Оглавление

[Оглавление 2](#_Toc40021963)

[Введение 3](#_Toc40021964)

[Основная часть 4](#_Toc40021965)

[Определение архитектуры «клиент – сервер» 4](#_Toc40021966)

[Основной функционал 5](#_Toc40021967)

[Функционал класса Data 5](#_Toc40021968)

[Функционал класса Server 6](#_Toc40021969)

[Функционал класса Client 9](#_Toc40021970)

[Тесты и работа программы 10](#_Toc40021971)

[Команда CREATE 11](#_Toc40021972)

[Команда READ 11](#_Toc40021973)

[Команда UPDATE 11](#_Toc40021974)

[Команда DELETE 12](#_Toc40021975)

[Команда SEARCH 12](#_Toc40021976)

[Команда FILTER 13](#_Toc40021977)

[Команда MAX COST 13](#_Toc40021978)

[Команда MIN SIZE 14](#_Toc40021979)

[Команда CHOSEN GENRE 14](#_Toc40021980)

[Заключение 15](#_Toc40021981)

[Список литературы 16](#_Toc40021982)

# Введение

Целью курсовой работы является реализация Информационно-поисковой системы – «Продажа музыкальной аудио-видео продукции». Необходимо реализовать клиент-сервер систему с базой данных. Система должна поддерживать стандартный функционал (CRUD – create, read, update, delete), то есть обеспечивать: создание, чтение, редактирование, удаление записей. Помимо этого будет реализован поиск и фильтр непосредственно для базы данных. Также, что немало важно для таких систем, необходимо обеспечить поддержку отчетности – выдачу отчета по критерию. Для реализации был выбран язык программирования C++.

# Основная часть

## Определение архитектуры «клиент – сервер»

Не все компьютеры и программы, находящиеся в рамках одной информационной сети обладают одинаковыми правами. Какие-то из них владеют ресурсами, а другие могут всего лишь обращаться к этим ресурсам. Компьютер (программу), управляющий ресурсами, определяют как сервер данного ресурса. Клиент и сервер какого-либо ресурса могут находиться в рамках одной вычислительной системы или сети.

Основной принцип технологии «клиент-сервер» заключается в разделении функций приложения на такие группы как:

* Функции управления ресурсами (файловой системой, базой данных).
* Ввод и отображения данных (взаимодействие с пользователем).
* Прикладные функции, обеспечивающие работоспособность первых двух групп.



Рисунок 1 – иллюстрация архитектуры «клиент-сервер»

## Основной функционал

Программа реализована с помощью функций и классов, разбитых на отдельные файлы.

Работа программы организована при помощи взаимодействия 3-х классов:

* class Data – объекты класса определяют информацию хранящуюся в базе
* class Server – объект класса осуществляет непосредственное управление базой дынных.
* class Client – объект класса позволяет взаимодействовать с программой – «сервером» с помощью набора команд.

### Функционал класса Data

Класс имеет набор приватных полей, которые отвечают за хранящуюся информацию. Так как для реализации была выбрана информационно-поисковая система (ИПС) – «Продажа музыкальной аудио-видео продукции», уместны такие значения как:

* id – идентификатор позиции в базе данных;
* name – наименование позиции;
* author – автор аудио-видео продукции;
* genre – жанр продукции;
* extension – разрешение реализуемой продукции;
* size – размер (в килобайтах) продукции;
* cost – цена товара;

Также реализован набор сеттеров\*, геттеров\* (прим. setName(…), getName()), которые отвечают за изменение и выдачу информации, хранящейся в приватных полях соответственно. Совершенно стандартный функционал для большинства типов хранения данных.

Определены перегрузки операторов:

* присваивания (=);
* потокового ввода (>>);
* потокового вывода (<<);

Перегрузка оператор присваивания была реализована для изменения полей дынных вовремя работы с базой. К примеру, во время удаления значения из базы, необходимо произвести сдвиг элементов в структуре хранения данных, то есть изменить значения идентификаторов позиций.

Перегрузки операторов ввода/вывода необходимы для взаимодействия с элементами в текстовой базе. В зависимости от функции базы данных, нужно вводить/выводить элементы в консоль/файл, соответственно для сокращения объема кодовой базы, были введены данные перегрузки операторов.

### Функционал класса Server

Данный класс является «основой» проекта. Он осуществляет прямое взаимодействие с базой данных, то есть является программой – «сервером». Класс имеет возможность перезаписи, добавления, удаления данных. Говоря простым языком, имеет «полную власть» над базой данных. Server универсален и подходит для управления базами, хранящими любые типы данных.

Данный класс имеет такие приватные поля как:

* baseName – имя файла базы данных
* currentId – указатель на последний идентификатор в базе
* base – основа для хранения базы данных в оперативной памяти.

В качестве структуры данных для base был выбран вектор (массив). Выбрана именно эта структура данных потому, что она поддерживает доступ по индексу элемента, нет никаких ограничений на хранимые данные, быстрое добавление элементов в конец структуры.

Определены конструктор и деструктор класса. Конструктор принимает имя файла базы данных в системе.

Server::Server(std::**string** \_baseName)

{

baseName = \_baseName;

loadBase(baseName);

}

Код 1. Конструктор класса Server

Конструктор вызывает функцию обеспечения loadBase(…), которая в свою очередь загружает текстовую базу данных в оперативную память, и выставляет указатель на последний элемент.

Реализован функционал CRUD – create, read, update, delete; посредством функций create(…), read(…), update(…), del(…).

* create – принимает на вход объект класса Data, осуществляет одновременную запись элемента в структуру данных базы во временной памяти и запись элемента в текстовую базу данных.
* read – принимает на вход целочисленную переменную id, функция возвращает строку базы данных по соответствующему идентификатору.
* update - принимает на вход целочисленную переменную id, а также переменную типа Data, заменяет строку с соответствующим id на введенную.
* del – принимает на вход целочисленную переменную id, удаляет из базы элемент с соответствующим идентификатором и сдвигает все элементы с большим значением id на одну позицию.

**for** (**int** i = id-1; i < base.size()-1; ++i) {

base[i+1].setId(i+1);

base[i] = base[i+1];

}

base.pop\_back();

Код 2. Часть функции del, показывающая сдвиг

Организована фильтрация и поиск, по заранее указанным полям.

Поиск ведется по полю name у данных. Это соответственно позволяет пользователю узнать, есть ли данная продукция в базе. Чтобы найти произведение, сервер обходит подгруженную в оперативную память базу посредством алгоритма линейного поиска.

Фильтрация ведется по полю cost. Ориентируясь на техническое задание, это поле является одним из самых релевантных для фильтрации.

std::**vector**<Data> Server::filter(**int** cost, **bool** (\*compareFunction)(**int**, **int**)){

std::**vector**<Data> filtered;

**for** (**const** **auto**& item: base) {

**if** (compareFunction(cost, item.getCost())) {

filtered.push\_back(item);

}

}

**return** filtered;

}

Код 3. Функция filter

Механизмы C++ позволяют передавать в качестве параметра функции другую функцию. Соответственно это помогает сократить кодовую базу. При организации клиента в функцию filter передается лямбда-выражение. Это помогает избежать создания лишних функции или блоков с условиями.

filtered = base.filter(bufferNum, [](**int** lhs, **int** rhs) -> **bool** { **return** lhs < rhs; });

Код 4. Пример использования функции filter

filter возвращает вектор с элементами типа Data, для последующего вывода в консоль.

Функции-отчеты представлены как:

* report10MaxCost() - выводит десять самых дорогих произведений в базе.
* report10MinSize() - выводит десять самых минимальных по размеру элементов реализуемой продукции.
* reportChosenGenre(...) - на вход принимает строку - выбранный жанр для отчета, далее находит все позиции с указанным жанром и выводит сформированный отчет в консоль.

Также у данного класса есть прикладная функция writeHat(…), которая принимает на вход поток вывода «шапки» базы. Эта функция помогает избежать повторения кода, к примеру, при выводе отчетов или перезаписи базы при удалении элемента.

Рисунок 2 - «шапка» базы

### Функционал класса Client

Посредством данного класса происходит «общение» пользователя с сервером.

Приватными полями данного класса выступают структуры данных типа map (C++). Выбрана такая структура данных в связи с тем, что структура switch не может принимать тип данных string. Поэтому ввод осуществляется посредством подмены команд соответствующими значениями словаря.

* commandMap – содержит основные команды первичного пользовательского ввода ({“CREATE”, 1});
* updateMap – содержит команды выбора поля при редактировании ({“NAME”, 1});
* filterKeyMap – содержит команды выбора направления фильтрации ({“MORE”, 1});

Еще одно приватное поле – baseName. В него записывается имя базы данных, с которой будет работать пользователь. Запись осуществляется либо внутри функции main посредством конструктора, либо во время работы функции userInterface. Если имя базы нигде не было указано, и пользователь отказался вводить его внутри функции userInterface, выдается сообщение об ошибке и работа программы прекращается.

userInterface() является основной функцией класса Client в ней осуществлен механизм пользовательского ввода. Внутри нее создается объект класса Server, с помощью которого происходит взаимодействие пользователя с базой данных.

Постоянный ввод команд обеспечивается с помощью бесконечного цикла do-while из которого можно выйти с помощью команды EXIT. Внутри цикла определены буферные переменные, в которые производится запись пользовательского ввода, в которые выводятся результаты работы функций класса Server. С каждой итерацией цикла система просит пользователя ввести команду для работы с базой. Далее, после введения команды, она передается в оператор switch, который ищет соответствие с доступными командами в своем теле. В случае отсутвия команды условие уходит в default, нде пользователя предупреждают о том, что такой команды нет.

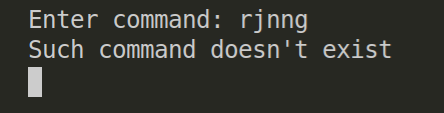


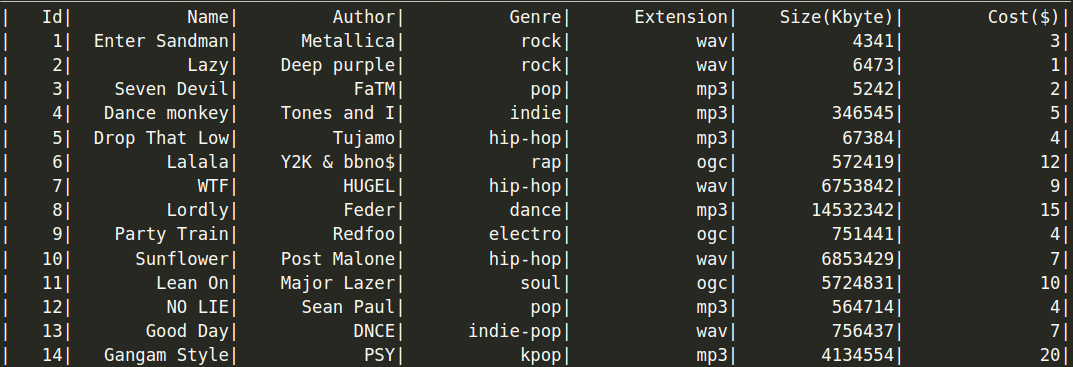
Рисунок 3 – иллюстрация обработки неправильного ввода

В случае успеха оператор switch переносит пользователя в следующий блок, где ему доступны дальнейшие действия для работы с базой. Внутри блоков также есть механизмы, которые перехватывают неправильный ввод и выводят пользователю сообщение об ошибке.

Система выстроена так, чтобы у пользователя была возможность создавать новые команды, путем быстрого добавления желаемой команды в тело оператора switch.

# Тесты и работа программы

База имеет вид таблицы, в столбцах которой уже есть некоторая информация.

Рисунок 4 – вид базы

Далее в зависимости от выбранных пользователем команд, вид базы меняется. По мере ввода команд система дает пользователю подсказки, что именно нужно вводить, какие поля нужны для фильтрации, какие для поиска и т.п.

## Команда CREATE

Данная команда посредством пользовательского ввода создает новую позицию в базе данных

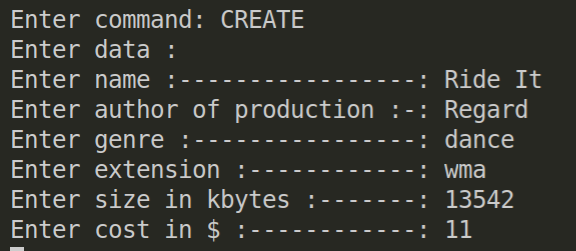
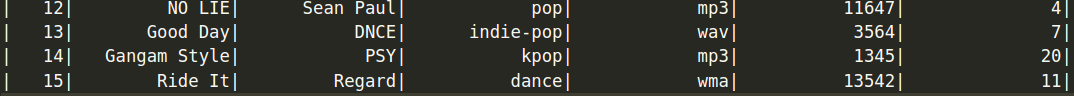


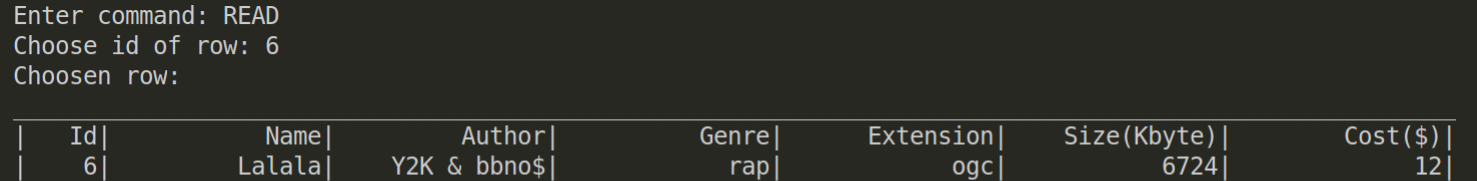
Рисунок 5 — пример пользовательского ввода команды CREATE

После ввода данных в базе появляется новая позиция с автоматически выставленным Id. Пользователь не может самостоятельно ввести Id во избежание появления двух идентичных записей в базе.

Рисунок 6 — появление новой записи в базе

## Команда READ

Посредством данной команды происходит чтение записей из базы. Пользователь вводит Id и выводится полная информация по позиции с соответствующим идентификатором. Для чтения выбран именно Id, потому что шансы ошибки ввода меньше. Так как Name и Author у продукции может повторяться, а Id — нет.

Рисунок 7 — применение команды READ

## Команда UPDATE

С помощью команды UPDATE пользователь может произвести замену полей в любой записи. Программа предлагает пользователю выбор определенного поля, соответственно полностью запись менять не нужно.

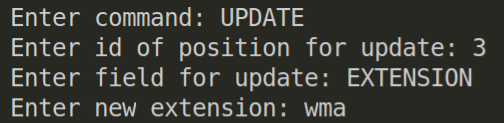
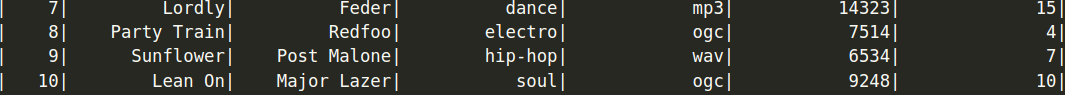
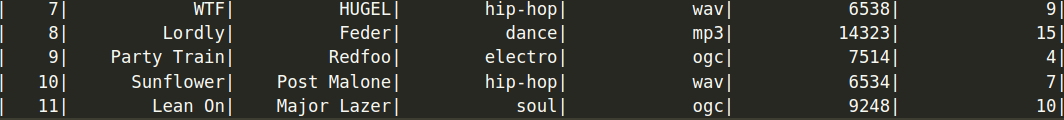


Рисунок 8 — применение команды UPDATE

Рисунок 9 — поле до/после

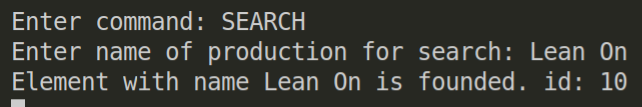
## Команда DELETE

Данная команда позволяет удалить любую запись из базы по ее уникальному идентификатору — Id. После выполнения данной операции все Id сдвигаются, и у каждой записи появляется новый Id.

Рисунок 10 — база до и после удаления 7-й записи

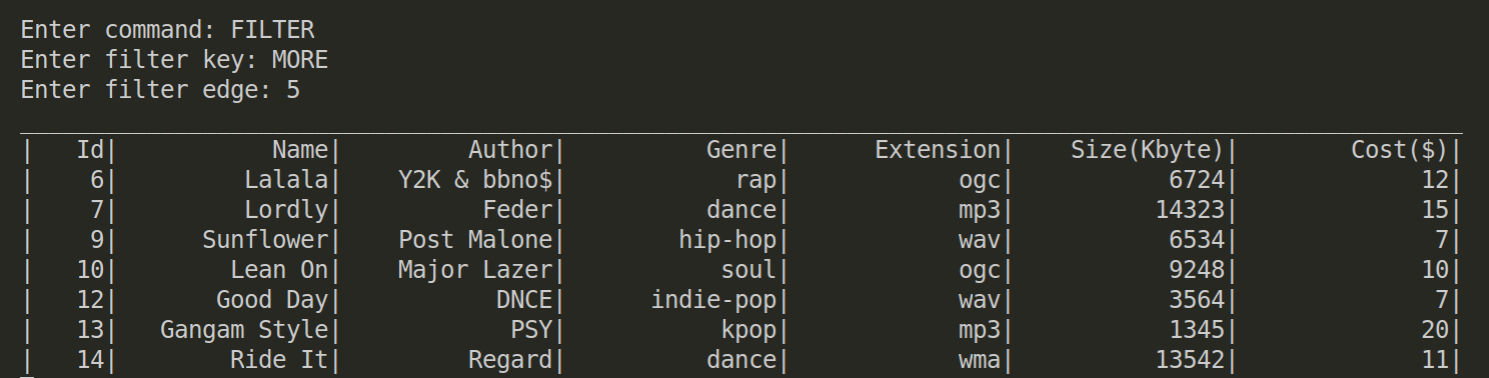
## Команда SEARCH

Посредством команды SEARCH производится поиск позиции в базе данных по полю NAME. В случае, если такой продукции нет, система выдает предупреждение об отсутствии, в случае успеха — id найденной позиции в базе.

Рисунок 11 - применение команды SEARCH

## Команда FILTER

Данная команда выводит таблицу записей в консоль в зависимости от ключа фильтра — {MORE, EQUAL, LESS}. То есть создает таблицу значений из элементов: больших, равных, меньших указанного. Фильтр работает по полю COST, так как предполагается то, что это одно из самых релевантных полей для системы продаж.

Рисунок 12 — пример работы команды FILTER

В данном случае выводятся все позиции, стоимость которых, больше 5 у.е. за штуку.

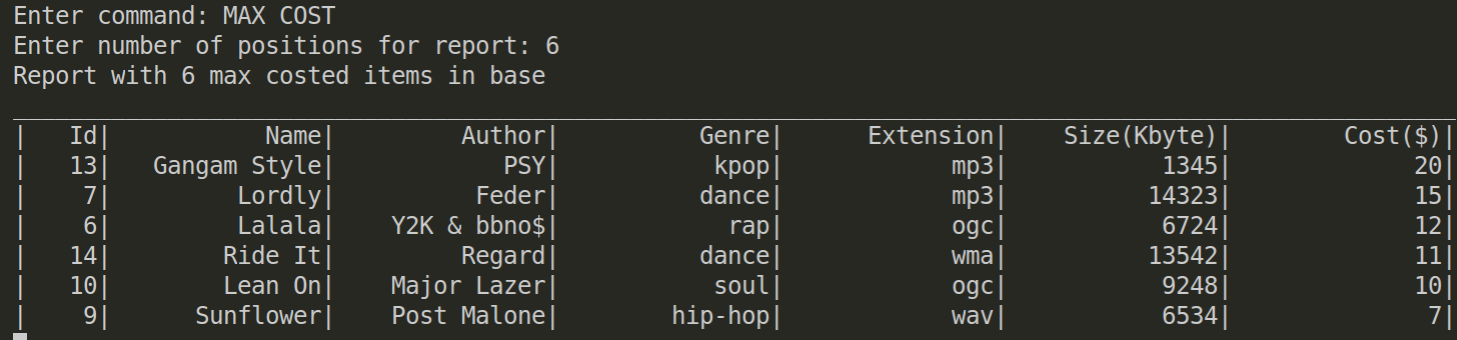
## Команда MAX COST

Также в базе представлены команды отчетов, одна из них — MAX COST. Пользователю необходимо ввести количество позиций для вывода, далее система с помощью стандартной библиотеки С++ - <algorithm> сортирует и выводит n первых позиций удовлетворяющих требованию — самые дорогие позиции в базе.

**sort**(report.begin(), report.end(), [](Data lhs, Data rhs) -> bool { return lhs.getCost() > rhs.getCost(); });

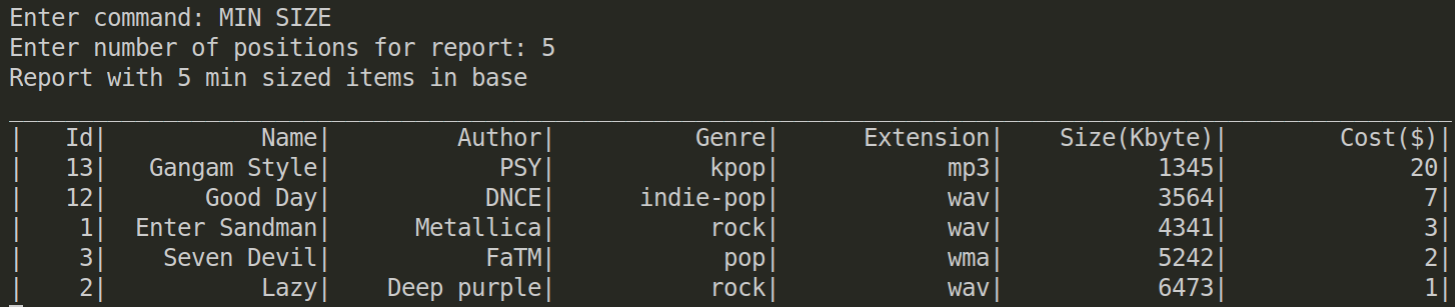
Код 5. Сортировка базы

В случае, если в базе меньше n позиций, система выводит предупреждение и отчет содержащий все отсортированные позиции в базе.

Рисунок 13 — отчет — n позиций с максимальной стоимостью

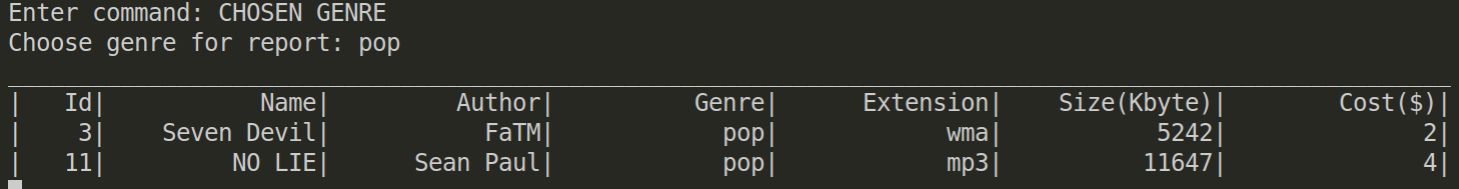
## Команда MIN SIZE

Команда работает по такому же принципу как и MAX COST. Разница только в направлении сортировки, так как создается отчет из n минимальных по размеру позиций.

Рисунок 14 — отчет — n позиций с минимальным размером

## Команда CHOSEN GENRE

Посредством команды CHOSEN GENRE создается отчет по выбранному жанру. То есть происходит полный обход базы, загруженной в оперативную память, и в отдельный вектор загружаются значения, удовлетворяющие условию — соответствующий жанр. В случае, если в базе нет такого жанра, выводится предупреждение об отсутствии.

Рисунок 15 — отчет — все позиции жанра pop

# Заключение

В ходе выполнения курсовой работы была реализована информационно-поисковая система (ИПС) - «Продажа музыкальной аудио-видео продукции» на языке C++. При разработке были продемонстрированы принципы построения приложений такого рода.

Выполнение курсовой работы помогло закрепить изученный материал и навыки, полученные в ходе лекций и на лабораторных работах, систематизировать знания языка и научиться применять их в поставленных задачах.

# Список литературы

1. Дьюхарст Программирование на C++ / Дьюхарст, Старк Стефан; , Кэти. - М.: ДиаСофт, 2015. - 272 c.  
2. Страуструп, Б. Язык программирования C++: Специальное издание / Б. Страуструп; Пер. с англ. Н.Н. Мартынов. — М.: БИНОМ, 2012. — 1136 c.

3. Раскин, Д. Интерфейс. Новые направления в проектировании компьютерных систем.: Символ-Плюс, 2007. - 272 с.