**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационных систем**

отчет

**по практической работе №1**

**по дисциплине «Программирование»**

|  |  |  |
| --- | --- | --- |
| Студент гр. 2373 |  | Чекстер А. А. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2023

**Цель работы.**

Изучение структур в c++, массивов структур, создание прототипа баз данный в c++ при помощи структур, добавление функционала.

**Основные теоретические положения.**

В языке c++, структура — композитный тип данных, инкапсулирующий без сокрытия набор значений различных типов. Порядок размещения значений в памяти задаётся при определении типа и сохраняется на протяжении времени жизни объектов, что даёт возможность косвенного доступа. Пример объявления структуры: struct str\_name

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.

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 имя\_объединения

{

    определения\_элементов

};

**Постановка задачи**

Необходимо создать программу, позволяющую:

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:**

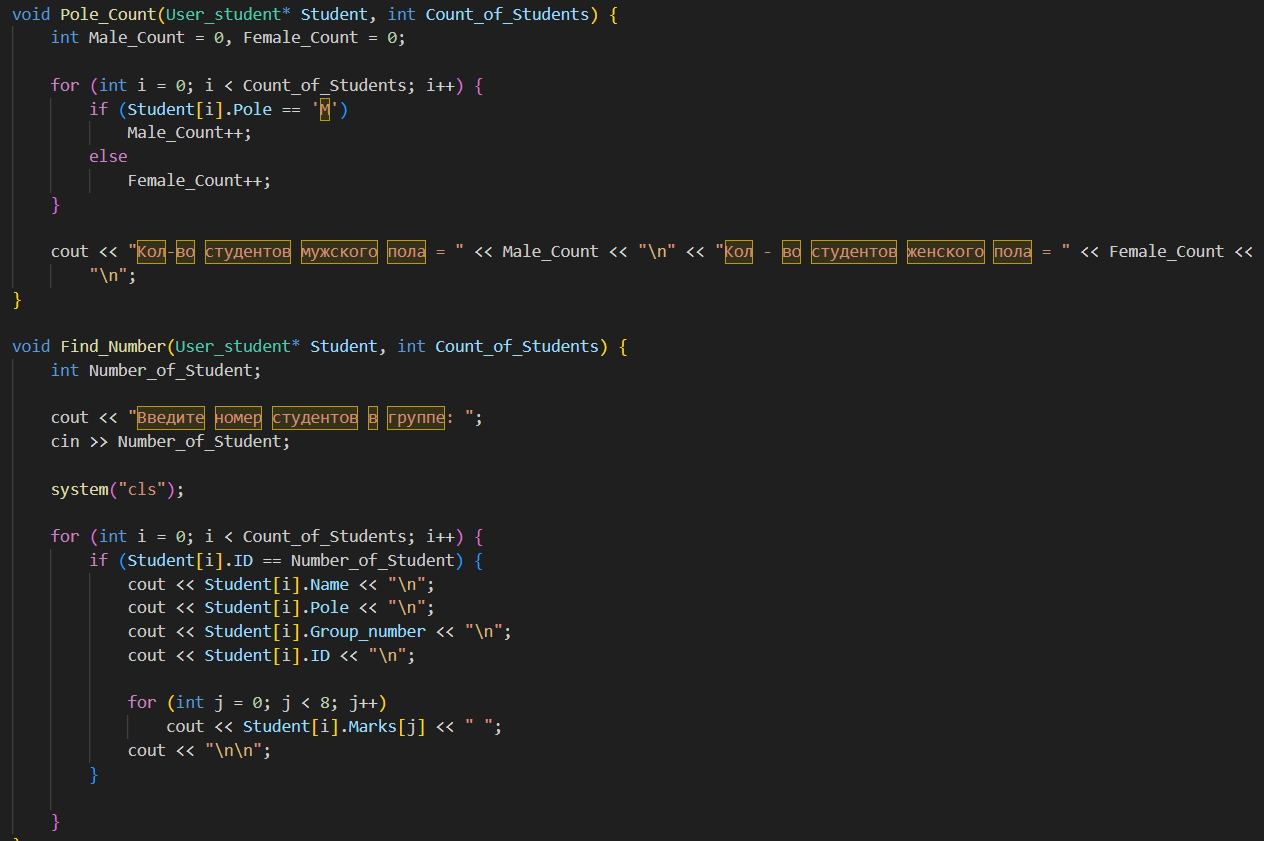
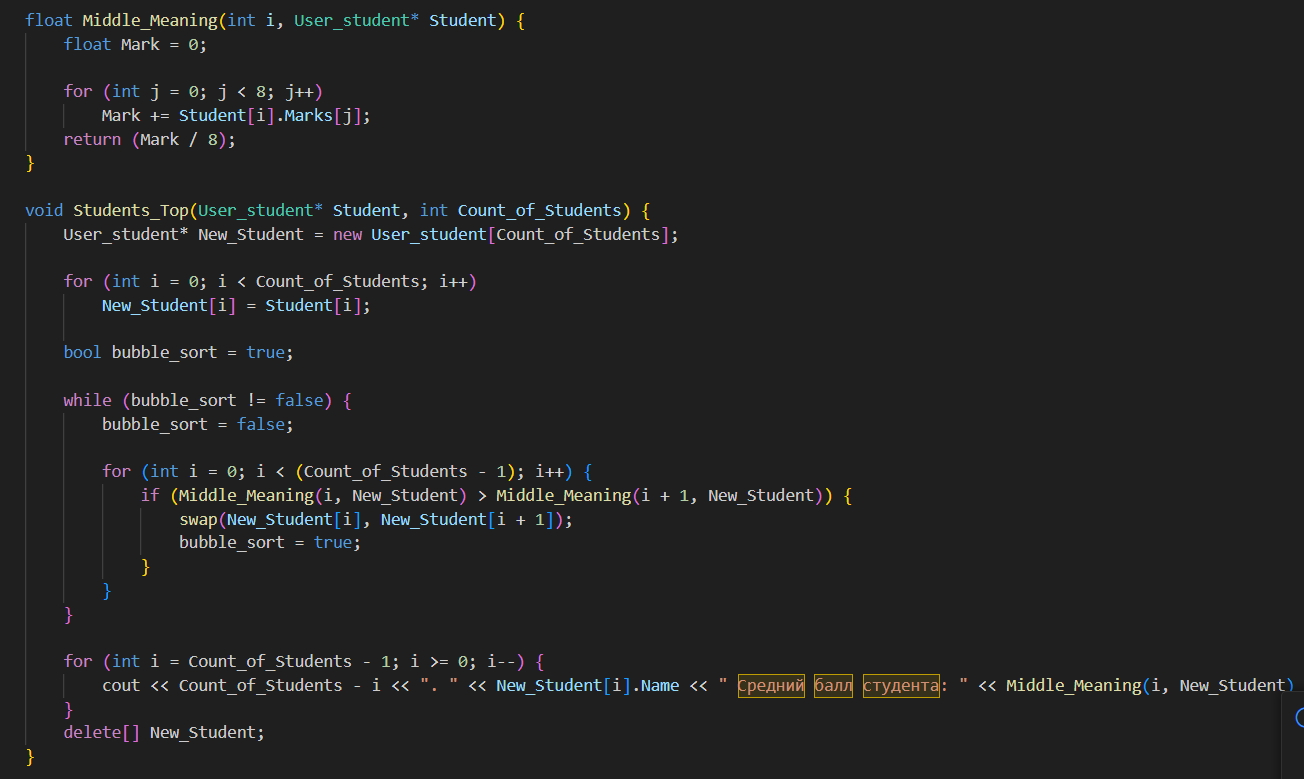
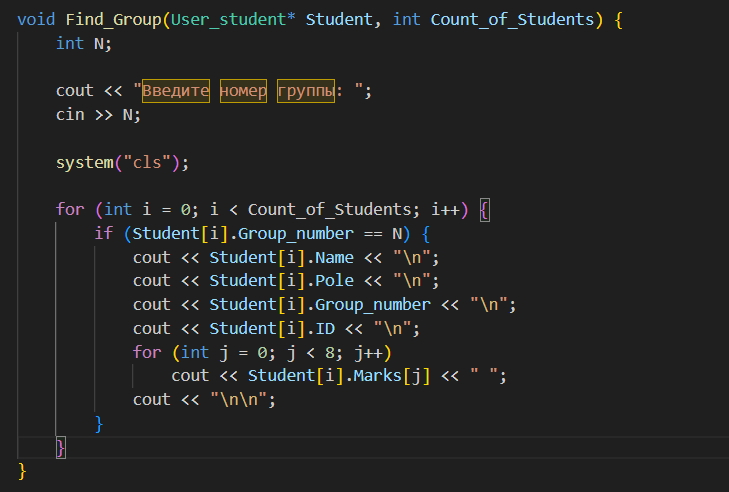
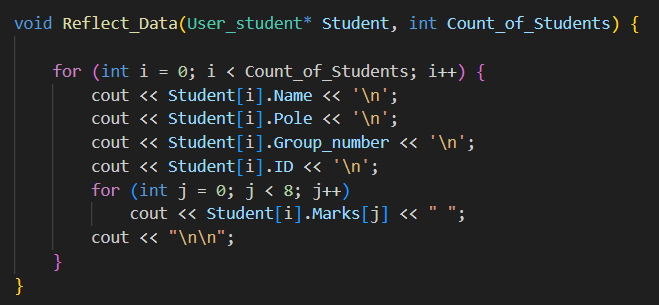
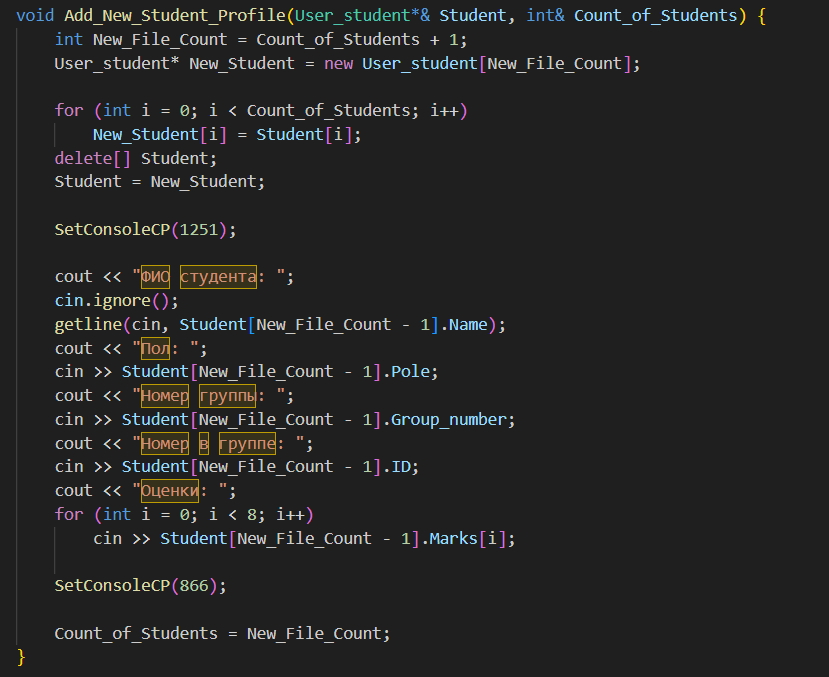
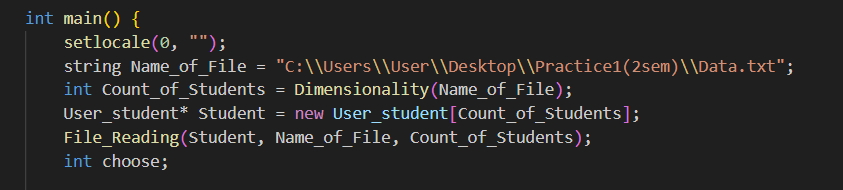
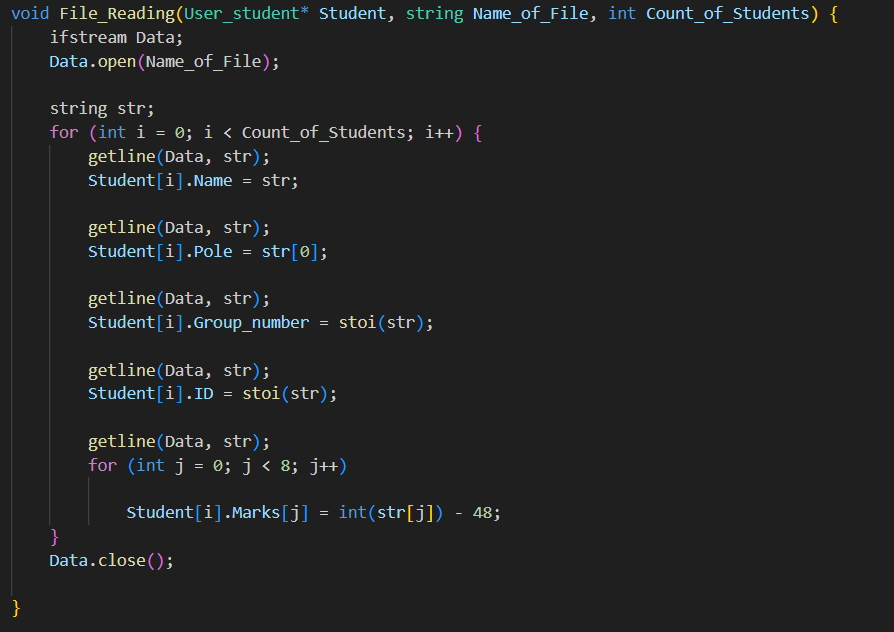
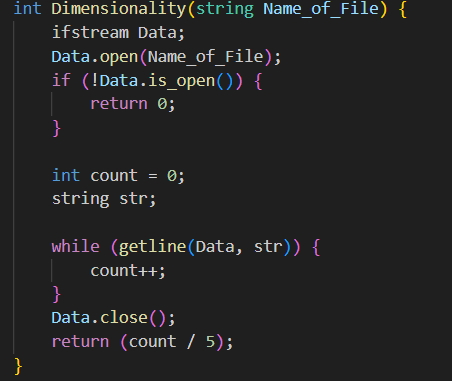
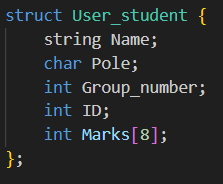
Далее просто считываем файл и полученные данные добавляем в массив структур.

1. Сначала массив расширяется на единицу, а затем в появившуюся ячейку записываются данные о новом студенте.
2. Пользователь вводит номер студента, данные которого хочет изменить, а затем меняет их.
3. Циклом for по очереди выводим всех студентов в списке.
4. Пользователь вводит номер группы и перебором выводятся студенты с соответствующим номером группы.
5. Сначала создаём функцию для высчитывания среднего значения оценок обучающегося.

Далее создаём копию исходного массива и сортируем его алгоритмом Bubble sort.

1. Перебором проходимся по всему массиву и увеличиваем счётчики обеих полов.
2. Делаем проверку оценок студентов. Если находим соответствие, то выводим в консоль.
3. Пользователь вводит номер и перебором выводятся студенты с соответствующим номеров в группе.

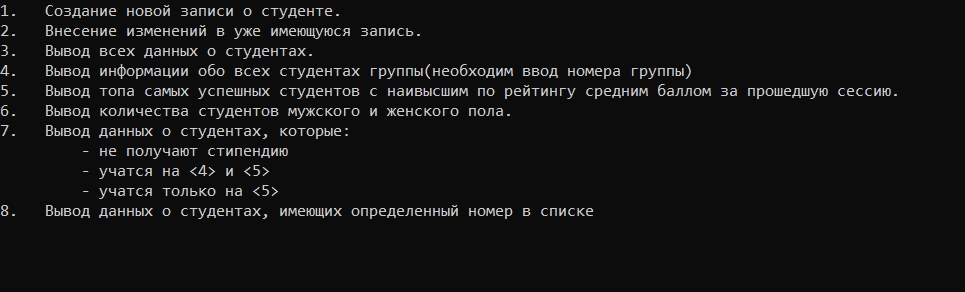
По завершению работы программы записываем получившиеся данные обратно в файл.



**Выводы.**

Мы научились использовать структуры c++, написали прототип баз данный c++, научились использовать функционал структур.

**Блок скриншотов работы программы**



Приложение А

рабочий код

#include <iostream>

#include <string>

#include <fstream>

#include <Windows.h>

using namespace std;

struct User\_student {

    string Name;

    char Pole;

    int Group\_number;

    int ID;

    int Marks[8];

};

void File\_Reading(User\_student\* Student, string Name\_of\_File, int Count\_of\_Students) {

    ifstream Data;

    Data.open(Name\_of\_File);

    string str;

    for (int i = 0; i < Count\_of\_Students; i++) {

        getline(Data, str);

        Student[i].Name = str;

        getline(Data, str);

        Student[i].Pole = str[0];

        getline(Data, str);

        Student[i].Group\_number = stoi(str);

        getline(Data, str);

        Student[i].ID = stoi(str);

        getline(Data, str);

        for (int j = 0; j < 8; j++)

            Student[i].Marks[j] = int(str[j]) - 48;

    }

    Data.close();

}

void File\_Writing(User\_student\* Student, string Name\_of\_File, int Count\_of\_Students) {

    ofstream Data;

    Data.open(Name\_of\_File);

    for (int i = 0; i < Count\_of\_Students; i++) {

        Data << Student[i].Name << '\n';

        Data << Student[i].Pole << '\n';

        Data << Student[i].Group\_number << '\n';

        Data << Student[i].ID << '\n';

        for (int j = 0; j < 8; j++)

            Data << Student[i].Marks[j];

        Data << '\n';

    }

    Data.close();

}

void Add\_New\_Student\_Profile(User\_student\*& Student, int& Count\_of\_Students) {

    int New\_File\_Count = Count\_of\_Students + 1;

    User\_student\* New\_Student = new User\_student[New\_File\_Count];

    for (int i = 0; i < Count\_of\_Students; i++)

        New\_Student[i] = Student[i];

    delete[] Student;

    Student = New\_Student;

    SetConsoleCP(1251);

    cout << "ФИО студента: ";

    cin.ignore();

    getline(cin, Student[New\_File\_Count - 1].Name);

    cout << "Пол: ";

    cin >> Student[New\_File\_Count - 1].Pole;

    cout << "Номер группы: ";

    cin >> Student[New\_File\_Count - 1].Group\_number;

    cout << "Номер в группе: ";

    cin >> Student[New\_File\_Count - 1].ID;

    cout << "Оценки: ";

    for (int i = 0; i < 8; i++)

        cin >> Student[New\_File\_Count - 1].Marks[i];

    SetConsoleCP(866);

    Count\_of\_Students = New\_File\_Count;

}

void Change\_Student\_Profile(User\_student\*& Student, int Count\_of\_Students) {

    int Find\_Group, Find\_ID;

    cout << "Введите номер группы: ";

    cin >> Find\_Group;

    cout << "Введите номер студента в группе: ";

    cin >> Find\_ID;

    int flag = -1;

    for (int i = 0; i < Count\_of\_Students; i++) {

        if ((Find\_Group == Student[i].Group\_number) && (Find\_ID == Student[i].ID)) {

            flag = i;

            break;

        }

    }

    SetConsoleCP(1251);

    if (flag == -1) {

        cout << "\nЭтот студент отсутствует\n\n";

    }

    else {

        int choose;

    Menu\_3: {

        system("cls");

        cout << Student[flag].Name << "\n";

        cout << Student[flag].Pole << "\n";

        cout << Student[flag].Group\_number << "\n";

        cout << Student[flag].ID << "\n";

        for (int i = 0; i < 8; i++)

            cout << Student[flag].Marks[i] << " ";

        cout << "\n\nИзмение содержание базы: \n" <<

            "1. ФИО студента\n" <<

            "2. Пол\n" <<

            "3. Номер группы\n" <<

            "4. Номер в группе\n" <<

            "5. Оценки\n" <<

            "6. Завершить\n";

        }

    cin >> choose;

    switch (choose) {

    case 1: {

        cout << "ФИО студента: ";

        cin.ignore();

        getline(cin, Student[flag].Name);

        goto Menu\_3;

    }

    case 2: {

        cout << "Пол: ";

        cin >> Student[flag].Pole;

        goto Menu\_3;

    }

    case 3: {

        cout << "Номер группы: ";

        cin >> Student[flag].Group\_number;

        goto Menu\_3;

    }

    case 4: {

        cout << " Номер в группе: ";

        cin >> Student[flag].ID;

        goto Menu\_3;

    }

    case 5: {

        cout << " Оценки: ";

        for (int i = 0; i < 8; i++)

            cin >> Student[flag].Marks[i];

        goto Menu\_3;

    }

    case 6: {

        break;

    }

    default: {

        cout << "\nВведите значение из меню\n\n";

        cin.clear();

        while (cin.get() != '\n');

        system("pause");

        goto Menu\_3;

    }

    }

    }

    SetConsoleCP(866);

}

void Reflect\_Data(User\_student\* Student, int Count\_of\_Students) {

    for (int i = 0; i < Count\_of\_Students; i++) {

        cout << Student[i].Name << '\n';

        cout << Student[i].Pole << '\n';

        cout << Student[i].Group\_number << '\n';

        cout << Student[i].ID << '\n';

        for (int j = 0; j < 8; j++)

            cout << Student[i].Marks[j] << " ";

        cout << "\n\n";

    }

}

void Find\_Group(User\_student\* Student, int Count\_of\_Students) {

    int N;

    cout << "Введите номер группы: ";

    cin >> N;

    system("cls");

    for (int i = 0; i < Count\_of\_Students; i++) {

        if (Student[i].Group\_number == N) {

            cout << Student[i].Name << "\n";

            cout << Student[i].Pole << "\n";

            cout << Student[i].Group\_number << "\n";

            cout << Student[i].ID << "\n";

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

                cout << Student[i].Marks[j] << " ";

            cout << "\n\n";

        }

    }

}

float Middle\_Meaning(int i, User\_student\* Student) {

    float Mark = 0;

    for (int j = 0; j < 8; j++)

        Mark += Student[i].Marks[j];

    return (Mark / 8);

}

void Students\_Top(User\_student\* Student, int Count\_of\_Students) {

    User\_student\* New\_Student = new User\_student[Count\_of\_Students];

    for (int i = 0; i < Count\_of\_Students; i++)

        New\_Student[i] = Student[i];

    bool bubble\_sort = true;

    while (bubble\_sort != false) {

        bubble\_sort = false;

        for (int i = 0; i < (Count\_of\_Students - 1); i++) {

            if (Middle\_Meaning(i, New\_Student) > Middle\_Meaning(i + 1, New\_Student)) {

                swap(New\_Student[i], New\_Student[i + 1]);

                bubble\_sort = true;

            }

        }

    }

    for (int i = Count\_of\_Students - 1; i >= 0; i--) {

        cout << Count\_of\_Students - i << ". " << New\_Student[i].Name << " Средний балл студента: " << Middle\_Meaning(i, New\_Student) << "\n";

    }

    delete[] New\_Student;

}

void Data\_of\_Students\_Stependia(User\_student\* Student, int Count\_of\_Students) {

    int choose;

Menu\_2: {

    system("cls");

    cout << "Данные каких студентов вы хотите получить?\n" <<

        "1. Студенты без степендии\n" <<

        "2. Студенты, учащиеся на '4' и '5'\n" <<

        "3. Студенты, учащиеся только на '5'\n" <<

        "4. Завершить\n";

    }

cin >> choose;

switch (choose) {

case 1: {

    system("cls");

    for (int i = 0; i < Count\_of\_Students; i++) {

        bool flag = true;

        for (int j = 0; j < 8; j++)

            if (Student[i].Marks[j] <= 3)

                flag = false;

        if (flag == false) {

            cout << Student[i].Name << "\n";

            cout << Student[i].Pole << "\n";

            cout << Student[i].Group\_number << "\n";

            cout << Student[i].ID << "\n";

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

                cout << Student[i].Marks[j] << " ";

            cout << "\n\n";

        }

    }

    system("pause");

    goto Menu\_2;

}

case 2: {

    system("cls");

    for (int i = 0; i < Count\_of\_Students; i++) {

        bool flag = true;

        for (int j = 0; j < 8; j++)

            if (Student[i].Marks[j] == 4)

                flag = false;

        if (flag == true) {

            cout << Student[i].Name << "\n";

            cout << Student[i].Pole << "\n";

            cout << Student[i].Group\_number << "\n";

            cout << Student[i].ID << "\n";

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

                cout << Student[i].Marks[j] << " ";

            cout << "\n\n";

        }

    }

    system("pause");

    goto Menu\_2;

}

case 3: {

    system("cls");

    for (int i = 0; i < Count\_of\_Students; i++) {

        bool flag = true;

        for (int j = 0; j < 8; j++)

            if (Student[i].Marks[j] == 4 or Student[i].Marks[j] == 5)

                flag = false;

        if (flag == true) {

            cout << Student[i].Name << "\n";

            cout << Student[i].Pole << "\n";

            cout << Student[i].Group\_number << "\n";

            cout << Student[i].ID << "\n";

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

                cout << Student[i].Marks[j] << " ";

            cout << "\n\n";

        }

    }

    system("pause");

    goto Menu\_2;

}

case 4: {

    break;

}

default: {

    cout << "\nВведите значение из меню\n\n";

    cin.clear();

    while (cin.get() != '\n');

    system("pause");

    goto Menu\_2;

}

}

}

void Pole\_Count(User\_student\* Student, int Count\_of\_Students) {

    int Male\_Count = 0, Female\_Count = 0;

    for (int i = 0; i < Count\_of\_Students; i++) {

        if (Student[i].Pole == 'М')

            Male\_Count++;

        else

            Female\_Count++;

    }

    cout << "Кол-во студентов мужского пола = " << Male\_Count << "\n" << "Кол - во студентов женского пола = " << Female\_Count <<

        "\n";

}

void Find\_Number(User\_student\* Student, int Count\_of\_Students) {

    int Number\_of\_Student;

    cout << "Введите номер студентов в группе: ";

    cin >> Number\_of\_Student;

    system("cls");

    for (int i = 0; i < Count\_of\_Students; i++) {

        if (Student[i].ID == Number\_of\_Student) {

            cout << Student[i].Name << "\n";

            cout << Student[i].Pole << "\n";

            cout << Student[i].Group\_number << "\n";

            cout << Student[i].ID << "\n";

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

                cout << Student[i].Marks[j] << " ";

            cout << "\n\n";

        }

    }

}

int Dimensionality(string Name\_of\_File) {

    ifstream Data;

    Data.open(Name\_of\_File);

    if (!Data.is\_open()) {

        return 0;

    }

    int count = 0;

    string str;

    while (getline(Data, str)) {

        count++;

    }

    Data.close();

    return (count / 5);

}

int main() {

    setlocale(0, "");

    string Name\_of\_File = "C:\\Users\\User\\Desktop\\Practice1(2sem)\\Data.txt";

    int Count\_of\_Students = Dimensionality(Name\_of\_File);

    User\_student\* Student = new User\_student[Count\_of\_Students];

    File\_Reading(Student, Name\_of\_File, Count\_of\_Students);

    int choose;

Menu: {cout << "1.   Создание новой записи о студенте.\n" <<

"2.   Внесение изменений в уже имеющуюся запись.\n" <<

"3.   Вывод всех данных о студентах.\n" <<

"4.   Вывод информации обо всех студентах группы(необходим ввод номера группы)\n" <<

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

"6.   Вывод количества студентов мужского и женского пола.\n" <<

"7.   Вывод данных о студентах, которые:\n" <<

"\t - не получают стипендию\n" <<

"\t - учатся на «4» и «5»\n" <<

"\t - учатся только на «5» \n" <<

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

    }

cin >> choose;

switch (choose) {

case 1: {

    system("cls");

    Add\_New\_Student\_Profile(Student, Count\_of\_Students);

    system("pause");

    goto Menu;

}

case 2: {

    system("cls");

    Change\_Student\_Profile(Student, Count\_of\_Students);

    system("pause");

    goto Menu;

}

case 3: {

    system("cls");

    Reflect\_Data(Student, Count\_of\_Students);

    system("pause");

    goto Menu;

}

case 4: {

    system("cls");

    Find\_Group(Student, Count\_of\_Students);

    system("pause");

    goto Menu;

}

case 5: {

    system("cls");

    Students\_Top(Student, Count\_of\_Students);

    system("pause");

    goto Menu;

}

case 6: {

    system("cls");

    Pole\_Count(Student, Count\_of\_Students);

    system("pause");

    goto Menu;

}

case 7: {

    system("cls");

    Data\_of\_Students\_Stependia(Student, Count\_of\_Students);

    system("pause");

    goto Menu;

}

case 8: {

    system("cls");

    Find\_Number(Student, Count\_of\_Students);

    system("pause");

    goto Menu;

}

case 9: {

    system("cls");

    File\_Writing(Student, Name\_of\_File, Count\_of\_Students);

    system("pause");

    goto Menu;

}

default: {

    cout << "\nВведите значение из меню\n\n";

    cin.clear();

    while (cin.get() != '\n');

    system("pause");

    goto Menu;

}

}

}