**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра «Информационные Системы и Технологий (ИСиТ)»**

отчет

**по лабораторной работе №1**

**по дисциплине «Программирование»**

Тема: «Структуры»

|  |  |  |
| --- | --- | --- |
| Студент гр. 0324 |  | Зимацкий С. Н. |
| 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.

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.

Необходимо подготовить собственный файл с тестовыми данными.

**Основные теоретические положения.**

## Определение структур

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;

};

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];}

В языке С++ отличие от С при определении переменной не нужно использовать слово **struct**.

## Инициализация структур

Инициализация структур аналогична инициализации массивов: в фигурных скобках передаются значения для элементов структуры по порядку. Так как в структуре student первым определено свойство, которое представляет тип **int**- число, то в фигурных скобках вначале идет число. И так далее для всех элементов структуры по порядку.

При определении переменной структуры ее можно сразу инициализировать, присвоив какое-нибудь значение:

struct student Ivan = {23, "Ivan", 0393};

Инициализировать структуру можно сразу после её объявления. 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}}

};

Все поля структурных переменных располагаются в непрерывной области памяти одно за другим. Общий объем памяти, занимаемый структурой, равен сумме размеров всех полей структуры. Для определения размера структуры следует использовать инструкцию **sizeof()**.

## Структуры как элементы структур

Кроме базовых примитивных типов данных как **int**или **char**, массивов и указателей в качестве элементов структуры можно использовать другие структуры.

struct Pizza

{

char name[20];

int price;

};

struct Client

{

char name[20];

char phone[20];

};

struct Order

{

struct Client client;

int count;

struct Pizza pizza;

};

Использованные здесь структуры отражают объекты реального мира. Структура **Order**представляет некоторого клиента, его элементы **client**, **count**  и **pizza** представляют клиента-заказчика (структура **client**), количество заданных пицц и заказанную пиццу (структура **pizza**) соответственно. Структура **Pizza**представляет собой пиццу, в которой определено три элемента: **name**- название, **price**- стоимость. Структура **Client** содержит информацию о клиенте-заказчике, его имя **name**и номер телефона **phone**.

Таким образом, получается, что структура **Order**имеет три элемента, но так клиент и пицца - составные элементы и сами содержат по два элемента, то при инициализации структуры **order** нужно передать 5 параметров:

Order order\_1 = {"Ivan", "+7 (111) 272-65-52", 2, "Hawaiian", 280};

Передача значений таким образом идет в том порядке, в котором элементы определены в структуре, в том числе во вложенных структурах.

## Массивы структур

В программировании очень часто используются такие конструкции, как массивы структур. Все форматы определения массива структур будут аналогичны определению массивов других типов:

struct Students students[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.

Ещё один пример использования массива структур:

struct person

{

int age;

char name[20];

};

int main(void)

{

struct person people[] = {23, "Tom", 32, "Bob", 26, "Alice", 41, "Sam"};

return 0;

}

В массиве people определено 4 объекта person. Хотя при инициализации в массив передается 8 значений {23, "Tom", 32, "Bob", 26, "Alice", 41, "Sam"}, но так как каждая структура состоит из двух элементов, то соответственно из этих значений по порядку получается 4 объекта person.

## Доступ к полям структур

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).

struct company

{

char name[20];

char country[30];

};

struct smartphone

{

char title[20];

int price;

struct company manufacturer;

};

int main(void)

{

struct smartphone phone = {"iPhone 8", 56000, "Apple", "USA"};

std::cout << phone.title;

std::cout << phone.price;

std::cout << phone.manufacturer.name;

return 0;

}

## 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».

## Использование указателей с массивами структур

Также как с массивами других типов с массивами структур можно использовать указатели:

struct person

{

int age;

char name[20];

};

int main(void)

{

struct person people[] = {23, "Tom", 32, "Bob", 26, "Alice", 41, "Sam"};

int n = sizeof(people)/sizeof(people[0]);

for(struct person \*p=people; p < people+n; p++)

{

cout << "name: " <<p->name << " age: " << p->age;

}

return 0;

}

Здесь в массиве **people** те же 4 элемента person. Для их просмотра создан указатель **\*p**, который устанавливается на начало массива **people**. И в цикле получаем элементы структур через этот указатель. После завершения каждой итерации указатель увеличивается на единицу, то есть перемещается в памяти на количество байт, которые занимает одна структура. И эти действия продолжаются пока указатель не дойдет до конца массива, который можно получить через выражение people+n.

## Структурные параметры функции

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).

## Указатели на структуру как параметры

При использовании структур в качестве параметров в функции следует учитывать, что при вызове функции для структуры, также как и для параметров типа **int** или **char**, выделяется память, в которую помещаются значения элементов структуры. То есть структура в функцию передается по значению, а это значит, что переданную в функцию структуру мы изменить не можем.

Для примера определим функцию, где попытаемся изменить структуру:

struct time

{

int hour;

int minute;

int second;

};

// прототип функции

void addminutes(struct time, int);

int main(void)

{

struct time current\_time = {17, 38, 10};

addminutes(current\_time, 21);

cout << current\_time.hour << " " << current\_time.minute << " " << current\_time.second << "\n|;

return 0;

}

void addminutes(struct time t, int minutes)

{

int h, d;

// прибавляем минуты

t.minute += minutes;

// если минут больше 59, делим на 60

if(t.minute >=60)

{

h = t.minute / 60;

// от минут вычитаем 60 \* h

t.minute -= 60 \* h;

// к часам прибавляем h

t.hour +=h;

}

// если часов больше 23, делим на 24

if(t.hour >=24)

{

d = t.hour / 24;

// от часов вычитаем d \* 24

t.hour -= 24 \* d;

}

cout << t.hour << " " << t.minute << " " << t.second);

}

Если мы запустим программу, то увидим, передаваемая структура **current\_time** в функцию **addminutes** после выполнения этой функции не изменится. Если действительно нужно изменить структуру в функции, то на надо передавать не саму структуру, а указатель на нее.

* Указатели на структуру как результаты функций

И еще один вариант использования структур в функции - это возвращение из функции указателя на структуру. Например, определим функцию, которая будет возвращать указатель на структуру time:

struct time

{

int hour;

int minute;

int second;

};

// прототип функции

struct time \* input(void);

int main(void)

{

struct time \* p\_time = input();

cout << p\_time->hour << " " << p\_time->minute << " " << p\_time->second << "\n";

free(p\_time); // освобождаем память

return 0;

}

struct time \* input()

{

// выделяем память для структуры

struct time \* p\_time = (struct time \*) malloc(sizeof(struct time));

// ввод значений

cout << "Enter hour: ";

cin >> &p\_time->hour;

cout << "Enter minutes: ";

cin >> &p\_time->minute;

cout << "Enter seconds: ";

cin << &p\_time->second;

return p\_time;

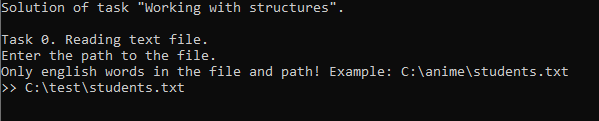
}

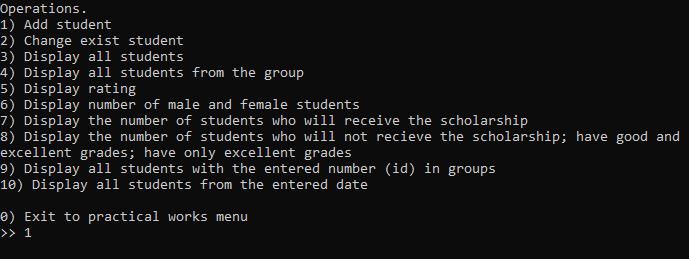
В функции **input** вначале выделяем память для структуры, получая указатель на выделенный блок памяти.

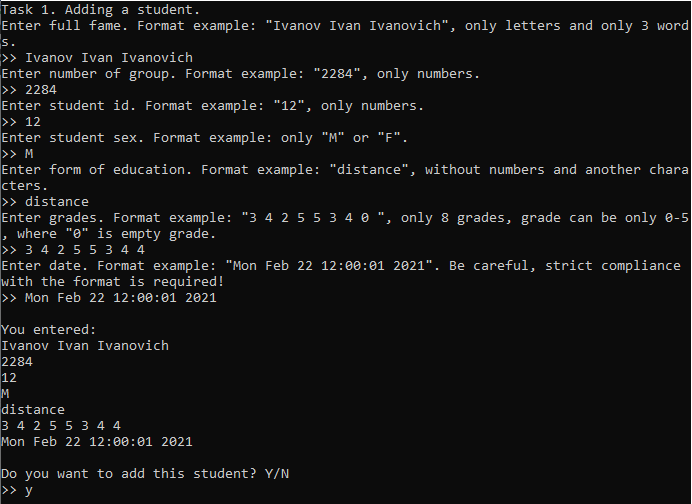
Потом поочередно вводим значения для элементов структуры и возвращаем указатель из функции.

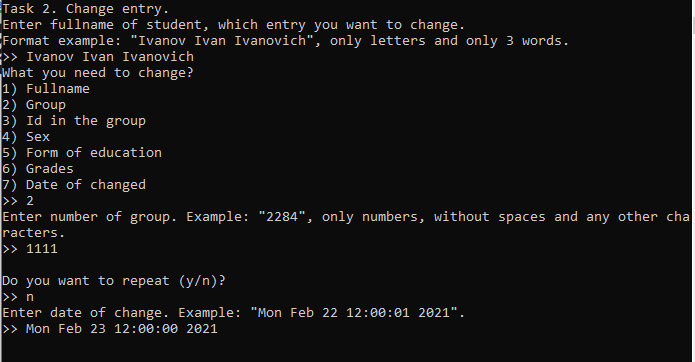
В функции main вызываем функцию **input**, получая из нее указатель. Используя полученный указатель, можно получить значения элементов структуры. И в конце освобождаем выделенную под структуру динамическую память.

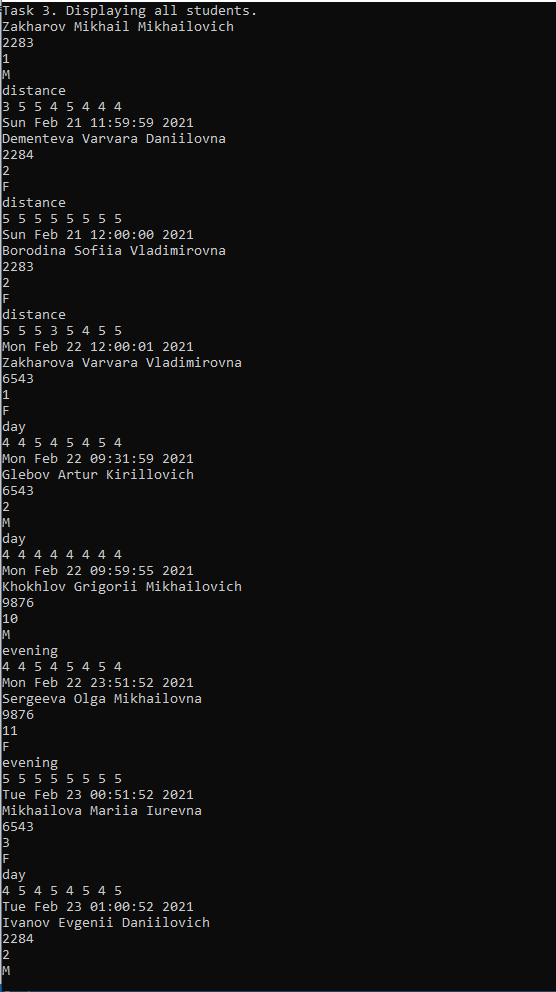
**Экспериментальные результаты.**

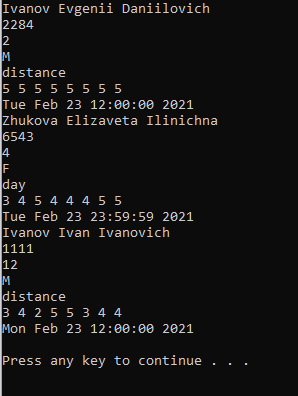


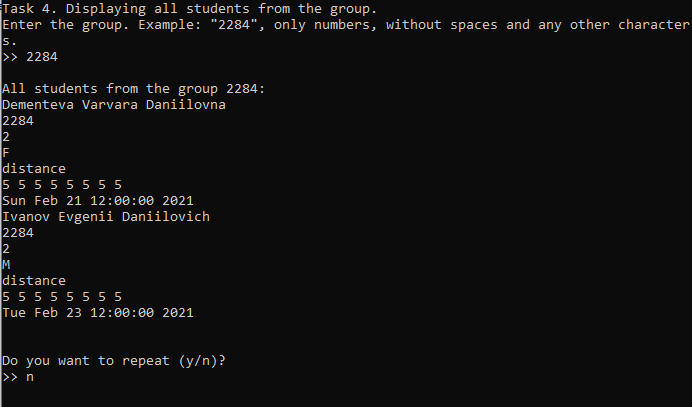


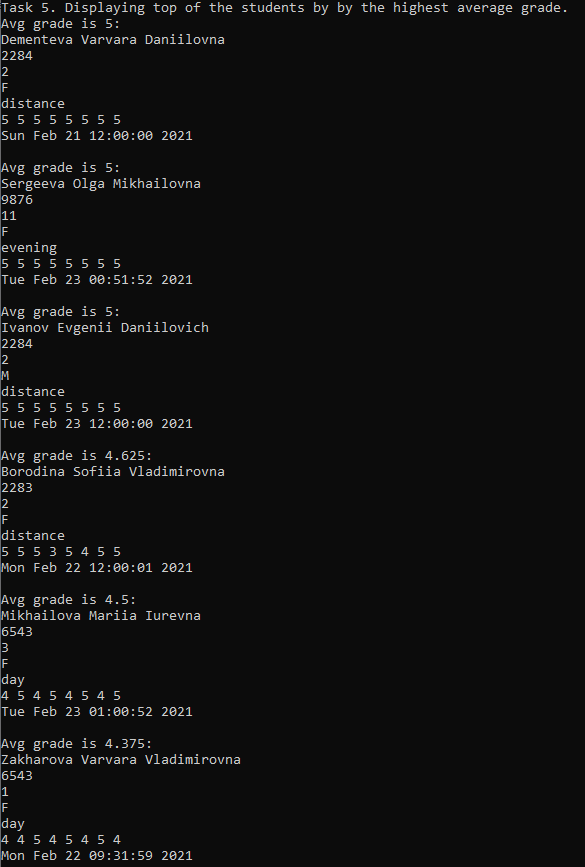


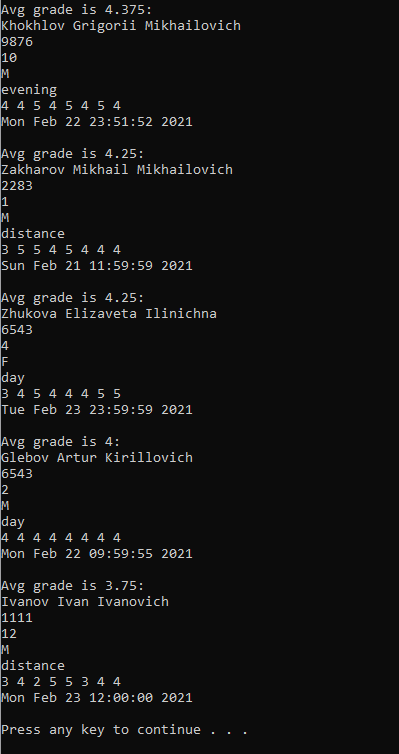


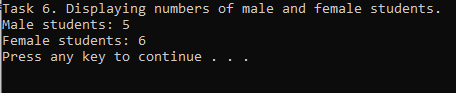


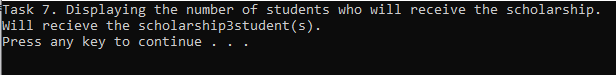


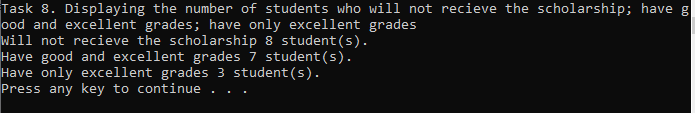


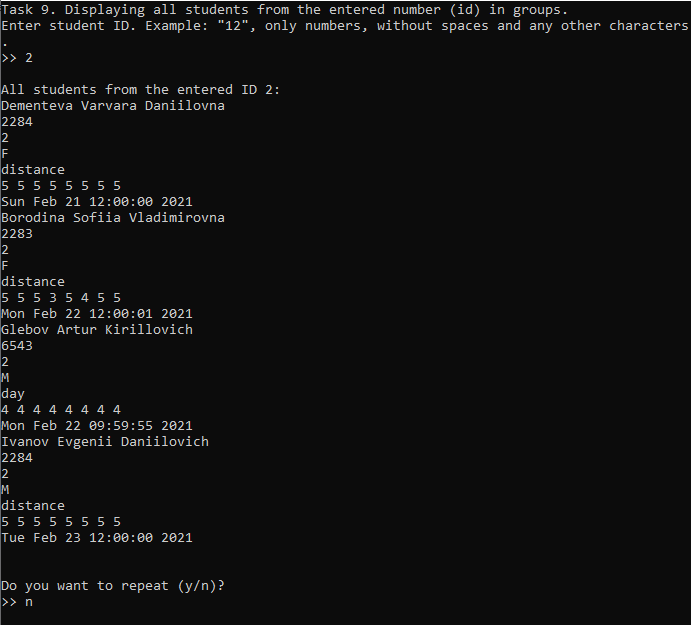












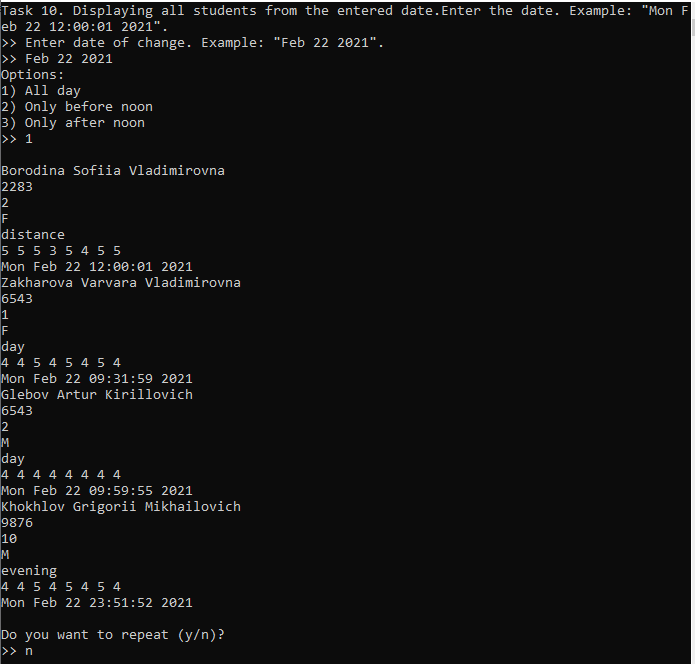


Рисунок 5 – Результат работы программы (полный код программы представлен в приложении А)

Приложение а  
полный код программы

#include <iostream>

#include <iomanip>

#include <math.h>

#include <string>

#include <fstream>

#include <regex>

using namespace std;

const int numOfStudentFields = 7;

struct timeHM {

string hour;

string minute;

string second;

};

struct date {

string dayOfWeek;

string month;

int dayOfMonth;

timeHM time;

int year;

};

struct student {

string fullName;

int group;

int id;

char sex;

string educationForm;

short grades[8];

date changedOn;

};

void practicalWork1();

int countOfRecords(ifstream \*);

student readStudentFromFile(ifstream \*);

student readStudentFromKeyboard();

student \* addStruct(student \*, int \*);

void addStudentsToFile(ofstream\*, student\*, int);

void printStudent(student);

int searchStudent(student \*, string, int);

void printTop(student \*, int);

int countOfMale(student \*, int);

int countOfFemale(student \*, int);

int countOfScholarships(student \*, int);

int countOfGoodStudents(student \*, int);

int countOfExcellentStudents(student \*, int);

void printStudentsFromDate(student \*, date, int);

void printStudentsFromDateBeforeNoon(student \*, date, int);

void printStudentsFromDateAfterNoon(student \*, date, int);

date readFullTimeFromKeyboard();

date readDateFromKeyboard();

bool isFullName(string);

bool isLetter(char);

bool isDay(string);

bool isWeek(string);

void printDate(date);

void printGrades(short \*);

bool choiseNextAction();

int main() {

while (true) {

cout << "Practical works: \n"

<< "\t1) Structures \n"

<< "Enter the number of practical work or enter 0 for close the program... \n>> ";

int input;

cin >> input;

switch (input) {

case 1:

practicalWork1();

break;

default:

goto Exit;

}

MainMenu:

continue;

}

Exit:

cout << "\nClosing the program... \n";

system("pause");

return 0;

}

void practicalWork1() {

system("CLS");

cout << "Solution of task \"Working with structures\". \n\n"

<< "Task 0. Reading text file. \n";

ifstream ifile;

ofstream ofile;

string path;

int numberOfRecords;

student \* students = 0;

do {

cout << "Enter the path to the file. \n"

<< "Only english words in the file and path! Example: C:\\anime\\students.txt \n>> ";

cin.ignore(32767, '\n');

getline(cin, path);

ifile.open(path);

if (!ifile.is\_open()) {

cout << "Error opening file! Please restart the program! \n";

continue;

}

ifstream \*pifile = &ifile;

numberOfRecords = countOfRecords(pifile);

if (!(numberOfRecords % numOfStudentFields == 0)) {

cout << "Error! Invalid number of lines in the file. Must be a multiple " << numOfStudentFields << ".\n";

continue;

}

numberOfRecords /= numOfStudentFields;

ifile.close();

ifile.open(path);

students = new student[numberOfRecords];

try {

for (int i = 0; i < numberOfRecords; i++) {

students[i] = readStudentFromFile(pifile);

}

}

catch (const char\* msg) {

cout << msg;

continue;

}

} while (!ifile.is\_open());

ifile.close();

bool isInMenu = true;

do {

system("CLS");

cout << "Operations. \n"

<< "1) Add student \n"

<< "2) Change exist student \n"

<< "3) Display all students \n"

<< "4) Display all students from the group \n"

<< "5) Display rating \n"

<< "6) Display number of male and female students \n"

<< "7) Display the number of students who will receive the scholarship \n"

<< "8) Display the number of students who will not recieve the scholarship; have good and excellent grades; have only excellent grades \n"

<< "9) Display all students with the entered number (id) in groups \n"

<< "10) Display all students from the entered date \n\n"

<< "0) Exit to practical works menu \n>> ";

int input;

cin >> input;

cout << endl;

switch (input) {

case 1:

do {

system("CLS");

cout << "Task 1. Adding a student. \n";

student newStudent = readStudentFromKeyboard();

cout << "\nYou entered: \n";

printStudent(newStudent);

cout << "\nDo you want to add this student? Y/N \n>> ";

char input;

cin >> input;

if (input == 'Y' || input == 'y') {

ofstream \* pofile = &ofile;

students = addStruct(students, &numberOfRecords);

students[numberOfRecords - 1] = newStudent;

ofile.open(path);

addStudentsToFile(pofile, students, numberOfRecords);

ofile.close();

}

else if (input == 'N' || input == 'n') {

continue;

}

else

cout << "Error! Invalid input. \n";

} while (choiseNextAction());

continue;

case 2:

{

system("CLS");

cout << "Task 2. Change entry. \n";

bool isRecordsChanged = false;

try {

cout << "Enter fullname of student, which entry you want to change. \n"

<< "Format example: \"Ivanov Ivan Ivanovich\", only letters and only 3 words. \n>> ";

string input;

cin.ignore(32767, '\n');

getline(cin, input);

if (isFullName(input)) {

int indexOfStudent = searchStudent(students, input, numberOfRecords);

if (indexOfStudent != -1) {

do {

cout << "What you need to change? \n"

<< "1) Fullname\n"

<< "2) Group\n"

<< "3) Id in the group\n"

<< "4) Sex\n"

<< "5) Form of education\n"

<< "6) Grades\n"

<< "7) Date of changed\n>> ";

int option;

cin >> option;

regex mask;

switch (option) {

case 1:

cout << "Enter fullName. Format example: \"Ivanov Ivan Ivanovich\", only letters and only 3 words. \n>> ";

cin.ignore(32767, '\n');

getline(cin, input);

if (isFullName(input)) { // first option of check

students[indexOfStudent].fullName = input;

isRecordsChanged = true;

cout << "Entry updated! \n";

}

else

cout << "Error! Invalid students FullName. Format example: \"Ivanov Ivan Ivanovich\", only letters and only 3 words. \n";

break;

case 2:

cout << "Enter number of group. Example: \"2284\", only numbers, without spaces and any other characters. \n>> ";

cin.ignore(32767, '\n');

getline(cin, input);

mask = "^[0-9]\*$"; // second option of check

if (regex\_search(input, mask)) {

students[indexOfStudent].group = stoi(input);

isRecordsChanged = true;

cout << "Entry updated! \n";

}

else

cout << "Error! Invalid number of group format. Example: \"2284\", only numbers, without spaces and any other characters. \n";

break;

case 3:

cout << "Enter student ID. Example: \"12\", only numbers, without spaces and any other characters. \n>> ";

cin.ignore(32767, '\n');

getline(cin, input);

if (regex\_search(input, mask)) {

students[indexOfStudent].id = stoi(input);

isRecordsChanged = true;

cout << "Entry updated! \n";

}

else

cout << "Error! Invalid student id format. Example: \"12\", only numbers, without spaces and any other characters. \n";

break;

case 4:

cout << "Enter student sex. Example: only \"M\" or \"F\", without spaces and any other characters. \n>> ";

cin.ignore(32767, '\n');

getline(cin, input);

mask = "^[MF|mf]{1}$";

if (regex\_search(input, mask)) {

students[indexOfStudent].sex = input[0];

isRecordsChanged = true;

cout << "Entry updated! \n";

}

else

cout << "Error! Invalid student sex format. Example: only \"M\" or \"F\", without spaces and any other characters. \n";

break;

case 5:

cout << "Enter student form of education. Example: \"distance\", without numbers, spaces and another characters. \n>> ";

cin.ignore(32767, '\n');

getline(cin, input);

mask = "^[A-Za-z]\*$";

if (regex\_search(input, mask)) {

students[indexOfStudent].educationForm = input;

isRecordsChanged = true;

cout << "Entry updated! \n";

}

else

cout << "Error! Invalid form of education format. Example: \"distance\", without numbers, spaces and another characters. \n";

break;

case 6:

cout << "Enter student grades. Example: \"3 4 2 5 5 3 4 0 \", grade can be only 0-5, where \"0\" is empty grade. \n>> ";

cin.ignore(32767, '\n');

getline(cin, input);

mask = "^[0-5]{1} {1}[0-5]{1} {1}[0-5]{1} {1}[0-5]{1} {1}[0-5]{1} {1}[0-5]{1} {1}[0-5]{1} {1}[0-5]{1}";

if (regex\_search(input, mask)) {

int j = 0;

for (int i = 0; i <= 14; i += 2) {

students[indexOfStudent].grades[j] = (int)input[i] - 48;

j++;

}

isRecordsChanged = true;

cout << "Entry updated! \n";

}

else

cout << "Error! Invalid grades format. Example: \"3 4 2 5 5 3 4 0 \", grade can be only 0-5, where \"0\" is empty grade. \n";

break;

case 7:

students[indexOfStudent].changedOn = readFullTimeFromKeyboard();

isRecordsChanged = true;

cout << "Entry updated! \n";

break;

default:

cout << "Error! Invalid input. \n";

break;

}

} while (choiseNextAction());

if (isRecordsChanged)

students[indexOfStudent].changedOn = readFullTimeFromKeyboard();

cout << "Entry updated! \n";

}

else

throw "Student was not found! \n";

}

else

throw "Error! Invalid students FullName. Format example: \"Ivanov Ivan Ivanovich\", only letters and only 3 words. \n";

}

catch (const char\* msg) {

cout << msg;

continue;

}

if (isRecordsChanged) {

ofstream \* pofile = &ofile;

ofile.open(path);

addStudentsToFile(pofile, students, numberOfRecords);

ofile.close();

}

}

break;

case 3:

system("CLS");

cout << "Task 3. Displaying all students. \n";

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

printStudent(students[i]);

cout << endl;

system("pause");

break;

case 4:

do {

system("CLS");

cout << "Task 4. Displaying all students from the group. \n";

cout << "Enter the group. Example: \"2284\", only numbers, without spaces and any other characters. \n>> ";

int group;

cin >> group;

bool isFound = false;

cout << "\nAll students from the group " << group << ": \n";

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

if (students[i].group == group) {

printStudent(students[i]);

isFound = true;

}

if (isFound)

cout << endl;

else

cout << "Students was not found! \n";

} while (choiseNextAction());

break;

case 5:

system("CLS");

cout << "Task 5. Displaying top of the students by by the highest average grade. \n";

printTop(students, numberOfRecords);

system("pause");

break;

case 6:

system("CLS");

cout << "Task 6. Displaying numbers of male and female students. \n";

cout << "Male students: " << countOfMale(students, numberOfRecords) << endl;

cout << "Female students: " << countOfFemale(students, numberOfRecords) << endl;

system("pause");

break;

case 7:

system("CLS");

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

cout << "Will recieve the scholarship " << countOfScholarships(students, numberOfRecords) << " student(s). \n";

system("pause");

break;

case 8:

system("CLS");

cout << "Task 8. Displaying the number of students who will not recieve the scholarship; have good and excellent grades; have only excellent grades \n";

cout << "Will not recieve the scholarship " << numberOfRecords - countOfScholarships(students, numberOfRecords) << " student(s). \n";

cout << "Have good and excellent grades " << countOfGoodStudents(students, numberOfRecords) << " student(s). \n";

cout << "Have only excellent grades " << countOfExcellentStudents(students, numberOfRecords) << " student(s). \n";

system("pause");

break;

case 9:

do {

system("CLS");

cout << "Task 9. Displaying all students from the entered number (id) in groups. \n";

cout << "Enter student ID. Example: \"12\", only numbers, without spaces and any other characters. \n>> ";

int id;

cin >> id;

bool isFound = false;

cout << "\nAll students from the entered ID " << id << ": \n";

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

if (students[i].id == id) {

printStudent(students[i]);

isFound = true;

}

if (isFound)

cout << endl;

else

cout << "Students was not found! \n";

} while (choiseNextAction());

break;

case 10:

do {

system("CLS");

cout << "Task 10. Displaying all students from the entered date.";

cout << "Enter the date. Example: \"Mon Feb 22 12:00:01 2021\". \n>> ";

date inputDate = readDateFromKeyboard();

cout << "Options: \n"

<< "1) All day \n"

<< "2) Only before noon \n"

<< "3) Only after noon \n>> ";

int input2;

cin >> input2;

cout << endl;

switch (input2) {

case 1:

printStudentsFromDate(students, inputDate, numberOfRecords);

break;

case 2:

printStudentsFromDateBeforeNoon(students, inputDate, numberOfRecords);

break;

case 3:

printStudentsFromDateAfterNoon(students, inputDate, numberOfRecords);

break;

default:

cout << "Error! Invalid input. \n";

break;

}

} while (choiseNextAction());

break;

case 0:

isInMenu = false;

break;

default:

cout << "Error! Invalid input. \n";

break;

}

} while (isInMenu);

}

int countOfRecords(ifstream \* file) {

string t;

int count = 0;

while (getline(\*file, t)) {

count++;

}

return count;

}

student readStudentFromFile(ifstream \* file) {

struct student newStudent;

string input;

getline(\*file, input);

if (isFullName(input)) // first option of check

newStudent.fullName = input;

else

throw "Error! Invalid students FullName. Format example: \"Ivanov Ivan Ivanovich\", only letters and only 3 words. \n";

getline(\*file, input);

regex mask("^[0-9]\*$"); // second option of check

if (regex\_search(input, mask))

newStudent.group = stoi(input);

else

throw "Error! Invalid number of group format. Example: \"2284\", only numbers, without spaces and any other characters. \n";

getline(\*file, input);

if (regex\_search(input, mask))

newStudent.id = stoi(input);

else

throw "Error! Invalid student id format. Example: \"12\", only numbers, without spaces and any other characters. \n";

getline(\*file, input);

mask = "^[MF|mf]{1}$";

if (regex\_search(input, mask))

newStudent.sex = input[0];

else

throw "Error! Invalid student sex format. Example: only \"M\" or \"F\", without spaces and any other characters. \n";

getline(\*file, input);

mask = "^[A-Za-z]\*$";

if (regex\_search(input, mask))

newStudent.educationForm = input;

else

throw "Error! Invalid form of education format. Example: \"distance\", without numbers, spaces and another characters. \n";

getline(\*file, input);

mask = "^[0-5]{1} {1}[0-5]{1} {1}[0-5]{1} {1}[0-5]{1} {1}[0-5]{1} {1}[0-5]{1} {1}[0-5]{1} {1}[0-5]{1}";

if (regex\_search(input, mask)) {

int j = 0;

for (int i = 0; i <= 14; i += 2) {

newStudent.grades[j] = (int)input[i] - 48;

j++;

}

}

else

throw "Error! Invalid grades format. Example: \"3 4 2 5 5 3 4 0 \", grade can be only 0-5, where \"0\" is empty grade. \n";

getline(\*file, input);

mask = "^[A-Za-z]{3} {1}[A-Za-z]{3} {1}[0-9]{2} {1}[0-9]{2}:{1}[0-9]{2}:{1}[0-9]{2} {1}[0-9]{4}$";

if (!regex\_search(input, mask))

throw "Error! Invalid date format. Example: \"Mon Feb 22 12:00:01 2021\". \n";

string sub = input.substr(0, 3); // Here and below individual cases of incorrect date are checked

if (isDay(sub))

newStudent.changedOn.dayOfWeek = sub;

else

throw "Error! Invalid day format. Example: \"Mon\" \"Tue\", \"Wed\" etc. \n";

sub = input.substr(4, 3);

if (isWeek(sub))

newStudent.changedOn.month = sub;

else

throw "Error! Invalid month format. Example: \"Jan\" \"Feb\", \"Mar\" etc. \n";

sub = input.substr(8, 2);

int intForCheck = stoi(sub);

if (intForCheck >= 1 && intForCheck <= 31)

newStudent.changedOn.dayOfMonth = intForCheck;

else

throw "Error! Invalid day of month format. Example: \"18\", 1-31. \n";

sub = input.substr(11, 2);

intForCheck = stoi(sub);

if (intForCheck >= 0 && intForCheck <= 24)

newStudent.changedOn.time.hour = sub;

else

throw "Error! Invalid hour format. Example: \"13\", 00-23. \n";

sub = input.substr(14, 2);

intForCheck = stoi(sub);

if (intForCheck >= 0 && intForCheck <= 59)

newStudent.changedOn.time.minute = sub;

else

throw "Error! Invalid minute format. Example: \"29\", 00-59. \n";

sub = input.substr(17, 2);

intForCheck = stoi(sub);

if (intForCheck >= 0 && intForCheck <= 59)

newStudent.changedOn.time.second = sub;

else

throw "Error! Invalid second format. Example: \"29\", 00-59. \n";

newStudent.changedOn.year = stoi(input.substr(20, 4));

return newStudent;

}

student readStudentFromKeyboard() {

struct student newStudent;

string input;

bool inputIsCorrect = false;

cin.ignore(32767, '\n');

do {

cout << "Enter full fame. Format example: \"Ivanov Ivan Ivanovich\", only letters and only 3 words. \n>> ";

getline(cin, input);

if (isFullName(input)) { // first option of check

newStudent.fullName = input;

inputIsCorrect = true;

}

else {

inputIsCorrect = false;

cout << "Error! Invalid students full fame. Format example: \"Ivanov Ivan Ivanovich\", only letters and only 3 words. \n";

}

} while (!inputIsCorrect);

do {

cout << "Enter number of group. Format example: \"2284\", only numbers. \n>> ";

getline(cin, input);

regex mask("^[0-9]\*$"); // second option of check

if (regex\_search(input, mask)) {

newStudent.group = stoi(input);

inputIsCorrect = true;

}

else {

inputIsCorrect = false;

cout << "Error! Invalid number of group format. Example: \"2284\", only numbers. \n";

}

} while (!inputIsCorrect);

do {

cout << "Enter student id. Format example: \"12\", only numbers. \n>> ";

getline(cin, input);

regex mask("^[0-9]\*$");

if (regex\_search(input, mask)) {

newStudent.id = stoi(input);

inputIsCorrect = true;

}

else {

inputIsCorrect = false;

cout << "Error! Invalid student id format. Example: \"12\", only numbers. \n";

}

} while (!inputIsCorrect);

do {

cout << "Enter student sex. Format example: only \"M\" or \"F\". \n>> ";

getline(cin, input);

regex mask("^[MF|mf]{1}$");

if (regex\_search(input, mask)) {

newStudent.sex = input[0];

inputIsCorrect = true;

}

else {

inputIsCorrect = false;

cout << "Error! Invalid student sex format. Example: only \"M\" or \"F\". \n";

}

} while (!inputIsCorrect);

do {

cout << "Enter form of education. Format example: \"distance\", without numbers and another characters. \n>> ";

getline(cin, input);

regex mask("^[A-Za-z]\*$");

if (regex\_search(input, mask)) {

newStudent.educationForm = input;

inputIsCorrect = true;

}

else {

inputIsCorrect = false;

cout << "Error! Invalid form of education format. Example: \"distance\", without numbers and another characters. \n";

}

} while (!inputIsCorrect);

do {

cout << "Enter grades. Format example: \"3 4 2 5 5 3 4 0 \", only 8 grades, grade can be only 0-5, where \"0\" is empty grade. \n>> ";

getline(cin, input);

regex mask("^[0-5]{1} {1}[0-5]{1} {1}[0-5]{1} {1}[0-5]{1} {1}[0-5]{1} {1}[0-5]{1} {1}[0-5]{1} {1}[0-5]{1}");

if (regex\_search(input, mask)) {

for (int i = 0, j = 0; i <= 14; i += 2) {

newStudent.grades[j] = (int)input[i] - 48;

j++;

}

inputIsCorrect = true;

}

else {

inputIsCorrect = false;

cout << "Error! Invalid grades format. Example: \"3 4 2 5 5 3 4 0 \", only 8 grades, grade can be only 0-5, where \"0\" is empty grade. \n";

}

} while (!inputIsCorrect);

do {

try {

cout << "Enter date. Format example: \"Mon Feb 22 12:00:01 2021\". Be careful, strict compliance with the format is required! \n>> ";

getline(cin, input);

regex mask("^[A-Za-z]{3} {1}[A-Za-z]{3} {1}[0-9]{2} {1}[0-9]{2}:{1}[0-9]{2}:{1}[0-9]{2} {1}[0-9]{4}$");

if (!regex\_search(input, mask))

throw "Error! Invalid date format. Example: \"Mon Feb 22 12:00:01 2021\". \n";

string sub = input.substr(0, 3); // Here and below individual cases of incorrect date are checked

if (isDay(sub))

newStudent.changedOn.dayOfWeek = sub;

else

throw "Error! Invalid day format. Example: \"Mon\" \"Tue\", \"Wed\" etc. \n";

sub = input.substr(4, 3);

if (isWeek(sub))

newStudent.changedOn.month = sub;

else

throw "Error! Invalid month format. Example: \"Jan\" \"Feb\", \"Mar\" etc. \n";

sub = input.substr(8, 2);

int intForCheck = stoi(sub);

if (intForCheck >= 1 && intForCheck <= 31)

newStudent.changedOn.dayOfMonth = intForCheck;

else

throw "Error! Invalid day of month format. Example: \"18\", 1-31. \n";

sub = input.substr(11, 2);

intForCheck = stoi(sub);

if (intForCheck >= 0 && intForCheck <= 24)

newStudent.changedOn.time.hour = sub;

else

throw "Error! Invalid hour format. Example: \"13\", 00-23. \n";

sub = input.substr(14, 2);

intForCheck = stoi(sub);

if (intForCheck >= 0 && intForCheck <= 59)

newStudent.changedOn.time.minute = sub;

else

throw "Error! Invalid minute format. Example: \"29\", 00-59. \n";

sub = input.substr(17, 2);

intForCheck = stoi(sub);

if (intForCheck >= 0 && intForCheck <= 59)

newStudent.changedOn.time.second = sub;

else

throw "Error! Invalid second format. Example: \"29\", 00-59. \n";

newStudent.changedOn.year = stoi(input.substr(20, 4));

inputIsCorrect = true;

}

catch (const char\* msg) {

cout << msg;

continue;

}

} while (!inputIsCorrect);

return newStudent;

}

student \* addStruct(student \* obj, int \* size) {

student \* tempObj = new student[\*size + 1];

for (int i = 0; i < \*size; i++) {

tempObj[i] = obj[i];

}

delete[] obj;

obj = tempObj;

\*size = \*size + 1;

return obj;

}

void addStudentsToFile(ofstream \* file, student \* students, int size) {

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

\*file << students[i].fullName << endl

<< students[i].group << endl

<< students[i].id << endl

<< students[i].sex << endl

<< students[i].educationForm << endl

<< students[i].grades[0] << ' ' << students[i].grades[1] << ' ' << students[i].grades[2] << ' ' << students[i].grades[3] << ' ' << students[i].grades[4] << ' ' << students[i].grades[5] << ' ' << students[i].grades[6] << ' ' << students[i].grades[7] << " \n"

<< students[i].changedOn.dayOfWeek << ' ' << students[i].changedOn.month << ' ' << students[i].changedOn.dayOfMonth << ' ' << students[i].changedOn.time.hour << ':' << students[i].changedOn.time.minute << ':' << students[i].changedOn.time.second << ' ' << students[i].changedOn.year << endl;

}

}

void printStudent(student input) {

cout << input.fullName << endl;

cout << input.group << endl;

cout << input.id << endl;

cout << input.sex << endl;

cout << input.educationForm << endl;

printGrades(input.grades);

printDate(input.changedOn);

}

int searchStudent(student \* students, string input, int size) {

int result = -1;

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

if (students[i].fullName == input) {

result = i;

break;

}

return result;

}

void printTop(student \* students, int size) {

float \* avgGrades = new float[size];

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

avgGrades[i] = (students[i].grades[0] + students[i].grades[1] + students[i].grades[2] + students[i].grades[3] + students[i].grades[4] + students[i].grades[5] + students[i].grades[6] + students[i].grades[7]) / (float)8;

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

float maxGrade = 0;

int indexOfMaxGrade = -1;

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

if (avgGrades[i] > 0 && avgGrades[i] > maxGrade) {

maxGrade = avgGrades[i];

indexOfMaxGrade = i;

}

cout << "Avg grade is " << avgGrades[indexOfMaxGrade] << ": \n";

printStudent(students[indexOfMaxGrade]);

avgGrades[indexOfMaxGrade] = 0;

cout << endl;

}

}

int countOfMale(student \* students, int size) {

int result = 0;

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

if (students[i].sex == 'M' || students[i].sex == 'm')

result++;

return result;

}

int countOfFemale(student \* students, int size) {

int result = 0;

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

if (students[i].sex == 'F' || students[i].sex == 'f')

result++;

return result;

}

int countOfScholarships(student \* students, int size) {

int result = 0;

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

if (students[i].educationForm == "day"

&& students[i].grades[0] > 3

&& students[i].grades[1] > 3

&& students[i].grades[2] > 3

&& students[i].grades[3] > 3

&& students[i].grades[4] > 3

&& students[i].grades[5] > 3

&& students[i].grades[6] > 3

&& students[i].grades[7] > 3)

result++;

return result;

}

int countOfGoodStudents(student \* students, int size) {

int result = 0;

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

if (students[i].grades[0] > 3

&& students[i].grades[1] > 3

&& students[i].grades[2] > 3

&& students[i].grades[3] > 3

&& students[i].grades[4] > 3

&& students[i].grades[5] > 3

&& students[i].grades[6] > 3

&& students[i].grades[7] > 3)

result++;

return result;

}

int countOfExcellentStudents(student \* students, int size) {

int result = 0;

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

if (students[i].grades[0] > 4

&& students[i].grades[1] > 4

&& students[i].grades[2] > 4

&& students[i].grades[3] > 4

&& students[i].grades[4] > 4

&& students[i].grades[5] > 4

&& students[i].grades[6] > 4

&& students[i].grades[7] > 4)

result++;

return result;

}

void printStudentsFromDate(student \* students, date input, int size) {

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

if (students[i].changedOn.month == input.month

&& students[i].changedOn.dayOfMonth == input.dayOfMonth

&& students[i].changedOn.year == input.year)

printStudent(students[i]);

}

void printStudentsFromDateBeforeNoon(student \* students, date input, int size) {

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

if (students[i].changedOn.month == input.month

&& students[i].changedOn.dayOfMonth == input.dayOfMonth

&& students[i].changedOn.year == input.year

&& stoi(students[i].changedOn.time.hour) < 12)

printStudent(students[i]);

}

void printStudentsFromDateAfterNoon(student \* students, date input, int size) {

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

if (students[i].changedOn.month == input.month

&& students[i].changedOn.dayOfMonth == input.dayOfMonth

&& students[i].changedOn.year == input.year

&& stoi(students[i].changedOn.time.hour) > 12)

printStudent(students[i]);

}

date readFullTimeFromKeyboard() {

bool isCorrected = false;

date tempDate;

do {

cout << "Enter date of change. Example: \"Mon Feb 22 12:00:01 2021\". \n>> ";

string input;

cin.ignore(32767, '\n');

getline(cin, input);

regex mask("^[A-Za-z]{3} {1}[A-Za-z]{3} {1}[0-9]{2} {1}[0-9]{2}:{1}[0-9]{2}:{1}[0-9]{2} {1}[0-9]{4}$");

if (!regex\_search(input, mask)) {

cout << "Error! Invalid date format. Example: \"Mon Feb 22 12:00:01 2021\". \n";

continue;

}

string sub = input.substr(0, 3); // Here and below individual cases of incorrect date are checked

if (isDay(sub))

tempDate.dayOfWeek = sub;

else {

cout << "Error! Invalid day format. Example: \"Mon\" \"Tue\", \"Wed\" etc. \n";

continue;

}

sub = input.substr(4, 3);

if (isWeek(sub))

tempDate.month = sub;

else {

cout << "Error! Invalid month format. Example: \"Jan\" \"Feb\", \"Mar\" etc. \n";

continue;

}

sub = input.substr(8, 2);

int intForCheck = stoi(sub);

if (intForCheck >= 1 && intForCheck <= 31)

tempDate.dayOfMonth = intForCheck;

else {

cout << "Error! Invalid day of month format. Example: \"18\", 1-31. \n";

continue;

}

sub = input.substr(11, 2);

intForCheck = stoi(sub);

if (intForCheck >= 0 && intForCheck <= 24)

tempDate.time.hour = sub;

else {

cout << "Error! Invalid hour format. Example: \"13\", 00-23. \n";

continue;

}

sub = input.substr(14, 2);

intForCheck = stoi(sub);

if (intForCheck >= 0 && intForCheck <= 59)

tempDate.time.minute = sub;

else {

cout << "Error! Invalid minute format. Example: \"29\", 00-59. \n";

continue;

}

sub = input.substr(17, 2);

intForCheck = stoi(sub);

if (intForCheck >= 0 && intForCheck <= 59)

tempDate.time.second = sub;

else {

cout << "Error! Invalid second format. Example: \"29\", 00-59. \n";

continue;

}

tempDate.year = stoi(input.substr(20, 4));

isCorrected = true;

} while (!isCorrected);

return tempDate;

}

date readDateFromKeyboard() {

bool isCorrected = false;

date tempDate;

do {

cout << "Enter date of change. Example: \"Feb 22 2021\". \n>> ";

string input;

cin.ignore(32767, '\n');

getline(cin, input);

regex mask("^[A-Za-z]{3} {1}[0-9]{2} {1}[0-9]{4}$");

if (!regex\_search(input, mask)) {

cout << "Error! Invalid date format. Example: \"Feb 22 2021\". \n";

continue;

}

string sub = input.substr(0, 3);

if (isWeek(sub))

tempDate.month = sub;

else {

cout << "Error! Invalid month format. Example: \"Jan\" \"Feb\", \"Mar\" etc. \n";

continue;

}

sub = input.substr(4, 2);

int intForCheck = stoi(sub);

if (intForCheck >= 1 && intForCheck <= 31)

tempDate.dayOfMonth = intForCheck;

else {

cout << "Error! Invalid day of month format. Example: \"18\", 1-31. \n";

continue;

}

tempDate.year = stoi(input.substr(7, 4));

isCorrected = true;

} while (!isCorrected);

return tempDate;

}

bool isFullName(string str) {

int countOfSpaces = 0;

int length = str.length();

for (int i = 0; i < length; i++) {

if (isLetter(str[i]))

continue;

else if ((str[i] == ' ') && (countOfSpaces < 2))

countOfSpaces++;

else {

return false;

}

}

return true;

}

bool isLetter(char c) {

if (

((c >= 65) && (c <= 90))

|| ((c >= 97) && (c <= 122))

)

return true;

else

return false;

}

bool isDay(string str) {

if (str == "Mon" || str == "mon"

|| str == "Tue" || str == "tue"

|| str == "Wed" || str == "wed"

|| str == "Thu" || str == "thu"

|| str == "Fri" || str == "fri"

|| str == "Sat" || str == "sat"

|| str == "Sun" || str == "sun"

)

return true;

else

return false;

}

bool isWeek(string str) {

if (str == "Jan" || str == "jan"

|| str == "Feb" || str == "feb"

|| str == "Mar" || str == "mar"

|| str == "Apr" || str == "apr"

|| str == "May" || str == "may"

|| str == "Jun" || str == "jun"

|| str == "Jul" || str == "jul"

|| str == "Aug" || str == "aug"

|| str == "Sep" || str == "sep"

|| str == "Oct" || str == "oct"

|| str == "Nov" || str == "nov"

|| str == "Dec" || str == "dec"

)

return true;

else

return false;

}

void printDate(date input) {

cout << input.dayOfWeek << ' ' << input.month << ' ' << input.dayOfMonth << ' ' << input.time.hour << ':' << input.time.minute << ':' << input.time.second << ' ' << input.year << endl;

}

void printGrades(short \* input) {

cout << input[0] << ' ' << input[1] << ' ' << input[2] << ' ' << input[3] << ' ' << input[4] << ' ' << input[5] << ' ' << input[6] << ' ' << input[7] << ' ' << endl;

}

bool choiseNextAction() {

cout << "\nDo you want to repeat (y/n)?" << "\n>> ";

char input;

while (true) {

cin >> input;

if (input == 'y' || input == 'Y')

return true;

else if (input == 'n' || input == 'N')

return false;

else

cout << "Wrong input! Try again...\n>> ";

}

}