Картина, която съдържа текст, графична колекция

Описанието е генерирано автоматичноТЕХНИЧЕСКИ УНИВЕРСИТЕТ – ВАРНА

Факултет по изчислителна техника и автоматизация

Катедра „ Компютърни науки и технологии”

**СЕМЕСТРИАЛНА ДОМАШНА РАБОТА**

по дисциплината „Базово програмиране”

на тема: „ информационна система, която поддържа студентска група”

Вариант 162

|  |  |
| --- | --- |
| Изготвил:Кристиян Красимиров Михнев | Проверил: |
| Специалност: КСТ |  |
| Група: 1Б |  |
| Факултетен номер: 21621353 |  |

2021

Съдържание

[I. Задание на проекта 3](#_Toc91106791)

[II. Анализ на решението 4](#_Toc91106792)

[1. Структура за данните в програмата 4](#_Toc91106793)

[2. Реализация на условие A 4](#_Toc91106794)

[3. Реализация на условие B 4](#_Toc91106795)

[4. Реализация на условие C 5](#_Toc91106796)

[5. Реализация на условие D 5](#_Toc91106797)

[6. Реализация на условие E 6](#_Toc91106798)

[7. Реализация на условие F 7](#_Toc91106799)

[8. Реализация на условие G 7](#_Toc91106800)

[9. Реализация на условие H (по желание) 8](#_Toc91106801)

[III. Упътване за употреба 9](#_Toc91106802)

[1. Впишете съответната част от проекта 9](#_Toc91106803)

[2. Впишете съответната част от проекта 9](#_Toc91106804)

[IV. Примерно действие на програмата 10](#_Toc91106805)

[1. Условие A 10](#_Toc91106806)

[2. Условие B 10](#_Toc91106807)

[3. Условие C 10](#_Toc91106808)

[4. Условие D 10](#_Toc91106809)

[5. Условие E 10](#_Toc91106810)

[6. Условие F 11](#_Toc91106811)

[7. Условие G 11](#_Toc91106812)

[8. Условие H (по желание) 11](#_Toc91106813)

Задание на проекта

**Студенти**

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 ниска**

**A** 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 факултетни номера, започващи с еднакви първи 6 цифри

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 студент 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

Структура за данните в програмата

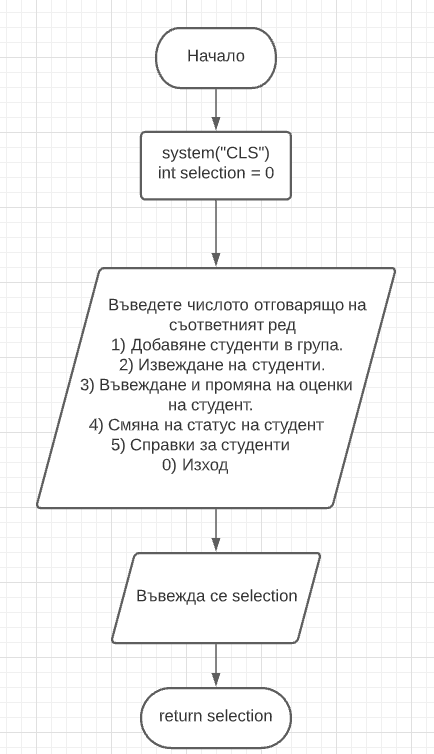
|  |  |  |
| --- | --- | --- |
| Структура | Обяснение | Примерени стойности |
| struct student {  int facultyId=0;  char name[30]="";  discipline disciplines[6];  char sex[10]="";  int age=0;  char status[10]="";  };  struct discipline {  char name[25];  int grade;  }; | Структурата държи данните на студента:  Факултетен номер(цяло число)  Име(масив от char)  Дисциплини(масив от дисциплини)  Пол(масив от char)  Възраст(цяло число)  Статус(масив от char)  Структурата държи данните за дисциплина  Име(масив от char)  Оценка(цяло число) | facultyId=21621353  name="Kristiyan Mihnev"  disciplines={}  sex="Мъжки"  age=19  status="завършил"  name="Базово Програмиране"  grade=6 |

Реализация на условие A

Анализ на алгоритъма, който трябва да се реализира

Изписват се всички опции на конзолата и се въвежда числото отговарящо на съответната опция след което функцията връща избраното число

Блок схема на алгоритъма



Функция с която е реализиран алгоритъма

int mainMenu(){

system("CLS");

int selection = 0

cout << "Въведете числото отговарящо на съответният ред" << endl;

cout << "1) Добавяне студенти в група." << endl;

cout << "2) Извеждане на студенти." << endl;

cout << "3) Въвеждане и промяна на оценки на студент." << endl;

cout << "4) Смяна на статус на студент." << endl;

cout << "5) Справки за студенти" << endl;

cout << "0) Изход"<<endl;

cin >> selection;

system("CLS");

return selection;}

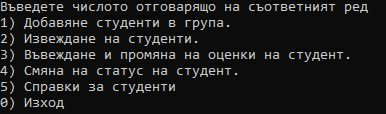
Функцията изчиства конзолата и изписва възможните опции на конзолата. След което се очаква въвеждане на цяло число отговарящо на съответната опция, след неговото въвеждане функцията го връща.

#### Входни данни на функцията

Функцията приема цяло число което съответства на определена опция

#### Изходни данни на функцията или данни, които се извеждат

Функцията извежда следните опции



Функцията връща въведеното от потребителя число

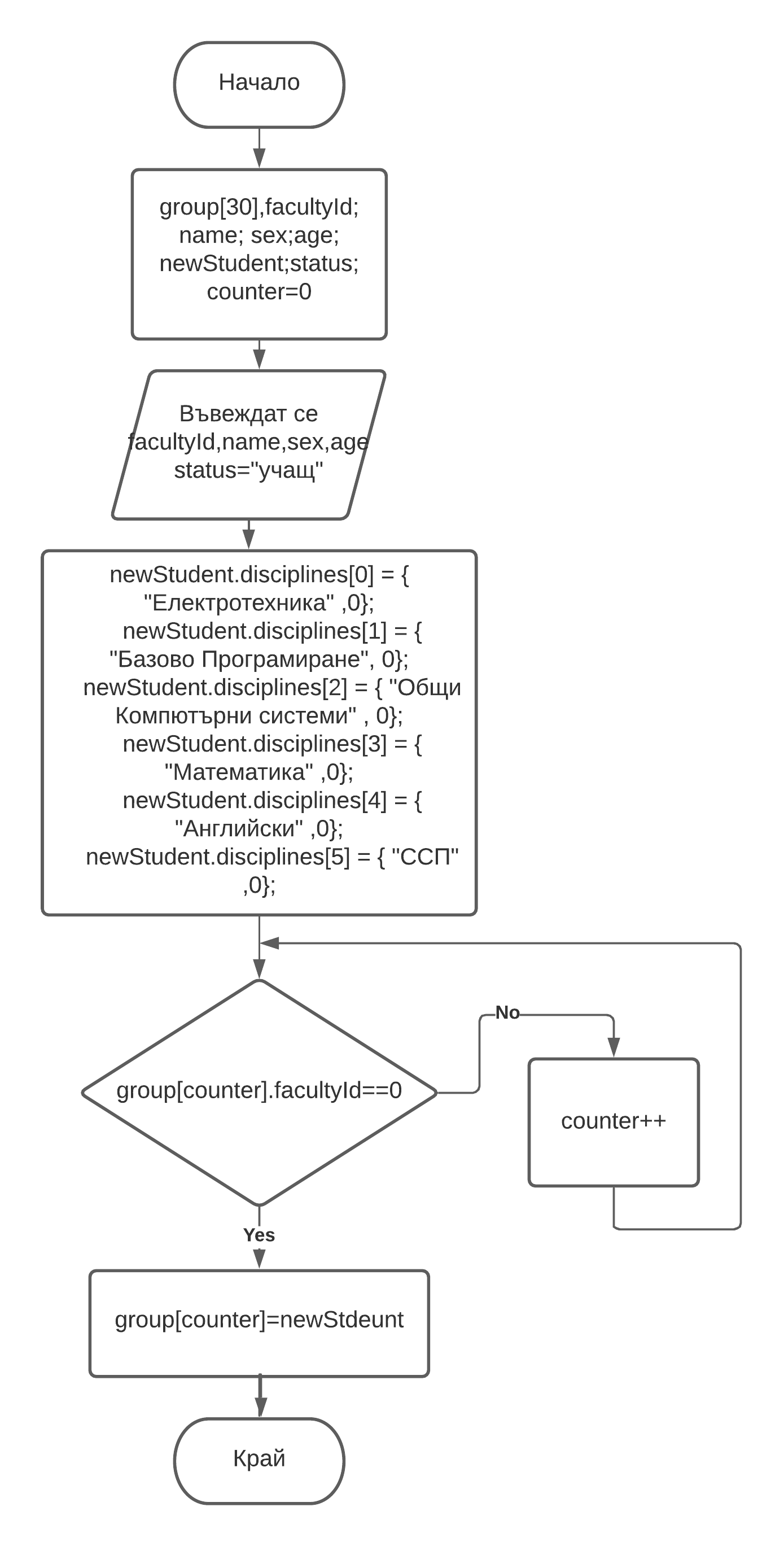
Реализация на условие B

Анализ на алгоритъма, който трябва да се реализира

Трябва да се реализира добавянето на студенти. Въвеждат се факултетен номер, име, пол и възраст на студента след което се присвояват към структурата. Статусът не се въвежда от потребителят но се задава автоматично при добавяне на студент, началната стойност е “учащ”. След присвояването на основните данни, се създават 6 дисциплини (Електротехника, Базово Програмиране, Общи Компютърни системи, Английски и ССП) със начална оценка 0. Те се присвояват към дисциплините на студента.

След попълването на всички данни на студента, се влиза в цикъл който обхожда студентите в групата докато не намери студент със факултетен номер=0 след което присвоява новият студент към него.

Блок схема на алгоритъма



Функция с която е реализиран алгоритъма

void createStudent(student group[]) {

int facultyN;

string name;

string sex;

unsigned int age;

cout << "Въвдете данните на студента в следния формат:" << endl;

cout << "Факултетен номер" << endl;

cin >> facultyN;

cout << "Име" << endl;

cin.ignore();

getline(cin, name);

cout << "Пол" << endl;

cin >> sex;

cout << "Възраст" << endl;

cin >> age;

student newStudent; стойности

newStudent.facultyId = facultyN;

strcpy\_s(newStudent.name, name.c\_str());

strcpy\_s(newStudent.sex, sex.c\_str());

string status = "учащ";

strcpy\_s(newStudent.status, status.c\_str());

newStudent.age = age;

newStudent.disciplines[0] = { "Електротехника" ,0};

newStudent.disciplines[1] = { "Базово Програмиране", 0};

newStudent.disciplines[2] = { "Общи Компютърни системи" , 0};

newStudent.disciplines[3] = { "Математика" ,0};

newStudent.disciplines[4] = { "Английски" ,0};

newStudent.disciplines[5] = { "ССП" ,0};

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

{

if (group[i].facultyId == 0)

{

group[i] = newStudent;

break;

}

}

}

Функцията приема данните на студента, създава и присвоява дисциплините и го добавя в групата

#### Входни данни на функцията

Функцията приема масив от студенти в който ще бъде добавен новият студент

#### Изходни данни на функцията или данни, които се извеждат

Функцията не извежда данни на конзолата. Тя приема на данни на студента и след като му ги присвои, го добавя в масива и приключва работа.

Реализация на условие C

Анализ на алгоритъма, който трябва да се реализира

За извеждане на всички студенти се минава през масива от студенти с цикъл, ако някой от студентите има факултетен номер = 0, то той се пропуска. Има брояч който се инкрементира след принтирането на всеки студент. Ако този брояч стигне число кратно на 5, то програмата се паузира до натискане на ентер, след което се изписват следващите 5 студента. Ако брояча е равен на 0 - няма студенти които да бъдат принтирани и се извежда съобщение

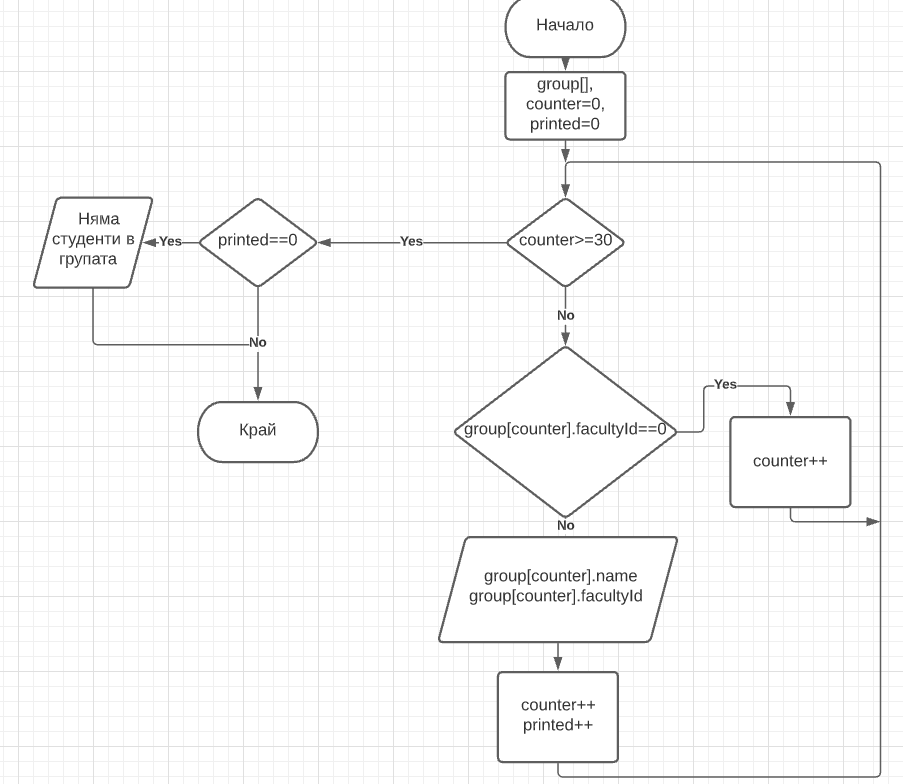
За извеждането на студенти с еднакви 6 цифри от факултетен номер. Въвеждат се 6-цигрено число от потребителя след което се завърта цикъл през елементите на масива ако първите 6 цифри от факултетния номер на студент съвпадат с цифрата въведена от потребителя, то този студент ще бъде изписан.

Извеждането на студенти е в следния формат

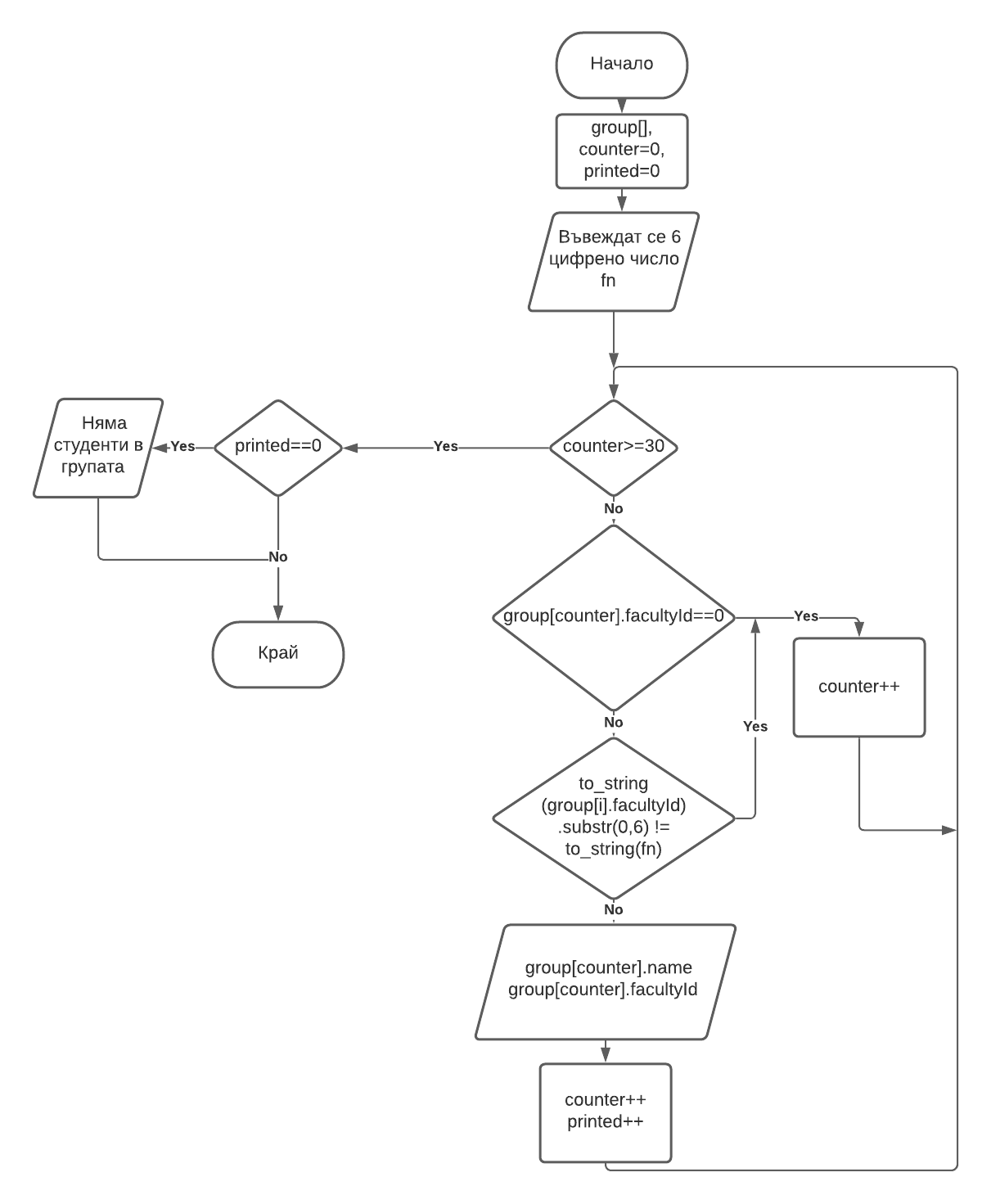
Име-факултетен номер

Блок схема на алгоритъма

За извеждане на всички студенти



За извеждане на студенти с еднакви 6 цифри на факултетния номер



Функция с която е реализиран алгоритъма

За извеждане на всички студенти

void printStudents(student group[]) {

int page = 1;

int start = 0;

int end = 5;

int printed = 0; cout << "страница: " << page<<endl;

cout << setw(26) << left << "Име на студент" << "факултетен номер" << endl;

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

{

if (group[i].facultyId == 0)

continue;

cout << printed+1 << ") " << setw(20) << left << group[i].name << " - " << setw(10) << group[i].facultyId << endl;

printed++;

if (printed%5==0)

{

system("pause");

system("CLS");

page++;

cout << "страница: " << page << endl;

cout << setw(26) << left << "Име на студент" << "факултетен номер" << endl;

}

}

if (printed == 0)

{

system("CLS");

cout << "Няма налични студенти" << endl;

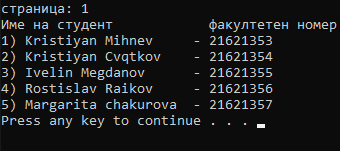
}

system("pause");

}

Използва се за принтиране на всички студенти във формат Име-факултетен номер

След нейното извикване



За извеждане на студенти с еднакви 6 цифри на факултетния номер

void findStudentByFn(student group[]) {

int fn;

do {

cout << "Въведете първите 6 цифри от факултетен номер." << endl;

cin >> fn;

} while (fn > 999999 || fn < 100000);

system("CLS");

cout << setw(26) << left << "Име на студент" << "факултетен номер" << endl;

int count = 0;

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

{

if (to\_string(group[i].facultyId).substr(0,6)==to\_string(fn))

{

count++;

cout << count << ") "<<setw(20)<<left << group[i].name << " - "<<setw(10) << group[i].facultyId<<endl;

}

}

if (count==0)

{

system("CLS");

cout << "Не са намерени студенти с този факултетен номер!"<<endl;

}

system("Pause");

system("CLS");

}

Използва се за намиране на всички студенти с еднакви на въведените първи 6 цифри от факултетния номер

#### Входни данни на функцията

Функцията приема масив от студенти от който ще бъдат изведени студентите

#### Изходни данни на функцията или данни, които се извеждат

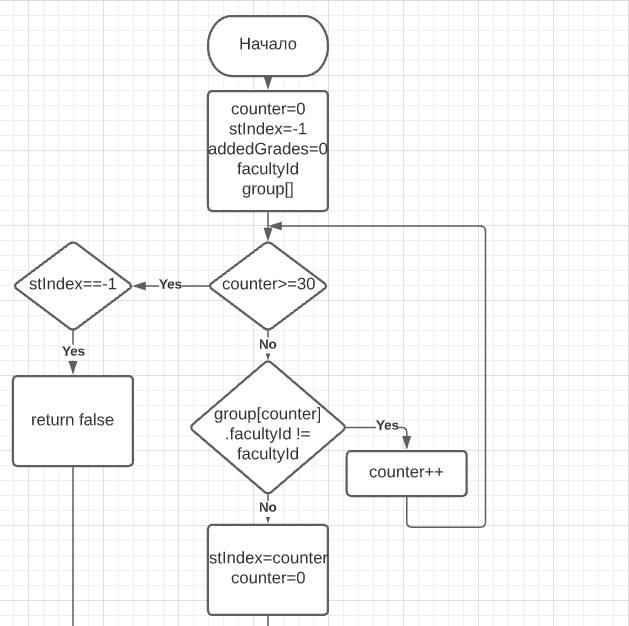
Функцията извежда студентите във формат Име - Факултетен номер

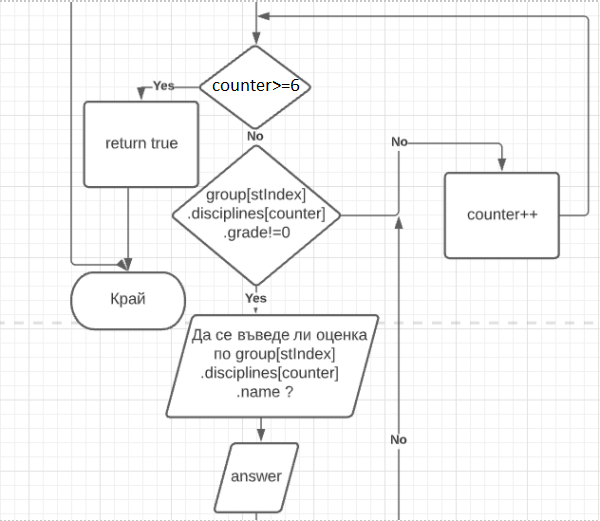
Реализация на условие D

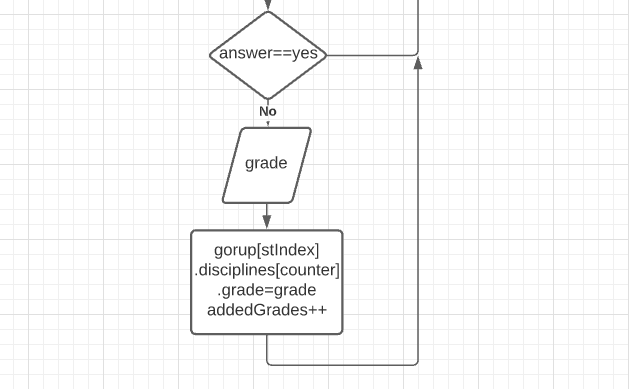
Анализ на алгоритъма, който трябва да се реализира

Цикъл минава през масива със студенти. Ако някой от студентите има факултетен номер=0, то той се пропуска. При намиране на студента с факултетен номер идентичен на този подаден на функцията се записва индекса му. Ако не е намерен студент се изписва съобщение. След намирането на студента се обхождат дисциплините му чрез цикъл и се пита дали потребителят иска да въведе оценка по съответната дисциплина. Ако студентът има оценка по някоя от дисциплините, то тя се пропуска. Ако всички оценки са попълнени се изкарва съобщение. Функцията връща булева стойност която се определя от това дали е намерен студент.

Блок схема на алгоритъма







Функция с която е реализиран алгоритъма

bool addGrades(int fn, student group[]) {

int stIndex=-1;

int addedGrades=0;

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

{

if (group[i].facultyId == 0)

continue;

if (group[i].facultyId == fn)

{

stIndex = i;

break;

}

}

if (stIndex==-1)

return false;

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

{

if (group[stIndex].disciplines[i].grade != 0)

continue;

int grade;

string answer;

cout << "Да се добави ли оцента по: " << group[stIndex].disciplines[i].name << "? (yes/no)"<<endl;

cin >> answer;

if (answer == "yes")

{

cout << "Въведете оценка по: " << group[stIndex].disciplines[i].name << endl;

cin >> grade;

group[stIndex].disciplines[i].grade = grade;

addedGrades++;

}

else

continue;

system("CLS");

}

if (addedGrades == 0)

cout << "Всички оценки на студента са въведени!" << endl;

else

cout << "Оценките са въведени успешно!"<<endl;

system("pause");

return true;

}

#### Входни данни на функцията

Функцията приема факултетен номер и масив от студенти

#### Изходни данни на функцията или данни, които се извеждат

Функцията връща дали е намерен студент и изписва съобщение

Ако е намерен- Оценките са въведени успешно

Ако е намерен но всички оценки са му въведени -

Всички оценки на студента са въведени!

Реализация на условие E

Анализ на алгоритъма, който трябва да се реализира

Смяна на статус:

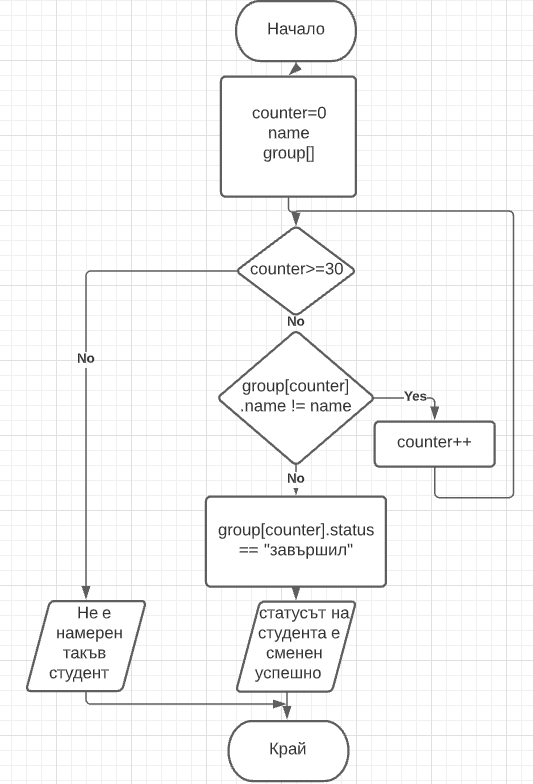
Минава се през студентите от масива чрез цикъл докато не е намерен студент със име идентично на въведеното от потребителят. След намирането на студента се сменя статусът му на завършил.

Промяна на оценки

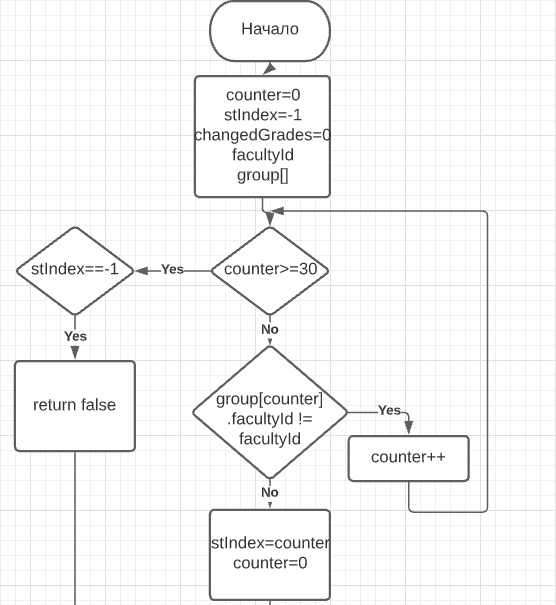
Минава се през студентите от масива чрез цикъл докато не се намери студент с идентичен факултетен номер на този, въведен от потребителя. След което се минава през оценките му като се пита дали да бъде пременена съответната оценка. При отговор да: се въвежда новата оценка, а при не: се преминава към следваща

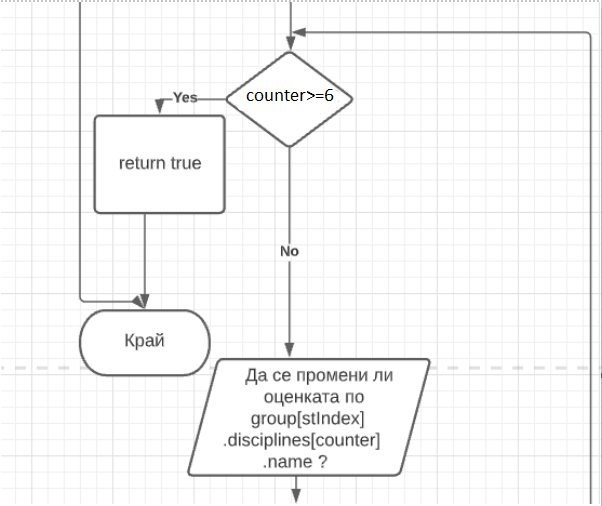
Блок схема на алгоритъма

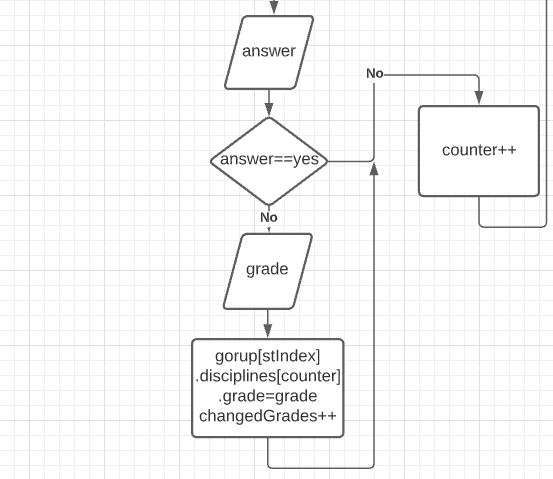
Смяна на статус на студент



Промяна на оценки







Функция с която е реализиран алгоритъма

* Смяна на статус

void changeStudentStatus(student group[]) {

cout << "Въведете име на студента." << endl;

char name[30];

cin.ignore();

cin.getline(name, 30);

bool found = false;

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

{

if (strcmp(group[i].name, name) == 0) {

strcpy\_s(group[i].status, "завършил");

system("CLS");

cout << "Статуса на " << group[i].name << " е сменен успешно." << endl;

found = true;

break;

}

}

if (!found)

{

system("CLS");

cout << "Не е намерен такъв студент." << endl;

}

system("pause");

}

Търси студент по име и ако той бъде намерен се сменя статусът му на “завършил”

* Смяна на оценки

bool editGrades(int fn, student group[]) {

int stIndex = -1;

int editedGrades = 0;

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

{

if (group[i].facultyId == 0)

continue;

if (group[i].facultyId == fn)

{

stIndex = i;

break;

}

}

if (stIndex == -1)

return false;

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

{

int grade;

string answer;

if (group[stIndex].disciplines[i].grade==0)

cout << "Студента няма оценка по този предмет. Да се добави ли? (yes/no)" << endl;

else

cout << "Да се промени ли оценката по: " << group[stIndex].disciplines[i].name << "? Текуща оценка: "<<group[stIndex].disciplines[i].grade<<"(yes/no)" << endl;

cin >> answer;

if (answer == "yes")

{

cout << "Въведете оценка по: " << group[stIndex].disciplines[i].name << endl;

cin >> grade;

group[stIndex].disciplines[i].grade = grade;

editedGrades++;

}

else

continue;

system("CLS");

}

if (editedGrades == 0)

cout << "Не са направени промени по оценките на студента." << endl;

else

cout << "Оценките са променени успешно!" << endl;

system("pause");

return true;

}

Търси се студент по факултетен номер и ако бъде намерен се дава опцията да се смени всяка от оценките му

#### Входни данни на функцията

* Смяна на статус

Функцията приема масив от студенти

* Промяна на оценки

Функцията приема факултетен номер и масив от студенти

#### Изходни данни на функцията или данни, които се извеждат

* Смяна на статус

Изписва се дали успешно е сменен статуса на студента

* Промяна на оценки

Изписва се дали успешно да променени оценките на студента

Реализация на условие F

Анализ на алгоритъма, който трябва да се реализира

* Сортиране на студентите в азбучен ред:

Подрежда се масива чрез сравняване на имената на два студента и при нужда, тяхното разменяне. След като са подредени, се извеждат с всички свои данни.

* Сортиране на студентите по възраст в низходящ ред.

Подрежда се масива чрез сравняване на възрастта на два студента и при нужда, тяхното разменяне. След като са подредени, се извеждат с всички свои данни.

* Сортиране на студентите по възраст във възходящ ред.

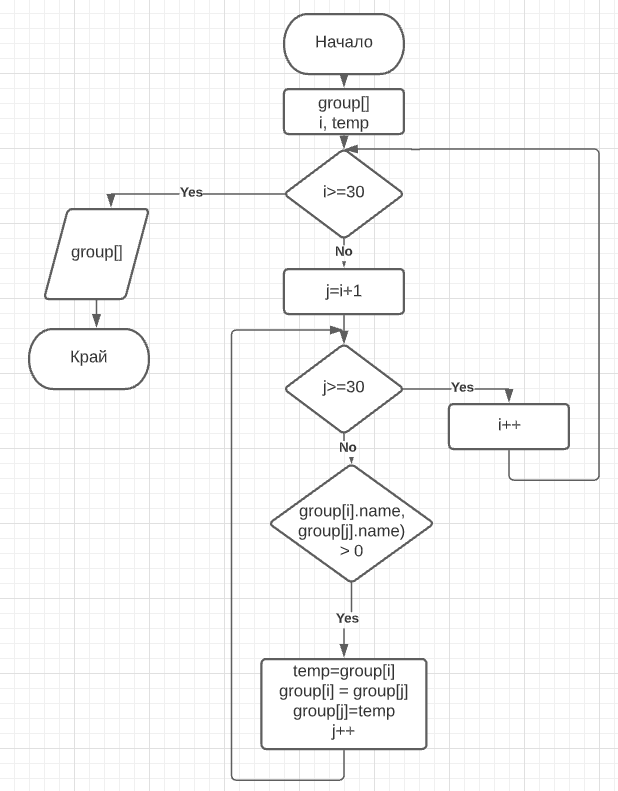
Подрежда се масива чрез сравняване на възрастта на два студента и при нужда, тяхното разменяне. След като са подредени, се извеждат с всички свои данни.

* Калкулира се и се запазва средният успех на първия студент в масива, ако се намери студент със по-голям среден успех, то той се запазва и това се повтаря до края на масива.
* Извеждане на студентите

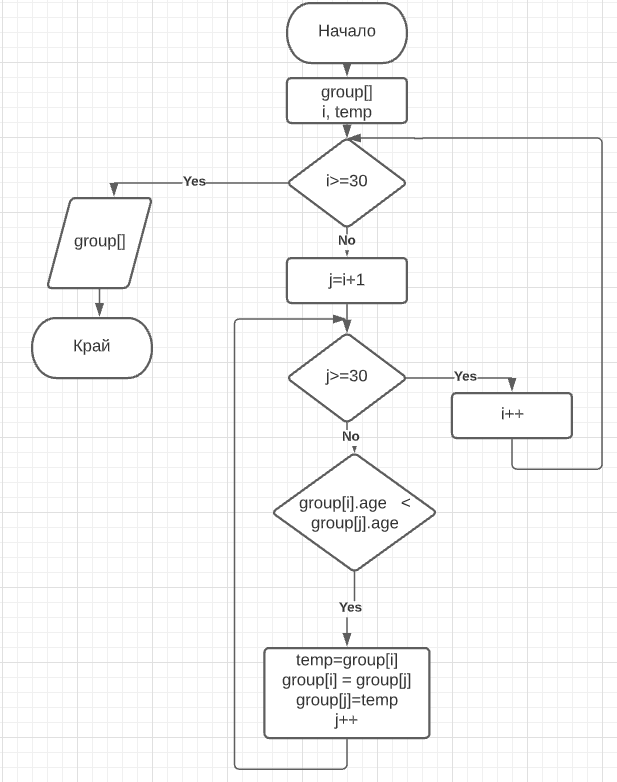
Вземат се сортираните студенти и се изписват със всички свои данни.

Блок схема на алгоритъма

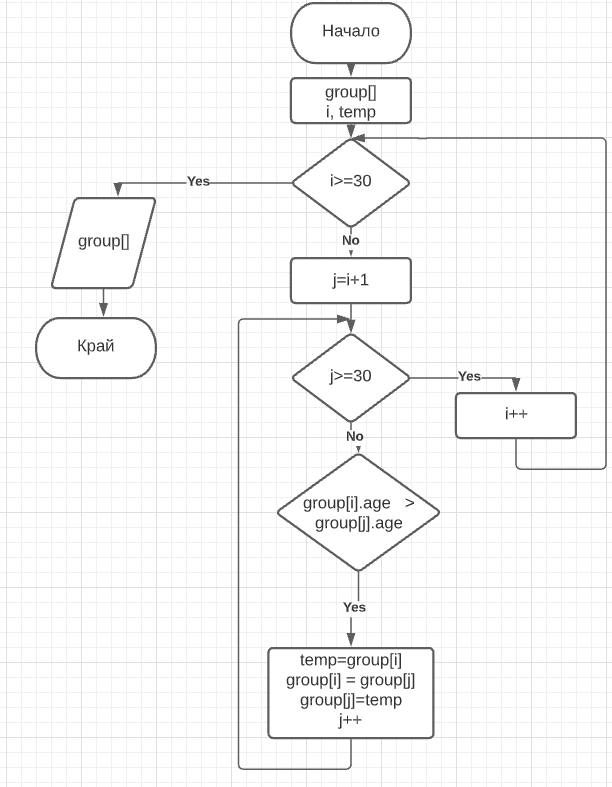
* Сортиране на студентите в азбучен ред:



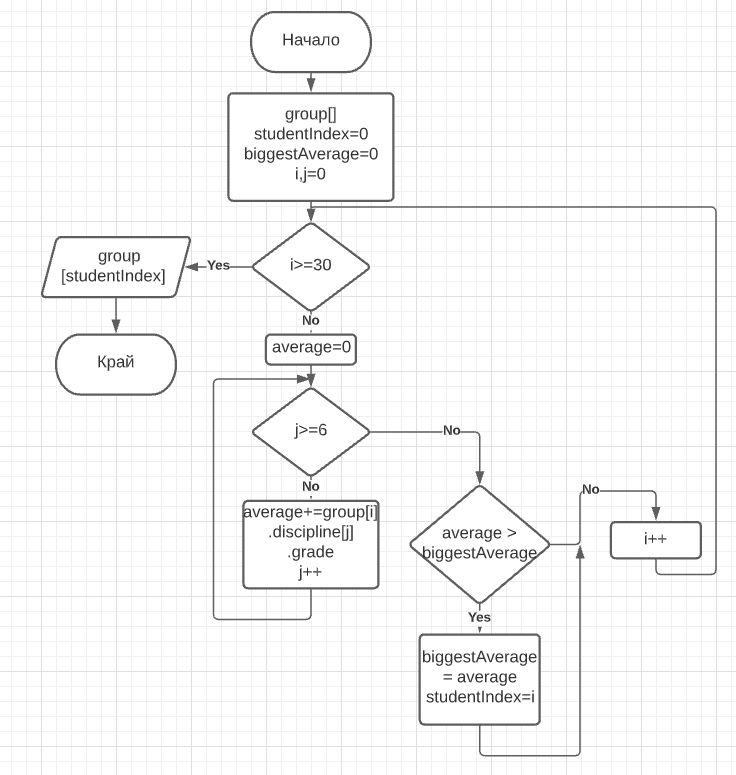
* Сортиране на студентите подредени по възраст в низходящ ред.



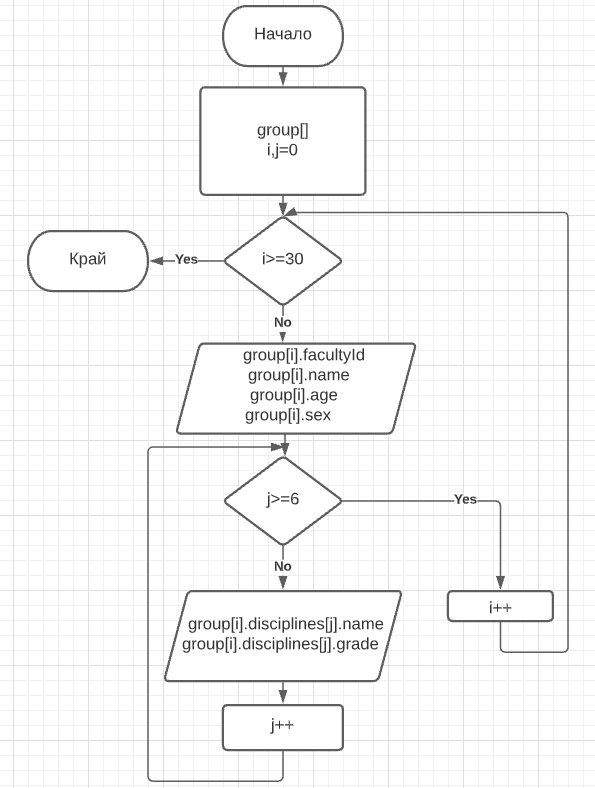
* Сортиране на студентите подредени по възраст във възходящ ред.



* Извеждане на студента с най-висок среден успех



* Извеждане на студенти с пълни данни



Функция с която е реализиран алгоритъма

* Сортиране на студентите в азбучен ред:

student\* sortByName(student group[]) {

student temp;

student\* sorted = group;

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

{

for (int j = i + 1; j < groupCount; j++)

{

if (strcmp(sorted[i].name, sorted[j].name) > 0)

{

temp = sorted[i];

sorted[i] = sorted[j];

sorted[j] = temp;

}

}

}

return sorted;

}

Поддава се масив със студенти на функцията и тя връща масив с подредени в азбучен ред студенти.

* Сортиране на студентите подредени по възраст в низходящ ред.

student\* sortByAgeDesc(student group[]) {

student temp;

student\* sortedGroup = group;

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

{

for (int j = i + 1; j < groupCount; j++)

{

if (sortedGroup[i].age < sortedGroup[j].age)

{

temp = sortedGroup[i];

sortedGroup[i] = sortedGroup[j];

sortedGroup[j] = temp;

}

}

}

return sortedGroup;

}

Поддава се масив със студенти на функцията и тя връща масив с подредени по възраст в низходящ ред студенти.

* Сортиране на студентите подредени по възраст във възходящ ред.

student\* sortByAgeDesc(student group[]) {

student temp;

student\* sortedGroup = group;

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

{

for (int j = i + 1; j < groupCount; j++)

{

if (sortedGroup[i].age > sortedGroup[j].age)

{

temp = sortedGroup[i];

sortedGroup[i] = sortedGroup[j];

sortedGroup[j] = temp;

}

}

}

return sortedGroup;

}

Поддава се масив със студенти на функцията и тя връща масив с подредени по възраст във възходящ ред студенти.

* Извеждане на студента с най-висок среден успех

student findStudentWithBiggestAvg(student group[]) {

int bestStudentIndex = 0;

double biggestAverage = 0;

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

{

if (group[i].facultyId == 0)

continue;

int average = 0;

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

{

average += group[i].disciplines[j].grade;

}

if (average > biggestAverage)

{

biggestAverage = average;

bestStudentIndex = i;

}

}

return group[bestStudentIndex];

}

Приема масив от студенти и връща студента с най-висок успех

* Извеждане на студенти с пълни данни

void printStudentsAllInfo(student group[], string sex) {

int printed = 0;

cout << setw(25) << left << "Име на студент" <<setw(16)<<left<< "Факултетен номер "<<setw(8)<<left <<"Пол"<<setw(2)<<left<<" Възраст"<<setw(19)<<left<<" Статус"<< endl<<endl;

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

{

if (group[i].facultyId == 0)

continue;

if (sex == "All")

{

cout << printed + 1 << ") " << setw(22) << left << group[i].name << setw(17) << group[i].facultyId << setw(9) << left << group[i].sex << setw(8) << left << group[i].age << setw(9) << left << group[i].status << endl;

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

{

cout << group[i].disciplines[j].name << " - " << group[i].disciplines[j].grade << " ; ";

}

cout << endl << endl;

printed++;

}

else

{

if (group[i].sex != sex)

continue;

cout << printed + 1 << ") " << setw(22) << left << group[i].name << setw(17) << group[i].facultyId << setw(9) << left << group[i].sex << setw(8) << left << group[i].age << setw(9) << left << group[i].status << endl;

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

{

cout << group[i].disciplines[j].name << " - " << group[i].disciplines[j].grade << " ; ";

}

cout << endl << endl;

printed++;

}

}

if (printed == 0)

{

system("CLS");

cout << "Няма налични студенти" << endl;

}

system("pause");

}

#### Входни данни на функцията

* Сортиране на студентите в азбучен ред:

Приема масив от студенти

* Сортиране на студентите по възраст в низходящ ред:

Приема масив от студенти

* Сортиране на студентите по възраст във възходящ ред:

Приема масив от студенти

* Извеждане на студент с най-висок среден успех

Приема масив от студенти

* Извеждане на всички данни на студентите

Приема масив от студенти и пол от който да бъдат изведени студентите

#### Изходни данни на функцията или данни, които се извеждат

* Сортиране на студентите в азбучен ред:

Връща сортиран масив от студенти

* Сортиране на студентите по възраст в низходящ ред:

Връща сортиран масив от студенти

* Сортиране на студентите по възраст във възходящ ред:

Връща сортиран масив от студенти

* Извеждане на студент с най-висок среден успех

Връща студент

* Извеждане на всички данни на студентите

Извежда всички студенти според критериите зададени от потребителя

Реализация на условие G

Анализ на алгоритъма, който трябва да се реализира

* запис

Записват се масива със студенти в двоичен файл

* Четене

Чете се двоичният файл и данните се записват в масив

Функция с която е реализиран алгоритъма

* Запис

void writeToFile(student group[30]) {

fstream file;

file.open(fileName, ios::binary | ios::out);

file.write((char\*)group, groupCount \* (sizeof(student)));

file.close();

}

* Четене

void readFromFile(student group[]) {

fstream file;

file.open(fileName, ios::binary | ios::in);

file.seekg(0L, ios::end);

long pos = (long)file.tellg()

file.seekg(0L, ios::beg);

file.close();

int n = pos / (sizeof(student));

file.open(fileName, ios::binary | ios::in);

file.read((char\*)group, n \* (sizeof(student)));

file.close();

}

Реализация на условие H (по желание)

Допълненията са добавени към съответната подточка

Упътване за употреба

Впишете съответната част от проекта

Очаквани входни данни

За добавяне на студент се очакват:

Факултетен номер - int

Име - string

Пол - string

Възраст - int

При невалидни данни програмата приключва

Очакван резултат от изпълнението на конкретния фрагмент от проекта

Създава се нов студент и се добавя в групата

Впишете съответната част от проекта

Очаквани входни данни

За извеждане на студенти с еднакви първи 6 цифри от факултетният номер

6-цифрено число - int

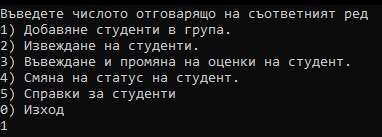
Очакван резултат от изпълнението на конкретния фрагмент от проекта

Извеждат се всички студенти чийто факултетни номера започват с тези 6 цифри

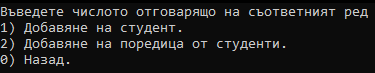
Примерно действие на програмата

Условие A

Снимка на изгледа с примерни входни данни

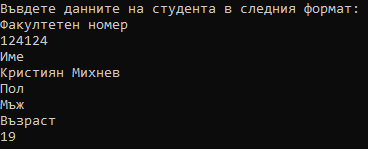


Снимка на изгледа с примерни изходни данни



Условие B

Снимка на изгледа с примерни входни данни



Снимка на изгледа с примерни изходни данни

Няма изходни данни

Условие C

Снимка на изгледа с примерни входни данни

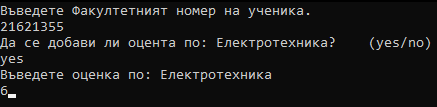


Снимка на изгледа с примерни изходни данни



Условие D

Снимка на изгледа с примерни входни данни



Снимка на изгледа с примерни изходни данни



Условие E

Снимка на изгледа с примерни входни данни



Снимка на изгледа с примерни изходни данни

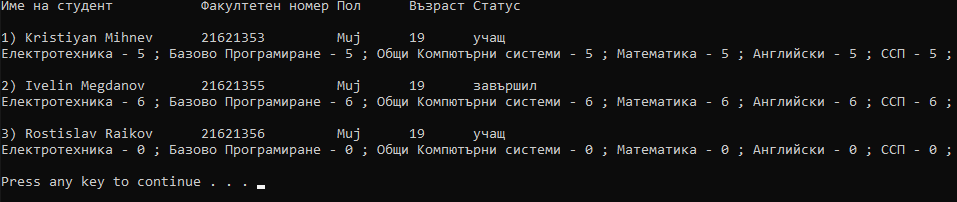


Условие F

Снимка на изгледа с примерни входни данни

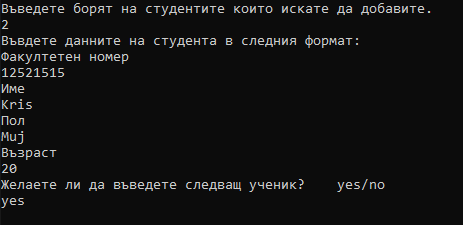


Снимка на изгледа с примерни изходни данни



Условие H (по желание)

Снимка на изгледа с примерни входни данни



Условие H (по желание)

Снимка на изгледа с примерни входни данни

