Увод

1. Документацията, описана в този файл, се отнася за създаване на програма, която реализира система за музикална библиотека. В нея се съдържат данните на потребители и песни, добавени от тях, както и реализацията на интерфейс, чрез който се изпълняват редица операции, избрани чрез команди от съответния потребител.
2. Идеята на програмата е при стартиране всеки да може да създаде свой профил, да влезе в системата чрез създадения от него профил и да има достъп до множество от функции при коректен вход на данните. Тези функции са свързани с промяна на данни на профила, добавяне на песни, създаване на плейлист под дадено име, зареждане и показване на песните от избран от потребителя плейлист.
3. Съдържание на документацията
4. Реализация на програмата
5. Интерфейс на програмата
6. Заключение

Линк към Github: https://github.com/jggrigorov/Musical\_collection\_project

1. Реализация на програмата

За създаването на песен, профил и плейлист използваме класовете съответно Song, Profile и Playlist. Чрез тях създаваме обектите, върху които се извършват операциите за създаване на песен, профил и плейлист, както и промяна в данните на обектите. Реализацията на класовете се извършва в отделни заглавни файлове, съответно Song.h, Profile.h и Playlist.h. Функциите, които формират функционалността и интерфейса на програмата, са изнесени в отделен заглавен файл, наречен Functions.h. Тъй като тези функции не са член-функции на нито един от посочените по-горе класове, използваме функции за достъп и мутатори на съответните класове, за да можем да реализираме функциите, формиращи интерфейса. За член-данни на класовете за използвани структурите от данни *std::vector* и *std::string*. С тях се описват имена на песни, имена на потребители и техните пароли, множество от жанрове, множество от плейлисти на даден потребител и други член-данни. Също така тези структури са използвани при изпълнение на функциите, които ще бъдат описани по-долу в документацията на проекта.

Реализация на класовете, използвани в програмата:

class Song {

private:

std::string name;

std::string author;

std::string genre;

std::string album;

int release\_year;

double rating;

void copy\_song(const Song&);

void delete\_song();

public:

Song();

Song(const Song&);

Song& operator=(const Song&);

~Song();

void set\_name(std::string);

void set\_author(std::string);

void set\_genre(std::string);

void set\_album(std::string);

void set\_release\_year(int);

void set\_rating(double);

std::string get\_name()const;

std::string get\_author()const;

std::string get\_genre()const;

std::string get\_album()const;

int get\_release\_year()const;

double get\_rating()const;

void print()const;

};

class Playlist {

private:

std::string name\_of\_playlist;

unsigned int maximum\_size\_of\_playlist;

std::vector<Song> songs\_in\_playlist;

void copy\_playlist(const Playlist&);

void delete\_playlist();

public:

Playlist();

Playlist(const Playlist&);

Playlist& operator=(const Playlist&);

~Playlist();

void set\_name\_of\_playlist(std::string);

void set\_maximum\_size\_of\_playlist(int);

void add\_song(Song);

std::string get\_name\_of\_playlist()const;

unsigned int get\_maximum\_size\_of\_playlist()const;

std::vector<Song> get\_songs\_in\_playlist()const;

void create\_playlist(std::vector<Song>&,std::vector<std::string>);

void load\_playlist(const std::string&);

void print()const;

};

class Profile {

private:

std::string profile\_name;

std::string password;

std::string full\_name;

std::string date\_of\_birth;

std::vector<std::string> genres;

std::vector<Playlist> playlists;

void copy\_profile(const Profile&);

void delete\_profile();

public:

Profile();

Profile(const Profile&);

Profile& operator=(const Profile&);

~Profile();

void set\_profile\_name(std::string);

void set\_password(std::string);

void set\_full\_name(std::string);

void set\_date\_of\_birth(std::string);

std::string get\_profile\_name()const;

std::string get\_password()const;

std::vector<std::string> get\_genres()const;

std::vector<Playlist> get\_playlist()const;

void add\_genre(std::string);

void remove\_genre(std::string);

void add\_playlist(Playlist);

void print()const;

};

1. Интерфейс на програмата

Както споменах по-рано в документа, функционалността и интефейса на програмата са изнесени в заглавния файл Functions.h. За да се осъществят някои функции, като например зареждане плейлист под дадено име и подреждане на песните на плейлист в азбучен ред, са използвани алгоритми за търсене и сортиране (съответно linear search и bubble sort). Командите в конзолния интерфейс се осъществяват чрез въвеждане на число. Тези команди са описани на конзолата и потребителят има избор коя функция да се изпълни.

1. Заключение

Разработката на програмата позволява оптимизации на някои алгоритми, както и редактиране на някои фрагменти от нея.