**Курсовая работа по дисциплине**

**«Программирование приложений с использованием библиотек»**

2018 г

# ЗАДАНИЕ

(ВСТАВИТЬ БЛАНК ЗАДАНИЯ)

Оглавление

[ЗАДАНИЕ 2](#_Toc518060049)

[ВВЕДЕНИЕ 4](#_Toc518060050)

[1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 6](#_Toc518060051)

[1.1 Язык прогаммирования C++ 6](#_Toc518060052)

[1.2 Стандартная библиотека шаблонов языка C++ 6](#_Toc518060053)

[1.2.1 Общие сведения 6](#_Toc518060054)

[1.2.2 Описание используемых в курсовой контейнеров STL 7](#_Toc518060055)

[1.2.3 Описание используемых в курсовой итераторов STL 7](#_Toc518060056)

[1.2.4 Описание используемых в курсовой алгоритмов STL 8](#_Toc518060057)

[1.3 Краткое описание используемой среды разработки 8](#_Toc518060058)

[2 ПРАКТИЧЕСКАЯ ЧАСТЬ 9](#_Toc518060059)

[2.1 Описание реализованных классов 9](#_Toc518060060)

[2.2 Краткое руководство пользователя 9](#_Toc518060061)

[ЗАКЛЮЧЕНИЕ 16](#_Toc518060062)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 17](#_Toc518060063)

[ПРИЛОЖЕНИЕ А. ЛИСТИНГ ПРОГРАММЫ 18](#_Toc518060064)

# ВВЕДЕНИЕ

В настоящее время, с развитием научно-технического прогресса и информационных технологий, сложность автоматизируемых предметных областей постоянно и неуклонно возрастает. Программист проводит всё своё рабочее время, по сути, в борьбе со сложностью программ, отлаживая свой код в поисках допущенных ошибок. Ради борьбы со сложностью эволюционируют и языки программирования.

После машинных кодов и перфокарт появились текстовые редакторы и язык ассемблера, что позволило писать чуть более сложные и объёмные программы при меньших трудозатратах. Далее появились процедурные языки высокого уровня - C, Паскаль и многие другие, что перевело технологии программирования на новый уровень.

Следующим крупным скачком на пути борьбы со сложностью стало появление объектно-ориентированного подхода и разработка соответствующих языков и компиляторов. Так C эволюционировал в C++.

Язык C включает достаточно аскетичную стандартную библиотеку, упрощающую работу с памятью, строками, файлами и т.д. Однако стандартная библиотека шаблонов STL (англ. Standard Templates Library), включенная в стандарт C++ и доступная в каждом компиляторе, не идёт ни в какое сравнение со стандартными инструментами, предоставляемыми предшественником.

При разработке на языке C программисты вынуждены постоянно писать код для выделения и очистки памяти под массивы, списки и другие контейнеры. Для каждого контейнера отдельно приходится хранить количество элементов. Для обработки элементов пишутся бесчисленные циклы for и while. Для написания универсальных процедур приходится прибегать к указателям типа void\*, что требует особой внимательности при преобразовании типов. Более того, в реальном коде неизбежно встречаются многократные повторения кусков кода, выполняющих сортировку, поиск и многие другие типовые операции обработки последовательностей.

К счастью для большинства программистов, разработчики стандарта языка C++ – в частности, его главный идеолог и основатель Бьёрн Страуструп – собрали вокруг себя команду профессионалов, которые определили основной круг типовых задач, решаемых в повседневном программировании, а также разработали начальную реализацию стандартной библиотеки.

В итоге, STL содержит средства для управления контейнерами – массивы, списки, стеки, очереди и т.д. – и значительное число типовых алгоритмов для работы с ними.

Данная работа посвящена изучению современных подходов к разработке программных систем на языке высокого уровня С++ с использованием стандартной библиотеки шаблонов. В качестве среды разработки используется бесплатно распространяемая система Dev С++.

Целью работы является закрепление и углубление знаний, полученных при изучении языка C++ и его стандартной библиотеки, а также развитие практических навыков при выборе представления исходных данных, использовании стандартной библиотеки при написании программ, тестировании, отладки программы и оформлении сопроводительной документации.

В качестве предметной области для демонстрации возможностей C++ и STL выбрана система формирования концертной программы по заявкам, в которой пользователи смогут просматривать, добавлять, редактировать и удалять песни, а также осуществлять поиск и голосовать за любимые композиции.

# ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

## Язык программирования C++

Предшественником языка C++ является императивный (процедурный) язык программирования C. Язык C++ значительно расширяет возможности своего предшественника и включает в себя сразу три парадигмы – процедурное, объектно-ориентированное и обобщенное программирование. При этом соблюдается обратная совместимость с предшественником. По сути, современный язык C++ – это три диалекта, весьма существенно отличающиеся друг от друга.

В последние годы, с внедрением стандартов 2011, 2014 и 2017 годов, наметилась тенденция к появлению четвёртого диалекта. Речь идёт о так называемом функциональном программировании, в котором, в противовес императивной парадигме, процесс вычисления трактуется как последовательное вычисление значений функций в математическом понимании, а не смена состояний в результате череды процедурных вызовов.

## Стандартная библиотека шаблонов языка C++

### Общие сведения

Стандартная библиотека, поставляемая в комплекте с компилятором, включает наиболее распространённые контейнеры и алгоритмы.

Механизм шаблонов встроен в компилятор языка C++, чтобы дать возможность программистам делать свой код короче за счет обобщенного программирования – речь идёт об упоминавшейся ранее парадигме обощённого программирования. Именно на этой парадигме и строится стандартная библиотека шаблонов STL.

Самые популярные коллекции из библиотеки это string, vector, list, queue, stack, map (multimap) и set (multiset).

### Описание используемых в курсовой контейнеров STL

В работе использованы такие контейнеры как string, vector и map. Контейнер string является обобщением только для строковых последовательностей – как правило, это строки из символов типа char и wchar\_t («Wide Char»). Он инкапсулирует в себе все рутинные операции по управлению памятью и является крайне удобной заменой использовавшимся в язык C символьным массивам.

Как и многие другие контейнеры, string доступен для обработки рядом обобщенных алгоритмов из библиотеки, а также имеет свои собственные методы, оптимизированные для работы именно со строками – например, поиск подстроки.

Контейнер vector представляет собой обобщение для массивов элементов произвольного типа. Его особенность в том, что он гарантирует последовательное расположение своих элементов в памяти и разрешает, таким образом, произвольный доступ к любому хранимому элементу, без перебора остальных.

Контейнер map является удобным обобщением для ассоциативного массива, или словаря. Он позволяет индексировать элементы хранимой последовательности любым пользовательским типом, поддерживающим операцию сравнения «<» («меньше») – в отличие от вектора, который индексирует элементы только целочисленным беззнаковым типом.

### Описание используемых в курсовой итераторов STL

В данной работе используется итератор доступа к ассоциативному массиву, в котором хранятся зарегистрированные пользователи системы. Глобальный итератор указывает на вошедшего в систему пользователя и олицетворяет собой текущую сессию.

Аналогичные итераторы используются в массиве userSong, хранящем связи между песнями и проголосовавшими за них пользователями, а также для поиска в словаре пользователей (пользователь представляет собой ассоциативную пару имя – пароль).

Временные итераторы произвольного доступа используются при поиске в массиве (векторе) песен.

### Описание используемых в курсовой алгоритмов STL

В данной работе используются следующие обобщенные алгоритмы:

* find() – поиск элемента;
* find\_if() – параметризированный поиск элемента;
* for\_each() – перебор элементов;
* sort() – сортировка последовательности;
* copy() – копирование элементов одной последовательности в другую.

## Краткое описание используемой среды разработки

Dev-C++ — [свободно](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) распространяемая [интегрированная среда разработки (англ. IDE, Integrated Development Environment) приложений](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8_%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9) для языков программирования [C](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F))/[C++](https://ru.wikipedia.org/wiki/C%2B%2B). В дистрибутив входит компилятор [MinGW](https://ru.wikipedia.org/wiki/MinGW). Сам Dev-C++ написан на [Delphi](https://ru.wikipedia.org/wiki/Delphi_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)).

Проект поддерживается [SourceForge](https://ru.wikipedia.org/wiki/SourceForge). Основатель проекта Колин Лаплас, компания Bloodshed Software. Некоторое время была доступна версия для Linux, но на текущий момент актуализирована только версия под [Windows](https://ru.wikipedia.org/wiki/Microsoft_Windows). В настоящее время оригинальный Dev-C++ не разрабатывается.

Продолжением разработки сейчас занимается компания Orwell.

# ПРАКТИЧЕСКАЯ ЧАСТЬ

## Описание реализованных классов

В ходе декомпозиции задачи вывялен ряд обособленных сущностей, реализованных в виде пользовательских классов:

* Song –полная информацию о песне;
* Concert –система приёма концертных заявок;
* FindSongByName – функциональный тип для фильтра песен по названию;
* FindSongBySinger – функциональный тип для фильтра песен по исполнителю;
* InvalidOrdinalError –исключение для обнаружения ситуации с вводом недопустимого значения порядкового номера;
* PrintSong – функциональный тип для безусловной распечатки песни, используется для вывода на печать всех доступных в системе песен;
* PrintAvailableSong – функциональный тип для условной распечатки песни, используется для вывода песен, за которые пользователь ещё не голосовал.

## Краткое руководство пользователя

Система формирования программы концерта по заявкам реализована в виде консольной программы Windows, то есть имеет интерфейс командной строки. Для запуска программы необходимо открыть терминал (cmd.exe) и выполнить исполняемый файл Concert.exe. Необходимо либо указать полный путь к файлу, либо предварительно перейти в каталог, где он хранится, с помощью команды cd.

Кроме того, программу можно запустить из Проводника Windows, дважды щёлкнув по пиктограмме исполняемого файла.

Для взаимодействия с пользователем используется текстовый интерфейс – на каждом этапе взаимодействия программа предлагает список из доступных действий и, в зависимости от текущего состояния, предлагает ввести либо номер пункта, либо запрашиваемое значение параметра.

После запуска программы откроется окно с общим меню, изображённое на рисунке 1.

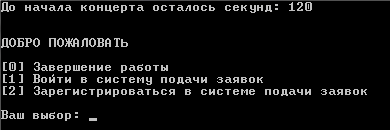


Рисунок 1 – Общее меню программы

Это меню доступно всем пользователям системы, в том числе не авторизованным, и содержит всего три пункта – войти в систему, зарегистрироваться и завершить работу.

В начале работы в системе не зарегистрировано ни одного пользователя, поэтому сперва необходимо зарегистрироваться, введя имя пользователя и пароль, как показано на рисунке 2.

sign-up.png

Рисунок 2 – Регистрация пользователя

В случае если такого пользователя нет в системе, регистрация пройдёт успешно и будет автоматически выполнен вход в систему с новыми учётными данными. В противном случае программа напишет сообщение об ошибке и предоставит возможность начать сначала.

До тех ор, пока программа выполняется, можно регистрировать нескольких пользователей, а также входить в систему и выходить из неё.

В случае если в системе уже зарегистрирован хотя бы один пользователь, можно войти в систему, выбрав пункт меню с номером [1], как показано на рисунке 3.

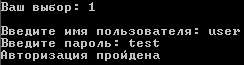


Рисунок 3 – Авторизация пользователя

После успешного завершения регистрации или авторизации, пользователю становится доступно меню формирования программы концерта, показанное на рисунке 4.

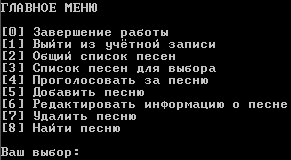


Рисунок 4 – Меню формирования программы концерта

Авторизованным пользователям доступны следующие пункты:

* Завершение работы приложения.
* Выход из текущей учётной записи.
* Печать на экране полного списка песен.
* Печать на экране песен, ещё не включённых в программу концерта.
* Выбор песни – голосование за включение её в программу.
* Добавление новой песни.
* Редактирование существующей песни.
* Удаление существующей песни.
* Поиск песни по названию или исполнителю.

Первый пункт завершает работу всего приложения так же, как и аналогичный пункт общего меню.

Следующий пункт позволяет, не завершая работы, выйти из текущей учётной записи. Впоследствии можно будет войти под этим же именем, либо под другим.

Пункт под номером [2] позволяет вывести общий список песен, включая те, за которые текущий пользователь уже голосовал. Примерный вид окна приложения с общим списком песен приведён на рисунке 5.

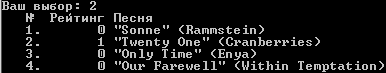


Рисунок 5 – Общий список песен

Пункт под номером [3] позволяет распечатать ещё не выбранные пользователем песни, за которые он может проголосовать. Песни, за которые он уже проголосовал (возможно, в предыдущие сеансы авторизации), в этом списке не отображаются.

Пункт 5 позволяет проголосовать за понравившуюся песню. Как показано на рисунке 6, при выборе пользователем этого пункта, программа выводит список доступных песен и предлагает выбрать номер интересующей песни.

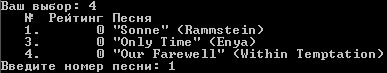


Рисунок 6 – Выбор интересующей песни

Как видно из рисунков 5 и 6, текущий пользователь уже проголосовал за песню «Twenty One» и теперь может выбрать любую из оставшихся трёх, но не ту же самую.

Далее следуют три служебные операции по формированию списка песен, доступных для включения в программу – добавление, редактирование и удаление песен из системы.

Для добавления песни необходимо выбрать пункт 5 пользовательского меню. Пример добавления новой песни показан на рисунке 7.

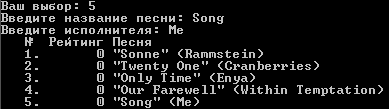


Рисунок 7 – Добавление новой песни

Редактирование информации о песне позволяет частично или полностью заменить данную позицию, введя новое название и исполнителя, как показано на рисунке 8.

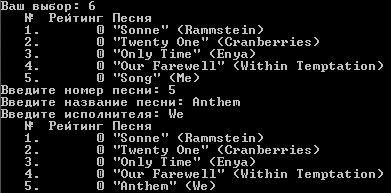


Рисунок 8 – Редактирование существующей песни

Приложение, при выборе пользователем пункта 6, выводит полный список песен и запрашивает сперва номер песни для редактирования, а затем новые значения названия и исполнителя выбранной песни. После ввода данных выводится обновлённый список песен.

При выборе пункта 7 аналогично предлагается указать номер песни для удаления, как показано на рисунке 9.

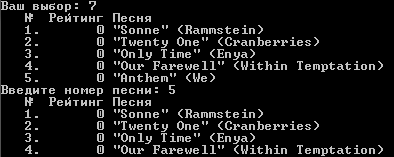


Рисунок 10 – Удаление песни из списка

После удаления также выводится обновлённый список песен.

Последний пункт меню актуален при большом количестве доступных песен и позволяет найти песню по названию или исполнителю. На рисунке 11 показано вспомогательное меню, где приложение предлагает пользователю выбрать интересующее поле для поиска.

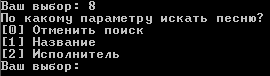


Рисунок 11 – Подменю «Поиск песни»

Для поиска по названию, как показано, на рисунке 12, следует выбрать пункт 1, а для поиска по исполнителю – пункт 2 (рисунок 13).

search-1.png

Рисунок 12 – Поиск песни по названию

search-2.png

Рисунок 13 – Поиск песни по исполнителю

После каждого действия пользователя, приложение проверяет оставшееся время для концерта, выводит его на экран и, по истечении отведённого времени, формирует концертную программу по результатам голосования. Песни, набравшие наибольшее число голосов, идут первыми. Количество песен для исполнения ограничено в демонстрационной программе двумя позициями, как показано на рисунке 14.

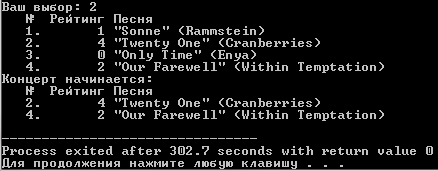


Рисунок 14 – Поиск песни по исполнителю

После завершения концерта завершается и сама программа.

По умолчанию концерт начинается через 5 минут после запуска программы. Если этого времени недостаточно для регистрации пользователей и формирования концертной программы, приложение можно запускать из командной строки, передавая ему в качестве параметра желаемое количество секунд до начала концерта.

# ЗАКЛЮЧЕНИЕ

В результате работы создана программа, демонстрирующая применение в повседневной разработке контейнеров и алгоритмов стандартной библиотеки шаблонов языка C++.

Приобретены практические навыки разработки приложений, использования языковых конструкций С++ и инструментов STL, отладки программ и составления сопроводительной документации. На базе практического применения, закреплены базовые знания, полученные ранее, при изучении теории.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Герберт Шилдт Искусство программирования на C++ / Герберт Шилдт. - М.: БХВ-Петербург, 2005. - 496 c.

2. Герб Саттер Решение сложных задач на С++ / Герб Саттер. - М.: Вильямс, 2015. - 400 c.

3. Александр Чиртик Программирование на C++. Трюки и эффекты / Александр Чиртик. - М.: "Издательство "Питер", 2010. - 352 c.

4. Роберт Лафоре Объектно-ориентированное программирование в С++ / Роберт Лафоре. - М.: Питер, 2015. - 928 c.

5. Айра Пол Объектно-ориентированное программирование на С++ / Айра Пол. - М.: Не указано, Бином, Невский Диалект, 2001. - 464 c.

6. Николаи М. Джосаттис Стандартная библиотека C++. Справочное руководство / Николаи М. Джосаттис. - М.: Вильямс, 2014. – 123 c.

7. ГОСТ 2.105-95 Общие требования к текстовым документам.

# ПРИЛОЖЕНИЕ А. ЛИСТИНГ ПРОГРАММЫ

**main.cpp**

#include <clocale>

#include <sstream>

#include "Concert.h"

// Точка входа приложения

int main(int argc, char\*\* argv) {

// Установка кириллицы в консоли

setlocale(LC\_CTYPE, "Russian");

// Получение таймаута из параметра командной строки

int timeout = 300; // по умолчанию 300 сек (5 мин)

if(argc > 1) {

std::istringstream iss;

iss.str(argv[1]);

int value = 0;

if(iss >> value) {

timeout = value;

}

}

// Основной объект приложения

Concert concert(timeout);

return concert.run();

}

**Song.h**

#ifndef SONG\_H

#define SONG\_H

#include <string>

// Песня

class Song

{

public:

// Конструктор

Song(const std::string& name = "",

const std::string& singer = "", size\_t id = 0);

// Методы доступа к полям класса

inline void setName(const std::string& value) {

name = value;

}

inline std::string getName() const {

return name;

}

inline void setSinger(const std::string& value) {

singer = value;

}

inline std::string getSinger() const {

return singer;

}

inline void setId(size\_t value) {

id = value;

}

inline size\_t getId() const {

return id;

}

inline size\_t getVotes() const {

return votes;

}

// Выбор песни (голосование)

inline size\_t vote() {

return ++votes;

}

// Обнулить результаты голосования

inline void resetVotes() {

votes = 0;

}

private:

// Название

std::string name;

// Исполнитель

std::string singer;

// Идентификатор

size\_t id;

// Количество голосов

size\_t votes;

};

#endif

**Song.cpp**

#include "Song.h"

Song::Song(const std::string& name, const std::string& singer, size\_t id)

: name(name), singer(singer), id(id), votes(0)

{

}

**Concert.h**

#ifndef CONCERT\_H

#define CONCERT\_H

#include <string>

#include <map>

#include <vector>

#include <list>

#include <iostream>

#include <iomanip>

#include <algorithm>

#include <chrono>

#include "Song.h"

// Функциональный тип для фильтра песен по названию

class FindSongByName {

public:

FindSongByName(const std::string& field) : field(field) {

}

bool operator()(const Song\* song);

private:

std::string field;

};

// Функциональный тип для фильтра песен по исполнителю

class FindSongBySinger {

public:

FindSongBySinger(const std::string& field) : field(field) {

}

bool operator()(const Song\* song);

private:

std::string field;

};

// Функция-предикат для сортировки песен по убыванию рейтинга

bool CompareSongsRating(const Song\* a, const Song\* b);

// Концерт

class Concert

{

// Исключение - недопустимый порядковый номер

class InvalidOrdinalError {

public:

InvalidOrdinalError(size\_t value) : value(value) {

}

inline size\_t getValue() const {

return value;

}

private:

size\_t value;

};

public:

// Конструктор

Concert(int timeout);

// Деструктор

~Concert();

// Основная рабочая функция системы приёма заявок

// (общее меню для всех пользователей)

int run();

public:

// Псевдонимы для типов

using Users = std::map< std::string, std::string >;

using Songs = std::vector< Song\* > ;

using UserSongs = std::vector< std::pair< const Song\*, Users::iterator > >;

private:

// Зарегистрированные пользователи

Users users;

// Песни, достпуные для выбора пользователями

Songs songs;

// Время запуска системы приём заявок

std::chrono::steady\_clock::time\_point startTime;

// Время до начала концерта

int timeout;

// Выбранные песни

UserSongs userSongs;

// Авторизованный в данный момент пользователь

Users::iterator currUser;

// Функциональный тип для безусловной распечатки песни

struct PrintSong {

void operator()(const Song\* song);

};

// Функциональный тип для условной распечатки песни

class PrintAvailableSong {

public:

PrintAvailableSong(UserSongs& userSongs, Users::iterator currUser)

: userSongs(userSongs), currUser(currUser) {

}

void operator()(const Song\* song);

private:

UserSongs& userSongs;

Users::iterator currUser;

};

private:

// Авторизация существующего пользователя

bool signIn();

// Регистрация нового пользователя

bool signUp();

// Меню для авторизованного пользователя

bool runAuth();

// Выбор песнипользователем

void selectSong();

// Добавление новой песни

void addSong();

// Редактирование информации о песне

void editSong();

// Удаление песни

void delSong();

// Поиск песни по выбранному полю

void findSong();

// Ввод данных пользователя

Users::value\_type inputUser();

// Ввода данных песни

Song\* inputSong();

// Выбор номера песни

size\_t inputIndex(size\_t id);

// Проверка оставшегося времени до концерта

// и его запуск, в случае истечения времени

bool checkTimeout();

// Параметризированный поиск песни

template<class Finder>

Songs::iterator findSong(const Finder& finder) {

return find\_if(songs.begin(), songs.end(), finder);

}

// Распечатка списка песен

template<class Container, class Print>

void printSongs(Container container, Print print) {

if(container.empty())

std::cout << "Список песен пуст" << std::endl;

else {

std::cout << std::setw(4) << "№" << " Рейтинг Песня" << std::endl;

std::for\_each(container.begin(), container.end(), print);

}

}

// Формирование программы концерта по результатам гоолования

template<size\_t ConcertSongsCount>

Songs createConcertProgram() {

Songs result(songs.size());

std::copy(songs.begin(), songs.end(), result.begin());

std::sort(result.begin(), result.end(), CompareSongsRating);

if(ConcertSongsCount < result.size())

result.resize(ConcertSongsCount);

return result;

}

};

#endif

**Concert.cpp**

#include "Concert.h"

using namespace std;

Concert::Concert(int timeout) : timeout(timeout) {

currUser = users.end();

// В любой конфигурации с "Debug" для удобства отладки

// в систему изначально добавляется набор условных пользователей

// и набор из 4 песен

// Для отключения этой возможности нужно пересобрать проект

// в любой из конфигураций с "Release"

#ifdef \_\_DEBUG\_\_

users["a"] = "a";

users["b"] = "b";

users["c"] = "c";

users["d"] = "d";

users["e"] = "e";

songs.push\_back(new Song("Sonne", "Rammstein", 1));

songs.push\_back(new Song("Twenty One", "Cranberries", 2));

songs.push\_back(new Song("Only Time", "Enya", 3));

songs.push\_back(new Song("Our Farewell", "Within Temptation", 4));

#endif

startTime = chrono::steady\_clock::now();

checkTimeout();

}

Concert::~Concert() {

for(auto song : songs) {

delete song;

}

}

int Concert::run() {

int counter = 0;

int menu = -1;

while (menu != 0) {

cout << "\n\nДОБРО ПОЖАЛОВАТЬ\n\n";

cout << "[0] Завершение работы\n";

cout << "[1] Войти в систему подачи заявок\n";

cout << "[2] Зарегистрироваться в системе подачи заявок\n";

cout << endl << "Ваш выбор: ";

cin.sync();

cin >> menu;

cout << endl;

switch(menu) {

case 1:

if(signIn() && runAuth())

menu = 0;

break;

case 2:

if(signUp() && runAuth())

menu = 0;

break;

}

if(counter++ == 10) {

cout << "emergency exit" << endl;

break;

}

if(menu != 0 && checkTimeout()) {

break;

}

} // while()

cin.get();

return 0;

}

bool Concert::signIn() {

Concert::Users::value\_type user = inputUser();

currUser = users.find(user.first);

if(currUser == users.end() || currUser->second != user.second) {

cout << "Неверное сочетание имени пользователя и пароля" << endl;

currUser = users.end();

return false;

}

cout << "Авторизация пройдена" << endl;

return true;

}

bool Concert::signUp() {

Concert::Users::value\_type user = inputUser();

Users::const\_iterator it = users.find(user.first);

if(it != users.end()) {

cout << "Такой пользователь уже существует" << endl;

return false;

}

currUser = users.insert(user).first;

cout << "Регистрация завершена" << endl;

return true;

}

bool Concert::runAuth() {

bool exit = false;

bool running = true;

int counter = 0;

int menu = -1;

while (running) {

cout << "\n\nГЛАВНОЕ МЕНЮ\n\n";

cout << "[0] Завершение работы\n";

cout << "[1] Выйти из учётной записи\n";

cout << "[2] Общий cписок песен\n";

cout << "[3] Список песен для выбора\n";

cout << "[4] Проголосовать за песню\n";

cout << "[5] Добавить песню\n";

cout << "[6] Редактировать информацию о песне\n";

cout << "[7] Удалить песню\n";

cout << "[8] Найти песню\n";

cout << endl;

cout << "Ваш выбор: ";

cin.sync();

cin >> menu;

switch(menu) {

case 0:

running = false;

exit = true;

break;

case 1:

running = false;

break;

case 2:

printSongs(songs, PrintSong());

break;

case 3:

printSongs(songs, PrintAvailableSong(userSongs, currUser));

break;

case 4:

selectSong();

break;

case 5:

addSong();

break;

case 6:

editSong();

break;

case 7:

delSong();

break;

case 8:

findSong();

break;

}

if(counter++ == 10) {

cout << "emergency exit" << endl;

break;

}

if(checkTimeout()) {

exit = true;

break;

}

} // while()

currUser = users.end();

return exit;

}

void Concert::selectSong() {

try {

printSongs(songs, PrintAvailableSong(userSongs, currUser));

size\_t index = inputIndex();

Song\* song = songs[index];

song->vote();

userSongs.push\_back(UserSongs::value\_type(song, currUser));

} catch(InvalidOrdinalError&) {

cout << "Недопустимое значение" << endl;

}

}

void Concert::addSong() {

Song\* song = inputSong(songs.size() + 1);

const auto it = find(songs.begin(), songs.end(), song);

if(it != songs.end()) {

cout << "Такая песня уже есть" << endl;

delete song;

return;

}

songs.push\_back(song);

printSongs(songs, PrintSong());

}

void Concert::editSong() {

try {

printSongs(songs, PrintSong());

size\_t index = inputIndex();

songs[index] = inputSong(index + 1);

printSongs(songs, PrintSong());

} catch(InvalidOrdinalError&) {

cout << "Недопустимое значение" << endl;

}

}

void Concert::delSong() {

try {

printSongs(songs, PrintSong());

size\_t index = inputIndex();

songs.erase(songs.begin() + index);

for(size\_t i = 0; i < songs.size(); i++)

songs[i]->setId(i + 1);

printSongs(songs, PrintSong());

} catch(InvalidOrdinalError&) {

cout << "Недопустимое значение" << endl;

}

}

void Concert::findSong() {

int menu = -1;

string s;

cout << "По какому параметру искать песню?\n";

cout << "[0] Отменить поиск\n";

cout << "[1] Название\n";

cout << "[2] Исполнитель" << endl;

cout << "Ваш выбор: ";

cin.sync();

cin >> menu;

cout << "Введите значение для поиска: ";

cin >> s;

Songs::iterator it = songs.end();

switch(menu) {

case 1:

it = findSong(FindSongByName(s));

break;

case 2:

it = findSong(FindSongBySinger(s));

break;

default:

return;

}

if(it == songs.end()) {

cout << "Песня не найдена" << endl;

return;

}

cout << "Песня найдена, её номер " << (\*it)->getId() << endl;

}

Concert::Users::value\_type Concert::inputUser() {

string name, pass;

cout << "Введите имя пользователя: ";

cin >> name;

cout << "Введите пароль: ";

cin >> pass;

return Users::value\_type(name, pass);

}

Song\* Concert::inputSong(size\_t id) {

string name, singer;

cout << "Введите название песни: ";

cin >> name;

cout << "Введите исполнителя: ";

cin >> singer;

return new Song(name, singer, id);

}

size\_t Concert::inputIndex() {

size\_t index;

cout << "Введите номер песни: ";

cin >> index;

if(0 == index || index > songs.size())

throw InvalidOrdinalError(index);

return index - 1;

}

void Concert::PrintSong::operator()(const Song\* song) {

cout << setw(4) << song->getId() << ". " << setw(7) << song->getVotes() << " \"" << song->getName() << "\" (" << song->getSinger() << ")" << endl;

}

void Concert::PrintAvailableSong::operator()(const Song\* song) {

Concert::UserSongs::iterator it = find(userSongs.begin(), userSongs.end(), Concert::UserSongs::value\_type(song, currUser));

if(it == userSongs.end()) {

cout << setw(4) << song->getId() << ". " << setw(7) << song->getVotes() << " \"" << song->getName() << "\" (" << song->getSinger() << ")" << endl;

}

}

bool Concert::checkTimeout() {

bool result = false;

auto now = chrono::steady\_clock::now();

auto delta = chrono::duration\_cast<chrono::seconds>(now - startTime).count();

auto rest = timeout - delta;

if(rest <= 0) {

cout << "Концерт начинается:" << endl;

Songs program = createConcertProgram<2>();

printSongs(program, PrintSong());

result = true;

} else {

cout << "До начала концерта осталось секунд: " << rest << endl;

}

return result;

}

bool FindSongByName::operator()(const Song\* song) {

return song->getName() == field;

}

bool FindSongBySinger::operator()(const Song\* song) {

return song->getSinger() == field;

}

bool CompareSongsRating(const Song\* a, const Song\* b) {

return a->getVotes() > b->getVotes();

}