Линк към хранилище в Github: <https://github.com/danisini/SUSI>

**Проект 13: СУСИ**

1. **Увод**

**Описание и идея на проекта**

Разглежданият проект реализира информационна система за обслужване на студенти. Тя съхранява и обработва необходимите данни във файл както за студентите, така и за предметите, които се изучават. Данните за студентите са име, факултетен номер, курс, специалност, група, статус( записан, прекъснал, завършил) и среден успех. Учебните дисциплини се характеризират с тип ( задължителна/ избираема). Спрямо специалността си всеки студент може да запише определени дисциплини, като за тях се пазят и оценките, получени при изпитите им, ако е положен такъв.

**Цели и задачи на разработката**

* Подходящо представяне на студентите, специалностите и дисциплините
* Подходящо представяне и на системата СУСИ, която ще обработва данни
* Изграждане на леснодостъпно меню.
* Реализиране на основни функционалости на проекта ( отваряне, затваряне, записване, помощно меню, изход от програмата)
* Реализиране на допълнителни фунцкионалости на проекта ( добавяне на студенти, добавяне на дисциплини, добавяне на оценки, отчети и др. )
* Опция за записване на промените само при написването на команди save/ save as

**Структура на документацията**

Документацията е структуриране в следните основни точки:

1. Увод
2. Проектиране
3. Реализация
4. **Проектиране**

По време на работата се стреми да се следват добрите принципи и практики на ООП дизайна. Реализирани са следните 6 класа:

* Subject
* Program
* Student
* Susi
* String
* Vector <T>

Те имат следната йерархия:

* **Клас Subject**: реализира учебните дисциплини

**Private** член-данни:

* String name, program; // име на дисциплината и специалността, в която се учи
* bool type; // задължителна/избираема
* size\_t year; // година, в която може да се учи
* double mark; // оценка на ученика

**Public** член-функции:

* void set\_mark(const double& mark\_){mark = mark\_;} – записва оценката от положения изпит
* const double get\_mark()const{return mark;} – взима оценката за предмета
* const size\_t get\_year()const{return year;} – взима годината, в която се учи предметът
* const String get\_program()const{return program;} – взима специалността, в която се учи дисциплината
* const String get\_name()const{return name;} – взима името на дисциплината
* const bool get\_type()const{return type;} – взима типа на предмета
* Subject(String, String, const bool&, const size\_t&); - конструктор
* void construct(String, String, const bool&, const size\_t&); - функция, инициализираща обект
* void print()const; - извежда данните за предмета
* Subject& operator=(const Subject&); - оператор=
* bool operator==(const Subject&); - оператор за проверка дали два предемета са еднакви
* **Клас Program**: реализира специалностите в университета

**Private** член-данни:

* String programName; // име на специалността
* Subject\*\* subjects; // пойнтър към масив от пойнтъри от предмети
* size\_t numSubjects; // брой предмети
* Vector <size\_t> indexOfSubjects; // динамичен масив от индексите на тези предмети от subjects, които са възможни за записване от тази специалност

**Private** член-функции – помощни функции за копиране и изтриване на памет

* void copyProgram(const Program&);
* void deleteProgram();

**Public** член-функции:

* Program(String, Subject\*\*, const size\_t&); // конструктор
* Program(const Program&); // копиращ конструктор
* Program& operator=(const Program&); // оператор за присвояване на обекти от тип Program
* ~Program(); // деструктор
* const String get\_program()const{return programName;} // връща името на специалността
* void set\_program(const String& programName\_){programName = programName\_;} // записва ново име на специалността. Използва се при прехвърляне в друга специалност.
* Subject\* get\_subject(const size\_t& i)const{return subjects[i];}// връща дисциплина с индекс i
* const size\_t get\_numSubjects()const{return numSubjects;} // връща броя на предметите в специалността
* void changeProgram(const Program&);// извършва прехвърляне в специалността
* **Клас String**: реализира динамичен масив от символи

**Private** член-данни:

* char\* str; - пойнтър към масив от символи
* size\_t sz; - големина на масива

**Private** член-функции:

* void delete\_string();
* void add\_char(const char);

**Public** член-функции:

* String(const char\* = ""); - конструктор
* String(const String&); - копиращ конструктор
* String& operator=(const String&); - оператор равно
* ~String(); - деструктор
* size\_t size()const; - метод за връщане на големината на масива
* bool operator==(const String&)const; - оператор ==
* bool operator<(const String&)const; - оператор <
* String operator+(const String&)const; - опеартор +, добавящ стринг
* String operator+(const char&)const; - опеартор +, добавящ стринг
* String& operator+=(const String&); - опеартор +=, добавящ стринг
* String& operator+=(const char&); - опеартор +=, добавящ стринг
* char& operator[](const size\_t&); - осигурява пряк достъп до елемент на масива
* bool operator<=(const String&)const; - оператор <=
* bool operator>=(const String&)const; - оператор >=
* bool operator>(const String&)const; - опеартор >
* const char\* stringToChar(){return str;} – преобразуваща член-фунцкия от String към char\*;
* friend std::istream& operator>>(std::istream& is, String& str); - приятелска функция за въвеждане на String;
* friend std::ostream& operator<<(std::ostream& os, const String& str); - приятелска функция за извеждане на String
* friend std::istream& getline(std::istream& is, String& str); - приятелска функция за въвеждане на String до въвеждане на нов ред
* **Клас Student:**

**Private** член-данни:

* size\_t fn, year, groupNum, status; // факултетен номер, година на следване, номер на група, статус
* String name; // име на студента
* double averMarks; // среден успех
* Program program; // име на специалността
* Vector<int> indexOfTakenOnes; // динамичен масив от индексите на тези предмети, които студентът е записал

**Public** член-функции:

* Student(); - конструктор по подразбиране
* Student(String, const size\_t&, const size\_t&, const size\_t&, const size\_t&, const Program&); - конструктор с параметри
* Student& operator=(const Student&); - оператор за присвояване
* const size\_t get\_groupNum()const{return groupNum;} - връща номера на групата
* const size\_t get\_year()const{return year;} - връща в кой курс е студента
* const size\_t get\_status()const{return status;} - връща статуса на студента
* const double get\_averMarks()const{return averMarks;} - връща средния успех на студента
* const size\_t get\_fn()const{return fn;} - връща факултетния номер на студента
* const String get\_name()const{return name;} - връща името на студента
* const String get\_programName()const{return program.get\_program();} - връща името на специалността, в която студентът учи
* void set\_program(const Program& program\_){program.changeProgram(program\_);} - извиква функция за променяне на специалността
* void set\_group(const size\_t& groupNum\_){groupNum = groupNum\_;} - сменя номера на групата
* void set\_year(const size\_t& year\_){year = year\_;} - прехвърля студента в следващ курс
* bool canChangeProgram(); - булева функция, показваща дали студентът покрива критериите за преместване в друга специалност
* bool canChangeYear(const size\_t&); - булева функция, показваща дали студентът може да премине в следващ курс
* void enroll(String, const size\_t&, const size\_t&, const Program&); - записване на нов студент
* void enrollIn(String, const double&); - добавя оценка на студента по предмет
* void advance(); - прехвърля студента в следващ курс
* void graduate(); - променя статуса на студента на „завършил“
* void interrupt(); - променя статуса на студента на „прекъснал“
* void resume(); - възстановява студентските права на студента
* void print(); - извежда справка за студента
* void report(); - извежда акадамеична справка за оценките на студента
* void addGrade(String, const double&); - добавя оценка по даден предмет
* void printForProtocol(String course); - принтира оценка на студент по даден предмет (използва се при протокол за предмети по специалности в клас Суси)
* bool is\_interrupted(); - проверка дали студентът не е прекъснал
* void saveSubjects(std::ofstream&); - записва предметите във файл
* void addSubj(String, const double&); - записване на нов предмет от студента
* bool hasSubject(String); - проверява дали студентът е записал даден предмет
* Клас Susi – класът реализира системата за обработка на данните за студентите, дисциплините и специалностите

**Private** член-данни:

* Student\*\* students; // указател към масив от указатели от студенти
* Subject\*\* subjects; // указател към масив от указатели от дисциплини
* Program\*\* programs; // указател към масив от указатели от специалности
* size\_t numPrograms; // брой специалности в системата
* size\_t numSubjects; // брой предмети в системата
* size\_t numStudents; // брой студенти в системата
* String filePathSubj; // път към отворения файл от предмети
* String filePathStud; // път към отворения файл от студенти

**Private** член-функции:

* void deleteStudent(); // помощна функция за триене на обект от тип Student
* void open(String); // отваряне на файл. Тук важно е да се отбележи, че системата работи с два типа файлове - за студенти и за предмети, като трябва да се отвори първо файл с предмети преди такъв със студенти.
* void save(); // записване на двата отворени файла
* void help(); // извежда всички възможни команди, техния синтаксис и предназначение
* void saveAs(String); // записва файловете на определено от потребителя място, като е важно да се отбележи, че тъй като се отварят два файла едновременно, то не се задава име, а само директория
* void deleteSusi(); // помощна функция за триене на обект от системата Суси
* size\_t countSpaces(String); // помощна функция за преброяване на интервалите при въвеждане с цел разграничаване на командите
* void protocol(String, const Program&); // извежда протокол за предмет по специалности
* void sortForProtocol(); // помощна функция за сортиране на данните за горепосочения протокол
* (public) Член-функции – тук е реализирана единствено голямата четворка(конструктор, копиращ конструктор, оператор= и деструктор)

1. **Реализация и уточнения**

Клас **Subject**:

* Два предмета са еднакви, ако имат еднакви имена.
* Ако type е със стойност 1, то предметът е задължително избираем, а иначе е избираем.

Клас **Program**:

* Прави се проверка дали не се трие нулева памет при триене на обект.
* При смяна на специалността се добавят предметите, които са били изучавани в предната специалност в динамична масив от индекси, който показва кои предмети са от специалността. Това се прави с цел улеснение на работата при клас Student, за да може да се обработват там и тези предмети, а не само новите (при академичната справка).
* Subject\* get\_subject(const size\_t& i)const{return subjects[i];} не е констатен, защото тази член-функция се изпозлва за вземане на предмет, върху който ще се извършват промени.

Клас **Student**:

* Тук не изпозлвам динамична памет пряко и затова нямам голяма четворка.
* Има методи, които помагат за проверка дали студент може да се мести в друга специалност или курс в зависимост от статуса си и взетите изпити.

Клас **Susi**:

* Направени са проверки дали файлът е отворен.
* Работи се с два типа файла – за студенти и за предмети. Първо се отваря такъв за предмети, а после – за студенти, защото така е по-удобно, тъй като в клас Student има поле Subject, а не обратното.
* Направена е проверка за това дали студентът принадлежи към специалност.
* Направена е проверка дали същестува специалността, в която студентът иска да се премести.
* Направена е и проверка за валидност на опцията за промяна( на курс, специалност и група)

**Още уточнения:**

* Класът String въвежда данните символ по символ
* Не съм използвал допълнителни източници
* using namespace std се изпозлва само в main.cpp с цел спазване на добрите практики.
* Допълнителните методи за копиране и триене са също с цел спазване на добрите практики.
* Освен това на много места параметрите на функциите са константни и по референция отново с цел спазване на добрите практики.
* Командата close() затваря и двата отворени файла(и за предмети, и за студенти)