**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: Типы данных, определяемые пользователем. Структуры

|  |  |  |
| --- | --- | --- |
| Студент гр. 2373 |  | Андреев Н.С. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2023

**Цель работы.**

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

**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. Чаще всего ограничиваются тем, как структуры представлены в языке С:

struct [имя\_типа] {

тип\_1 элемент\_1;

тип \_2 элемент\_2;

…

тип\_k элемент\_k;

} [ список\_описателей ];

Описание структуры начинается ключевым словом struct. 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.

При описании структуры память для размещения данных не выделяется. Работать с описанной структурой можно только после того, как будет определена переменная (переменные) этого типа данных, только при этом компилятор выделит необходимую память.

Для инициализации структуры значения ее элементов перечисляют в фигурных скобках в порядке их описания:

struct complex{

float real, im;

} data [2][2] = {

{{1,1}, {2,2}},

{{3,3}, {4,4}}

};

Для того чтобы записать данные в структурную переменную, необходимо каждому полю структуры присвоить определенное значение. Для этого необходимо использовать оператор ‘’ («точка»):

struct Stack { // Cтек

float arr[100];

short topIndex;

};

…

Stack stack; // Объявляем переменную типа Stack

Stack.arr[0] = 1;

…

При доступе к определенному полю его следует рассматривать как обычную переменную, тип данных которой соответствует типу этого поля. Поля структур могут участвовать в качестве операндов любых выражений, допускающих использование операндов соответствующего типа данных.

Копирование данных из одной структурной переменной в другую осуществляется простой операцией присваивания, независимо от количества полей и размера структуры (это можно делать только в том случае, когда обе переменные одного и того же типа).

В программировании очень часто используются такие конструкции, как массивы структур. Например, сведения о студентах некоторой учебной группы можно хранить в массиве студентов:

t\_Student Gruppa\_N [30];

Был определен 30-элементный массив, каждый элемент которого предназначен для хранения данных одного студента. Получение доступа к данным некоторого студента из группы *N* осуществляется обычной индексацией переменной массива. Поскольку поля структуры могут быть любого типа данных, то они в свою очередь могут быть другой структурой или массивом других структур:

struct Stud

{

char FN[100];

short listNumber;

};

struct Group

{

int groupNumber;

short students;

Stud stud[30];

};

Но в структуре поля нельзя использовать элемент, тип которого совпадает с типом самой структуры, так как рекурсивное использование структур запрещено.

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.

Через указатели можно работать с отдельными полями структур. Для доступа к полю структуры через указатель используется оператор ‘’.

Структуры можно использовать в качестве параметров функций, как и обычные переменные. Для структур поддерживаются все три механизма передачи данных: по значению, через указатели и по ссылке.

Предпочтительно использование передачи структуры по указателю или ссылке:

void WriteStudent ( t\_Student \*S )

{

cout << "Фамилия: " << S -> Fam << endl;

cout << "Имя: " << S -> Name << endl;

cout << "Год рождения: " << S -> Year << endl;

if ( S -> Sex )

cout << "Пол: " << "М\n";

else

cout << "Пол: " << "Ж\n";

cout << "Средний балл: " << S -> Grade << endl;

}

Фактической передачи данных в функцию не осуществляется. Дополнительные затраты памяти для создания локальной переменной небольшие – это адрес памяти (4 байта, независимо от размера самой структуры). Вызов такой функции будет происходить быстрее, а расход памяти будет существенно меньше, чем при передаче данных по значению.

**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). Данные о студентах хранятся в файле.

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 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»;

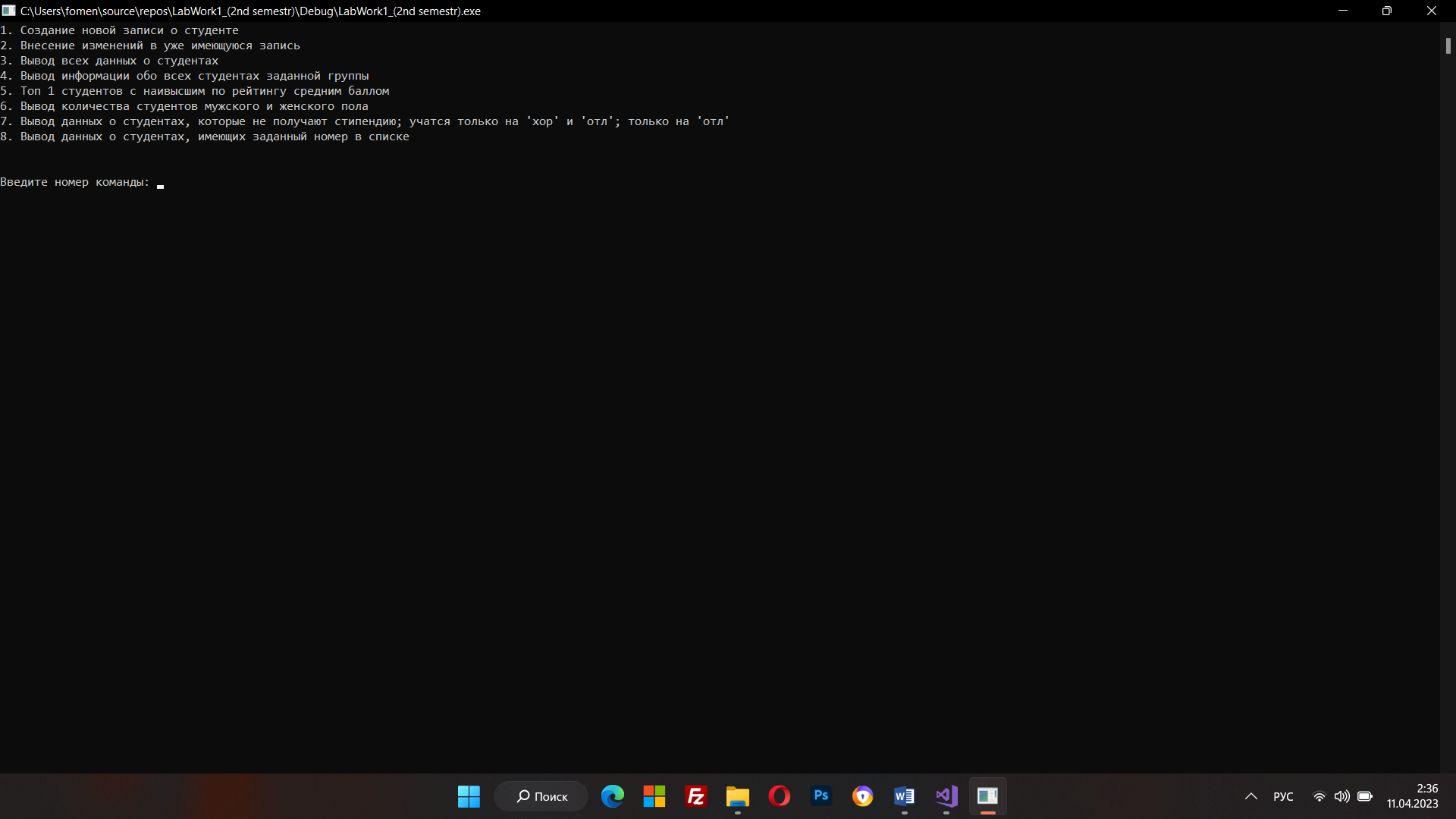
8.   Вывод данных о студентах, имеющих номер в списке – *k*.

**Выполнение работы.**

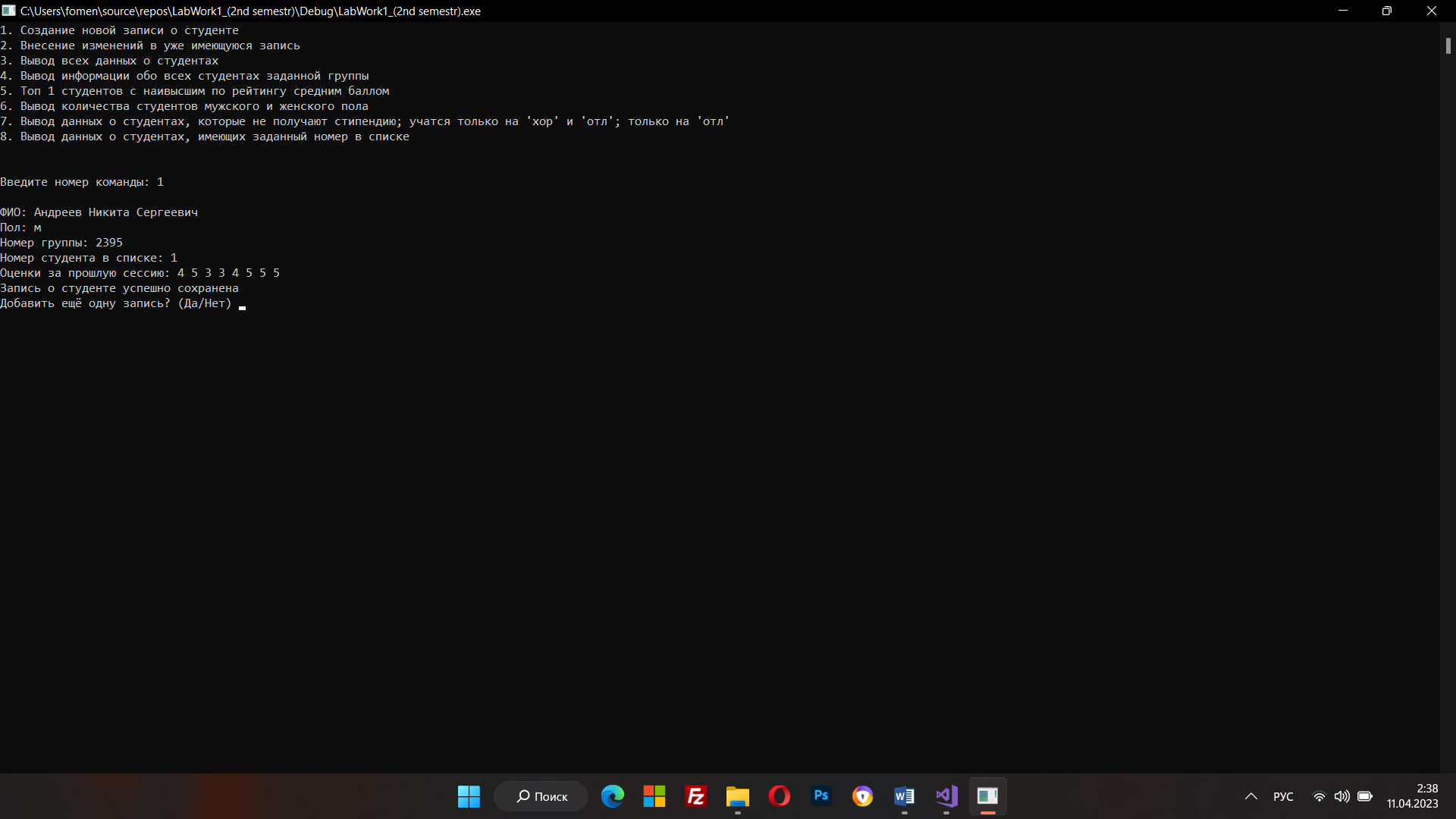
Код программы представлен в приложении А.

Блок описания кода и использованных алгоритмов

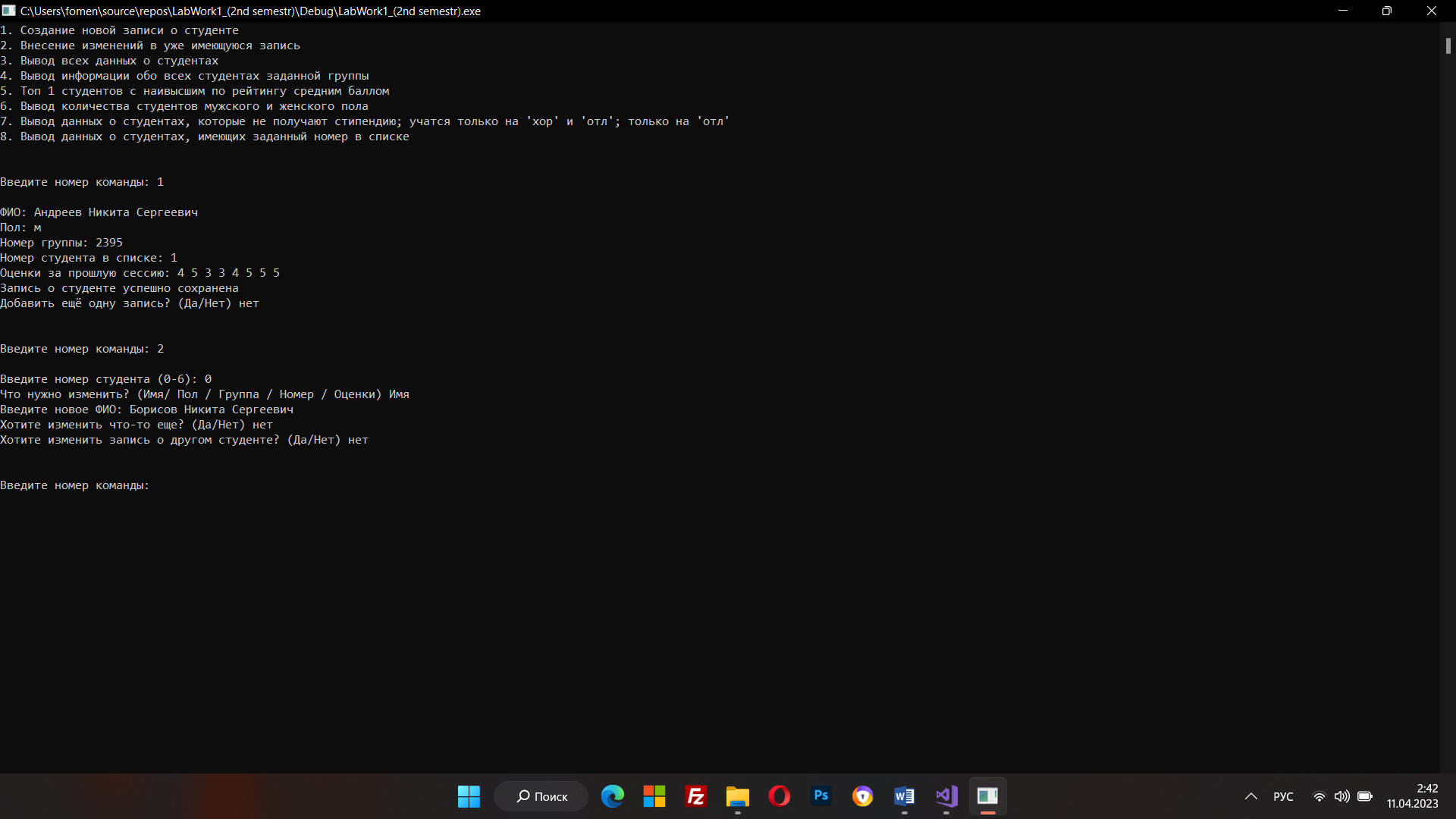
1. При запуске программы появляется окно, в котором выводится меню допустимых операций под номерами. Пользователю предлагается ввести любой номер операции от 1 до 8:



2. Операция 1 – создание новой записи. Пользователю предлагается по очереди внести сведения о студенте. Введённые данные сохраняются в структуру.



3. Операция 2 – внесение изменений в запись. Пользователю предлагается изменить либо 1 пункт записи, либо всю запись. Чтобы определить, какую запись необходимо изменить, вводится номер группы и порядковый номер нужного студента.

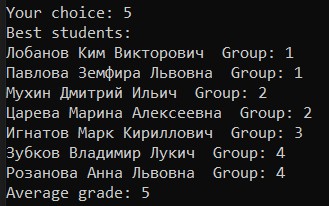


5. Операции 3 – 8 – вывод данных о студентах, классифицированных по определённому признаку.

3 – вывод данных о всех студентах.

4 – вывод данных о студентах выбранной группы. На экран выводится порядковый номер, ФИО и оценки студентов.

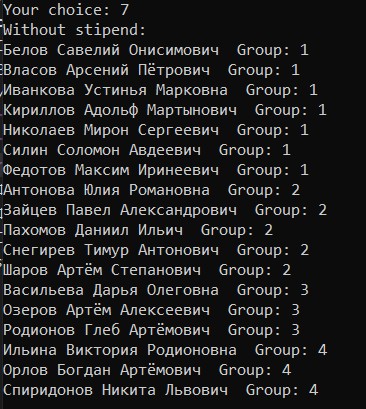
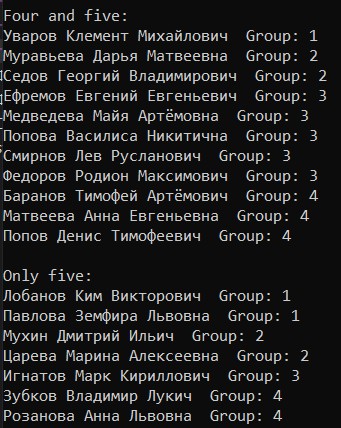
5 – список студентов с наивысшим средним баллом. Выводятся ФИО, номер группы и средний балл лучших студентов.



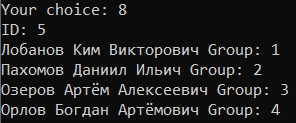
6 – вывод количества студентов мужского и женского пола.



7 – вывод данных о студентах с различной успеваемостью (те, кто не получает стипендию; учится на «хорошо» и «отлично»; учится только на «отлично»).

8 – список студентов с номером k в группе (k вводится пользователем с клавиатуры).



6. Операция 9 – повторный вывод меню на экран, 0 – выход из программы.

**Выводы.**

Структуры позволяют сделать простой и удобной работу с данными различных типов, которые связаны друг с другом определённым образом (например, информация о ФИО, группе и оценках студента). При этом есть возможность анализировать и изменять части структуры по отдельности, а также создавать массивы структур для работы с большим количеством упорядоченных данных.

Приложение А

рабочий код

#include <iostream>

#include <fstream>

#include <string>

#include <Windows.h>

using namespace std;

const int SIZE\_ = 50;

struct student

{

string FullName;

char sex;

int group;

int num;

int grades[8];

};

void CreateNewList(student\*, string);

void ShowFile(string);

void CopyInStruct(student\*, string);

void ChangeList(student\*, string);

void DeleteAll(string);

void WriteGroup(student\*, int, string);

void WriteBestStudents(student\*, string);

void WriteMaleFemale(student\*, string);

void WriteGrades(student\*, string);

void WriteStudentsNum(student\*, string);

int CountStudents(string);

//const int SIZE = 50;

int main()

{

setlocale(0, "");

SetConsoleCP(1251);// установка кодовой страницы win-cp 1251 в поток ввода

SetConsoleOutputCP(1251); // установка кодовой страницы win-cp 1251 в поток вывода

//const int SIZE = 50;

string path = "myFile.txt";

struct student Student[SIZE\_];

struct student S;

int size = CountStudents(path);

cout << "1. Создание новой записи о студенте\n2. Внесение изменений в уже имеющуюся запись\n3. Вывод всех данных о студентах\n"

<< "4. Вывод информации обо всех студентах заданной группы\n5. Топ " << size / 6 << " студентов с наивысшим по рейтингу средним баллом\n"

<< "6. Вывод количества студентов мужского и женского пола\n7. Вывод данных о студентах, которые не получают стипендию; учатся только на 'хор' и 'отл'; только на 'отл'\n"

<< "8. Вывод данных о студентах, имеющих заданный номер в списке\n";

int command = 1;

while (command)

{

cout << "\n\nВведите номер команды: ";

cin >> command;

switch (command)

{

case 0:

break;

case 1:

{

if (size >= SIZE\_)

cout << "Файл полностью заполнен!" << endl;

else

{

cout << endl;

CreateNewList(&S, path);

CopyInStruct(Student, path);

}

break;

}

case 2:

{

cout << endl;

CopyInStruct(Student, path);

ChangeList(Student, path);

break;

}

case 3:

{

cout << endl;

ShowFile(path);

break;

}

case 4:

{

CopyInStruct(Student, path);

int N;

cout << "\nВведите номер группы: ";

cin >> N;

WriteGroup(Student, N, path);

break;

}

case 5:

{

cout << endl;

WriteBestStudents(Student, path);

break;

}

case 6:

{

cout << endl;

WriteMaleFemale(Student, path);

break;

}

case 7:

{

cout << endl;

WriteGrades(Student, path);

break;

}

case 8:

{

cout << endl;

WriteStudentsNum(Student, path);

break;

}

}

}

}

int CountStudents(string path)

{

ifstream fin(path);

if (fin.is\_open())

{

int count = 0;

string str;

while (!fin.eof())

{

getline(fin, str);

count++;

}

fin.close();

int n;

n = count / 5;

return n;

}

else

return 0;

}

void DeleteAll(string path)

{

ofstream fout;

fout.open(path);

fout.close();

}

void ShowFile(string path)

{

ifstream fin;

fin.open(path);

if (!fin.is\_open())

cout << "Не удалось открыть файл" << endl;

else

{

int N;

N = CountStudents(path);

if (N == 0)

cout << "\nФайл пуст" << endl;

else

{

string str;

int count = 0;

while (!fin.eof())

{

getline(fin, str);

count++;

if (count % 5 == 0)

cout << str << "\n\n";

else

cout << str << endl;

}

fin.close();

cout << "Всего в файле в файле " << N << " студентов" << endl;

}

}

}

void CreateNewList(student\* S, string path)

{

bool F;

do

{

cout << "ФИО: ";

string Name1, Name2, Name3, FullName;

cin >> Name1 >> Name2 >> Name3;

FullName = Name1 + " " + Name2 + " " + Name3;

S->FullName = FullName;

cout << "Пол: ";

cin >> S->sex;

cout << "Номер группы: ";

cin >> S->group;

cout << "Номер студента в списке: ";

cin >> S->num;

cout << "Оценки за прошлую сессию: ";

for (int i = 0; i < 8; i++)

cin >> S->grades[i];

cin.clear();

bool flag = true;

for (int i = 0; i < 8; i++)

{

if (S->grades[i] == 2)

flag = false;

}

if (flag)

{

ofstream fout;

fout.open(path, ios::app);

if (!fout.is\_open())

cout << "Ошибка чтения файла " << endl;

else

{

fout << "\n";

fout << S->FullName << "\n";

fout << S->sex << "\n" << S->group << "\n" << S->num << "\n";

for (int i = 0; i < 8; i++)

{

if (i != 7)

fout << S->grades[i] << " ";

else

fout << S->grades[i];

}

cout << "Запись о студенте успешно сохранена" << endl;

}

fout.close();

}

else

cout << "Запись не может быть сохранена, т. к. студент должен быть отчислен" << endl;

string YesOrNot;

cout << "Добавить ещё одну запись? (Да/Нет) ";

cin >> YesOrNot;

if (YesOrNot == "Да")

F = true;

else

F = false;

} while (F);

}

void CopyInStruct(student\* Student, string path)

{

int size = CountStudents(path);

ifstream fin;

fin.open(path);

if (!fin.is\_open())

cout << "\nОшибка!\n";

else

{

for (int i = 0; i < size; i++)

{

string str = " ";

int A, B;

getline(fin, str);

(Student + i)->FullName = str;

fin >> str;

(Student + i)->sex = str[0];

fin >> A;

(Student + i)->group = A;

fin >> B;

(Student + i)->num = B;

for (int j = 0; j < 8; j++)

{

int C = 0;

fin >> C;

(Student + i)->grades[j] = C;

}

getline(fin, str);

}

}

fin.close();

}

void ChangeList(student \*Student, string path)

{

int number;

int size = CountStudents(path);

string parameter;

bool yes, yes1;

do

{

cout << "Введите номер студента (0-" << size - 1 << "): ";

cin >> number;

do

{

cout << "Что нужно изменить? (Имя/ Пол / Группа / Номер / Оценки) ";

cin >> parameter;

if (parameter == "Имя")

{

string NewName1, NewName2, NewName3, NewFullName;

cout << "Введите новое ФИО: ";

cin >> NewName1 >> NewName2 >> NewName3;

NewFullName = NewName1 + " " + NewName2 + " " + NewName3;

(Student + number)->FullName = NewFullName;

}

else if (parameter == "Пол")

{

char NewSex;

cout << "Введите новый пол: ";

cin >> NewSex;

(Student + number)->sex = NewSex;

}

else if (parameter == "Группа")

{

int NewGroup;

cout << "Введите новый номер группы: ";

cin >> NewGroup;

(Student + number)->group = NewGroup;

}

else if (parameter == "Номер")

{

int NewNum;

cout << "Введите новый номер в списке: ";

cin >> NewNum;

(Student + number)->num = NewNum;

}

else if (parameter == "Оценки")

{

int NewGrades[8];

cout << "Введите новые оценки (через пробел): ";

for (int i = 0; i < 8; i++)

{

cin >> NewGrades[i];

(Student + number)->grades[i] = NewGrades[i];

}

}

string yes\_or\_no;

cout << "Хотите изменить что-то еще? (Да/Нет) ";

cin >> yes\_or\_no;

if (yes\_or\_no == "Да")

yes1 = true;

else

yes1 = false;

} while (yes1);

string yes\_or\_no;

cout << "Хотите изменить запись о другом студенте? (Да/Нет) ";

cin >> yes\_or\_no;

if (yes\_or\_no == "Да")

yes = true;

else

yes = false;

} while (yes);

ofstream fout(path);

for (int i = 0; i < size; i++)

{

fout << (Student + i)->FullName << "\n";

fout << (Student + i)->sex << "\n" << (Student + i)->group << "\n" << (Student + i)->num << "\n";

for (int j = 0; j < 8; j++)

{

if ((j == 7) && (i == (size - 1)))

fout << (Student + i)->grades[j];

else if (j == 7 && i != (size - 1))

fout << (Student + i)->grades[j] << "\n";

else

fout << (Student + i)->grades[j] << " ";

}

}

fout.close();

}

void WriteGroup(student\* S, int N, string path)

{

int size = CountStudents(path);

int count = 0;

cout << endl;

for (int i = 0; i < size; i++)

{

if ((S + i)->group == N)

{

count++;

cout << (S + i)->FullName << "\n";

cout << (S + i)->sex << "\n";

cout << (S + i)->group << "\n";

cout << (S + i)->num << "\n";

for (int j = 0; j < 8; j++)

cout << (S + i)->grades[j] << " ";

cout << "\n\n";

}

}

if (count == 0)

cout << "Студентов из этой группы не найдено\n";

}

void WriteBestStudents(student\* S, string path)

{

CopyInStruct(S, path);

int size = CountStudents(path);

float arr\_grades[SIZE\_];

int sum\_grade;

double curr\_grade;

for (int i = 0; i < size; i++)

{

sum\_grade = 0;

for (int j = 0; j < 8; j++)

{

sum\_grade += (S + i)->grades[j];

}

curr\_grade = sum\_grade / 8;

arr\_grades[i] = curr\_grade;

}

int arr\_numb[SIZE\_];

for (int i = 0; i < size; i++)

{

arr\_numb[i] = i;

}

for (int i = 0; i < size; i++)

{

for (int j = 0; j < size - 1; j++)

{

if (arr\_grades[i] > arr\_grades[j + 1])

{

swap(arr\_grades[i], arr\_grades[j + 1]);

swap(arr\_numb[i], arr\_numb[j + 1]);

}

}

}

cout << "Студенты с наилучшей успеваемостью: \n\n";

for (int i = 0; i < size / 6; i++)

{

cout << (S + arr\_numb[i])->FullName << "\n" << (S + arr\_numb[i])->sex << "\n" << (S + arr\_numb[i])->group << "\n" << (S + arr\_numb[i])->num << "\n";

for (int j = 0; j < 8; j++)

{

if ((j == 7) && (i == (size - 1)))

cout << (S + arr\_numb[i])->grades[j];

else if (j == 7 && i != (size - 1))

cout << (S + arr\_numb[i])->grades[j] << "\n\n";

else

cout << (S + arr\_numb[i])->grades[j] << " ";

}

}

}

void WriteMaleFemale(student\* S, string path)

{

CopyInStruct(S, path);

int size = CountStudents(path);

int man = 0, woman = 0;

for (int i = 0; i < size; i++)

{

if ((S + i)->sex == 'м')

man += 1;

else if ((S + i)->sex == 'ж')

woman += 1;

}

cout << "Количество парней в списке: " << man << endl;

cout << "Количество девушек в списке: " << woman << endl;

}

void WriteGrades(student\* S, string path)

{

CopyInStruct(S, path);

int size = CountStudents(path);

string arr\_value[SIZE\_];

for (int i = 0; i < size; i++)

{

bool low = false;

int sum = 0;

for (int j = 0; j < 8; j++)

{

if ((S + i)->grades[j] <= 3)

{

low = true;

break;

}

else

sum += (S + i)->grades[j];

}

if (low)

{

arr\_value[i] = "low";

continue;

}

if (sum == 5 \* 8)

arr\_value[i] = "high";

else

arr\_value[i] = "middle";

}

cout << "Студенты, учащиеся на 5:\n\n";

for (int i = 0; i < size; i++)

{

if (arr\_value[i] == "high")

{

cout << (S + i)->FullName << "\n";

cout << (S + i)->sex << "\n";

cout << (S + i)->group << "\n";

cout << (S + i)->num << "\n";

for (int j = 0; j < 8; j++)

{

if ((j == 7) && (i == (size - 1)))

cout << (S + i)->grades[j];

else if (j == 7 && i != (size - 1))

cout << (S + i)->grades[j] << "\n\n";

else

cout << (S + i)->grades[j] << " ";

}

}

}

cout << "Студенты, учащиеся на 4 и 5:\n\n";

for (int i = 0; i < size; i++)

{

if (arr\_value[i] == "middle")

{

cout << (S + i)->FullName << "\n";

cout << (S + i)->sex << "\n";

cout << (S + i)->group << "\n";

cout << (S + i)->num << "\n";

for (int j = 0; j < 8; j++)

{

if ((j == 7) && (i == (size - 1)))

cout << (S + i)->grades[j];

else if (j == 7 && i != (size - 1))

cout << (S + i)->grades[j] << "\n\n";

else

cout << (S + i)->grades[j] << " ";

}

}

}

cout << "Студенты, не получающие стипендию:\n\n";

for (int i = 0; i < size; i++)

{

if (arr\_value[i] == "low")

{

cout << (S + i)->FullName << "\n";

cout << (S + i)->sex << "\n";

cout << (S + i)->group << "\n";

cout << (S + i)->num << "\n";

for (int j = 0; j < 8; j++)

{

if ((j == 7) && (i == (size - 1)))

cout << (S + i)->grades[j];

else if (j == 7 && i != (size - 1))

cout << (S + i)->grades[j] << "\n\n";

else

cout << (S + i)->grades[j] << " ";

}

}

}

}

void WriteStudentsNum(student\* S, string path)

{

CopyInStruct(S, path);

int size = CountStudents(path);

int gr, num\_in\_gr;

cout << "Введите номер группы: ";

cin >> gr;

cout << "\nВведите номер студента в группе: ";

cin >> num\_in\_gr;

cout << endl;

for (int i = 0; i < size; i++)

{

if ((S + i)->group == gr && (S + i)->num == num\_in\_gr)

{

cout << (S + i)->FullName << "\n";

cout << (S + i)->sex << "\n";

cout << (S + i)->group << "\n";

cout << (S + i)->num << "\n";

for (int j = 0; j < 8; j++)

{

if ((j == 7) && (i == (size - 1)))

cout << (S + i)->grades[j];

else if (j == 7 && i != (size - 1))

cout << (S + i)->grades[j] << "\n\n";

else

cout << (S + i)->grades[j] << " ";

}

}

}

}