постановка задачи.**

Необходимо создать массив структур, содержащий информацию о студентах: ФИО, пол, номер группы, номер в списке группы, оценки за прошедшую сессию (всего 3 экзамена и 5 дифференцированных зачетов).

Написать функции, реализующие операции со структурами (ввод данных с клавиатуры):

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 польователем.

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

6.   Вывод количества студентов мужского и женского пола.

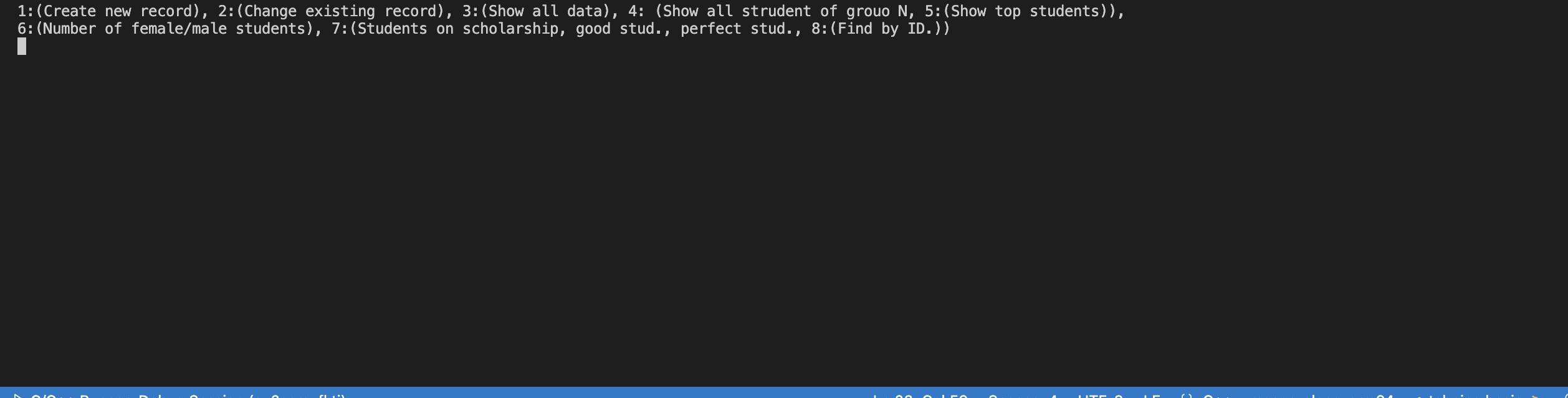
7.   Вывод данных о студентах, которые не получают стипендию; учатся только на «хорошо» и «отлично»; учатся только на «отлично»;

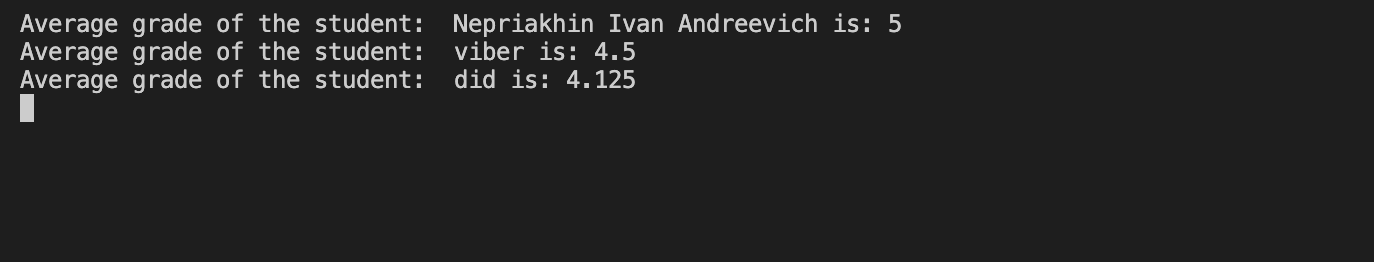
8.   Вывод данных о студентах, имеющих номер в списке – *k*.

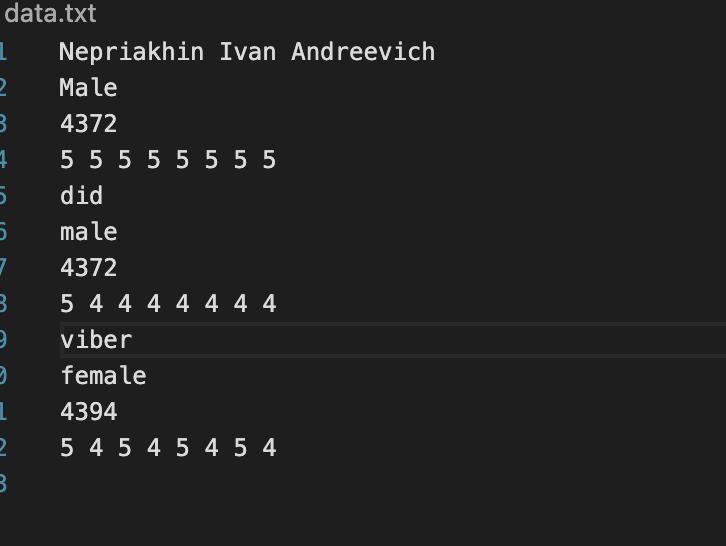
**Выполнение работы.**

Код программы представлен в приложении А.

При запуске выводится главное меню:



Функция подсчета средней оценки:  


Данные хранятся в .txt файле в следующем виде:  


**Выводы.**

В ходе практической работы были изучены основы использования структур в языке программирования C. Были реализованы функции для ввода, вывода и обработки данных, хранящихся в структурах. Стало ясно, что структуры являются удобным инструментом для хранения и обработки разнородных данных, однако при их использовании нужно учитывать ограничения по сравнению с объектно-ориентированным подходом.

Приложение А

рабочий код

#include <iostream>

#include <fstream>

using namespace std;

struct Student {

int id;

string fullname;

string sex;

unsigned int group;

int marksForExamsAndTests[8];

int get(void) {

return 4;

}

};

void studentDataWriteToFile(const Student \*writeStudent, int opt = 0) {

ofstream fileToWriteInto("data.txt", ios::app);

if (fileToWriteInto.is\_open()) {

fileToWriteInto << writeStudent->fullname << '\n';

fileToWriteInto << writeStudent->sex << '\n';

fileToWriteInto << writeStudent->group << '\n';

for (int grade = 0; grade < 8; grade++) {

fileToWriteInto << writeStudent->marksForExamsAndTests[grade] << ' ';

}

fileToWriteInto << '\n';

fileToWriteInto.close();

}

else {

cout << "Could not open the file, check if it exists." << endl;

}

}

void createRecord(void) {

bool isBadStudent = false;

Student newStudent;

cin.clear();

getline(cin, newStudent.fullname);

getline(cin, newStudent.sex);

cin >> newStudent.group;

for (int gradePos = 0; gradePos < 8; gradePos++) {

cin >> newStudent.marksForExamsAndTests[gradePos];

if (newStudent.marksForExamsAndTests[gradePos] <= 2 || newStudent.marksForExamsAndTests[gradePos] > 5) {

isBadStudent = true;

}

}

if (!isBadStudent) {

studentDataWriteToFile(&newStudent);

}

else {

cout << "Unfortuantely, this student is getting expelled" << endl;

if (newStudent.sex == "Male") {

cout << "Welcome to the Russian army!" << endl;

}

}

};

int findNumberOfStudents(void) {

int numberOfStudents = 0;

string studentRecord;

ifstream fileReadFrom("data.txt");

if (fileReadFrom.is\_open()) {

while (getline(fileReadFrom, studentRecord)) {

numberOfStudents++;

}

fileReadFrom.close();

return numberOfStudents / 4;

}

else {

return -1;

}

}

void showStudentN() {

int studentID;

cout << "Enter student number: ";

cin >> studentID;

cin.clear();

cin.ignore();

int studentArraySize = findNumberOfStudents();

vector<Student> studentList(studentArraySize);

ifstream fileToReadFrom("data.txt");

for (int studentNumber = 0; studentNumber < studentArraySize; studentNumber++) {

getline(fileToReadFrom, studentList[studentNumber].fullname);

//cout << studentList[studentNumber].fullname << endl;

getline(fileToReadFrom, studentList[studentNumber].sex);

//cout << studentList[studentNumber].sex << endl;

fileToReadFrom >> studentList[studentNumber].group;

fileToReadFrom.ignore();

//cout << studentList[studentNumber].group << endl;

for (int grade = 0; grade < 8; grade++) {

fileToReadFrom >> studentList[studentNumber].marksForExamsAndTests[grade];

fileToReadFrom.ignore();

//cout << studentList[studentNumber].marksForExamsAndTests[grade] << endl;

}

fileToReadFrom.ignore();

if (studentID == (studentNumber + 1)) {

cout << "Student's data: " << endl;

cout << "Fullname: " << studentList[studentNumber].fullname << endl;

cout << "Sex: " << studentList[studentNumber].sex << endl;

cout << "Group: " << studentList[studentNumber].group << endl;

for (int grade = 0; grade < 8; grade++) {

cout << "Grade " << grade + 1 << ": " << studentList[studentNumber].marksForExamsAndTests[grade] << endl;

}

}

if (studentNumber > studentArraySize) {

cout << "No student with such number" << endl;

}

}

fileToReadFrom.close();

}

int main(void) {

cout << sizeof(Student) << endl;

int subactionForPeformance;

while (true) {

system("clear");

puts("1:(Create new record), 2:(Change existing record), 3:(Show all data), 4: (Show all strudent of grouo N, 5:(Show top students)), ");

puts("6:(Number of female/male students), 7:(Students on scholarship, good stud., perfect stud., 8:(Find by ID.))");

int action;

cin >> action;

cin.clear();

cin.ignore();

switch (action) {

case 1:

system("clear");

cout << "Input data: " << endl;

cin.clear();

createRecord();

getchar();

getchar();

break;

case 2:

system("clear");

changeRecord();

system("pause");

break;

case 3:

system("clear");

cout << "Whole data: " << endl;

showAllStudents();

getchar();

break;

case 4:

system("clear");

showStudentOfGroupN();

getchar();

getchar();

break;

case 5:

system("clear");

showTopStudents();

getchar();

break;

case 6:

system("clear");

countByGender();

getchar();

break;

case 7:

{

int option = 0;

//cout << "What students would you like to see?" << endl;

//cin >> option;

//cin.clear();

rangeStudentsByPerformance(option);

getchar();

}

break;

case 8:

system("clear");

showStudentN();

getchar();

break;

case 9:

system("clear");

return 0;

default:

cout << "Invalid option, please try again." << endl;

}

}

return 0;

}