**Документация на Наталия Николова за 5 тема Личен календар**

**4 група, 0MI0800100**

**Линк към Github:** <https://github.com/nataliaNikolova13/project>

**Глава 1. Увод**

**1.1. Описание и идея на проекта**

Проектът представлява календар с планирани срещи. С предварително зададени данни се записва час, като на една дата и час може да съществува единствена среща.

**1.2. Цел и задачи на разработката**

Целта на проекта е да се реализира един напълно работещ календар, в който могат да се записват, отменят и променят срещи, както и да се извеждат справки за натовареност, дневен график и свободни часове.

**Глава 2. Преглед на предметната област**

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

Един от основните външни алгоритми, които използвам, е сортирането чрез пряка селекция.

Приложила съм основните принципи на ООП.

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

Най-сериозният проблем и трудност на задачата е да се валидира единствеността на срещите и да няма записани дублиращи се срещи.

Най-сложният метод, които трябва да се реализира, е намирането на свободни часове.

**2.3. Подходи, методи (евентуално модели и стандарти) за решаване на поставените проблемите**

Поставените задачи са класовете и методите, които трябва да се реализират.

Първият клас Appointment създава обектите на срещите, а класа Calendar създава масив от срещите.

Записването на среща става като се създаде обект в Appointment класа и като се добави с масива от класа Calendar чрез оператора +=. При реализирането на оператора се прави проверка дали вече не съществува среща в същия диапазон. Ако съществува, срещата не се записва в календара. Подаването на информацията може да става както  директно от потребителя с оператор за вход, така и зададено ръчно.

Отмяната на среща става чрез оператора -= по подаване на име и се изважда от календара.

Извеждането на дневната програма става като първо всички обекти в масива се сортират по начален час в низходящ ред (за това сортиране се взема предвид само и единствено началния час и се използва сортиране чрез пряка селекция). После масива се филтрира по зададена дата и се извеждат всички записани часове във файла "appointmentsByDay.txt". Те са подредени по час, заради горното сортиране.

Промяната на срещата става в класа Appointment. По подадени данни може да се промени всичко в срещата. За да се осигури, че часа на срещата не се дублира с друга вече съществуваща, срещата се изважда от календара, прави се промяната на часа и после пак се добавя в календара. От проверките за записване на час се осигурява единственост на срещата.

Търсенето на среща се реализира по въведена ключова дума, която се търси в името или съобщението. Всяка среща, в която се съдържа дадената ключова дума се извежда във файла "appointmentsSearchResult.txt". Ключовата дума може да се зададе от потребителя, както и в самата програма.

Извеждането на натовареността се реализира по зададен интервал от време. Програмата минава ден по ден и намира броя на срещите във всеки ден. Създава се двумерен масив, в които на нулева позиция се записва деня, а на първа броя срещи. Масивът се сортира по броя срещи със сортиране чрез пряка селекция във възходящ ред. Статистиките за дадения период се извеждат във файл.

Намирането на свободен час се реализиран по зададен интервал. Първо се проверява за свободни дни, ако съществува такъв, срещата се записва на първия възможен празен ден с най-ранния час спрямо зададения интервал. Ако няма свободен ден почват да се проверяват ден по ден, като преди това масива се сортира по начален час. Минавам отново през масива и ако в дадения ден има дупка с необходимата дължина се извежда съобщение.

**2.4 Потребителски (функционални) изисквания (права, роли, статуси, диаграми, ...) и качествени (нефункционални) изисквания (скалируемост, поддръжка, ...)**

Private са всички декларации на член данни и в двата класа и функциите свързани със заделянето на динамичната памет в класа с Календара. (Resize, reallocate, copy)

Public са всички останали функции и оператори.

Потребителят има достъп до всичко, което не е private. Срещите могат да бъдат запазени и чрез подаване на информация от потребителя от терминала.

**Глава 3. Проектиране**

**5.1. Обща архитектура – ООП дизайн**

Проектът се реализира чрез 2 класа, които са в два .cpp файла, където реализирам функциите, методите и операторите, 2 заглавни .hpp файла, където декларирам функциите, методите и операторите, и main.cpp файл, където тествам програмата си.

**Глава 4. Реализация, тестване**

**4.1. Реализация на класове**

Програмата се реализира чрез два класа. Първият клас Appointment създава обекти, които представляват записан час, с всичките необходими характеристики - име, съобщение, дата, месец, година, начално и крайно време. Втория клас Calendar прави масив от обекти(записаните часове). За да се запише час в този масив, времето, когато се провежда трябва да е единствено, ако се припокрива с друг час се взема предвид първия записан и другия не се зачита, за да няма дублиране.

**4.2. Управление на паметта и алгоритми. Оптимизации.**

Имената на часовете и съдържанието на съобщенията се заделят динамично.

Освен това масива с обектите също се заделя динамично. Реализирам голяма четворка, която се грижи за правилното заделяне на паметта. Конструкторът заделя масив с малка начална дължина (в случая 2). Когато се достигне максималната дължина на масива, Resize функцията увеличава размера на масива по 2. Reallocate изтрива заделената памет.

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

В функцията testProgramm се тества проекта. Създават се няколко срещи, които ако отговарят на условието за единственост се добавят в календара. Отдолу се тестват и всички останали методи.

**Глава 5. Заключение**

**5.1. Обобщение на изпълнението на началните цели**

Реализирала съм календар със срещи, където могат да се записват, отписват и променят срещи. Както и да се извеждат справки за натовареност за период, дневен календар и намиране на среща по подадена ключова дума. Извежда се и съобщение за свободен час в даден период.

**5.2. Насоки за бъдещо развитие и усъвършенстване**

Друго бъдещо подобрение би могло да бъде по-динамичното подаване на информация от страна на потребителя. И в двата класа информацията може да се подава от конзолата, но много от данните, които се използват за методите, са зададени ръчно. Може да се създаде нещо тип меню, където потребителя може да избира от всичките налични методи и интерактивно да избира и въвежда всичко.

Друго подобрение би било, когато един час се повтаря, да се дава на потребителя възможност да промени желания от него час или дата. Както е направено в момента ако съществува записан час по това време, новия час директно просто не се добавя в календара.