******

***Факултет по изчислителна техника и автоматизация***

***Катедра: СИТ***

***Дисциплина: Базово програмиране***

***Семестриална домашна работа***

**Изготвил: Анатоли Яворов Димов**

**Фак. Номер: 20621662**

***Условие на задачата***

**Да се напише компютърна програма, реализираща информационна система, която поддържа студентска група  
(факултетен номер, Име, Списък с дисциплини, пол, възраст, статус). Програмата съхранява и обработва данни  
за студентите в групата. Максималния брой студенти в една група е 30. Всеки студент има 5 оценки по  
дисциплините (Име, оценка) от 1 семестър.**  
Базова задача – Сложност нискаA. Меню за избор на функциите в програмата. (7 седмица)  
B. Добавяне на студенти в група (7-8 седмица)  
a. Добавяне на един студент в групата;  
b. Добавяне на списък от студенти. Въвежда се цяло число n и след него n на брой студенти  
C. Извеждане на всички студенти на екрана (8-9 седмица)  
a. Извеждане на студенти с минимална оценка по БП  
b. Извеждане на студенти от дадена група  
Допълнение Първо – Сложност средна (+ Базова задача)D. Въвеждане на оценки на студент: (9-10 седмица)  
a. Ако присъства студент с въведен факултетен номер, се разрешава добавяне на оценки за всички  
дисциплини (преди да се въведе оценка се пита дали да се добави, ако вече има оценка по тази  
дисциплина се преминава към следващата);  
b. Ако студента не присъства в групата трябва да се изведе пояснително съобщение;  
c. Ако всички оценки на студента са попълнени трябва да се изведе подходящо съобщение  
Допълнение Второ – Сложност висока (+ Базова задача + Допълнение Първо)E. Промяна на студентите в групата (10-11 седмица)  
a. Въвежда се името на студент и се променя статуса му на завършил.  
b. Въвежда се факултетен номер на студен и се позволява промяна на оценките му (преди да се  
въведе оценка се пита дали да се промени оценката по тази дисциплина).  
Допълнение Трето – Сложност висока (+ Базова задача + Допълнение Първо + Допълнение Второ)F. Справки за студентите в групата с под меню (11-12 седмица)  
a. Извеждане на студентите в подреден ред по:  
i. Името на студент;  
ii. По възраст на студент.  
b. Извеждане на най-младите студенти по пол (Мъж, Жена)  
c. Извеждане на студента с най-висок среден успех  
G. Данните в програмата да могат да се запазват във файл между две стартирания на програмата.  
Допълнение Четвърто – (за допълнителни точки)H. Допълнителни условия:  
a. За точка B при добавянето на един студент да се даде възможност, да се избере дали да се въведе  
нов или да прекъсне въвеждането.  
b. За точка C да се реализира изписването по 5 студента на страница. Пример: В групата има 30  
студента. Извеждат се 5 студента избира се следваща страница извеждат се следващите 5  
студента и т.н.  
c. За точка G да се записват данните в двоичен файл

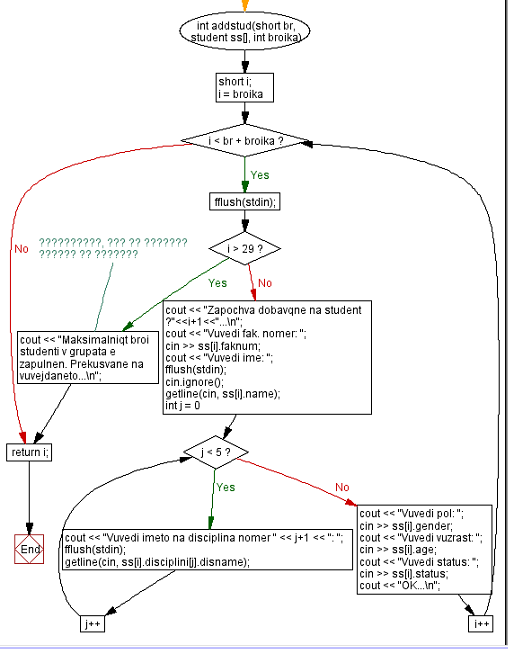
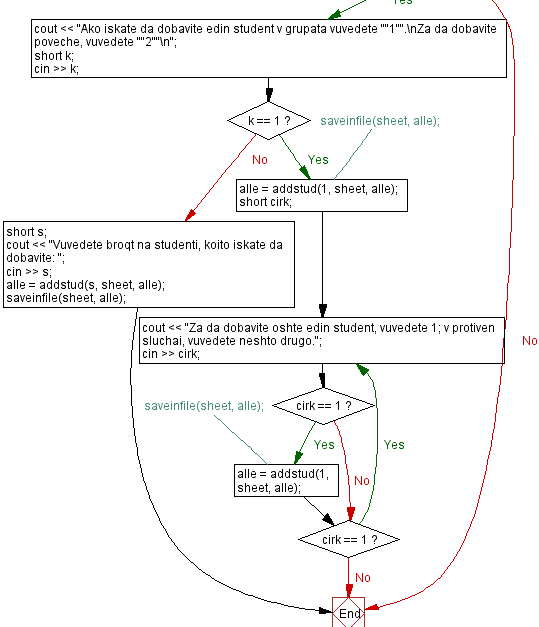
***Анализ на решението***

Информационната система е масив от структури. Основната структура съдържа променливи, повечето от тип стринг, една от тип short – за възрастта и масив от структура за дисциплини, изградена от име на дисциплината и оценка по дисциплината.

***А)*** Менюто е реализирано в main()-а на програмата. Използват се текстови инструкции, променлива - a, която се въвежда и от нея зависи към коя от точките от **B** до **G** ще се продължи. Подменютата или функциите, които ще се изпълнят зависят от стойността на променливата a, проверяват се случаите за стойности на a чрез множество от if-ове.

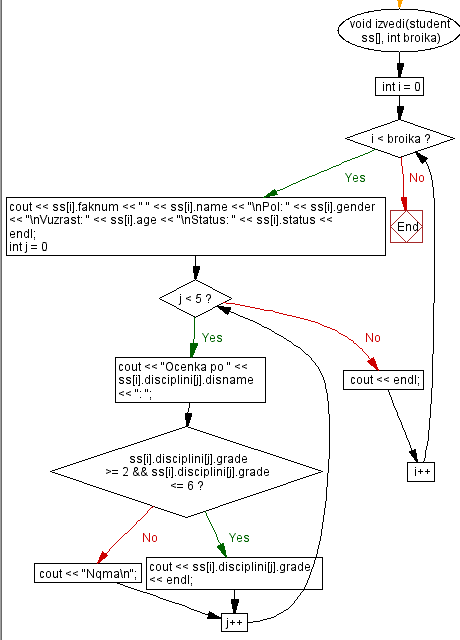
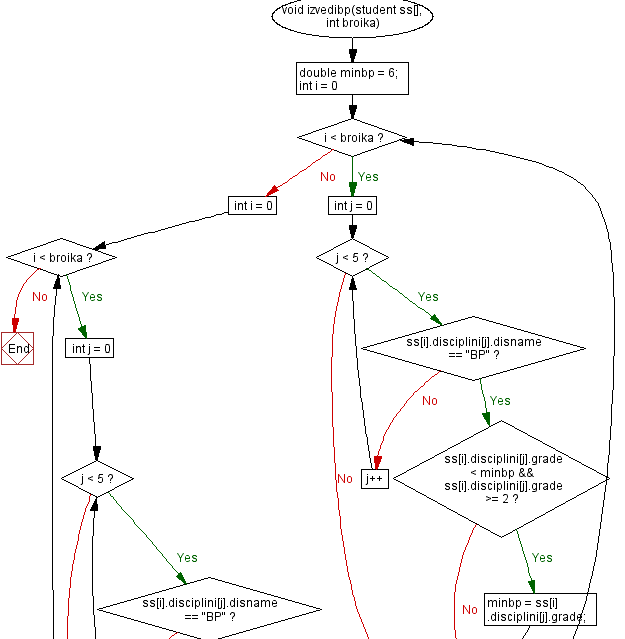
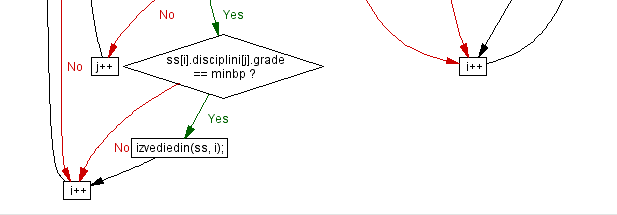
***B)*** При а=1 се започва изпълняването на B) условието. Реализира се подменю, от което се избира дали да се добави един студент или множество от студенти. И в двата случая се използва обща функция тип int, която приема за параметри броят на студентите, които ще бъдат добавени към масива sheet; самият масив за да бъде обработен и общият брой на елементите в масива. Връща обновеният общ брой на елементите в масива. Основата на функцията е цикъл for, който се върти от стойността на броя на елементите в масива, до същата+броят, който ще бъде добавен, за да се записва всичко в правилна клетка на масива. Започва се с проверка дали не се надвишава общият брой на студентите, в структурата. Ако е на път да бъде надвишен се извежда, че се прекъсва въвеждането и цикълът се спира.

Ако цикълът не прекъсне се започва поетапно добавяне на студент в клетка i на масива. Използва се основно cin, но при името на дисциплината и името на студеента се ползва getline, понеже могат да се състоят от повече от една дума и ще са необходими интервали. Функцията връща стойността на I като новата стойност на броят на елементите в масива.

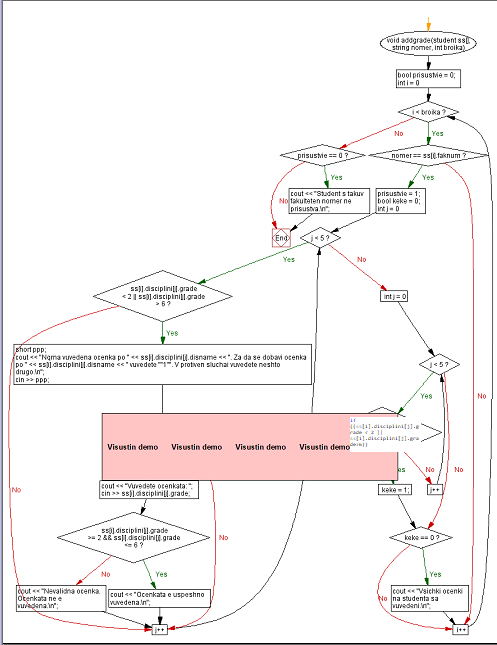


***C)*** Използвани са две отделни, но подобни функции за реализирането на подточките. Извеждането в първата се състои с void функция, приемаща масивът и броят на елементите в него. Завърта се фор от 0 до броят на елементите и се извеждат поетапно променливите от структурата, като за оценките има условие да са валидни.

При втората подточка се ползва друга функция войд, подават се същите променливи. Обхождат се всички студенти със същия фор и се открива минималната оценка по БП чрез сравнения. След това се повтаря нов фор, който извежда всички студенти с оценка, равна на минималната по БП.



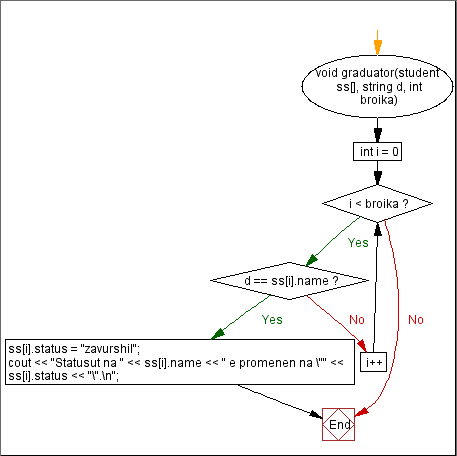
***D)*** Въвежда се факултетен номер – стринг, който се подава към функция войд, заедно с масива и бр. на ел. на масива. Функцията изпълнява и трите подточки на точката от задачата. Използва се булева променлива за да се следи присъствието на факултетния номер в групата със студенти. Започва обхождане на всичките студенти. Ако номерът, подаден към функцията, и номерът на някой от студентите съвпадат:

- булевата променлива взима стойност 1

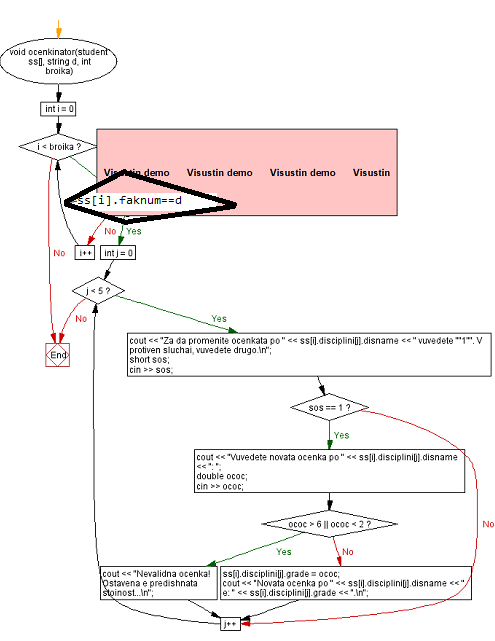
- започват се обхождане на дисциплините на студента. Ако няма валидна стойност за оценка по дисциплина(не е въведена/ не е въведена правилно) се дава избор за въвеждане на оценка по предмета.

Започва още едно обхождане на дисциплините за да се провери дали всички оценки са въведени. Ако има невъведена оценка, друг булев оператор става 1.

След края на цикъла се проверява – ако последният булев оператор е 0, да се изведе съобщение, показващо че всичките оценки на студента са въведени.

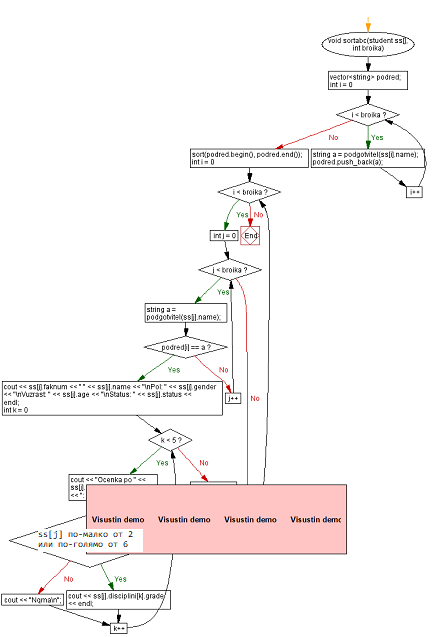
***Е)***

***а.***

***б.***

***F)***

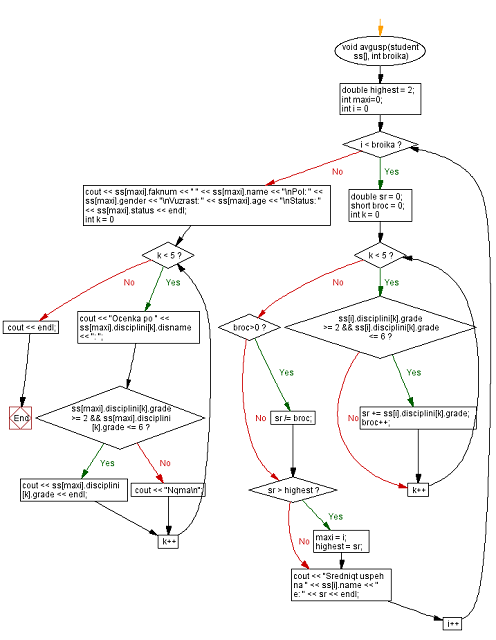
***a.*** Записват се всичките „преработени“ имена на студенти във вектор. Преработените имена са стрингове, съдържащи имената на студентите без интервали и само с малки букви, за да не стане грешка при азбучното им сортиране. Векторът се сортира. След това се използва троен фор. Обхожда се всеки елемент от вектора и се търси името на чий студент принадлежи. Извеждат се данните за студента.



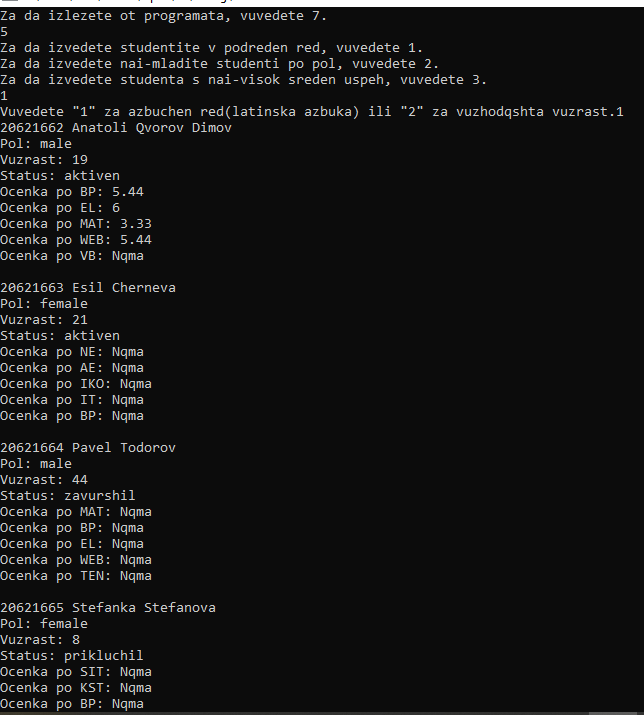
За втората под-подточка се използва подобен алгоритъм.

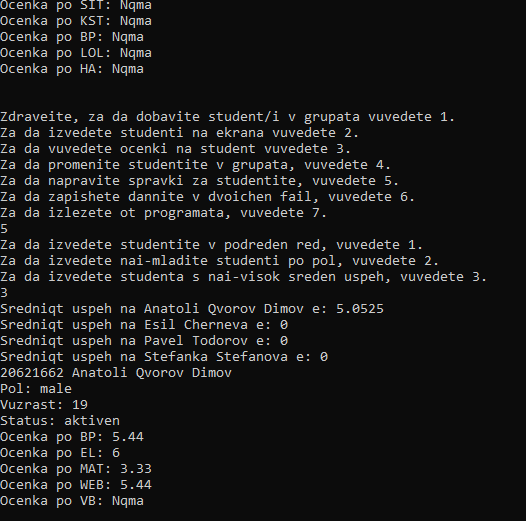
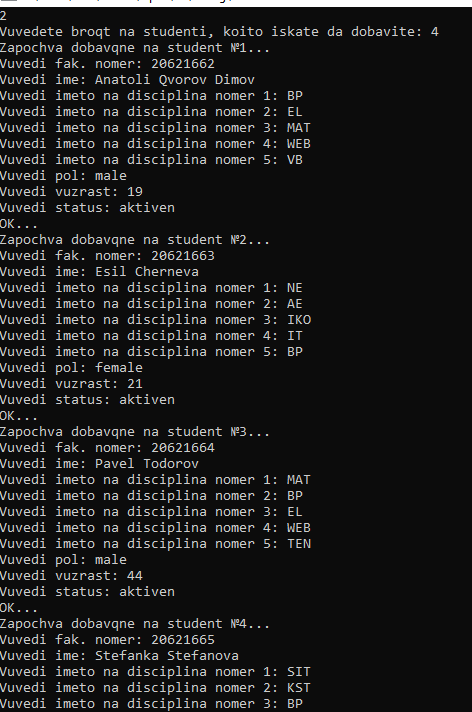
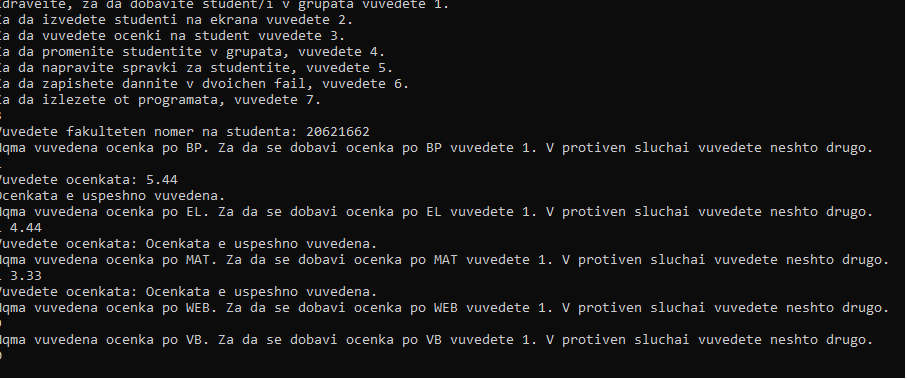
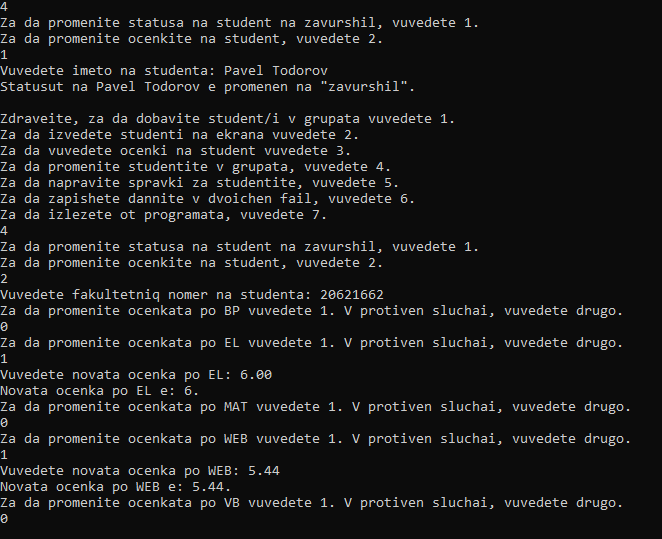
***b.*** Създават се два вектора – единият съдържа възрастите на мъжете, а другият – на жените. Сортират се възходящо. Обхождат се всички студенти, проверява се полът им и спрямо пола се проверява, дали възрастта им е равна на минималната(първият елемент в съответния вектор). Ако е така се извеждат данните за студент/ката.

***c.*** Обхождат се дисциплините на всички студенти и се пресмята средния им успех. Ако се открие по-висок от досегашния най-висок среден успех, запазваме позицията на студента и назначаваме нов макс. среден успех. Когато обходим всички студенти и сме открили позицията в масива на този с най-висок успех, извеждаме данните му.



***G.*** Оставих стандартни функции за запазване и четене на структура във\със двоичен файл. Но винаги при второто повикване на функцията за четене, цялата програма просто crash-ва. Отделих доста време, но не можах да диагностицирам и да оправя проблемът, затова не съм изпълнил условието.

***Примери за входни и изходни данни***



***Код на програмата***

***https://pastebin.com/BbDaG1dD***

#include<iostream>

#include<algorithm>

#include<string>

#include<cstring>

#include<cmath>

#include<cstdlib>

#include<vector>

#include<cctype>

#include<iomanip>

#include<fstream>

#include<cmath>

#define endl "\n"

using namespace std;

const string filename = "Students.dat";

struct Disc //структура, съдържааща име наа дисциплина и оценката по нея

{

string disname;

double grade;

};

struct student //основна структура - студент. съдържа всички характеристики за студента

{

string faknum; //фак.номер

string name; //име

Disc disciplini[6];//дисциплини с имена и оценки

string gender;

short age;

string status;

};

fstream fp; //файлов поток

void saveinfile(student sheet[], int n) //функция за запазване на структурата във файл. не е използвана по-нататък в програмата, понеже има проблеми с файловата обработка, които не успях да диагностицирам

{

fp.open(filename, ios::binary | ios::out); //отваря се двоичен файл, декларира се, че ще се "пише" във файла

if (!fp) { cout << "\n Error in file \n"; exit(1); } //съобщение за грешка, ако не се отвори правилно

fp.write((char\*)sheet, sizeof(student)\*n); //стандартен начин за записване на масив от структури

fp.close();

}

// n е броят на елементите в масива

int loadfile(student sheet[]) //функция за прочитане на информацията от двоичен файл

{

long pos; int n = 0, i; student b;

fp.open(filename, ios::binary | ios::in);

if (!fp) { cout << "\n file not exist\n"; return n; }

fp.seekg(0l, ios::end);

pos = fp.tellg(); //променливата ни дава "стойността" на края на файла -

fp.close();

n = pos / (sizeof(student)); //използваме я за да разберем от колко елементи се състои масивът във файла

fp.open(filename, ios::binary | ios::in);

if (!fp) { cout << "\n Error in file \n"; exit(1); }

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

{

fp.read((char\*)&b, sizeof(student)); //четем структурите една по една, записваме ги в b, а b записваме в sheet - основният ни масаив.

sheet[i] = b;

}

fp.close();

return n; //връщаме броят на елементите в масива

}

int addstud(short br, student ss[], int broika) //добавяне на елементи в масива

// подават се колко студенти ще бъдат добавени, основният масив и общият брой на елементите в него

{

short i;

for (i = broika; i < br + broika; i++) //цикълът за въвеждане започва от броя на елементите до същият+колко иска потребителят да добави

// за да се записва всичко в правилна клетка на масиваа

{

fflush(stdin);

if (i > 29) {

cout << "Maksimalniqt broi studenti v grupata e zapulnen. Prekusvane na vuvejdaneto...\n"; //прекъсване, ако се надвиши лимита на групата

break;

}

cout << "Zapochva dobavqne na student №"<<i+1<<"...\n";

cout << "Vuvedi fak. nomer: "; cin >> ss[i].faknum;

cout << "Vuvedi ime: "; fflush(stdin); cin.ignore(); getline(cin, ss[i].name);

for (int j = 0; j < 5; j++)

{

cout << "Vuvedi imeto na disciplina nomer " << j+1 << ": ";

fflush(stdin); getline(cin, ss[i].disciplini[j].disname);

}

cout << "Vuvedi pol: "; cin >> ss[i].gender;

cout << "Vuvedi vuzrast: "; cin >> ss[i].age;

cout << "Vuvedi status: "; cin >> ss[i].status;

cout << "OK...\n";

}

return i;

}

void izvediedin(student ss[], int i) //извежда конкретен студент, като се подава позицията му в масива

{

cout << ss[i].faknum << " " << ss[i].name << "\nPol: " << ss[i].gender << "\nVuzrast: " << ss[i].age << "\nStatus: " << ss[i].status << endl;

for (int j = 0; j < 5; j++)

{

cout << "Ocenka po " << ss[i].disciplini[j].disname << ": ";

if (ss[i].disciplini[j].grade >= 2 && ss[i].disciplini[j].grade <= 6) cout << ss[i].disciplini[j].grade << endl; //Ако променливата на оценката не е от 2 до 6 или няма въведена оценка, или е въведена неправилно

else cout << "Nqma\n";

}

cout << endl;

}

void izvedi(student ss[], int broika) //функцията извежда всички студенти от групата, подава се масивът и броят на елементите в него

{

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

{

cout << ss[i].faknum << " " << ss[i].name << "\nPol: " << ss[i].gender << "\nVuzrast: " << ss[i].age << "\nStatus: " << ss[i].status << endl;

for (int j = 0; j < 5; j++)

{

cout << "Ocenka po " << ss[i].disciplini[j].disname << ": ";

if (ss[i].disciplini[j].grade >= 2 && ss[i].disciplini[j].grade <= 6) cout << ss[i].disciplini[j].grade << endl; //Ако променливата на оценката не е от 2 до 6 или няма въведена оценка, или е въведена неправилно

else cout << "Nqma\n";

}

cout << endl;

}

}

void izvedibp(student ss[], int broika) //Открива учениците с най-ниска оценка по БП и ги извежда на екрана

{

double minbp = 6;

for (int i = 0; i < broika; i++) //цикълът открива минималната оценка по БП, като обхожда всички студенти и търси БП в дисциплините

{

for (int j = 0; j < 5; j++) {

if (ss[i].disciplini[j].disname == "BP") {

if (ss[i].disciplini[j].grade < minbp && ss[i].disciplini[j].grade >= 2) {

minbp = ss[i].disciplini[j].grade;

}

break; //прекъсва единствено разглеждането на оценки за студента, понеже дисциплина БП е открита

}

}

}

for (int i = 0; i < broika; i++) // ако се наамери ученик, който е с минимална оценка по БП, то той се извежда

{

for (int j = 0; j < 5; j++)

{

if (ss[i].disciplini[j].disname == "BP")

{

if (ss[i].disciplini[j].grade == minbp)izvediedin(ss, i);

break;

}

}

}

}

void addgrade(student ss[], string nomer, int broika) //функцията добавя оценки на студент, параметрите са основният масив; стринг факултетен номер и броят на елементите в масива

{

bool prisustvie = 0; //променлива, показваща дали има студент с въведеен фак. номер в маасива.

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

{

if (nomer == ss[i].faknum) //проверка

{

prisustvie = 1;

bool keke = 0; //показва дали има невъведени оценки на студент

for (int j = 0; j < 5; j++)

{

if (ss[i].disciplini[j].grade < 2 || ss[i].disciplini[j].grade > 6)

{

short ppp;

cout << "Nqma vuvedena ocenka po " << ss[i].disciplini[j].disname << ". Za da se dobavi ocenka po " << ss[i].disciplini[j].disname << " vuvedete ""1"". V protiven sluchai vuvedete neshto drugo.\n";

cin >> ppp;

if (ppp == 1)

{

cout << "Vuvedete ocenkata: ";

cin >> ss[i].disciplini[j].grade;

if (ss[i].disciplini[j].grade >= 2 && ss[i].disciplini[j].grade <= 6) cout << "Ocenkata e uspeshno vuvedena.\n";

else cout << "Nevalidna ocenka. Ocenkata ne e vuvedena.\n";

}

}

}

for (int j = 0; j < 5; j++)

{

if ((ss[i].disciplini[j].grade < 2 || ss[i].disciplini[j].grade>6))

{

keke = 1; //има невъведена оценка

break;

}

}

if (keke == 0)cout << "Vsichki ocenki na studenta sa vuvedeni.\n";

}

}

if (prisustvie == 0)cout << "Student s takuv fakulteten nomer ne prisustva.\n";

}

void graduator(student ss[], string d, int broika) //проста функция, променяща статусът на ученик спрямо името му.

{

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

{

if (d == ss[i].name)

{

ss[i].status = "zavurshil";

cout << "Statusut na " << ss[i].name << " e promenen na \"" << ss[i].status << "\".\n";

break;

}

}

}

void ocenkinator(student ss[], string d, int broika)//функция, променяща оценките на студент, параметърът d е факултетен номер

{

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

{

if (ss[i].faknum == d) //otkrivaa se studentut

{

for (int j = 0; j < 5; j++)

{

cout << "Za da promenite ocenkata po " << ss[i].disciplini[j].disname << " vuvedete ""1"". V protiven sluchai, vuvedete drugo.\n";

short sos; cin >> sos;

if (sos == 1)

{

cout << "Vuvedete novata ocenka po " << ss[i].disciplini[j].disname << ": ";

double ococ; cin >> ococ;

if (ococ > 6 || ococ < 2)cout << "Nevalidna ocenka! Ostavena e predishnata stoinost...\n";

else {

ss[i].disciplini[j].grade = ococ;

cout << "Novata ocenka po " << ss[i].disciplini[j].disname << " e: " << ss[i].disciplini[j].grade << ".\n";

}

}

}

break;

}

}

}

string podgotvitel(string purvi) //функцията се използва при сортирането на студенти по азбучен ред

// подава ѝ се стринг, който е име на студент и се връща същия, само че с малки букви и без интервали. направих функцията, защото при сравняваане по ascii код

// главните букви винаги са "по-малки" от малките, а това създава обърквания при по-сложни имена, които съдържат "van","дьо" и подобни...

{

purvi.erase(remove\_if(purvi.begin(), purvi.end(), isspace), purvi.end()); //премахват се интервалите

transform(purvi.begin(), purvi.end(), purvi.begin(), ::tolower);//всички букви се правят малки

return purvi;

}

void sortabc(student ss[], int broika) //извежда всички студенти на екрана, сортирани по латински/английски азбучен ред

{

vector<string> podred; //записвам обработените имена на студентите във вектор, за по-лесна обработка

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

{

string a = podgotvitel(ss[i].name);

podred.push\_back(a);

}

sort(podred.begin(), podred.end()); //сортирам всичките имена по азбучен ред

for (int i = 0; i < broika; i++) //троен цикъл. не е най-бързият вариант, но работи

//обхожда се всеки студент

{

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

{

string a = podgotvitel(ss[j].name); //извеждат се студентите според реда им в сортирания вектор

if (podred[i] == a)

{

cout << ss[j].faknum << " " << ss[j].name << "\nPol: " << ss[j].gender << "\nVuzrast: " << ss[j].age << "\nStatus: " << ss[j].status << endl;

for (int k = 0; k < 5; k++)

{

cout << "Ocenka po " << ss[j].disciplini[k].disname << ": ";

if (ss[j].disciplini[k].grade >= 2 && ss[j].disciplini[k].grade <= 6) cout << ss[j].disciplini[k].grade << endl;

else cout << "Nqma\n";

}

cout << endl; break;

}

}

}

}

void sortage(student ss[], int broika) //извеждат се всички студенти, сортирани по възраст.

//алгоритъмът е същият като сортирането по азбучен ред

{

vector<short> ages;

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

ages.push\_back(ss[i].age);

sort(ages.begin(), ages.end());

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

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

if (ages[i] == ss[j].age)

{

cout << ss[j].faknum << " " << ss[j].name << "\nPol: " << ss[j].gender << "\nVuzrast: " << ss[j].age << "\nStatus: " << ss[j].status << endl;

for (int k = 0; k < 5; k++)

{

cout << "Ocenka po " << ss[j].disciplini[k].disname << ": ";

if (ss[j].disciplini[k].grade >= 2 && ss[j].disciplini[k].grade <= 6) cout << ss[j].disciplini[k].grade << endl;

else cout << "Nqma\n";

}

cout << endl; break;

}

}

void polovete(student ss[], int broika)// извежда най-м

{

vector<short> fages, mages;

for (int i = 0; i < broika; i++) //векторите се запълват с възрастите на жените и възрастите на мъжете

{

if (ss[i].gender == "male")mages.push\_back(ss[i].age);

if (ss[i].gender == "female")fages.push\_back(ss[i].age);

}

sort(fages.begin(), fages.end()); sort(mages.begin(), mages.end());//сортира се, за да се открие най-ниската възраст

for (int j = 0; j < broika; j++) //обхожда се всеки студент и се извеждааат най-младите по възраст от пола

{

if (ss[j].gender == "male" && ss[j].age == mages[0])

{

cout << ss[j].faknum << " " << ss[j].name << "\nPol: " << ss[j].gender << "\nVuzrast: " << ss[j].age << "\nStatus: " << ss[j].status << endl;

for (int k = 0; k < 5; k++)

{

cout << "Ocenka po " << ss[j].disciplini[k].disname << ": ";

if (ss[j].disciplini[k].grade >= 2 && ss[j].disciplini[k].grade <= 6) cout << ss[j].disciplini[k].grade << endl;

else cout << "Nqma\n";

}

cout << endl;

}

if (ss[j].gender == "female" && ss[j].age == fages[0])

{

cout << ss[j].faknum << " " << ss[j].name << "\nPol: " << ss[j].gender << "\nVuzrast: " << ss[j].age << "\nStatus: " << ss[j].status << endl;

for (int k = 0; k < 5; k++)

{

cout << "Ocenka po " << ss[j].disciplini[k].disname << ": ";

if (ss[j].disciplini[k].grade >= 2 && ss[j].disciplini[k].grade <= 6) cout << ss[j].disciplini[k].grade << endl;

else cout << "Nqma\n";

}

cout << endl;

}

}

}

void avgusp(student ss[], int broika) //открива се студента с най-високият среден успех и се извежда на екрана

{

double highest = 2; int maxi=0;

for (int i = 0; i < broika; i++) //открива се най-високият среден успех

{

double sr = 0; short broc = 0;

for (int k = 0; k < 5; k++)

{

if (ss[i].disciplini[k].grade >= 2 && ss[i].disciplini[k].grade <= 6)

{

sr += ss[i].disciplini[k].grade; //открива се средният успех част 1

broc++;

}

}

if (broc>0)sr /= broc; //октирва се средният успех част 2

if (sr > highest) { //ако се открие по-висок от досегашния най-висок среден успех, запазваме позицията на студента и назначаваме нов макс. среден успех.

maxi = i; highest = sr;

}

cout << "Sredniqt uspeh na " << ss[i].name << " e: " << sr << endl;//извежда се средният успех на всеки студент за улеснение и

//проверка за коректност на програмата, може редът да се закоментари

}

//извеждаме студентът с най-висок среден успех

cout << ss[maxi].faknum << " " << ss[maxi].name << "\nPol: " << ss[maxi].gender << "\nVuzrast: " << ss[maxi].age << "\nStatus: " << ss[maxi].status << endl;

for (int k = 0; k < 5; k++)

{

cout << "Ocenka po " << ss[maxi].disciplini[k].disname << ": ";

if (ss[maxi].disciplini[k].grade >= 2 && ss[maxi].disciplini[k].grade <= 6) cout << ss[maxi].disciplini[k].grade << endl;

else cout << "Nqma\n";

}

cout << endl;

}

int main() //мейн, съдържаащ менюто

{

cout << "Za mujki i jenski pol molq izpolzvaite \"male\" i \"female\"!\n"; //пояснение

cout << endl;

student sheet[32]; //основния масив от структури

int alle = 0;//loadfile(sheet);

short a; //променлива за работаа с менюто

do {

cout << "\nZdraveite, za da dobavite student/i v grupata vuvedete ""1"". \nZa da izvedete studenti na ekrana vuvedete ""2"".\nZa da vuvedete ocenki na student vuvedete ""3"".\nZa da promenite studentite v grupata, vuvedete ""4"".\n";

cout << "Za da napravite spravki za studentite, vuvedete ""5"".\nZa da zapishete dannite v dvoichen fail, vuvedete 6.\nZa da izlezete ot programata, vuvedete 7.\n";

cin >> a;

if (a == 1)

{

cout << "Ako iskate da dobavite edin student v grupata vuvedete ""1"".\nZa da dobavite poveche, vuvedete ""2""\n";

short k; cin >> k;

if (k == 1)

{

alle = addstud(1, sheet, alle); /\*saveinfile(sheet, alle);\*/

short cirk;

do

{

cout << "Za da dobavite oshte edin student, vuvedete 1; v protiven sluchai, vuvedete neshto drugo.";

cin >> cirk;

if (cirk == 1) { alle = addstud(1, sheet, alle); /\*saveinfile(sheet, alle);\*/ }

} while (cirk == 1);

}

else {

short s;

cout << "Vuvedete broqt na studenti, koito iskate da dobavite: "; cin >> s;

alle = addstud(s, sheet, alle); saveinfile(sheet, alle);

}

}

if (a == 2)

{

cout << "Za da izvedete vsichki studenti na ekrana, vuvedete ""1"".\nZa da izvedete studenti s minimalna ocenka po BP, vuvedete ""2"".\n";

short k; cin >> k;

if (k == 1)izvedi(sheet, alle);

if (k == 2)izvedibp(sheet, alle);

}

if (a == 3)

{

string fakfak;

cout << "Vuvedete fakulteten nomer na studenta: ";

cin >> fakfak;

addgrade(sheet, fakfak, alle);

/\*saveinfile(sheet, alle);\*/

}

if (a == 4)

{

cout << "Za da promenite statusa na student na zavurshil, vuvedete ""1"".\nZa da promenite ocenkite na student, vuvedete ""2"".\n";

short k; cin >> k;

if (k == 1)

{

string umora; cout << "Vuvedete imeto na studenta: "; getline(cin, umora); getline(cin, umora);

graduator(sheet, umora, alle);

/\*saveinfile(sheet, alle);\*/

}

if (k == 2)

{

string fakulnum; cout << "Vuvedete fakultetniq nomer na studenta: "; cin >> fakulnum;

ocenkinator(sheet, fakulnum, alle);

/\*saveinfile(sheet, alle);\*/

}

}

if (a == 5)

{

cout << "Za da izvedete studentite v podreden red, vuvedete ""1"".\nZa da izvedete nai-mladite studenti po pol, vuvedete ""2"".\nZa da izvedete studenta s nai-visok sreden uspeh, vuvedete ""3"".\n";

short k; cin >> k;

if (k == 1)

{

cout << "Vuvedete \"1\" za azbuchen red(latinska azbuka) ili \"2\" za vuzhodqshta vuzrast.";

short skt; cin >> skt;

if (skt == 1)sortabc(sheet, alle);

if (skt == 2)sortage(sheet, alle);

}

if (k == 2)

{

polovete(sheet, alle);

}

if (k == 3)

{

avgusp(sheet, alle);

}

}

if (a == 6) {

cout << "Ne e sigurno dali raboti kato horata, no masivut e zapisan...\n";

saveinfile(sheet, alle);

}

} while (a != 7);

return 0;

}