Проект по Обектно Ориентирано Програмиране

Персонален календар

Изготвил: Благовест, група 8, 7MI0800003

1. **Увод**
   1. **Описание и идея на проекта**

Проектът представлява конзолна програма, която чрез команди от потребителския вход създава и управлява календари, запазвани във файлове. Идеята е тези календари да служат като място за записване на информация за ангажименти, която може да бъде извличана и обработвана по различни начини.

* 1. **Цел и задачи на разработката**

Целта на проекта е чрез ООП логика да бъде изградено конзолно приложение, което поддържа въвеждане на потребителски команди в диалогов режим.

Поддържаните операции са следните:

* Запазва час за среща
* Отменяне на час за среща
* Извеждане на хронологичен списък с всички срещи за даден ден
* Промяна на среща
* Търсене на среща по низ, съдържащ се в името или коментара на срещата
* Извеждане на статистика за натовареност: по дадени начална и крайна дата се извежда списък с дните от седмицата, подредени по критерия “брой заети часове”
* Намиране на свободно място за среща: по дадена дата и желана продължителност на срещата търси дата, на която е възможно да се запази такава среща, като търси само в предварително зададен интервал от часове

За да бъдат разгледани всички потенциални случаи, при всяка операция се валидират въведените потребителски данни. Ако има проблем операцията се прекъсва и потребителя бива известен.

1. **Проектиране**
   1. **Обща архитектура – ООП дизайн**

Реализацията на проекта е постигната чрез 2 класа – *Meeting* и *Calendar*.

*Meeting* е базовият клас, описващ среща със следните атрибути:

    char \*name; - име на срещата

    char \*comment; - коментар към срещата

    char date[11]; - дата на срещата във формата *dd-mm-yyyy*

    char startTime[6]; - начало на срещата във формата *hh-mm*

    char endTime[6];   - край на срещата във формата *hh-mm*

*Calendar* е клас, използващ обекти от тип *Meeting*, за да пресъздаде обект, съдържащ всички срещи на едно място. Неговите атрибути са следните:

char \*filename; - име на файла, в който се съхранява календара

    Meeting \*\*meetings; - масив от срещи (избягваме дефолтен констр.)

    size\_t size; - брой на записаните срещи

    size\_t capacity; - капацитет на календара (за брой срещи)

И за двата класа е реализирана голяма четворка, но дефолтният конструктор е заместен от такива с параметри:

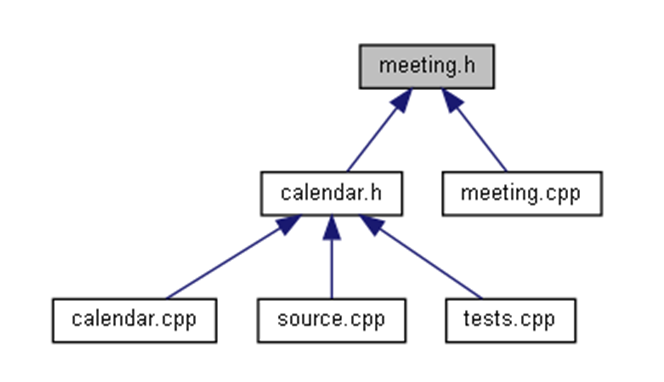
Обекти от тип *Meeting* могат да се създават само с конструктор с всички параметри:

Meeting(const char\*, const char\*, const char\*, const char\*, const char\*);

Обекти от тип *Calendar* могат както с всички параметри, така и само с името на файла, в който ще се съхранява информацията:

Calendar(const char\*);

Calendar(const char\*, Meeting\*\*, size\_t, size\_t);



**Фиг. 1** Обща структура на файловите връзки

Фиг. 1 описва връзката между файловете с декларации, имплеметационните файлове, main файла и файлът със тестове, които използват разработеният продукт за тестване *doctest.h*.

1. **Реализация, тестване**
   1. **Реализация на класове**

Клас *Meeting* съдържа в себе си следните функции:

* Голяма четворка (с конструктор с параметри)
* Мутатори (Гетъри и Сетъри)
* Функции за управление на паметта
* Функции за валидация на входните данни (статични, за да бъдат използвани самостоятелно)
* Предефинирани оператори (външни функции и една приятелска)

Интересните от тези функции са най-вече операторите:

friend istream& operator>>(istream&, Meeting\*&); - извлича данни от потока

ostream& operator<<(ostream&, const Meeting&); - вмъква данни в потока

bool operator==(const Meeting&, const Meeting&); - сравнява две срещи

bool operator!=(const Meeting&, const Meeting&); - използва горната функция

bool operator<(const Meeting&, const Meeting&); - проверява коя среща е по-напред във времето

Клас *Calendar* съдържа в себе си следните функции:

* Голяма четворка (с конструктори с параметри)
* Мутатори (Гетъри)
* Функции за управление на паметта
* Функции за добавяне и махане на срещи
* Функции за извеждане на данни на потребителския изход (и във файл)
* Функции за промяна на някоя от срещите по определен критерий

По – интересни функции:

void resize(size\_t); - променя капацитета на календара в зависимост от процентната заетост

void sort(); - сортира (намества) последната добавена среща

bool save(); - запазва текущия вид на календара (сортиран) във файл

void saveByWorkHours(const char\*, const char\*); - спрямо начална и крайна дата извежда всички срещи в периода, сортирани по заетост

bool isThereTime(const char\*, const char\*, size\_t, const char\*, const char\*);

- намира точно кога в даден период може да се намери време за среща с определени критерии

Метода *isThereTime(…)* беше реализиран за конкретен ден, както беше зададено първоначално по условие. След промяната на условието, по подадени две дати, определящи седмица (предполагаемо 7 дена една от друга), срещите през този период се комбинират по дни и после се извиква предишната функционалност за съответния ден. Предишната версия е налична в *Github*, предпоследния *commit*.

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

Управлението на паметта на базово ниво и в двата класа се изпълнява от методите *copyFrom(const \_\_&)*  и *free()*.

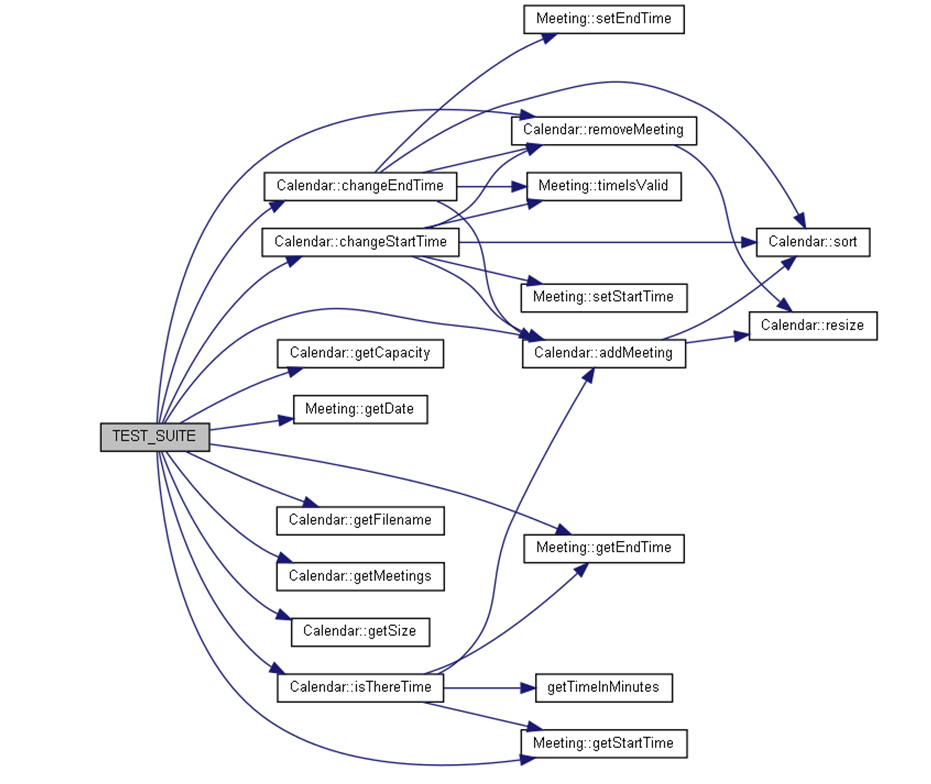
В класа *Calendar*, за да се избегне дефолтен конструктор на *Meeting* и също така да се спести честото копиране на стрингове, което е бавна операция, се използва динамичен масив от указатели към срещи, за да може чрез присвояване на указатели да се постигне по-голяма ефективност.

За изчистване на диалоговия прозорец се използва функцията *system(“clear”)*, взета от *<stdlib.h>*.

* 1. **Тестване**

Чрез използване на *doctest.h* са имплементирани тестове на всички функционалности на двата класа, които нямат поведение, дефинирано единствено във потребителския изход.

Чрез следващата Фиг. 2 ще се покажат функциите, тествани от *tests.cpp*:



**Фиг. 2** Връзки между тестовия файл и функциите, които биват извиквани

1. **Заключение и бъдещи подобрения**

Успешно бе реализирана програма, работеща със календари и файлове. Функционалностите, макар и чрез сравнително семпъл интерфейс, дават една добра база за един потенциално по-пълен и изчистен проект.

Неща, които биха могли да се подобрят:

* Да се използват готови класове за член данните от тип *Time* и *Date* (дава по-голяма ефективност, прегледност на кода и по-смислена логика)

Нeщо, което би могло да се добави:

* Поддържане на дни от седмицата, както и функция за това кой ден от седмицата е подадената като аргумент дата

**Използвана литература:**

* *doctest.h* – C++ testing framework, documentation link here: [Link](https://github.com/doctest/doctest)

**Връзка към хранилище, съдържащо реализацията на проекта:** [Github](https://github.com/just8do8it/Computer-Science-at-FMI/tree/main/Object%20Oriented%20Programming/Personal%20calendar)

--------------------------------------------------------------------------------------------------------------------

**Съдържание**

Увод………………………………………...стр. 1

Проектиране………………………………..стр. 2

Реализация, тестване………………………стр.3

Заключение и бъдещи подобрения……….стр.6

Използвана литература……………………стр. 6