**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:** **типы данных, определяемые пользователем»**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 0324 |  | Жигалова Д.А. |
| Преподаватель |  | Глущенко А.Г |

Санкт-Петербург

2021

# **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.

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 ];

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

struct Student

{

int age;

char name[10];

int group;

};

После определения структуры можно её использовать. Для начала можно определить объект структуры - по сути обычную переменную, которая будет представлять описанную ранее структуру.

struct Student Ivan;

Здесь определена переменная Ivan, которая представляет структуру Student.

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.

Еще один способ определения структуры представляет ключевое слово typedef:

typedef struct

{

int age;

char name[10];

} person;

В конце определения структуры после закрывающей фигурной скобки идет ее обозначение - в данном случае person. В дальнейшем можно использовать это обозначение для создания переменной структуры.

person tom = {23, "Tom"};

Еще один способ определить структуру представляет применение препроцессорной директивы #define:

#define PERSON struct {int age; char name[20];}

int main(void)

{

PERSON tom = {23, "Tom"};

return 0;

}

В данном случае директива define определяет константу PERSON, вместо которой при обработке исходного кода препроцессором будет вставляться код структуры struct {int age; char name[20];}

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.

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.

Передачу структур в функции по значению необходимо использовать аккуратно:

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;

}

Вызов такой функции сопровождается дополнительным расходом памяти для создания локальной переменной S и дополнительными затратами времени на физическое копирование данных из аргумента в параметр S. Учитывая то, что объем структур может быть очень большим, эти дополнительные затраты вычислительных ресурсов могут быть чрезмерными.

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.

Передача по ссылке по эффективности эквивалентна передаче данных через указатель. Однако, поскольку при передаче данных по ссылке все адресные преобразования берет на себя компилятор, существенно упрощается программирование действий со структурами. При использовании ссылочных параметров структурных типов доступ к членам структуры осуществляется обычным способом – с помощью оператора «точка».

Недостатком этих способов является то, что случайные изменения значений полей структуры внутри функции отразятся на значении аргумента после окончания работы функции. Если необходимо предотвратить изменения переданных по адресу аргументов, можно при определении соответствующего параметра объявить его константой (использовать спецификатор const).

# **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), форма обучения\*, отметка времени о внесении или изменении данных\*. Ввод и изменение данных обо всех студентах должен осуществляться в файл students\*.

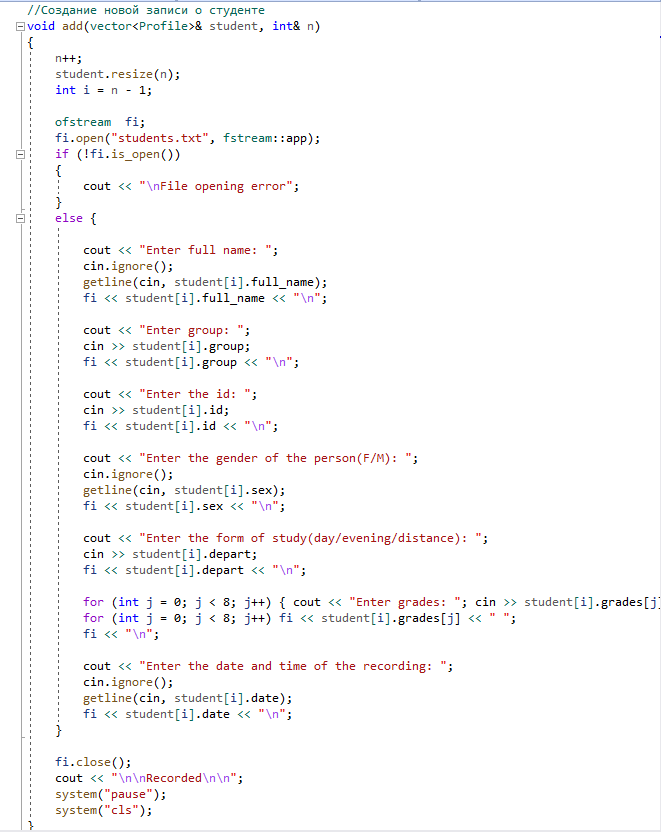
Написать функции, реализующие операции со структурами (ввод данных с клавиатуры):

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. Определение количества студентов, которые будут получать стипендию (стипендия начисляется, если у студента нет троек и очная форма обучения).\*
8. Вывод данных о студентах, которые не получают стипендию; учатся только на «хорошо» и «отлично»; учатся только на «отлично»;
9. Вывод данных о студентах, имеющих номер в списке – k.
10. Вывод всех записей, сделанных в день, который введет пользователь. Вывод всех записей, сделанных после полудня. Вывод всех записей, сделанных до полудня.\*

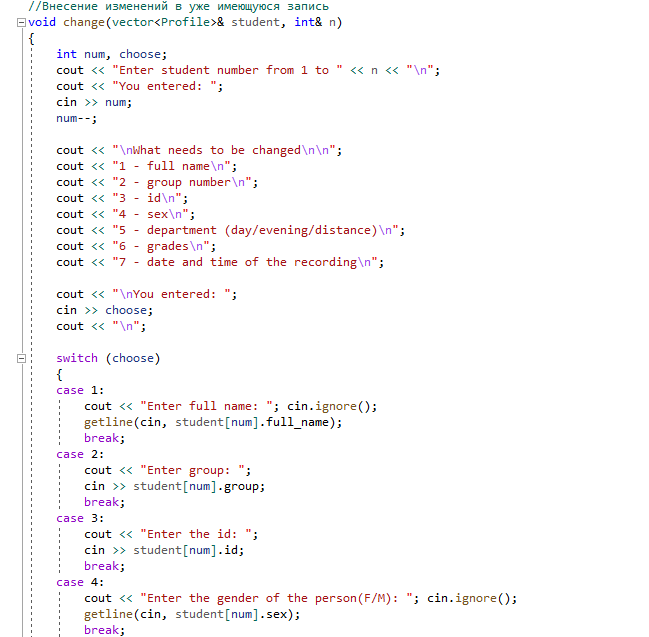
**Выполнение работы**

Для решения поставленных задач была создана программа на языке программирования C++. Итоговый код программы представлен в приложении А, а результат работы в приложении B.

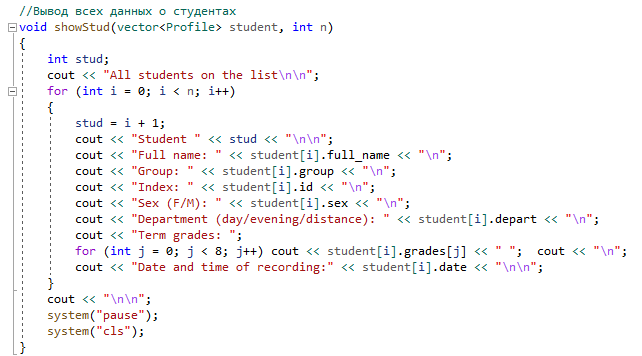
Для решения первой задачи была создана функция add, которая с клавиатуры позволяла вводить данные о новом студенте:



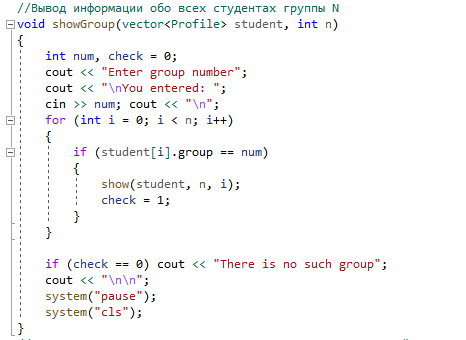
Во второй задаче для редактирования входного текста была создана функция change.



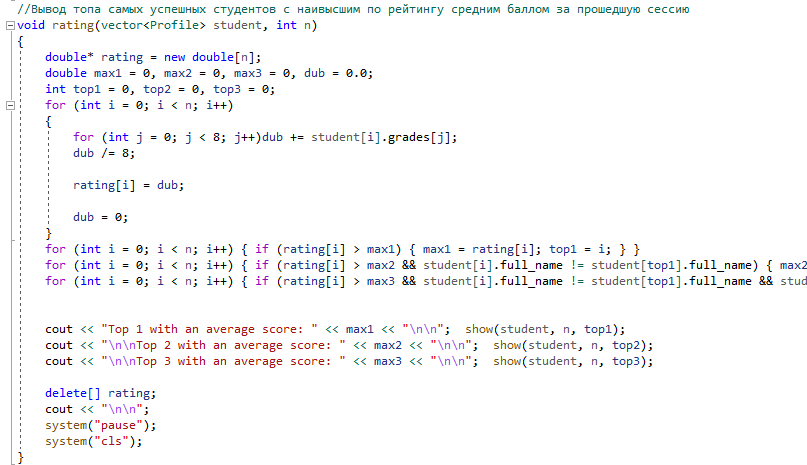
В третьей задаче была создана функция, которая позволяла вывести на экран данные обо всех студентах.



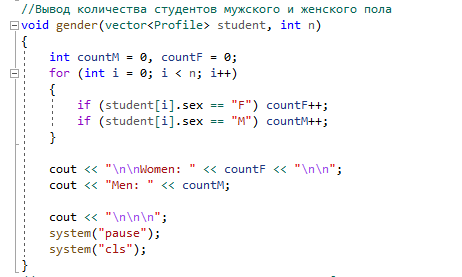
В четвертой задаче также была создана функция, которая выводила данные о студентах группы N.



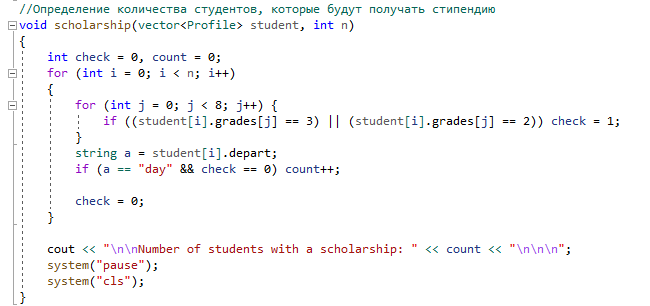
В пятой задаче также была создана функция вывода топа самых успешных студентов с наивысшим по рейтингу средним баллом за прошедшую сессию.



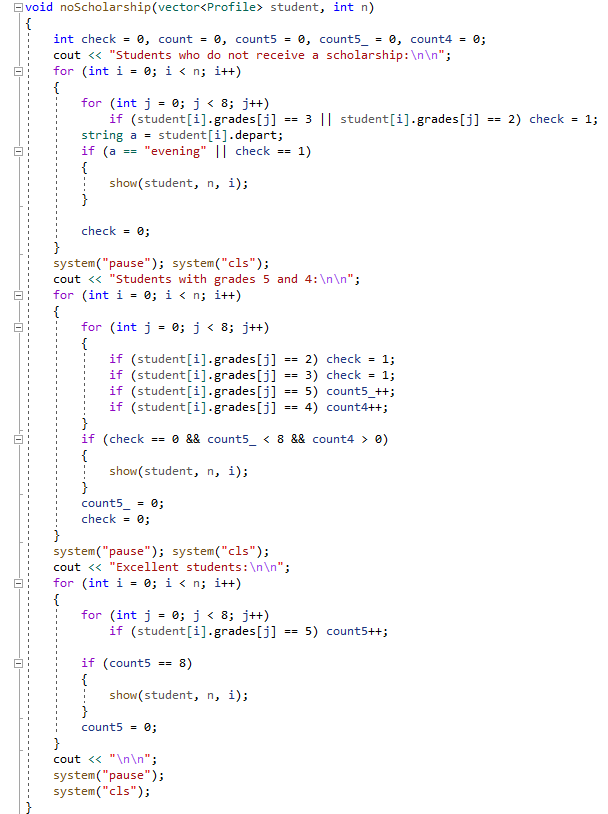
В шестой задаче была создана функция вывода количества студентов мужского и женского пола.



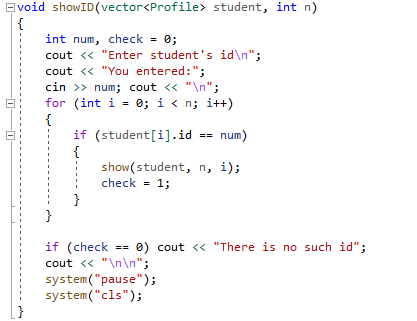
В седьмой задаче была создана функция определения количества студентов, которые будут получать стипендию.



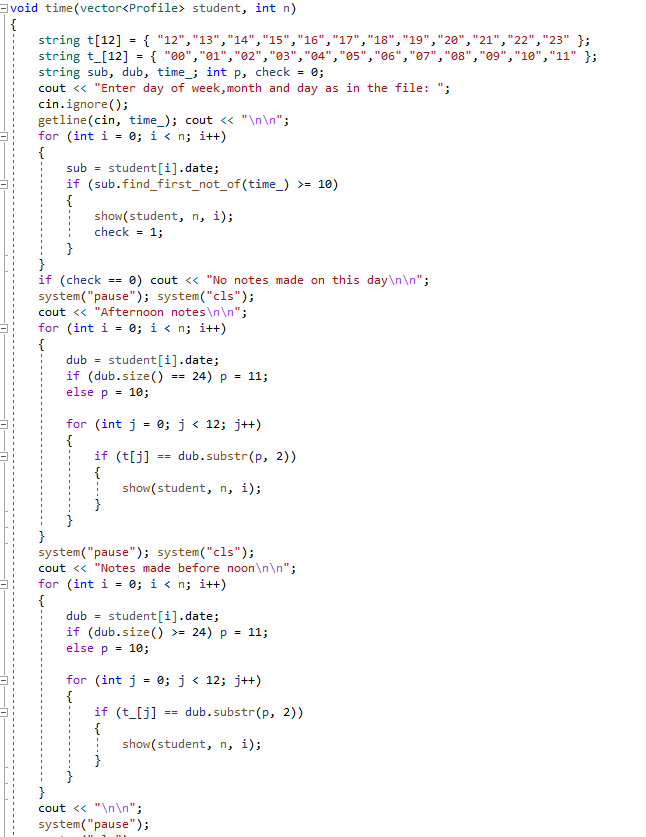
В восьмой задаче была создана функция вывода данных о студентах, которые не получают стипендию; учатся только на «хорошо» и «отлично»; учатся только на «отлично».



В девятой задаче была создана функция вывода данных о студентах, имеющих номер в списке – k.



В десятой задаче была создана функция вывода всех записей, сделанных в день, который введет пользователь. Вывод всех записей, сделанных после полудня. Вывод всех записей, сделанных до полудня.



**Вывод**

Изучены структуры и их организация. Получены практические навыки работы со структурами. Структура – это совокупность переменных, объединенных одним именем, предоставляющая общепринятый способ совместного хранения информации. Объявление структуры приводит к образованию шаблона, используемого для создания объектов структуры. Стоит отметить, что структуры относятся к типам значений, и поэтому ими можно оперировать непосредственно, а не по ссылке. Следовательно, для работы со структурой вообще не требуется переменная ссылочного типа, а это означает в ряде случаев существенную экономию оперативной памяти.

# **ПРИЛОЖЕНИЕ А**

**ПОЛНЫЙ КОД ПРОГРАММЫ**

#include <vector>

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

struct Profile

{

string full\_name;

string sex;

int group;

int id;

int grades[8];

char depart[9];

string date;

};

//Данные о студентах с id i

void show(vector<Profile> student, int n, int i)

{

cout << "Full name: " << student[i].full\_name << "\n";

cout << "Group: " << student[i].group << "\n";

cout << "Index: " << student[i].id << "\n";

cout << "Sex (F/M): " << student[i].sex << "\n";

cout << "Department (day/evening/distance): " << student[i].depart << "\n";

cout << "Term grades: ";

for (int j = 0; j < 8; j++) cout << student[i].grades[j] << " "; cout << "\n";

cout << "Date and time of recording:" << student[i].date << "\n\n";

}

//Вывод всех данных о студентах

void showStud(vector<Profile> student, int n)

{

int stud;

cout << "All students on the list\n\n";

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

{

stud = i + 1;

cout << "Student " << stud << "\n\n";

cout << "Full name: " << student[i].full\_name << "\n";

cout << "Group: " << student[i].group << "\n";

cout << "Index: " << student[i].id << "\n";

cout << "Sex (F/M): " << student[i].sex << "\n";

cout << "Department (day/evening/distance): " << student[i].depart << "\n";

cout << "Term grades: ";

for (int j = 0; j < 8; j++) cout << student[i].grades[j] << " "; cout << "\n";

cout << "Date and time of recording:" << student[i].date << "\n\n";

}

cout << "\n\n";

system("pause");

system("cls");

}

//Создание новой записи о студенте

void add(vector<Profile>& student, int& n)

{

n++;

student.resize(n);

int i = n - 1;

ofstream fi;

fi.open("students.txt", fstream::app);

if (!fi.is\_open())

{

cout << "\nFile opening error";

}

else {

cout << "Enter full name: ";

cin.ignore();

getline(cin, student[i].full\_name);

fi << student[i].full\_name << "\n";

cout << "Enter group: ";

cin >> student[i].group;

fi << student[i].group << "\n";

cout << "Enter the id: ";

cin >> student[i].id;

fi << student[i].id << "\n";

cout << "Enter the gender of the person(F/M): ";

cin.ignore();

getline(cin, student[i].sex);

fi << student[i].sex << "\n";

cout << "Enter the form of study(day/evening/distance): ";

cin >> student[i].depart;

fi << student[i].depart << "\n";

for (int j = 0; j < 8; j++) { cout << "Enter grades: "; cin >> student[i].grades[j]; }

for (int j = 0; j < 8; j++) fi << student[i].grades[j] << " ";

fi << "\n";

cout << "Enter the date and time of the recording: ";

cin.ignore();

getline(cin, student[i].date);

fi << student[i].date << "\n";

}

fi.close();

cout << "\n\nRecorded\n\n";

system("pause");

system("cls");

}

//Запись нового студента в файл

void recording(vector<Profile>& student, int& n)

{

ofstream Rec;

Rec.open("students.txt");

if (!Rec.is\_open())

{

cout << "\nFile opening error";

}

else {

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

{

Rec << student[i].full\_name << "\n";

Rec << student[i].group << "\n";

Rec << student[i].id << "\n";

Rec << student[i].sex << "\n";

Rec << student[i].depart << "\n";

for (int j = 0; j < 8; j++) Rec << student[i].grades[j] << " ";

Rec << "\n";

Rec << student[i].date << "\n";

}

}

Rec.close();

}

//Внесение изменений в уже имеющуюся запись

void change(vector<Profile>& student, int& n)

{

int num, choose;

cout << "Enter student number from 1 to " << n << "\n";

cout << "You entered: ";

cin >> num;

num--;

cout << "\nWhat needs to be changed\n\n";

cout << "1 - full name\n";

cout << "2 - group number\n";

cout << "3 - id\n";

cout << "4 - sex\n";

cout << "5 - department (day/evening/distance)\n";

cout << "6 - grades\n";

cout << "7 - date and time of the recording\n";

cout << "\nYou entered: ";

cin >> choose;

cout << "\n";

switch (choose)

{

case 1:

cout << "Enter full name: "; cin.ignore();

getline(cin, student[num].full\_name);

break;

case 2:

cout << "Enter group: ";

cin >> student[num].group;

break;

case 3:

cout << "Enter the id: ";

cin >> student[num].id;

break;

case 4:

cout << "Enter the gender of the person(F/M): "; cin.ignore();

getline(cin, student[num].sex);

break;

case 5:

cout << "Enter the form of study(day/evening/distance): "; cin.ignore();

cin >> student[num].depart;

break;

case 6:

for (int j = 0; j < 8; j++) { cout << "Enter grades: "; cin >> student[num].grades[j]; }

break;

case 7:

cout << "Enter the date and time of the recording: "; cin.ignore();

getline(cin, student[num].date);

break;

}

recording(student, n);

cout << "\nData changed\n\n";

system("pause");

system("cls");

}

//Вывод информации обо всех студентах группы N

void showGroup(vector<Profile> student, int n)

{

int num, check = 0;

cout << "Enter group number";

cout << "\nYou entered: ";

cin >> num; cout << "\n";

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

{

if (student[i].group == num)

{

show(student, n, i);

check = 1;

}

}

if (check == 0) cout << "There is no such group";

cout << "\n\n";

system("pause");

system("cls");

}

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

void rating(vector<Profile> student, int n)

{

double\* rating = new double[n];

double max1 = 0, max2 = 0, max3 = 0, dub = 0.0;

int top1 = 0, top2 = 0, top3 = 0;

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

{

for (int j = 0; j < 8; j++)dub += student[i].grades[j];

dub /= 8;

rating[i] = dub;

dub = 0;

}

for (int i = 0; i < n; i++) { if (rating[i] > max1) { max1 = rating[i]; top1 = i; } }

for (int i = 0; i < n; i++) { if (rating[i] > max2 && student[i].full\_name != student[top1].full\_name) { max2 = rating[i]; top2 = i; } }

for (int i = 0; i < n; i++) { if (rating[i] > max3 && student[i].full\_name != student[top1].full\_name && student[i].full\_name != student[top2].full\_name) { max3 = rating[i]; top3 = i; } }

cout << "Top 1 with an average score: " << max1 << "\n\n"; show(student, n, top1);

cout << "\n\nTop 2 with an average score: " << max2 << "\n\n"; show(student, n, top2);

cout << "\n\nTop 3 with an average score: " << max3 << "\n\n"; show(student, n, top3);

delete[] rating;

cout << "\n\n";

system("pause");

system("cls");

}

//Вывод количества студентов мужского и женского пола

void gender(vector<Profile> student, int n)

{

int countM = 0, countF = 0;

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

{

if (student[i].sex == "F") countF++;

if (student[i].sex == "M") countM++;

}

cout << "\n\nWomen: " << countF << "\n\n";

cout << "Men: " << countM;

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

system("pause");

system("cls");

}

//Определение количества студентов, которые будут получать стипендию

void scholarship(vector<Profile> student, int n)

{

int check = 0, count = 0;

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

{

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

if ((student[i].grades[j] == 3) || (student[i].grades[j] == 2)) check = 1;

}

string a = student[i].depart;

if (a == "day" && check == 0) count++;

check = 0;

}

cout << "\n\nNumber of students with a scholarship: " << count << "\n\n\n";

system("pause");

system("cls");

}

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

void noScholarship(vector<Profile> student, int n)

{

int check = 0, count = 0, count5 = 0, count5\_ = 0, count4 = 0;

cout << "Students who do not receive a scholarship:\n\n";

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

{

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

if (student[i].grades[j] == 3 || student[i].grades[j] == 2) check = 1;

string a = student[i].depart;

if (a == "evening" || check == 1)

{

show(student, n, i);

}

check = 0;

}

system("pause"); system("cls");

cout << "Students with grades 5 and 4:\n\n";

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

{

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

{

if (student[i].grades[j] == 2) check = 1;

if (student[i].grades[j] == 3) check = 1;

if (student[i].grades[j] == 5) count5\_++;

if (student[i].grades[j] == 4) count4++;

}

if (check == 0 && count5\_ < 8 && count4 > 0)

{

show(student, n, i);

}

count5\_ = 0;

check = 0;

}

system("pause"); system("cls");

cout << "Excellent students:\n\n";

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

{

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

if (student[i].grades[j] == 5) count5++;

if (count5 == 8)

{

show(student, n, i);

}

count5 = 0;

}

cout << "\n\n";

system("pause");

system("cls");

}

//Вывод данных о студентах, имеющих номер в списке

void showID(vector<Profile> student, int n)

{

int num, check = 0;

cout << "Enter student's id\n";

cout << "You entered:";

cin >> num; cout << "\n";

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

{

if (student[i].id == num)

{

show(student, n, i);

check = 1;

}

}

if (check == 0) cout << "There is no such id";

cout << "\n\n";

system("pause");

system("cls");

}

//Вывод всех записей, сделанных в день, который введет пользователь

void time(vector<Profile> student, int n)

{

string t[12] = { "12","13","14","15","16","17","18","19","20","21","22","23" };

string t\_[12] = { "00","01","02","03","04","05","06","07","08","09","10","11" };

string sub, dub, time\_; int p, check = 0;

cout << "Enter day of week,month and day as in the file: ";

cin.ignore();

getline(cin, time\_); cout << "\n\n";

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

{

sub = student[i].date;

if (sub.find\_first\_not\_of(time\_) >= 10)

{

show(student, n, i);

check = 1;

}

}

if (check == 0) cout << "No notes made on this day\n\n";

system("pause"); system("cls");

cout << "Afternoon notes\n\n";

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

{

dub = student[i].date;

if (dub.size() == 24) p = 11;

else p = 10;

for (int j = 0; j < 12; j++)

{

if (t[j] == dub.substr(p, 2))

{

show(student, n, i);

}

}

}

system("pause"); system("cls");

cout << "Notes made before noon\n\n";

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

{

dub = student[i].date;

if (dub.size() >= 24) p = 11;

else p = 10;

for (int j = 0; j < 12; j++)

{

if (t\_[j] == dub.substr(p, 2))

{

show(student, n, i);

}

}

}

cout << "\n\n";

system("pause");

system("cls");

}

int main() {

ifstream fin;

fin.open("students.txt");

int n = 0, task, task2, task3;

while (!fin.eof()) {

if (fin.get() == '\n')

n++;

}

n = n / 7;

fin.close();

cout << "Number of students on the list: " << n << "\n\n";

vector<Profile>student(n);

ifstream f;

f.open("students.txt");

if (!f.is\_open())

{

cout << "\nFile opening error";

}

else {

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

{

getline(f, student[i].full\_name);

f >> student[i].group; f.ignore();

f >> student[i].id; f.ignore();

getline(f, student[i].sex);

f >> student[i].depart; f.ignore();

for (int j = 0; j < 8; j++) { f >> student[i].grades[j]; }

f.ignore(2);

getline(f, student[i].date);

}

}

f.close();

do

{

cout << "Enter number of task\n\n";

cout << "1. Create a new student record\n";

cout << "2. Making changes to an existing record\n";

cout << "3. Display of all student data\n";

cout << "4. Display of information about all students of group N\n";

cout << "5. Finding the most successful students\n";

cout << "6. Display of the number of male and female students\n";

cout << "7. Determining the number of students who will receive the scholarship\n";

cout << "8. Conclusion of data on students who: do not receive a scholarship, study only for good and excellent, study only excellently\n";

cout << "9. Displays data about students who have a number in the list - k\n";

cout << "10. Displays all entries made on the day that the user enters, displays all entries made in the afternoon, displays all entries made before noon.\n";

cout << "To exit, enter 0\n";

cout << "\n";

cout << "You entered: ";

cin >> task;

system("cls");

switch (task)

{

case 1:

add(student, n);

break;

case 2:

change(student, n);

break;

case 3:

showStud(student, n);

break;

case 4:

showGroup(student, n);

break;

case 5:

rating(student, n);

break;

case 6:

gender(student, n);

break;

case 7:

scholarship(student, n);

break;

case 8:

noScholarship(student, n);

break;

case 9:

showID(student, n);

break;

case 10:

time(student, n);

break;

case 0:

break;

}

} while (task != 0);

return 0;

}

# **ПРИЛОЖЕНИЕ B**

**РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ**

