**Курсов проект по**

**Обектно-ориентирано**

**програмиране.**

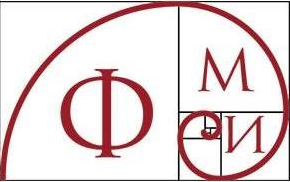
**Тема:** Информационна система за обслужване на студенти СУСИ.

**Изготвил:** Алкан Ридван Мустафа

**Специалност:** Информационни системи,I курс, I група.

**ФН:** 72018

***Github:*** [alkan369](https://github.com/alkan369)



**Факултет по Математика и Информатика**

**Съдържание**:

**Глава 1:Увод**

1.1.Описание и идея на проекта

1.2.Цел и задачи на разработката.

1.3.Структура на документацията

**Глава 2: Преглед на предметната област**

2.1.Основни дефиниции,концепции и алгоритми,които ще бъдат използвани.

2.2.Дефиниране на проблеми и сложност на поставената задача

2.3.Подходи,методи за решаване на поставените проблеми

**Глава 3: Проектиране**

3.1.Обща архитектура – ООП дизайн

3.2.Диаграми(на структура и поведение – по обекти,слоеве с най-важните извадки от кода)

**Глава 4: Реализация,тестване**

4.1.Реализация на класове

4.2.Управление на паметта и алгоритми.Оптимизации.

4.3.Планиране,описание и създаване на тестови сценарии(примери)

**Глава 5: Заключение**

5.1.Обобщение на изпълнението и началните идеи

5.2.Бъдещо развитие и усъвършенстване

**Използвана литература**

**Глава 1 : Увод**

**1.1.Описание и идея на проекта.**

Проектът представлява информационна система за обслужване на студенти.Проектът трябва да има възможността да запише студент с факултетен номер в специалност,да може да го отписва,да му сменя специалността,да има възможността да прекъсне следването на студента и евентуално по-късно пак да продължи следването му,също така студентът да има право да запише избираем предмет,да може студентът да се явява на изпити и да получи оценки,да може да се смята средния успех на студента и др.Също така програмата трябва да може да запише(сереализира) данните на студента във файл и да чете от файл(десереализира).

**1.2.Цел и задачи на разработката.**

Целта е самата програма да бъде един вид симулатор на СУСИ,където се запазват данните на съответен студент,като се предостави възможността тези данни да бъдат променяни ако е нужно.В програмата да може от командния ред да се въвеждат и осъществяват следните операции :

* Да може програмата да отваря даден файл и да чете от него
* Да може да затваря даден файл след като е приключена работата с него
* Да може да запазва данни във вече отворен файл
* Да може да запазва данни във някой друг файл по желание
* Да поддържа команда („Help“),което показва какви команди поддържа програмата
* Да поддържа команда за изход,която прекратява/приключва програмата
* Да поддържа команда за записване на студент в 1 курс с име и факултетен номер
* Да поддържа команда за записване на студент в следващ курс
* Да поддържа команда ,която дава възможност на студент да смени група,специалност или курс
* Да поддържа команда,която отбелязва студента като завършил
* Да поддържа команда ,която маркира студента като прекъснал
* Да поддържа команда,която възстановява студентските права на прекъснал студент
* Да поддържа команда,която извежда справка за даден студент
* Да поддържа команда,която извежда справка за всички студенти от дадена специалност и курс
* Да поддържа команда,която дава възможност на студент да запише избираема дисциплина
* Да поддържа команда,която добавя оценка по даден предмент на съответния студент
* Да поддържа команда,която извежда протоколи за всички студенти ,записали дадена дисциплина

Да поддържа команда,която извежда академична справка за оценките на даден студент и среден успех

**1.3.Структура на документацията**

Документацията е структурирана по начин,по който ще се представи всеки отделен клас и всички член-данни и методи на съответния клас.Както и ще бъдат представени срещнатите проблеми повреме на разработката на проекта както и методите на тяхното преодоляване.Ще се представи и общата архитектура на целия проект,тестове върху отделните класове(юнит тестове).Ще се обърне внимание и на възможните бъдещи оптимизации,които могат да бъдат направени.

**Глава 2 : Преглед на предметната област**

**2.1.Основни дефиниции и концепции и алгоритми ,които ще бъдат използвани.**

Главното място ,където ще се изпълняват всички команди на програмата,е **класът SUSI**,който се води главния клас,където са включени всички останали класове,които участват в изграждането на цялостната програма.Ще се използват основните концепции на ООП-то както и главните принципи при работа с динамична памет

**2.2.Дефиниране на проблеми и сложност на поставената задача.**

В задачата се изисква за всеки студент да се съхранява следната информация :

* Име с произволна дължина
* Факултетен номер
* Текущо записан курс,специалност,група
* Статус(записан,прекъснал,завършил)
* Среден успех от следването до момента

Учебни дисциплини,които се характеризират с име и тип(задължителна или избираема)

Всяка специалност да има списък от дисциплини и курс,в който могат да бъдат записвани

За всеки студент да се пазят всички записани дисциплини и оценки,по които са положени изпити.

**2.3.Подходи ,методи за решаване на поставените проблеми.**

Първоначално бяха създадени 2 класа (Стринг и Вектор) ,които представляват динамични контейнери ,на които ще се базират други класове,които са фундаментални за програмата.

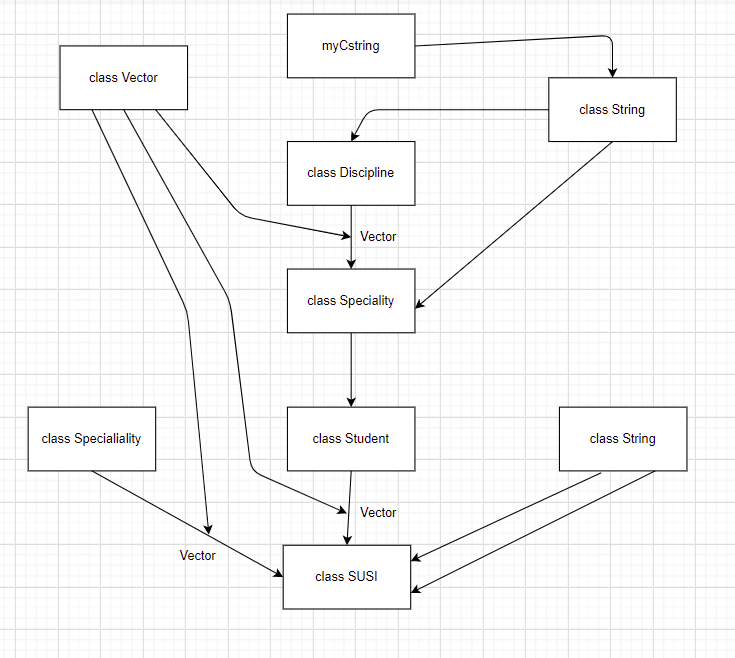
**Глава 3: Проектиране**

**3.1. Обща архитектура – ООП дизайн.**

Програмата се състои от собственоръчно създаден Стринг и Вектор,myCstring(в него се състоят функциите myStrCpy(за копиране на един низ в друг),myStrcmp(сравняване на два низа един с друг),myStrlen(намиране на дължината на един низ).

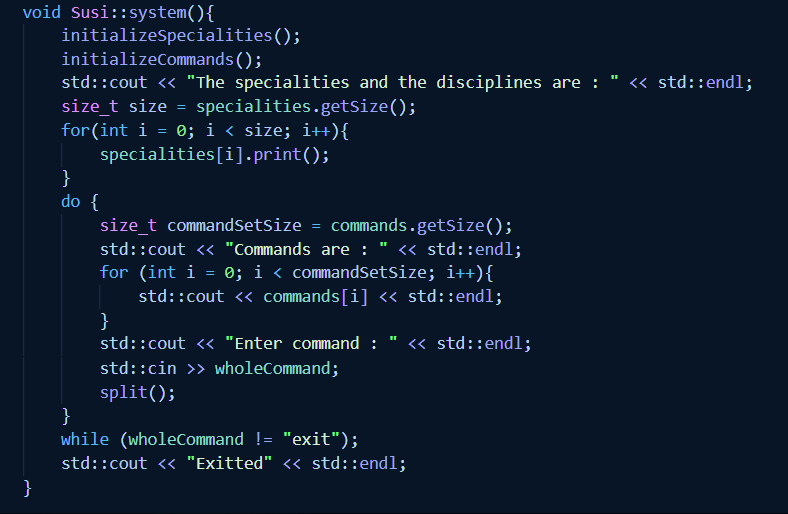
Както и клас Студент,клас Специалност и клас Дисциплина,и **клас СУСИ**,където са обединени всички останали класове.

**3.2 Диаграми (на структура и поведение – по обектимслоеве с най-важните извадки от кода)**



На диаграмата е представена архитектурата на цялата програма.

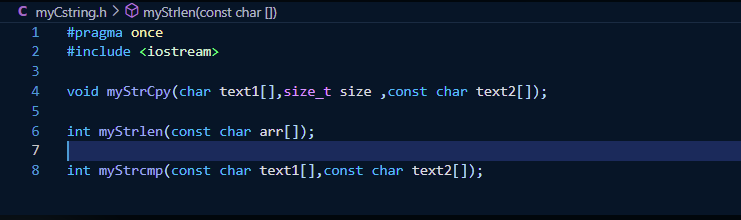
myCstring се използва само в класа Стринг.Класът Стринг и класът Вектор се използват в клас Дисциплина,а самият клас Дисциплина се използва като Вектор(вектор от дисциплини) в класа Специалност,използва се и клас Стринг.В класа Студент участва клас Специалност като член-данна.Класа СУСИ е мега клас ,където участват като член-данни клас Специалност(Вектор от специалности), две член данни от клас Стринг,както и клас Студент(Вектор от студенти).

Четенето на команди става по следния начин: 

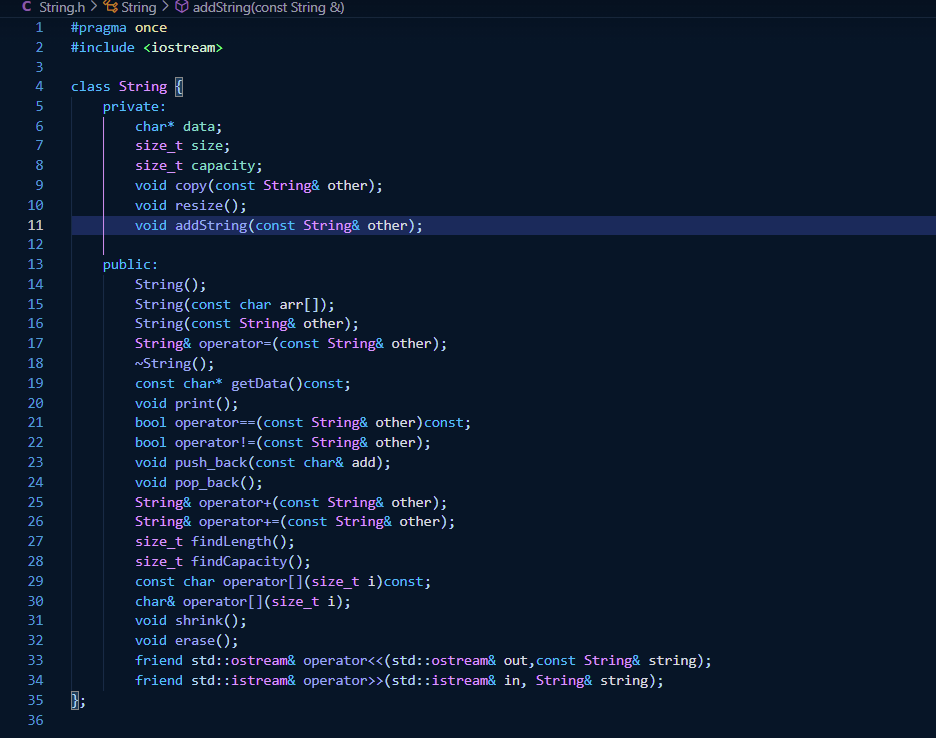
С един do while цикъл обхожда командния ред,докато не се въведе “exit”,след което програмата приключва,а ключовите команди от цялото съобщение се отделят в метода “split();”.

**Глава 4: Реализация,тестване**

4.1.Реализация на класове.

**myCstring :** 

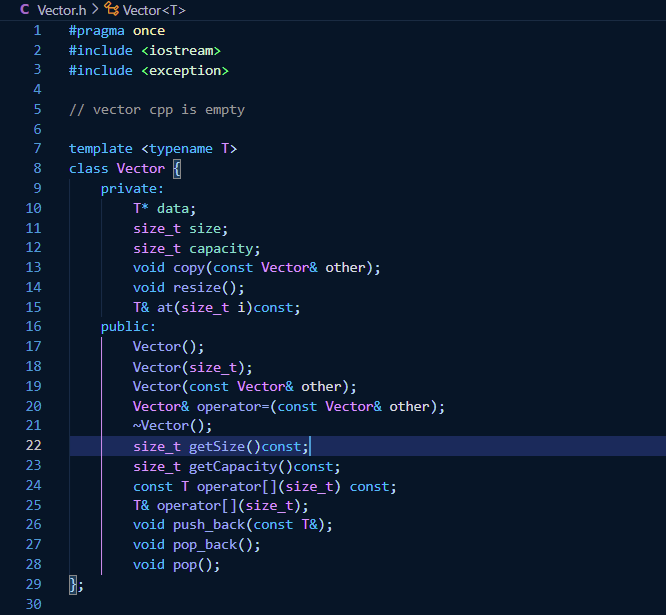
Това не е клас ,по-скоро са 3 глобални функции: myStrCpy,който приема 2 символни низа и размер и копира втория символен низ в първия(копира size елемента от втория в първия).myStrlen приема символен низ и измерва неговата дължина и го връща като резултат.myStrcmp сравнява два символни низа(дали са еднакви) и връща число като резултат : ако върнатия резултат е 0,това означава ,че низовете са еднакви,в противен случай (!= 0) символните низове са различни.

**Клас Стринг :** 

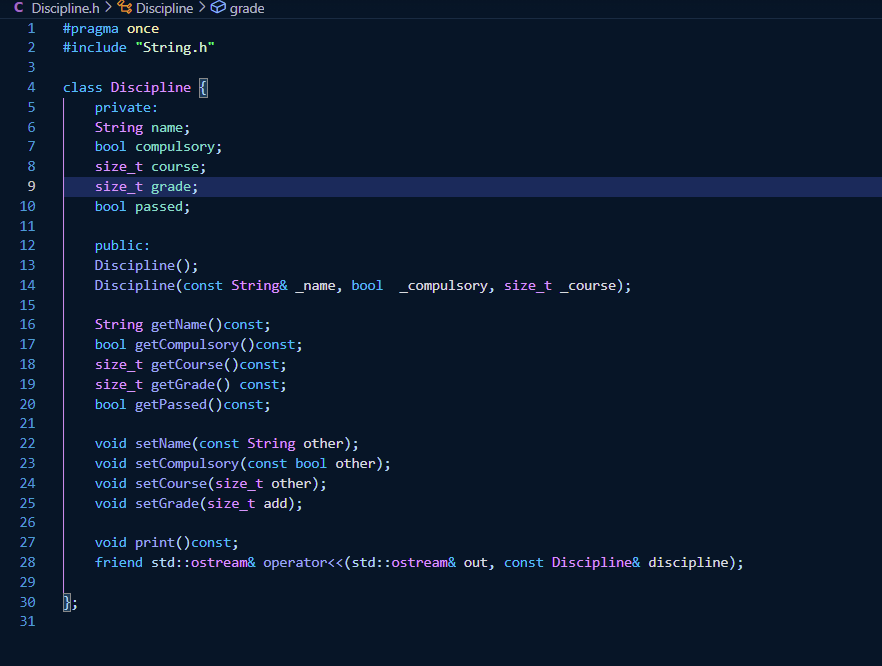
Класът стринг има член-данни char\* data(указател от тип char), цяло неотрицателно число за размер и капацитет.В класа има работа с динамична памет и затова е имплементирана **Голяма Четворка**.В дефолтния конструктор,капацитетът е равен на 8,а размерът на 0 и се създава динамичен масив с размер 8.Констуркторът с параметри приема символен низ,създава се динамичен масив с капацитет размерът на символния низ \* 2 и се копира подадения низ.Копи конструкторът създава динамичен масив и копира размера и капацитета на подадения Стринг и създава динамичен масив,след което копира данните му 1 по 1.

Операторът = прави същото,но преди това се прави проверка дали двата обекта не сочат един и същи адрес,т.е a = a,което в резултат ще унищожи обекта.Поради това,че се повтаря код общата част от копи конструктора и оператор = е изнесена в private частта в отделен метод copy(const String& other); .

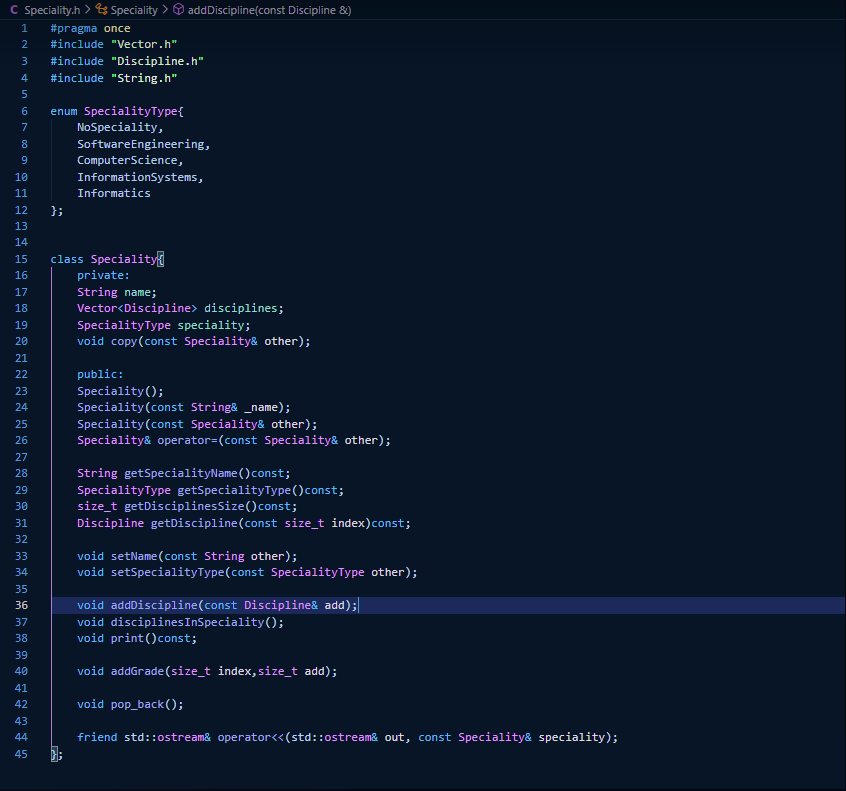
Накрая има и деструктор ,който изтрива динамичната памет след като вече не я използваме.Класът има също така гетър за символния низ(използва се в fstream).Има метод за печатане на символния низ,оператор == и != (за проверка дали подадените Стрингове за съответно еднакви и различни),метод push\_back(const chat& add) и pop\_back() ,които добавят и махат символ от края на масива.Предефинирани оператор + и оператор +=,които свързват два Стринга.Методи findLength() и findCapacity(),които връщат размера и капацитета на Стринга.Предефинирани оператор [] (единия за четене на отделните части от масива,а другия за промяна на отделните части от масива).Метод shrink()(сваля капацитета до размера + 1) и медор erase()(изтрива данните на вече създадения динамичен низ и създава нов празен със същия капацитет и нулира размера).Предефинирани са и оператор << и оператор >> за входен и изходен поток.

**Клас Вектор :** 

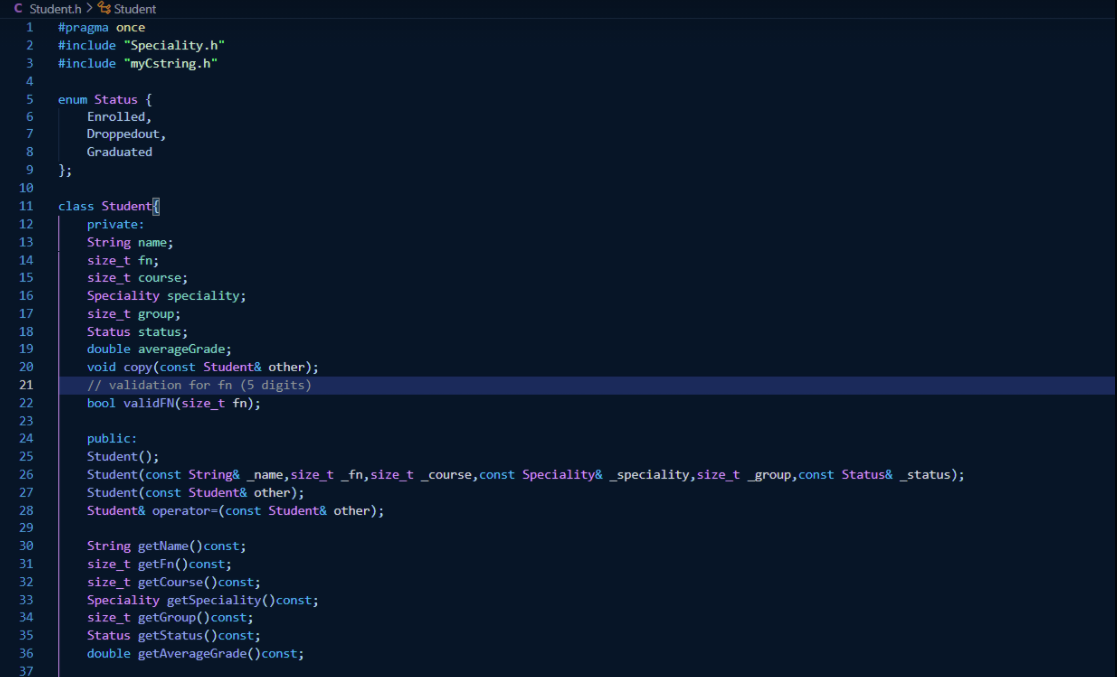
Класът Вектор е шаблонен клас и има подобни член данни като Стринга само,че data е от тип Т,т.е се създава динамичен масив от тип Т.Съответно пак са имплементирани **Голяма Четворка**(работим с динамична памет).Дефолтния конструктор приравнява капацитета на 8,а размера на 0 и създава празен динамичен масив с капацитет 8.Конструкторът с параметри приема размер от цяло неотрицателно число и създава празен масив с капацитет равен на това число.Съответно копи конструкторът приравнява капацитета на този на подадения Вектор и създава динамичен масив ,и копира елементита 1 по 1 от подадения Вектор(при всеки добавен елемент размерът се увеличава + 1 като започва от 0).Съответно оператор = се прави провека дали адресите на двата обекта са еднакви ,т.е а = а ,за да не се изтрие обекта.Общия код е изнесен в private частта от класа в метод copy(const Vector& other); . Деструкторът изтрива динамичния масив тогава когато вече не ни трябва.Създадени са методи(гетъри),които връщат размера и капацитета на Вектора.Предефинирани са операторите [] ,както за четене поотделно на всеки един елемент от Вектора,така и за промяна на всеки един елемент от Вектора поотделно.Класът има също и методите : push\_back(const T& add) за добавяне на елемент от тип Т в края на масива, pop\_back(); за триене на последния елемент от масива и метод pop(); ,който трие данните на масива и създава нов динамичен масив със същия капацитет и нулиран размер.

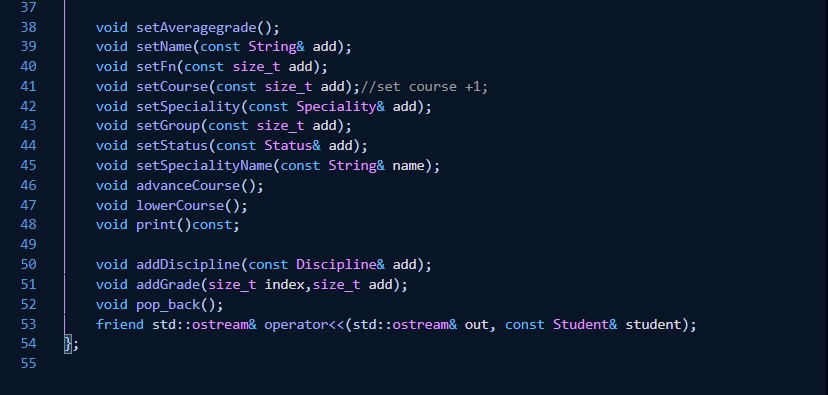
**Клас Дисциплина :** 

Класът има член-данни Стринг за името на дисциплината,булева променлива дали е задължителна дисциплина,цяло неотрицателно число за това в кой курс се провежда/записва специалността,цяло неотрицателно число за оценката по дисциплината и булева променлива дали специалността е мината.Методите на класа са Дефолтен конструктор и констурктор с параметри,при който не се подават оценка и булев израз дали е мината дисциплината.По default оценката е 2 и булевата променлива passed е false,оценката се променя чрез сетъра setGrade(size\_t add); (там след като се промени оценката автоматично се сменя променливата дали дисциплината е мината или не).Класът има и принт метод,както и предефиниран оператор за изходен подок <<,както и гетъри за отделните член-данни,така и сетъри.

**Клас Специалност :** 

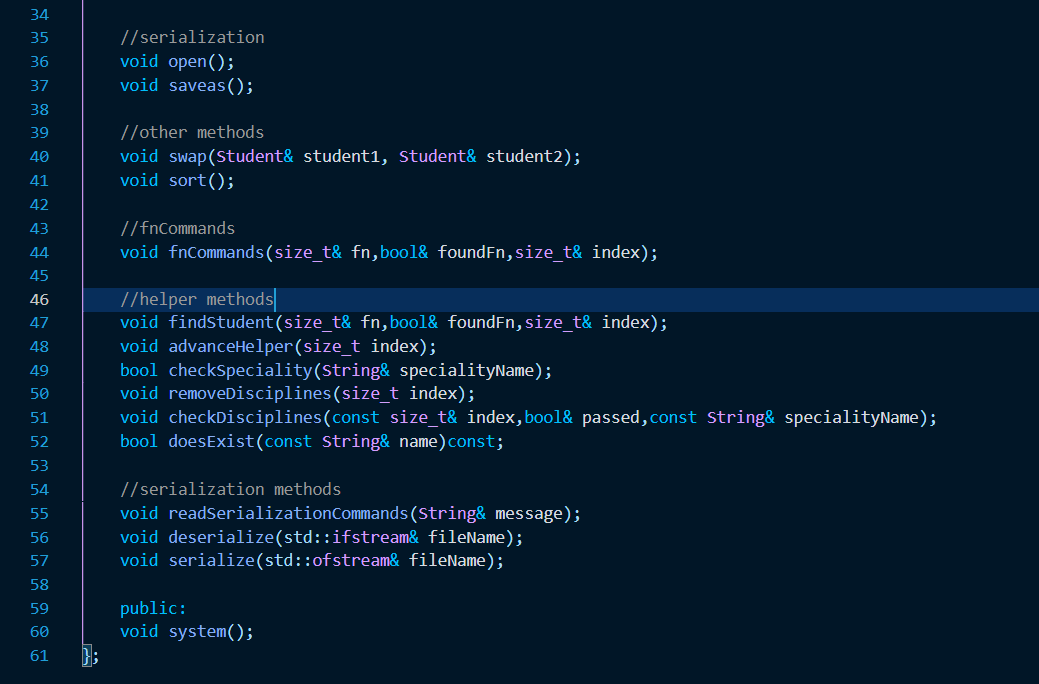
Класът има член-данни Стринг за името(произволна дължина),дисциплини(Вектор от дисциплини),вид на специалността(specialityType),който е enum(ограничаваме системата СУСИ до 4 специалности).Дефолтния конструктор дава името на специалността “No Name”, и за specialityType = NoSpeciality,конструкторът с параметри приема Стринг за името и създава специалност с това име и пак specialityType = Nospeciality.Конструкторът за копиране и оператор = копират подадената Специалност other и общия код е изведен в прайвет частта на класа в метод copy(const Speciality& other),където обектът присвоява името на специалността,вида специалност(specialityType) и вектора пт дисциплини на подадената специалност.Класът има гетъри за името нас специалността,за вида на специалността,за броя дисциплини,които има в тази специалност и гетър за достъпване на определена специалност по подаден индекс.Класът има също и сетъри за име и за вид на специалността(тук се зареждат определените дисциплини в зависимост от това какъв вид е специалността).Метод addDiscipline(const Discipline& add); за добавяне на специалност.Метод print()const; за печатане на специалността,метод за добавяне на оценка на съответната специалност по подаден индекс addGrade(size\_t index,size\_t add); ,метод за изтриване на специалност pop\_back() (изтрива последната специалност от Вектора).Класът има и предефиниран оператор << за изходен поток.

**Клас Студент :** 



Класът Студент се състои от член-данни Стринг за име(с произволна дължина),цяло неотрицателно число за факултетния номер,за курса както и за групата на студента.Класът има също и член-данна специалност от тип Специалност,статус от тип Статус(enum дали студентът е прекъснал,записан или завършил),среден успех от тип double.Дефолтния конструктор на студента дава стойност за името “No name”,факултетен номер = 10000,група = 0, статус = прекъснал и среден успех = 0.Конструкторът с параметри приема Стринг за името,цяло неотрицателно число за факултетния номер,курс и група,специалност от тип Специалност,статус от тип Статус, а средния успех се приравнява на 0.В конструктора се прави проверка дали факултетния номер е валиден тоест е 5-цифрено число,ако не е,то се хвърля грешка.Копиращия конструктор и оператора = копират подадения студент,като общия код е пренесен в прайвет частта в метод с име copy(const Student& other); .Класът има гетъри за името на студента,за факултетния номер,за курса,за специалността,за групата,за статуса и за средния успех.Класът има и сетъри пресмятане на средния успех на студента като се изчислят всички оценки от всички дисциплини и се раздели на броя на дисциплините,сетър за името,за факултетния номер(прави се проверка дали факултетния номер е валиден,т.е 5-цифрено число и ако не е 5-цифрено,то се хвърля грешка),сетър за курса,за специалноста,за групата,за статуса,за име на специалността(сменя името на специалността).В класа се съдържат и методите advanceCourse(); увеличава курса на студента + 1, lowerCourse(); намаля курса на студента -1, метод print(); за печатане.Също и методи addDiscipline(const Discipline& add);,който добавя нова дисциплина в края на вектора от дисциплини,addGrade(size\_t index,size\_t add); ,който добавя подадена оценка по съответната дисциплина по подаден индекс.Накрая в класа има и предефиниран оператор << за изходен поток.

**Клас СУСИ :** 

****

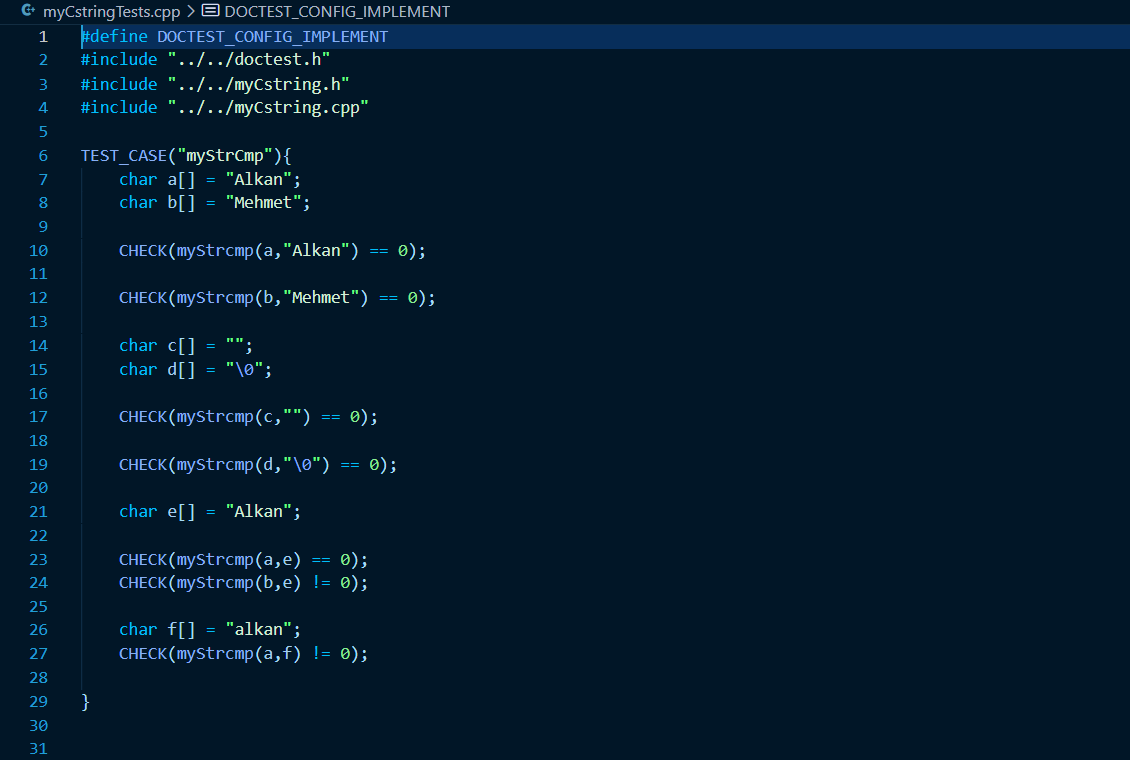
Класът СУСИ е мега класът,където са включени всички останали класове и където се случва цялото действие на програмата.Всички член-данни и методи на класа са прайвет освен методът system(); .Член-данните са вектор от студенти,вектор от специалности(те са предварително заредени с определен брой дисциплини),вектор от стрингове за командите(тук са предварително заредени командите,които системата поддържа),Стринг за цялата въведена команда и Стринг за ключовата дума от командата ,въз основа на която се определя коя команда да се изпълни.Методите initializeSpecialities(); и initializeCommands(); зареждат специалностите и командите с съответните дисциплини и команди,които се поддържат от програмата.Методът split(); разделя ключовата дума от цялата въведена команда.Булевият containsSpace(); проверява дали въведеното съобщение има интервал,методът amountOfSpaces(); проверява колко интервала се съдържа в подаденото съобщение.От методът commandCases(); се решава коя команда да се изпълни в зависимост от подадената ключова дума.Командните методи за СУСИ са : enroll(); записва в 1 курс даден студент с факултетен номер в дадена специалност в дадена група.advance(); записва студента в следващ курс,graduate() отбелязва студента като завършил,interrupt(); маркира студента като прекъснал,resume(); възстановява студентските права на определен студент,print(); извежда справка за определен студент,printall(); извежда справка за всички студенти в дадена специалност и курс,enrollin(); записва съответния студент в избираема дисциплина,addgrade(); добавя оценка на съответния студент по съответния предмет,protocol(); извежда протоколи за всички студенти,записани в дадена специалност(като факултетните номера на студентите са сортирани в нарастващ ред),report(); извежда академична справка за оценките на даден студент и неговия среден успех.Методите за сериализациа са : open(); ,който отваря вече създаден файл и чете данните от него(десереализира) или създава нов празен и saveas(); ,който запазва данните от програмата във файл(сереализира).Методът system(); е единствения ,който се намира в public частта на класа и там се подават съобщенията и се извикват нужните методи,които допринасят за правилното изпълнение на цялостната програма.

4.2.Управление на паметта и алгоритми.Оптимизации.

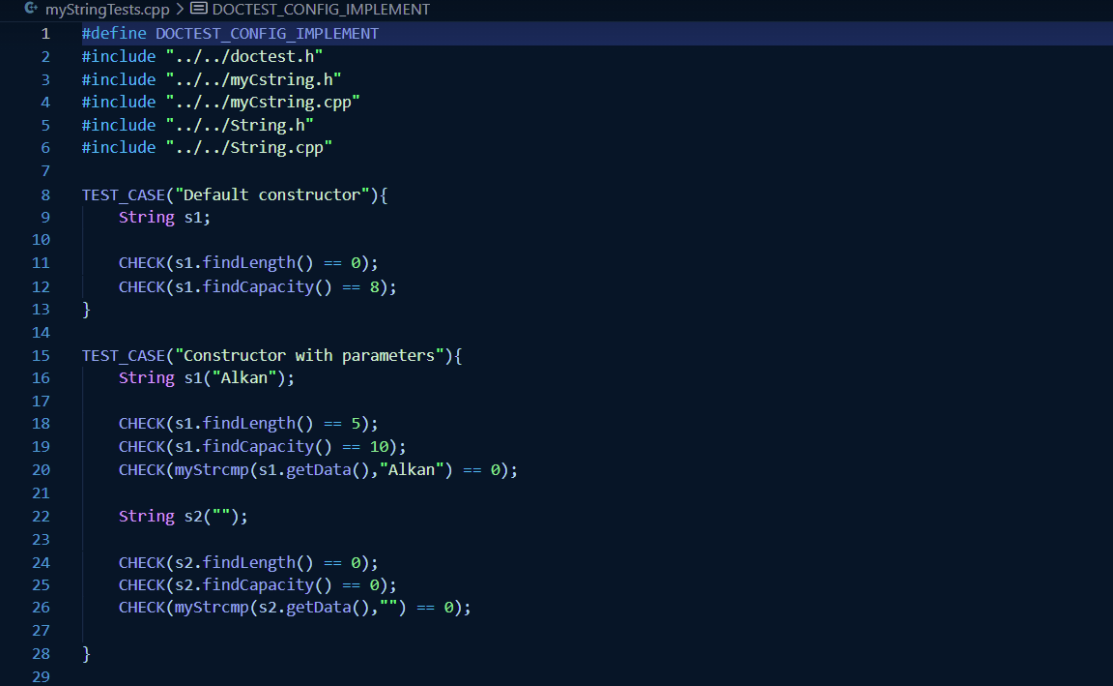
В програмата управлението на паметта се базира на написаните класове Вектор и Стринг,използваната памет е динамична и когато паметта вече не е нужда,тя се изтрива от деструкторите от само себе си.Тоест ние почти не се занимаваме с управлението на паметта в програмата,нито с нейното заделяне,нито с нейното изтриване,т.е е вдигнато нивото на абстракция.В метода printall(); в класа СУСИ за сортиране е използван алгоритъмът bubble sort(методът на мехурчето).Някои от многото направени оптимизации по програмата са : в класа Студент за името char\* name е заменен със String name,което ни дава възможността да махнем деструктора и да не се занимаваме със заделяне,управление и изтриване на памет,тъй като вече имаме класове за това (Стринг и Вектор).По този начин е вдигнато нивото и на абстракцията.Всички асътри(от библиотеката cassert) са заменени с изключения,които се хвърлят на подходящите за това места.Всички методи,които съдържат повтарящ се код или имат нещо общо помежду си са преместени в отделен общ метод.

4.3.Планиране,описание и създаване на тестови сценарии(създаване на примери)

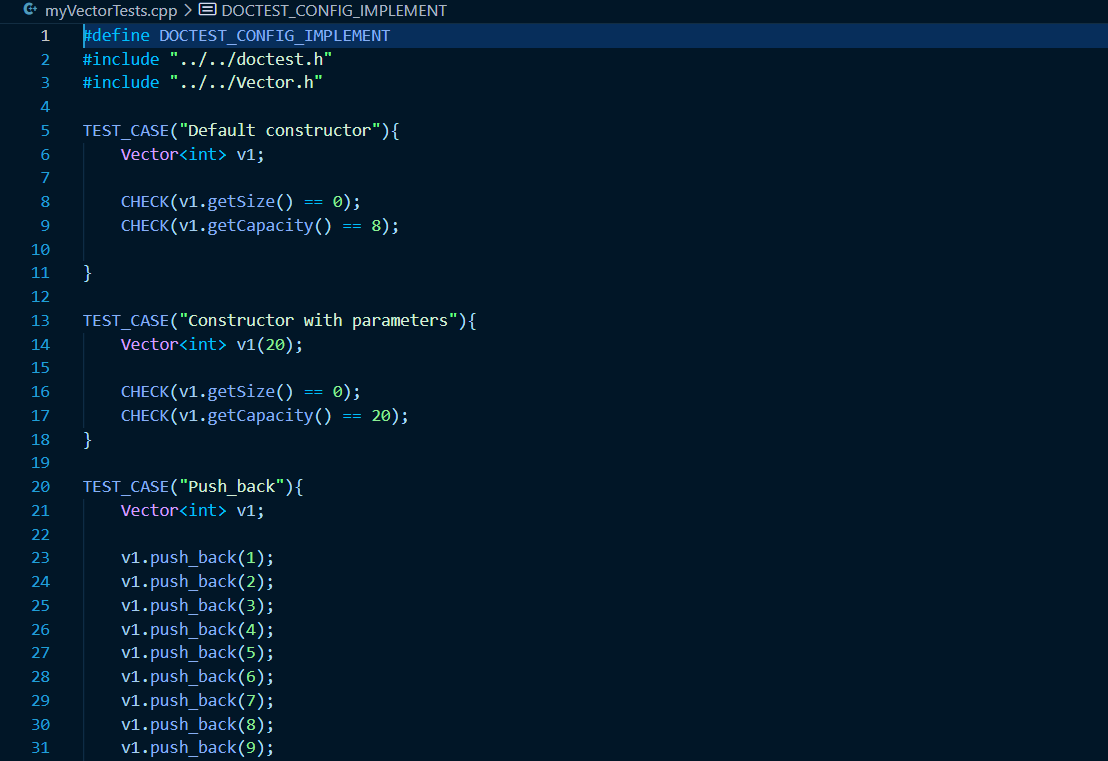
Направени са юнит тестове,за myCstring и за класовете Стринг, Вектор, Дисциплина,Специалност и Студент.Тоест на всички класове,които участват в мега класа СУСИ.За направата на тестовете е използван doctest.h.

**myCstring :**

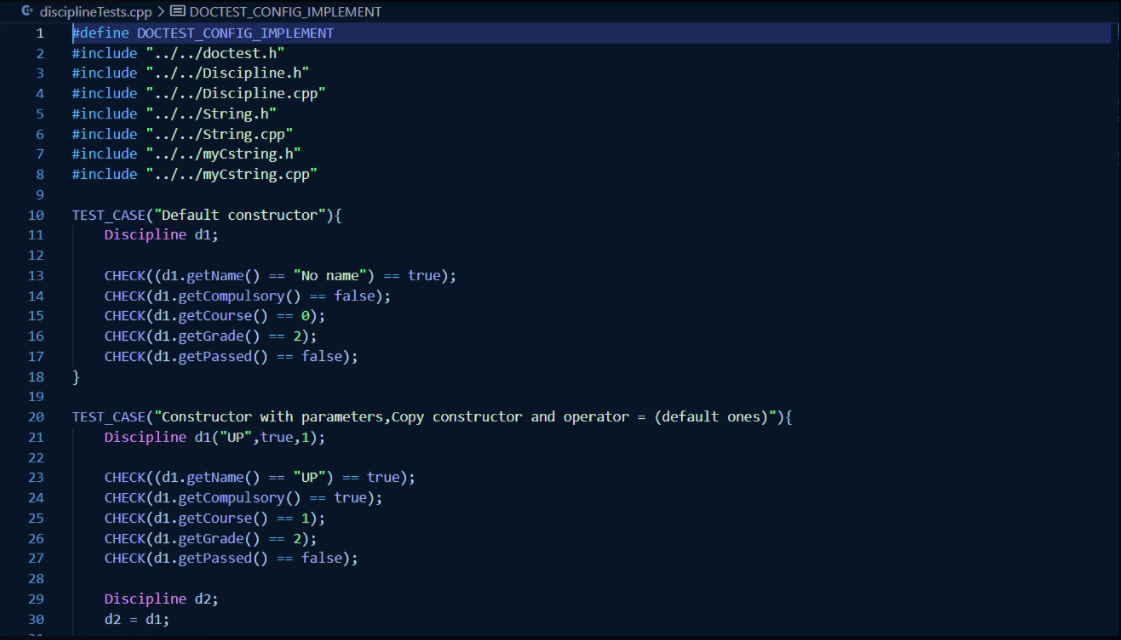
Това са част от тестовете,които са направени за myCstring,общо тестовете ,направени за него са 16 като всички са минати успешно.

**Клас Стринг :** 

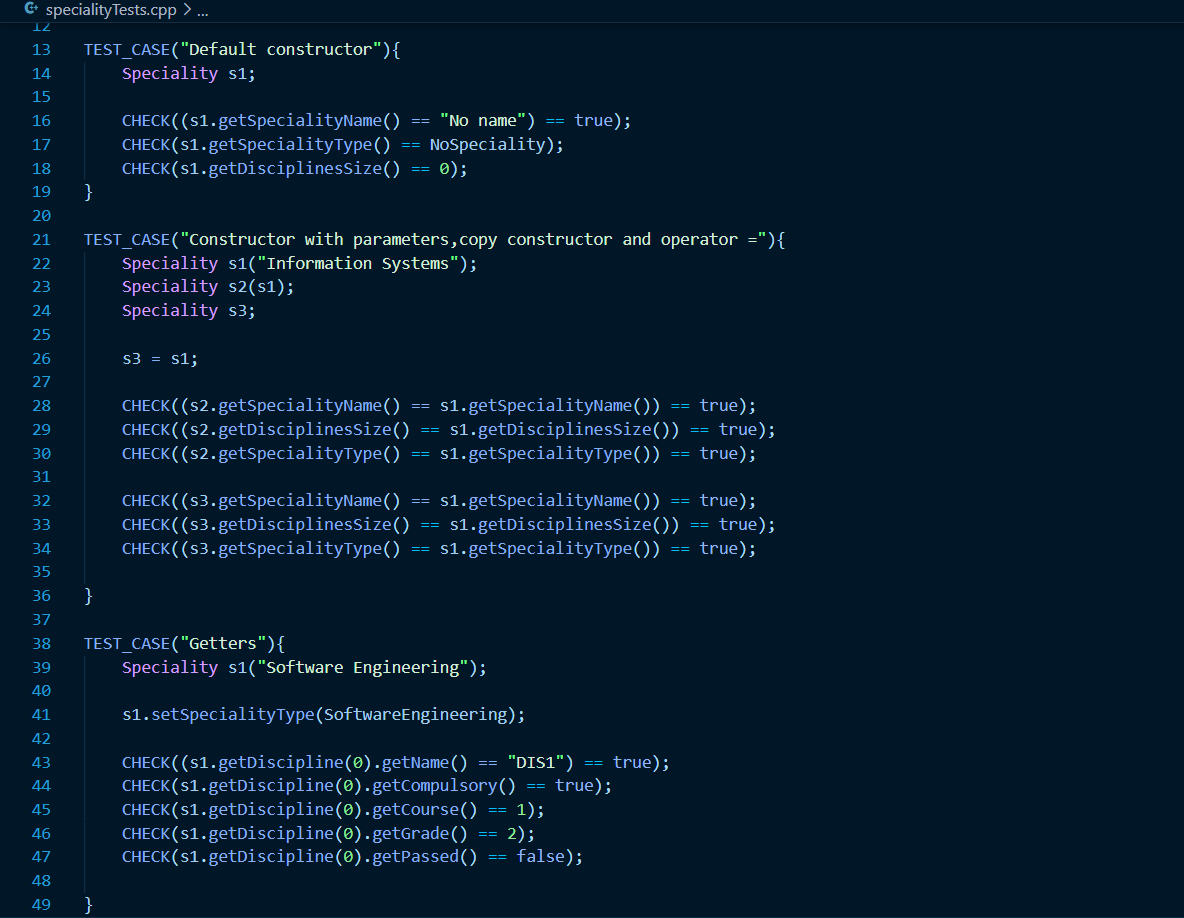
Това са част от тестовете ,направени за класа Стринг,общо тестовете,направени за този клас са 55 като всички са минати успешно.

**Клас Вектор :**

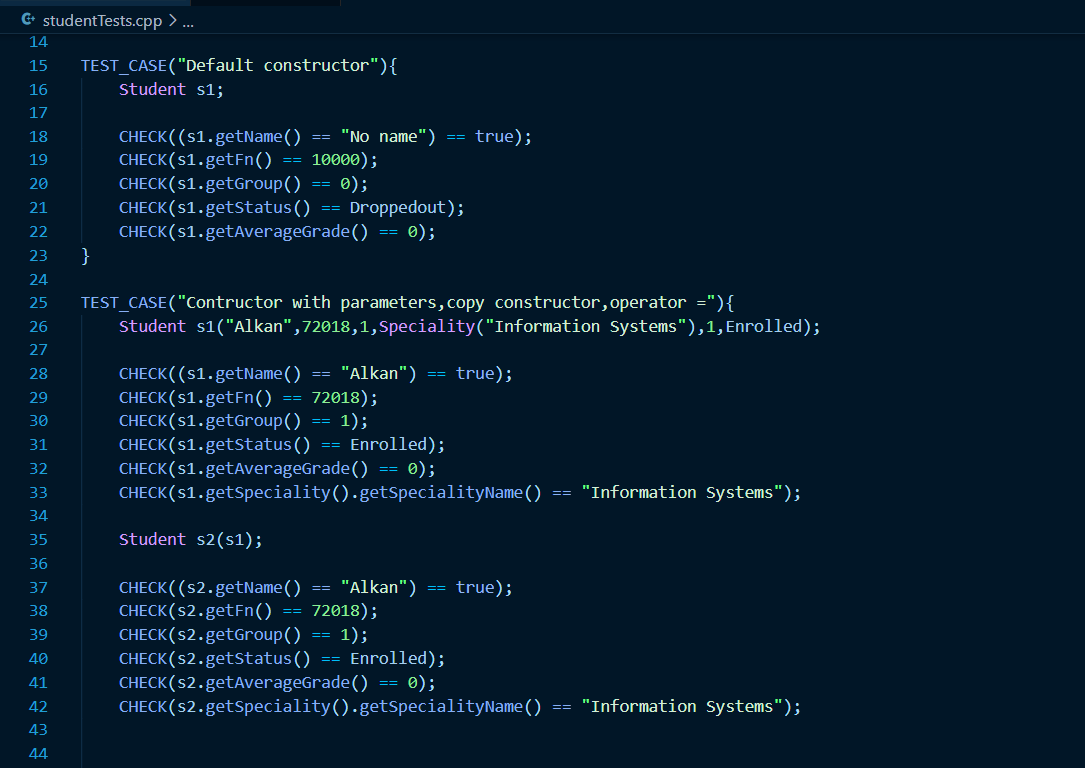
Това са част от направените тестове за класа Вектор,общо всички направение тестове за този клас са 43 като всички са минати успешно.

**Клас Дисциплина :** 

Това са част от направените тестове за класа Дисциплина,като всички направени тестове за този клас са общо 32 като всички тестове са преминати успешно.

**Клас Специалност :**

Това са част от направените тестове за клас Специалност,като общо направените тестове са 20 и всички са преминати успешно.

**Клас Студент :**

Представените тестове са само част от всички,направени за класа Студент,общо всички направени са 50 и всички са преминати успешно.

Като цяло всички направени юнит тестове върху всички имат общ брой 200 и всички са преминати успешно.

**Глава 5 : Заключение**

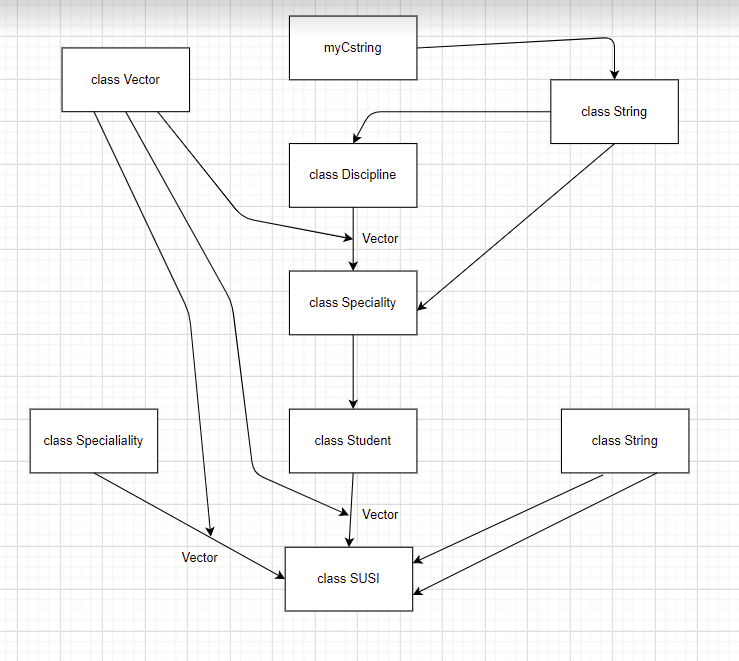
5.1.Обобщение на изпълнението на началните цели

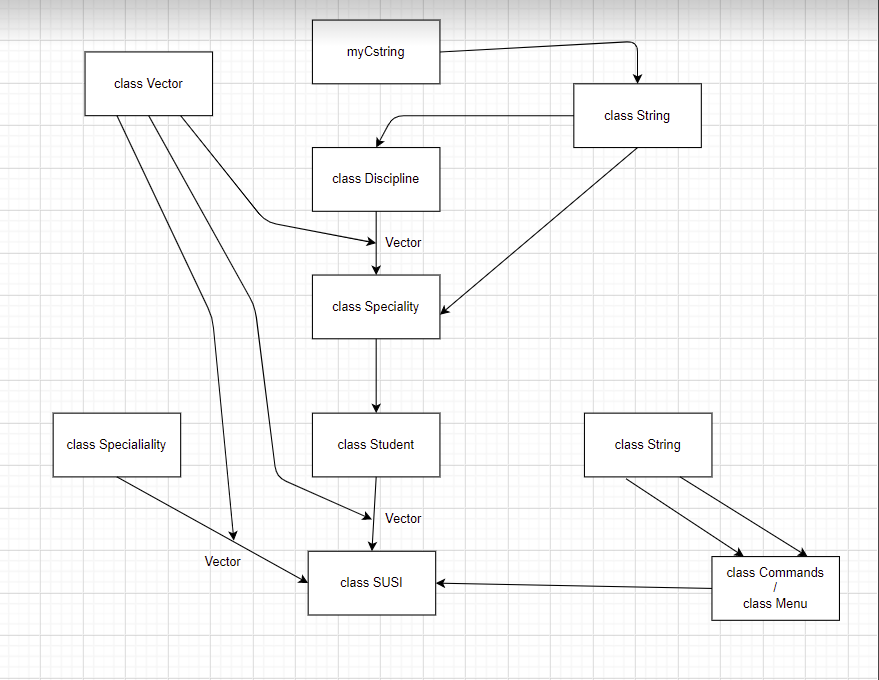
Като цяло всички всички начални цели бяха изпълнени,само някои от тях бяха променени и то с цел оптимизация на програмата и тн.

5.2.Насоки за бъдещо развитие и усъвършенстване

Едно от нещата,които може да се направи с цел промяна и оптимизация на текущата програма е : самите команди(които се намират в клас СУСИ) и четенето на командите да се изнесат в отделен клас Команди/клас Меню(class Commands/class Menu) това ще скъси много кода,който се намира в срр и хедър файла на клас СУСИ.

Съответно самата програма да придобие следния вид :

Текущ вид : 

Евентуален оптимизиран в бъдещето вид : 

**Използвана литература :**

**…..**