Кафедра математического и аппаратного обеспечения

информационных систем

КУРСОВОЙ ПРОЕКТ

По дисциплине “Технологии и методы программирования”

на тему:

Проектирование и реализация класса «Театр»

Выполнил: студент гр. ИВТ 21-22

Туктин К.В. \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Научный руководитель

к.ф.-м.н., доцент

Филиппов В.П. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc153360053)

[Глава 1 ТЕОРИТИЧЕСКАЯ ЧАСТЬ 5](#_Toc153360054)

[1.1 Основная информация об языке программирования C++ 5](#_Toc153360055)

[1.2 Описание подключаемых библиотек и их основных функций 5](#_Toc153360056)

[1.3 Описание предметной области 8](#_Toc153360057)

[Глава 2 ПРАКТИЧЕСКАЯ ЧАСТЬ 12](#_Toc153360058)

[2.1 Описание структуры пользовательского меню 12](#_Toc153360059)

[2.2 Алгоритмы и методы в программе 15](#_Toc153360060)

[2.3 Демонстрация работы программы 32](#_Toc153360061)

[ЗАКЛЮЧЕНИЕ 36](#_Toc153360062)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 37](#_Toc153360063)

[ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ 39](#_Toc153360064)

# ВВЕДЕНИЕ

Оценка актуальности НИР. На сегодняшний день отмечается рост интереса к развлекательным и культурно-просветительским заведениям, таким как музеи, театры, кинотеатры, и библиотеки. Информационные технологии успешно решают проблемы в различных сферах, включая работу с клиентами. Однако в театральной области Российской Федерации замечается недостаточное использование высоких технологий, несмотря на высокий спрос. Внутренние потребности театров включают в себя обработку данных, работу с клиентами и анализ продаж, что требует современных технологий [8]. Для решения этой проблемы Александрийский театр разработал свою CRM-систему, обеспечивающую эффективное взаимодействие с клиентами и анализ продаж [9]. Это подчеркивает актуальность создания программного обеспечения для театральных организаций. Разработка такого ПО требует особого внимания к процессу разработки, основанному на архитектурном подходе, модульности написания и экономии ресурсов системы

На фоне этого можно выделить ключевую проблему исследования в рамках курсовой работы, а именно, разработку структурированного и гибкого класса "Театр" на языке C++, способного эффективно управлять информацией о залах, постановках, свободных и забронированных местах и других аспектах театральной деятельности и предоставлять пользователю посредством интерфейсных методов возможность удобной работы с объектами этого класса через консоль.

Цели курсовой работы:

* закрепление полученных в процессе курса «технологии и методы программирования» навыков разработки ПО с использованием принципов и методов объектно-ориентированного программирования (ООП) через написание программного продукта, решающего проблему данной НИР;
* освоение архитектурного (модульного) подхода к разработке ПО;
* проектирование и реализация класса для процессов управления данными театра, позволяющее осуществлять ввод и вывод информации о предстоящих постановках, составление расписания, удобную работу с большим набором данных, бронирование мест и быстрый поиск.

Задачи курсовой работы:

* исследовать и описать предметную область;
* организовать связь между полями основного класса и структур данных;
* реализовать основные методы работы с данными, указанными в требовании к программе;
* осуществить возможности функций для администрирования и бронирования списка постановок;
* разработать пользовательское консольное меню, позволяющее в полной мере проверить работу с объектами класса «Театр»;
* протестировать программу на работу с некоторым набором входных данных.

# ТЕОРИТИЧЕСКАЯ ЧАСТЬ

## Основная информация об языке программирования C++

C++ является одним из наиболее популярных объектно-ориентированных языков программирования. Он был создан Бьярном Страуструпом в начале 1980-х годов как расширение языка C. C++ сочетает в себе процедурное и объектно-ориентированное программирование, предоставляя высокоуровневые абстракции и возможность написания эффективного кода на низком уровне [6].

Компилятор — это программа, которая преобразует исходный код на языке программирования в машинный код, понятный компьютеру. Visual Studio — это интегрированная среда разработки (IDE) от Microsoft, которая включает в себя компилятор C++. Она обеспечивает удобное создание, отладку и сопровождение программного обеспечения [10]. Именно редактор кода MS Visual Studio будет использоваться для написания и отладки программы курсового проекта.

ООП— парадигма, в которой программа организуется вокруг объектов, взаимодействующих между собой. ООП включает в себя принципы наследования, инкапсуляции и полиморфизма. Наследование позволяет создавать новые классы на основе существующих, инкапсуляция скрывает реализацию объектов, а полиморфизм обеспечивает возможность использования одних и тех же имён для разных функций и объектов разных типов [7].

## Описание подключаемых библиотек и их основных функций

C++ является мощным и гибким языком программирования, который обладает широким набором библиотек, предназначенных для различных сфер программирования. В отличие от некоторых других языков, C++ предоставляет разработчикам доступ к обширной коллекции стандартных и сторонних библиотек, что делает его библиотечным языком программирования высокого уровня.

Перейдём к описанию библиотек, подключаемых к приложению, и прописанных в них функций и структур данных, используемых в программе.

Библиотека <iostream> — это библиотека в языке программирования C++, предоставляющая основные средства для ввода и вывода данных. Она включает в себя объекты и функции, обеспечивающие взаимодействие с консолью, такие как, например, cin (стандартный ввод) и cout (стандартный вывод).

Функции fill() и width() используются для управления форматированием вывода. fill устанавливает символ заполнения, width – минимальную ширину поля вывода.

Библиотека <string> — это библиотека в C++, предоставляющая возможности для работы со строками (последовательностями символов). Строки могут быть изменяемыми и предоставляют удобные методы для работы с текстовой информацией, что очень полезно для реализации решения поставленных задач, которые связанны с работой с теми или иными строковыми данными [11].

Из библиотеки <string> будут взяты следующие функции:

istream& getline(istream& is, string& str, char delim), считывающую строку из входного потока is в строку str до символа разделителя delim и возвращающую ссылку на входной поток.

size\_t find(const string& str, size\_t pos = 0) const noexcept , ищущую подстроку str в строке, начиная с позиции pos и возвращающую позицию первого вхождения или string::npos, если ничего не найдено.

string& erase(size\_t pos = 0, size\_t count = npos), удаляющую count символов, начиная с позиции pos и возвращающую ссылку на строку.

bool empty() const noexcep, возвращающая истину, если строка пуста, и ложь в противном случае.

size\_t length() const noexcep, возвращающая количество символов в строке.

Библиотека <vector> — это библиотека C++, реализующая динамический массив, который может изменять свой размер во время выполнения программы, что оказывается очень удобным и полезным, когда дело касается изменения массива данных, хранящихся в программе [12].

Указанная библиотека предоставляет приложению следующие функции:

void push\_back(const T& value), добавляющую элемент value в конец вектора.

bool empty() const noexcept, возвращающую истину, если вектор пуст, и ложь в противном случае.

void clear() noexcept, очищающую содержимое вектора.

Для работы с файлами было также решено подключить библиотеку <fstream>. Типы ifstream, ofstream, fstream предоставляют возможности для ввода, вывода и одновременного ввода/вывода файлов соответственно [13].

Эта библиотека предоставляет следующий полезный функционал:

bool is\_open() const noexcept проверяет, открыт ли файл.

bool eof() const noexcept проверяет, достигнут ли конец файла.

void close() noexcept закрывает файл.

Стоит таже обратить внимание на заголовочный файл в Microsoft Windows API — <Windows.h>. Он предоставляет доступ к различным функциям для работы с операционной системой Windows. Например, для решения задач курсовой работы были взяты следующие функции [14].

Функция void GetLocalTime(LPSYSTEMTIME lpSystemTime) получает текущее системное время и записывает его в структуру SYSTEMTIME. А поля wDay, wMonth, wYear в получившейся строке содержат текущие день, месяц и год соответственно. Эта функция нужная для проверки корректности данных даты, так как в информационно системе (ИС) театра могут храниться только грядущие постановки, а никак не прошедшие.

Функции SetConsoleCP(UINT wCodePage), SetConsoleOutputCP(UINT wCodePage) устанавливают кодовую страницу для консольного ввода и вывода.

Кодировка Windows-1251 планируется к использованию в программе, так как она способна поддерживать кириллические символы, введённые с клавиатуры [15].

## Описание предметной области

Театр — это уникальная организация, направленная на подготовку, организацию и показ спектаклей, а также предоставление сопутствующих услуг в области сценического искусства. Основной задачей театра является формирование и удовлетворение потребностей зрителей в высококачественном развлечении и культурном опыте. В представленном описании выделены основные функции театра, которые можно систематизировать следующим образом:

Требования к ИС театра:

* удобная система ведения репертуара;
* возможность подключения и управления различными залами и сценами;
* функции удобной ориентации в массиве данных о постановках, хранящихся в ИС театра;
* формирование расценок на билеты и создание схем залов;
* систематизация процесса бронирования билетов и управление информацией о местах.

Разрабатываемый абстрактный класс «Театр» должен будет воплощать в себе все основные свойства автоматизированной информационной системы для реализации деятельности подобных театров, и при этом он должен быть достаточно гибким, чтобы с помощью него можно было определить целый ряд всевозможных театров со своими количеством залов, их параметрами, со своими постановками и своим названием. Предполагается, что система будет предоставлять следующий функционал:

Афиша:

* дата и время проведения постановки;
* детали о каждой постановке, включающие жанр и цену на билет;
* информация о зале проведения постановки, количестве свободных мест.

Залы театра:

* количество зрительских залов театров;
* определения характеристик каждого из залов (количества мест).

Управление залами и сценами:

* возможность подключения и управления различными залами, сценами и площадками;
* создание и редактирование схем залов для продажи билетов.

Бронирование и обработка заказов:

* систематизация процесса бронирования мест в зале:
* уточнение информации о свободных и занятых местах.

В рамках разработки абстракции театра, основываясь на её ИС, проведено абстрагирование предметной области с использованием инфологической модели. Инфологическая модель представляет собой структуру данных, отображающую ключевые аспекты театральной деятельности с точки зрения информационных связей между сущностями. В системе любого проекта сущность представлена в виде экземпляров. Каждый из экземпляров сущности будем называть также объектом класса "Театр". В рамках исследуемой предметной области сущностями могут быть такие театры или сценические заведения, как "Театр Оперы и Балета", "Филармония", "Русский драматический театр" и тому подобные.

У всякой сущности есть свои атрибуты, которые их характеризуют, описывая их составляющие. В данном случае приведём атрибуты сущности «Театр», к которым будут относится наименование театра, количество залов, массив залов и список постановок. Атрибуты также могут называться полями. Сущность "Театр" можно охарактеризовать и некоторыми другими полями, но для решаемой научно-исследовательской проблемы это не требуется, поэтому достаточно обойтись предложенным набором атрибутов. Атрибуты образующие тот или иной экземпляр сущности одинаковы для каждого из них, но их значения могут отличаться. Например, у театров «Театр Оперы и Балета» и «Русский драматический театр» есть один и тот же атрибут «Количество залов», но значения этого атрибута у каждого из этих двух театров отличается.

Отдельно стоит упомянуть о типах связей между представленными сущностями. Между сущностями "Театр" и "Постановка" установлена связь "1:М" (один-ко-многим), так как каждый театр может проводить несколько постановок, но каждая постановка принадлежит только одному театру. Между сущностями "Театр" и "Зал" также установлена связь "1:М", так как у театра может быть несколько залов, но каждый зал принадлежит только одному театру. Между сущностями "Постановка", "Дата" и "Время" установлены связи "М:1" (многие-к-одному), так как одна постановка может иметь несколько дат и времени проведения, но каждая конкретная дата и время принадлежат только одной постановке. Между сущностью "Зал" и "Постановка" установлена связь "М:1", так как одна постановка может проходить в нескольких залах, но каждый зал принадлежит только одной постановке в конкретный момент времени [16]

Для наглядности представления связей между структурами данных класса «Театр» приведём ER-диаграмму (см. рисунок 1.1). Для построения ER-диаграммы класса воспользуемся нотацией Мартина, поскольку она обеспечивает компактность и информативность.

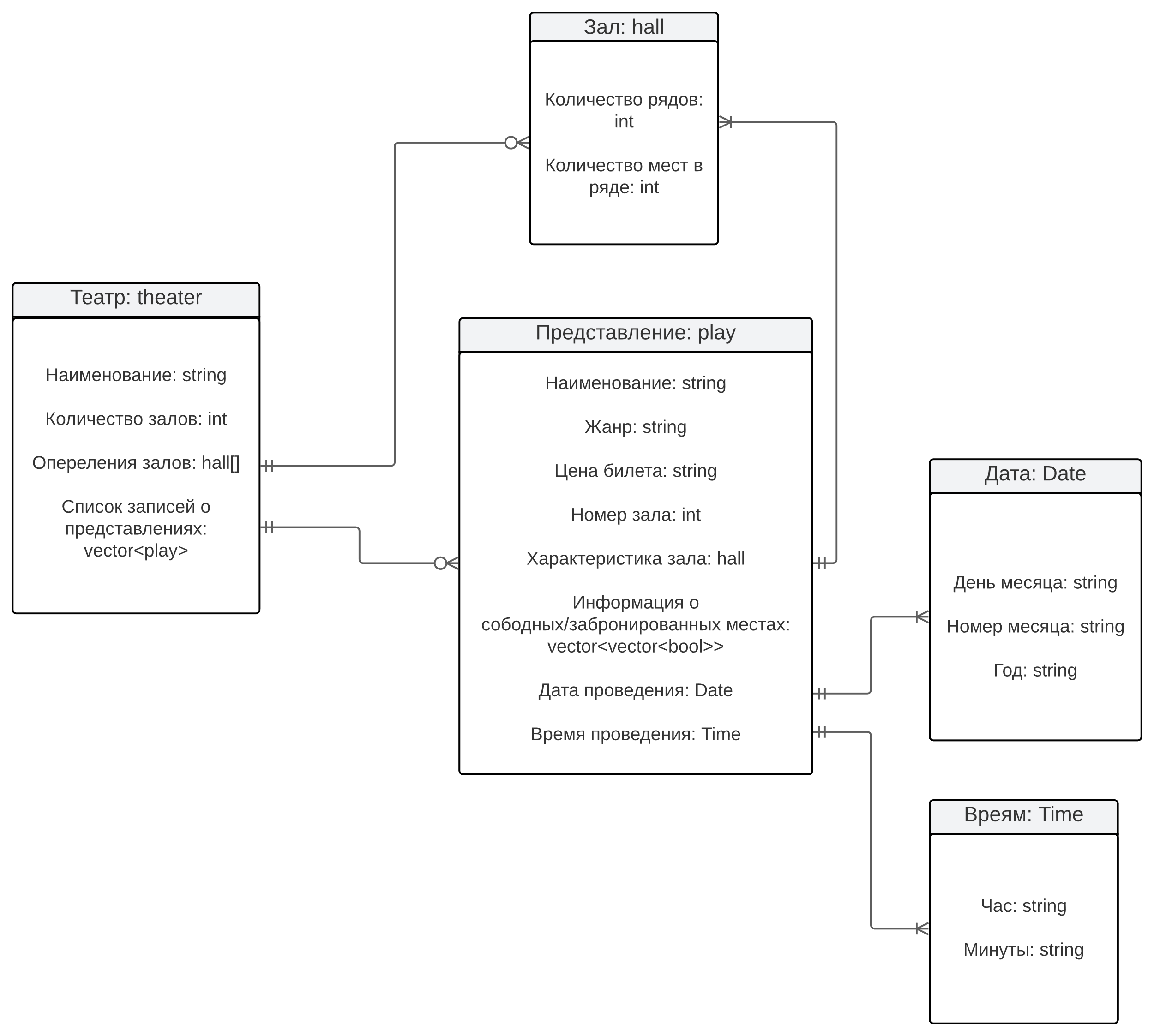


Рисунок . — ER-диаграмма данных класса «Театр»

Представленная ER-диаграмма наглядно иллюстрирует структуру связей данных проектируемой абстракции театра и позволяет перейти непосредственно к разработке программного кода курсовой работы.

# ПРАКТИЧЕСКАЯ ЧАСТЬ

## Описание структуры пользовательского меню

Поскольку конечный результат должен быть доступным и понятным в использовании широкому кругу лиц, необходимо сделать возможным управление разрабатываемого класса без взаимодействия с кодом проекта. Для этого в программе должны быть приписаны пользовательские меню. Во время исследования предметной области работы было принято соглашение о том, что пользователь будет взаимодействовать с классом theater посредством консольного ввода и вывода данных. Из-за обилия функционала класса разобьём пользовательское меню на три смысловые составляющие:

* меню создания объекта класса theater (см. рисунок 2.1);
* меню работы с основными функциями созданного объекта (см. рисунок 2.2);
* меню проверки работы перегруженных операторов класса (см. рисунок 2.3).

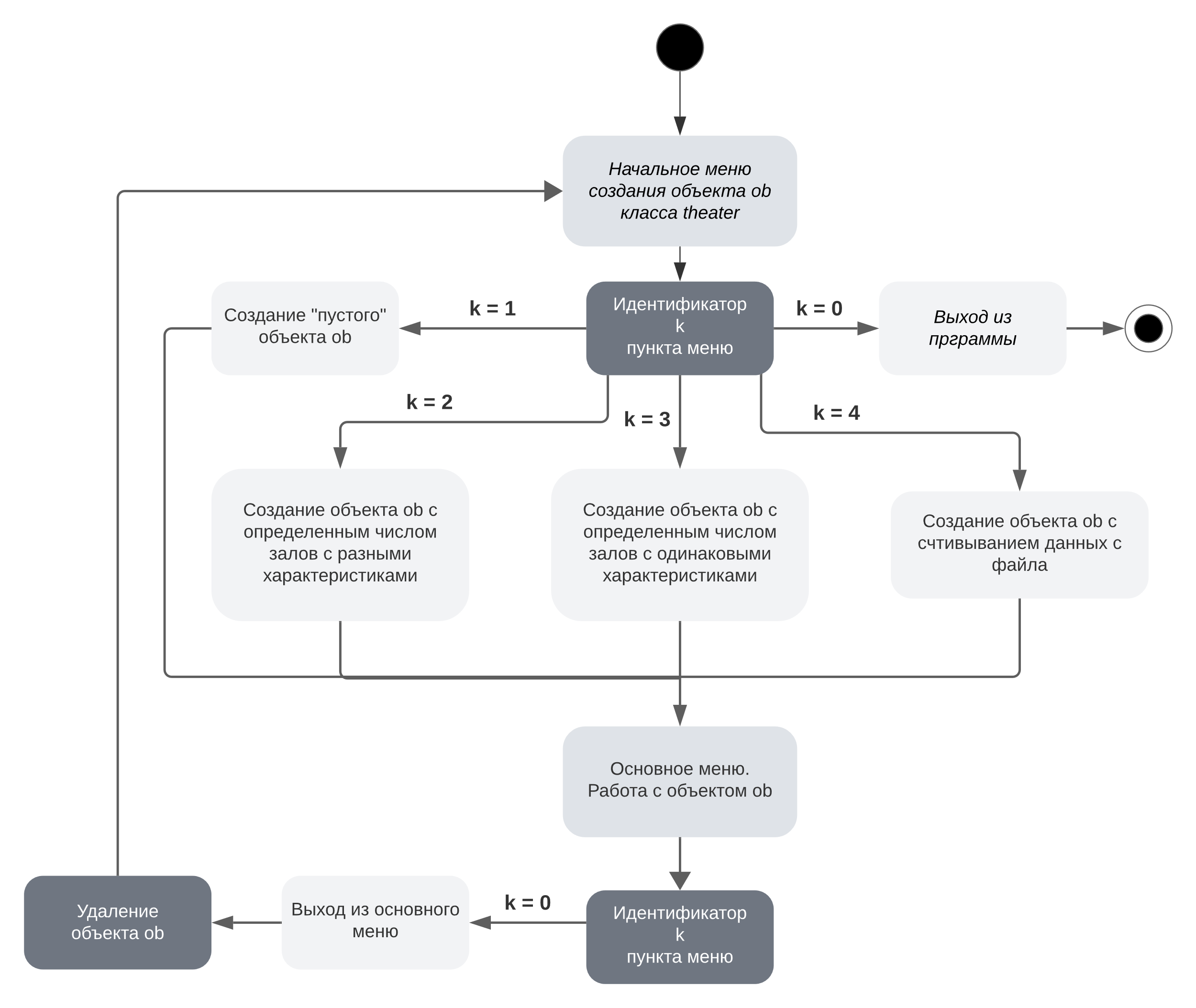
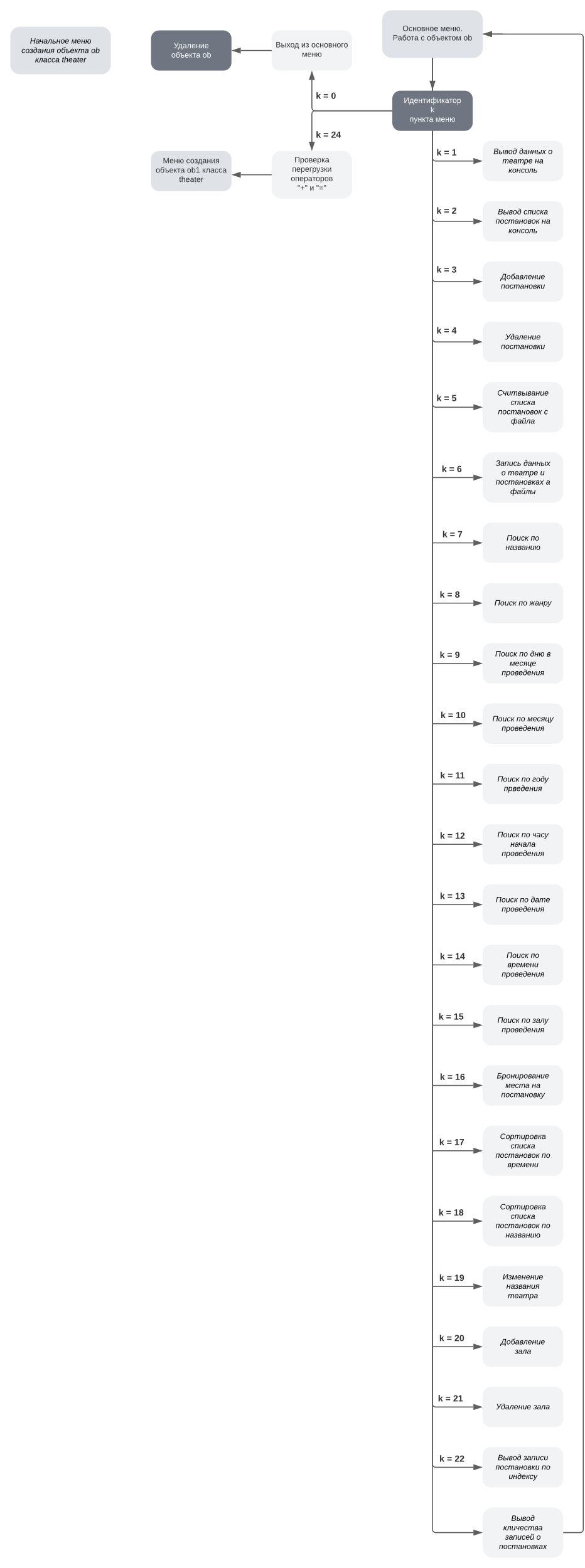




Рисунок . — Схема начального пользовательского меню



k = 23

Рисунок . — Схема основного пользовательского меню



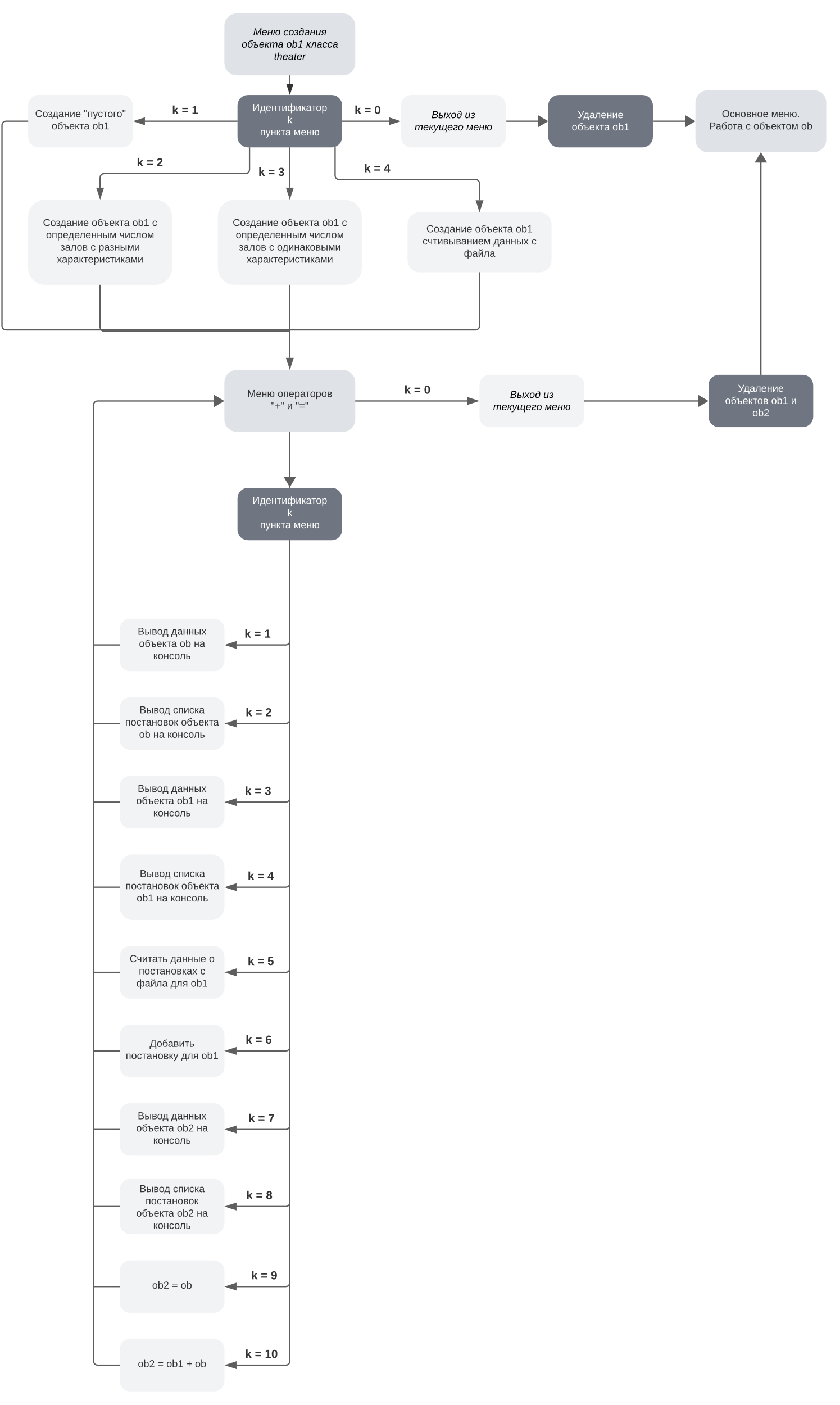


Рисунок . — Схема меню проверки перегрузки операторов

## Алгоритмы и методы в программе

После проектирование приложения театра, можно переходить непосредственно к реализации его функционала. Дадим определение структурным типам, являющихся составными типами. К ним относятся уже оговорённые ранее структуры зала (hall), даты (Date), времени (Time) и записи постановки (play).

Для начала введём структурный тип данных для залов. Его именем будет hall. В качестве полей пользовательского типа hall были выбраны были выбраны целочисленные встроенные типы данных, отождествлённые соответственно с количеством рядов и мест в одном ряде. Помимо этого здесь представлены и функции-члены структуры: конструктор с параметрами, задающими поля структуры, перегрузка оператора присваивания и конструктор по умолчанию, нужный как в этом, так и в остальных определяемых в программе пользовательских типах для создания объекта или переменной этого типа без прямой инициализации для случаев, когда неизвестно, какое значение должен иметь этот объект. В случае залов значениями по умолчанию являются ноль рядов и ноль мест в ряде зала. Отдельно выделим оператор присваивания, который в дальнейшем может пригодится для изменения или переноса значения. Он принимает ссылку на объект правого операнда, чтобы не вызывать лишний раз конструктор копирования и изменяет правый операнд, делая их равными полям левого операнда.

Теперь рассмотрим пользовательские структуры даты и времени. Структура даты Date состоит из трёх переменных полей типа string, которое будут выражать натуральное число, не превышающее разумных пределов. Решение хранить дату в виде строк было принято для того, так как осуществлять писок по неполным данным ведённой даты проще работая с функциями класса string, а также для того, чтобы при введении в поля даты некорректных символов, не являющихся цифрами, программа не выходила из строя, и этот ввод можно было бы идентифицировать с помощью функций работы со строками. Помимо этого в определении Date имеются следующие функции-члены: конструктор по умолчанию, конструктор с параметрами, принимающем строки, эквивалентные соответственно порядковому номеру дня в месяце, номеру месяца в году и году определяемой даты, перегрузка оператора «=», которая здесь нужна по той же причине, что и для типа hall, и перегрузка оператора сравнения дат, которая проверяет, совпадают ли поля экземпляров структуры. Перегрузка оператора «==» нужна для сокращения записи сравнения переменных типа для постановок.

Структура времени нужна для определения времени проведения постановки. Она состоит из строковых переменных, представляющих собой час и минуты начала сценического представления. Такое определение полей структуры Time обусловлено тем же, что и в случае с типом Date. Аналогично структуре Date здесь имеются следующие функции-члены: конструктор по умолчанию, конструктор с параметрами, принимающими значения часа и минут, перегрузка оператора присваивания и перегрузка оператора сравнения.

Теперь же, когда было дано определение элементарных пользовательских структур данных, перейдём к реализации более сложной структуры, описывающей запись планируемой в театре постановки. Она уже будет содержать в себе определённые ранее пользовательские типы. Описание записи постановке имеет следующий вид (см. рисунок 2.4):

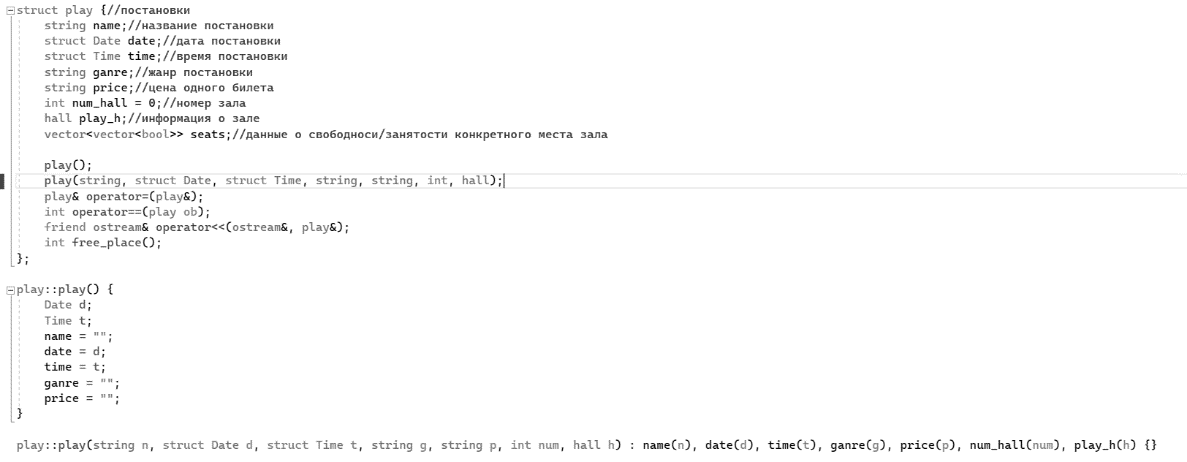


Рисунок . — Определение структуры play в коде программы

Из рисунка видно, что запись о постановке struct play содержит поле имени строкового типа (name), поля даты (date) типа Date и времени (time) типа Time проведения постановки, поле жанра строкового типа (ganre), поле цены (price) так же строкового типа, что обусловлено необходимостью предотвращения поломки программы при введении нечисловых данных в эту область памяти, поле номера зала целочисленного типа (num\_hall), обозначающего порядковый номер зала театра, в котором будет проводиться постановка, а также определение этого зала, содержащейся в переменной play\_h типа hall, которая определяется автоматически по номеру зала в театре и его характеристикам, и поле вектора с именем seats, содержащего векторы, хранящие логические переменные типа bool, необходимый для хранения данных о забронированных местах на указанную постановку, где true обозначит свободное место, false — забронированное.

Отдельное внимание обратим на функции-члены, содержащиеся в структуре записи о постановке. В первую очередь стоит описать конструкторы. Конструктор по умолчанию не создаёт полноценную переменную, а лишь определяет пустую запись. Конструктор с параметрами принимает 7 аргументов в следующем порядке: название постановки, дата её проведения, время начала её проведения, её жанр, цену билета, номер зала и параметры зала, хранящиеся в передаваемом аргументе типа hall. Все эти значения он напрямую инициализирует каждому полю структуры.

Так же как и для предыдущих пользовательских типов данных, в представлении рассматриваемой структуры имеются и функции-члены, перегружающие операторы присваивания «=» и сравнения «==».

Если в случае с оператором присваивания всё понятно: для операнда, стоящего слева, поразрядно, в рамках каждого поля, присваиваются значения операнда, расположенного справа, то в случае с оператором сравнения стоит пояснить, что эквивалентными (равными) записями о постановках считаются такие переменные, у которых одновременно совпадает и наименование, и время проведения (дата и время), и место проведения (номер зала), и жанр постановки. Конечно же это равенство имеет смысл только в области действия одного театра.

Далее определим немаловажный оператор вывода записи о постановке в консоль. Как можно было заметить, функция перегрузки оператора вывода является дружественной для структуры play. Перегрузка вывода первым аргументом принимает ссылку на поток типа ostream, определённым в классе ios, вторым аргументом она принимает ссылку на выводимую структурную переменную данного типа. Она также возвращает ссылку на поток, чтобы было возможно записывать вывод переменных типа play в ряд. Первым делом в консоль выводиться наименование спектакля, затем ставиться разделитель, чтобы показать, что все дальнейшие параметры относятся к постановке с выведенным именем. Отдельное внимание стоит обратить на назначение функций fill и width, которые здесь выводят дополнительный ноль в случае, если значения дня, месяца даты или значения часа и минуты содержат лишь один символ. Для читаемости и красоты вывода все характеризующие постановку параметры выводятся через табуляцию. Особое внимание стоит обратить на вывод количества свободных мест, необходимый, чтобы пользователь/клиент уже на старте мог сделать оценку того, сможет ли он занять нужное ему место в зале или нет. Это делается возможным благодаря функции free\_place, которая перебирает все места в зале и считает только те позиции вектора, которые равны true (см. рисунок 2.5).

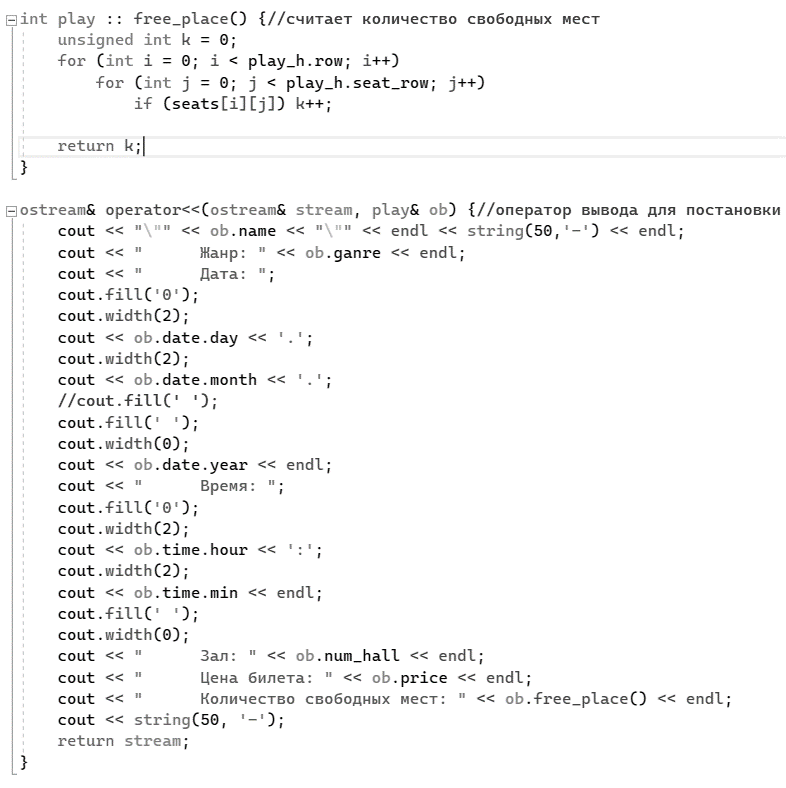


Рисунок . — Перегрузка оператора вывода для play, free\_place()

После того как были определены все составляющие класса theater можно перейти к рассмотрению структуры устройства самого театра (см. рисунок 2.6).

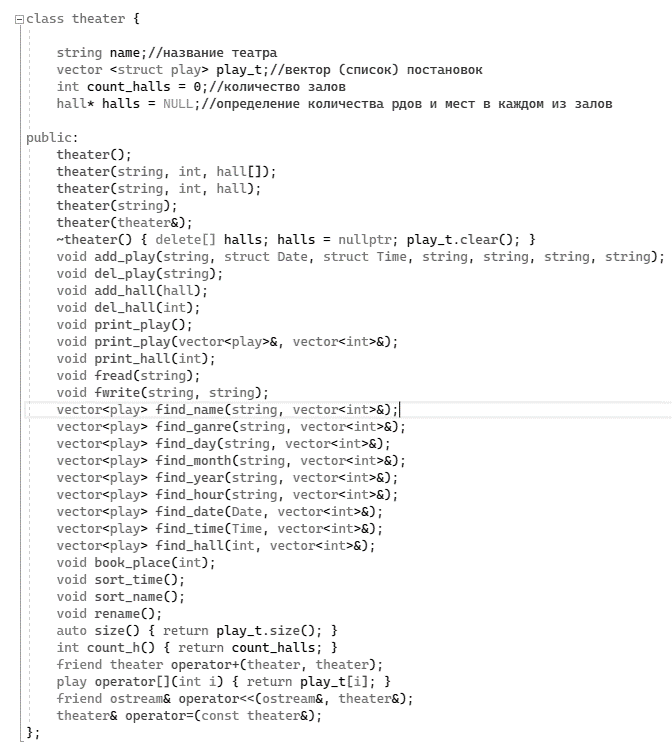


Рисунок . — Описание закрытой части класса theater

В закрытой части класса theater, хранятся следующие поля: наименование театра, список планируемых постановок, хранящихся в векторе vector<play> play\_t, количество залов, составляющих театр и определение каждого из этих залов, хранящееся в переменных типа hall, расположенных в свою очередь в массиве, ссылку на первый элемент которого хранит указатель halls. Заметим, что порядковый номер каждого зала начинается с единицы и соответствует номеру его расположения в массиве залов театра. Тот факт, что эти поля являются закрытыми от произвольного доступа к данным, хранящимся в ИС обусловлен ограничением возможности хаотического управления данными объекта класса, без их контроля его функциями.

Прежде чем начинать описывать функционал класса для работы с данными, необходимо эти данные определить и присвоить полям для хранения в объекте. С этой задачей призваны справиться конструкторы класса. Для разрабатываемого класса theater реализовано пять различных конструкторов. Сущность их назначения в программе хоть и одинакова, однако количество аргументов и способы инициализации полей различны.

В первую очередь стоит определить конструктор по умолчанию, который создаёт театр с ничего не содержащим набором данных. Такой конструктор нужен для определения объектов класса theater в тех случаях, когда заранее неизвестно, какие данные следует хранить в объекте такого класса, хотя его уже нужно определять в программе.

Теперь же определим один из конструкторов с параметрами (см. рисунок 2.7), который принимает в качестве аргументов наименование театра, количество залов и указатель на первый элемент массива залов.

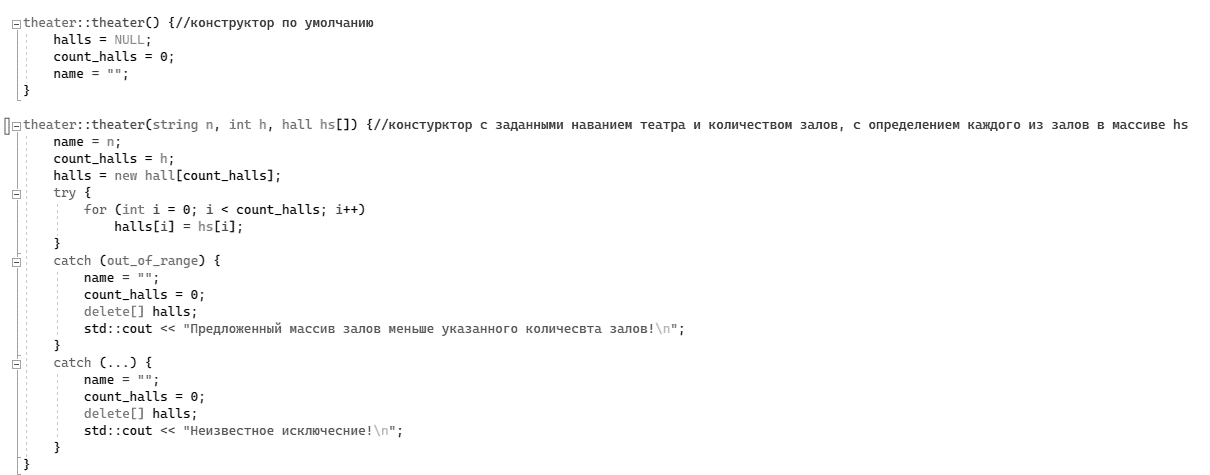


Рисунок . — Описание конструктора с определением залов

Обработчик исключительных ситуаций в функции необходим в случае, если размер передаваемого массива меньше заявленного количество залов. Тогда динамическая память, выделяемая под массив залов, удаляется и инициализация объекта не происходит. В благоприятных случаях происходит примитивное присваивание. Логично будет следующим шагом определить конструктор класса theater, инициализирующий объект театра с определённым количеством залов, равным count, с одинаковыми характеристиками тогда риск выхода за пределы массива почти сводится к нулю.

Наличие конструктора копирования обусловлено тем, что при инициализации объекта класса выделяется динамическая память, на начало которой указывает halls, поэтому необходимо напрямую прописывать инициализацию копии, чтобы указатель копии хранил адрес уже на другую область памяти для хранения данных о залах, относящуюся только к этой копии. В противном случае будет происходить примитивное поразрядное присваивание полям копии полей данных оригинала, и указатель halls копии будет указывать на ту же область памяти, что и этот же указатель оригинала, а это в свою очередь может привести к непредсказуемой работе программы.

Перед тем, как описывать реализацию последнего конструктора класса, необходимо прописать следующую очень важную функцию, которая будет использована в коде программы ещё не раз (см. рисунок 2.8)

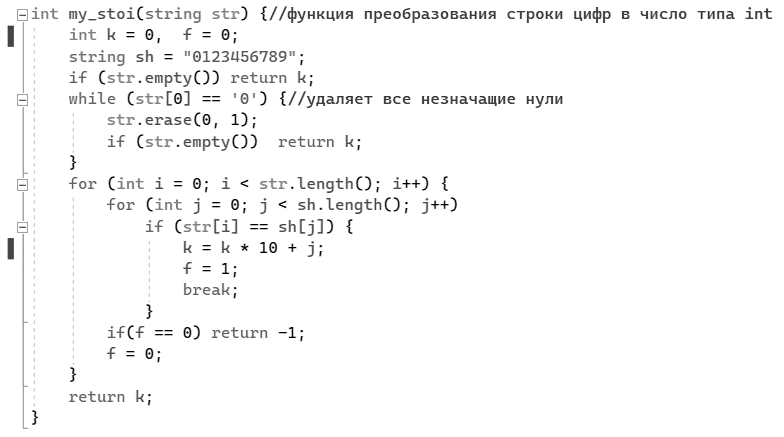


Рисунок . — Описание функции my\_stoi()

Назначение функции my\_stoi() заключается в преобразовании строки в целочисленное значение. Для обеспечения такого функционала внутри функции прописана строка sh, в которой содержатся символы от 0 до 9, то есть индекс каждого из этих символов в строке соответствует значению этого символа. Начальное значение k равно 0, то есть пустая стока для данной функции равна 0. Для ненулевой строки функция подключает цикл, в котором для каждого символа строки str проверяет, есть и она в строке sh. Если рассматриваемый символ является цифрой, то разряд числа k увеличивается на 10 и к этому значению прибавляется значение индекса возвращённого значения. Если же рассматриваемого символа в строке нет, что определяется по флажку, то полученная строка не может быть преобразована в целое неотрицательное число, поэтому программа возвращает значение -1.

Теперь определим конструктор, считывающий данные о театре с файла (см. рисунок 2.9). Функция в качестве аргумента получает название файла, типа string, обозначающий имя файла, вызывает конструктор класса ifstream, открывающий поток ввода с файла, затем проверяет, может ли быть открыт этот файл, с помощью функции is\_open(), в случае ошибки открытия файла произойдёт выброс исключения, и программа завершит выполнение, иначе продолжается считывание с файла в соответствии со строго определённой формой хранения данных: в первой ячейке хранится наименование театра, во второй – количество залов, в третьей - определения каждого из залов: сначала идёт количество рядов, затем через пробел количество мест в ряде. Если в файле указанное количество залов и количество определений залов не совпадают или в файле нет необходимых разделителей, то выбрасывается исключение. В лучшем случае функция приступает к считыванию информации о рядах и залах. Функция dig призвана определить, не содержаться ли в значении количеств недопустимых символов, если программа их обнаружит, то попросит пользователя вручную ввести эти данные в консоль.

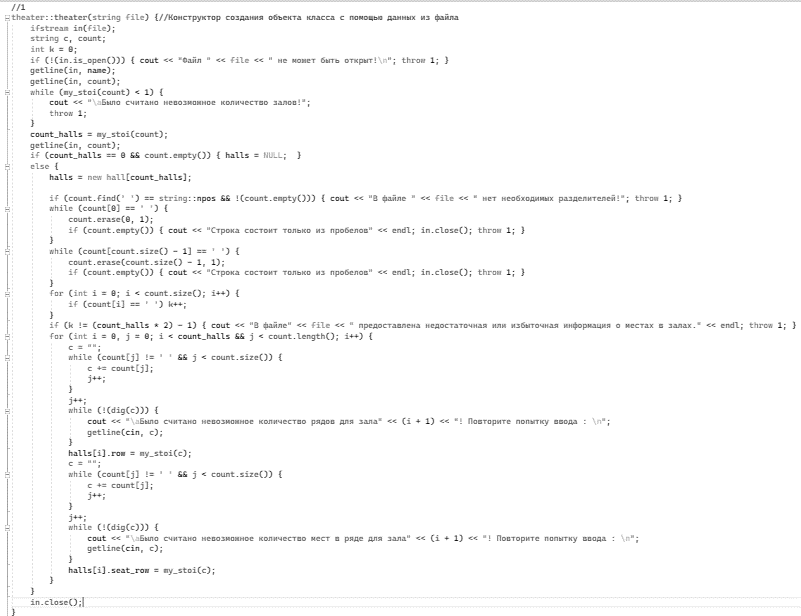


Рисунок . — Описание конструктора, считывающего данные с файла

Функция dig основана на той же концепции, что и функция my\_stoi(), только её задача более ограничена, так как она должна просто идентифицировать, содержатся ли в строке только цифры (true) или имеются иные символы (false).

Теперь перейдём к рассмотрению функций сортировок списка постановок. Функция сортировки по времени sort\_time, устанавливается по умолчанию. Идея сортировки заключается в том, чтобы найти запись постановки, которая планируется позднее всех, и переместить её на последнюю позицию, затем уменьшить диапазон поиска на один элемент массива и искать более позднее представление в рамках выделенного подмножества. В тех случаях, когда постановки планируются в одно и то же время, постановки сортируются по номерам залов следующим образом, постановка с большим номером зала ставится ниже постановки с меньшим номером. В классе также имеется функция сортировки по полю названия постановки sort\_name, сортирующая тем же способом, что и sort\_time(), но постановки будут расположены уже в лексикографическом порядке их наименований, то есть по алфавиту.

Отдельно следует рассмотреть функцию добавления постановки (рисунок 2.10). Основной задачей этой функции является проверка переданных ей данных о постановке на корректность, и требование от пользователя изменить их на допустимые. Аргументам этой функции соответствуют ранее указанные атрибуты записи о постановке. Будем рассматривать каждый аргумент по мере его появления в листинге кода функции. Первым делом программа проверяет на корректность предложенную дату (Date) проведения. Для этого она должна знать текущую дату, которая берётся из функции cur\_date, которая, в свою очередь, обращается к ОС. Далее в функции проверяется, чтобы дата не имела в своём значении недопустимых символов и не являлась прошлой. Здесь и во многих других местах кода предусмотрена возможность введения месяца с помощью слов. Для того, чтобы понять, состоит ли строка только из буквенных символов используется функция only\_let, (см. рисунок 2.11). Слова, обозначающие месяц могут таже вводиться в любом регистре, так как их идентификация происходит с помощью сравнения строк, прописные буквы которых преобразованы в строчные. Это является возможным благодаря функции lowercase (см. рисунок 2.11) которая, принимая строку, возвращает ту же самую строку, но в пониженном регистре, если в ней имелись прописные буквы.

В функции only\_let() для идентификации того, что рассматриваемый символ является буквой русского либо латинского алфавита, используется строка А, в которой перечислены все эти буквы. Если при рассмотрении какого-то конкретного символа, он не встречается в этой строке, то флажок f не увеличивается, а значит конечное значение флажка не совпадёт с длиной строки. Если же все символы являются буквами, то значение f равно количеству символов. Функция lowercase, если встречает в строке прописную букву, заменяет её строчной буквой из строки a с тем же индексом.

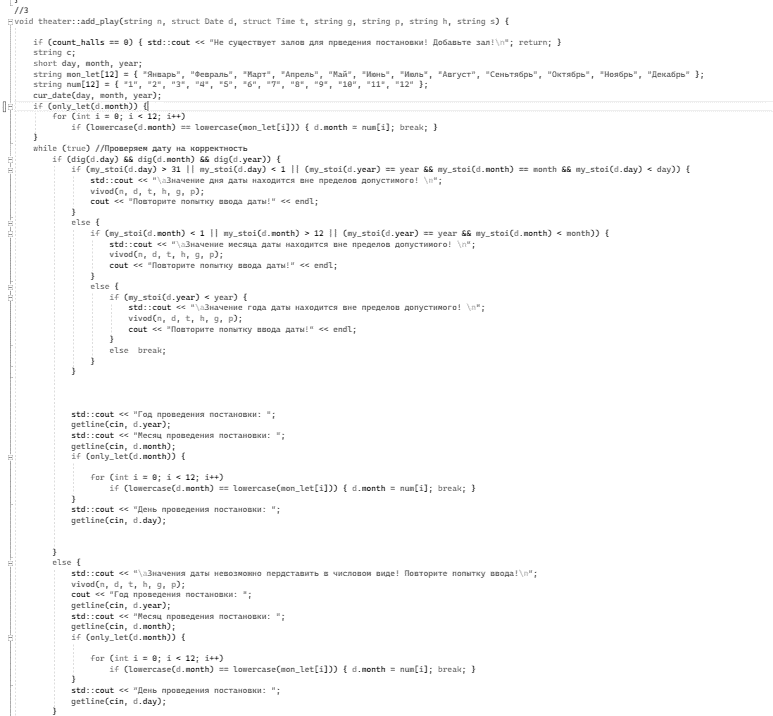


Рисунок . — Начало функции add\_play()

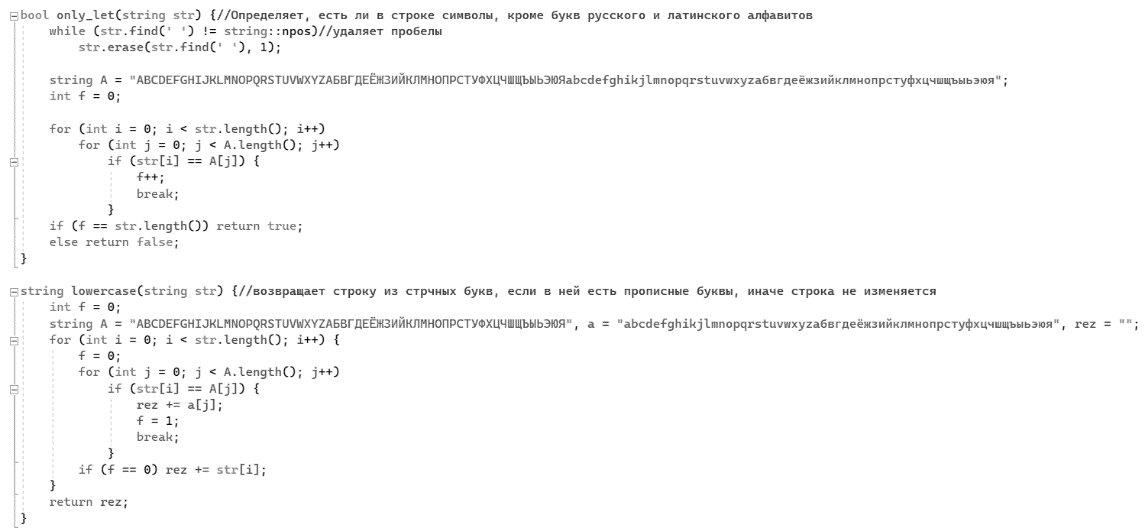


Рисунок 2.11 — Функции lowercase и only\_let

Далее функция add\_play проверяет на допустимость значения часа и минут. Проверка жанра, цены и номера зала осуществляется аналогичным образом. Только в случае проверки жанра применяется функция only\_let\_cap которая является дополнением функции only\_let, но с дополнительной проверкой на начало строки с прописной буквы, поскольку жанр должен содержать прописную букву только в начале. А для проверки цены добавляется функция dig\_price, которая допускает наличие одной точки в строке цены. Функция add\_play не допускает добавления постановки, которая будет происходить одновременно с уже имеющейся в списке в одном и том же зале (см. рисунок 2.12).

Здесь также происходит проверка забронированных мест и создания вектора, хранящего информацию о свободных и забронированных местах (см. рисунок 2.12). Это необходимо в первую очередь для определения забронированных мест на постановки, считанных из файла. У каждого забронированного места есть своя форма записи в файле: {номер ряда}.{номер места}, эти обозначения разделены между собой пробелами. Именно поэтому здесь вызываются исключения, так как в случаях, когда считываются некорректные забронированные места, считывание с файла должно прекратиться. В противном случае все последующие данные будут считываться некорректно. После прохождения всех представленных проверок создаётся переменная типа play, она добавляется в вектор списка постановок и этот вектор сортируется по времени с помощью sort\_time.

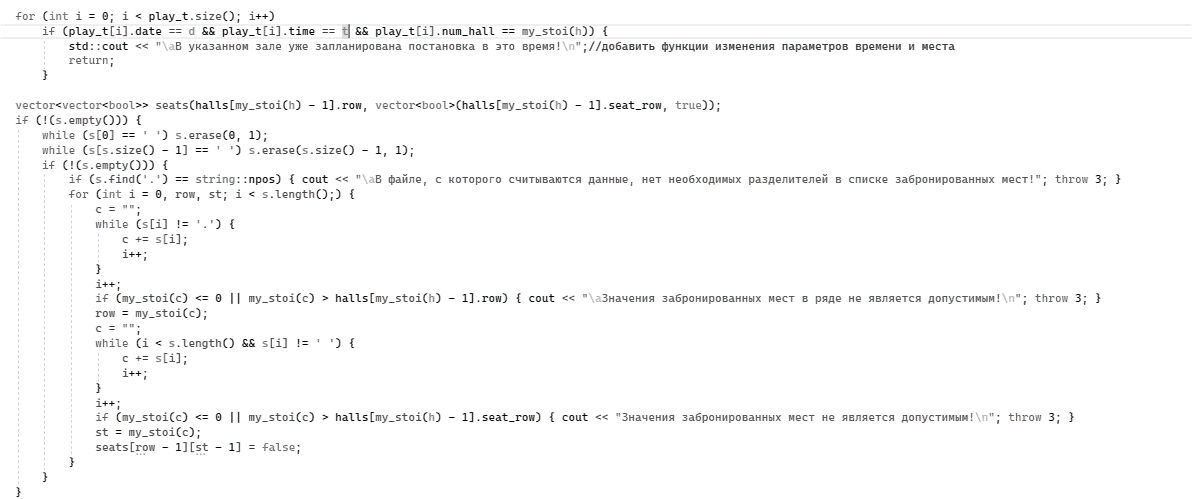


Рисунок . — Конец функции add\_play()

Функция del\_play удаляет запись о постановке по её порядковому номеру в списке. В случае, если введённый индекс постановки выходит за пределы массива или массив списка постановок пуст, функция возвращает ошибку, в лучшем случае она уточняет у пользователя, действительно ли он хочет удалить запись, и при положительном ответе удаляет запись.

Функция add\_hall добавляет зал в театр. Она выделяет память под новый динамический массив. В его конец добавляет переданную в качестве аргумента переменную типа hall. Удаляет старый динамический массив и присваивает halls адрес на первый элемент нового массива залов. Функция del\_hall позволяет удалить зал из театра. Но перед этим она проверяет возможность этого действия, так как если на какую-то постановку забронированы места, то удалить нельзя. При благоприятном результате функция перезаписывает в новый массив все залы старого массива, пропуская удаляемый элемент, удаляет старый массив и присваивает halls указатель на первый элемент нового массива.

Перегрузка оператора вывода информации о театре, как и в случае с play, является функцией, дружественной к классу theater. Она окаймляет вывод информации о театре сверху и снизу, выводит все атрибуты через табуляцию, среди который данные о залах и их характеристиках: количество рядов и мест в ряде.

Функция считывания списка постановок с файла fread (см. рисунок 2.13) принимает на вход наименование файла и структурно действует так же, как и конструктор считывания данных о театре с файла. Она также проверяет форму хранения данных: наличие у записи даты точек, наличие у записи времени двоеточия. В случае, когда строка с забронированными местами имеет недопустимые значения, она закрывает поток ввода из файла и выбрасывает исключение. Она использует функцию add\_play для добавления постановки.

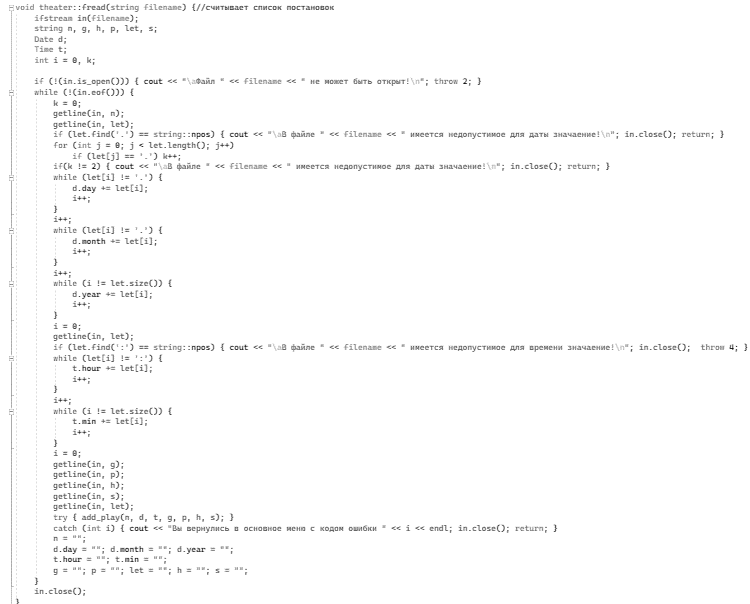


Рисунок . Описание функции считывания списка постановок с файла

Функция fwrite (см. рисунок 2.14) принимает два аргумента, обозначающих названия файлов: в первый файл записываются данные о театре, во второй - список постановок. Запись в файл происходит в соответствии с необходимой формой записи так, чтобы с получившихся файлов программа в дальнейшем смогла считать данные.

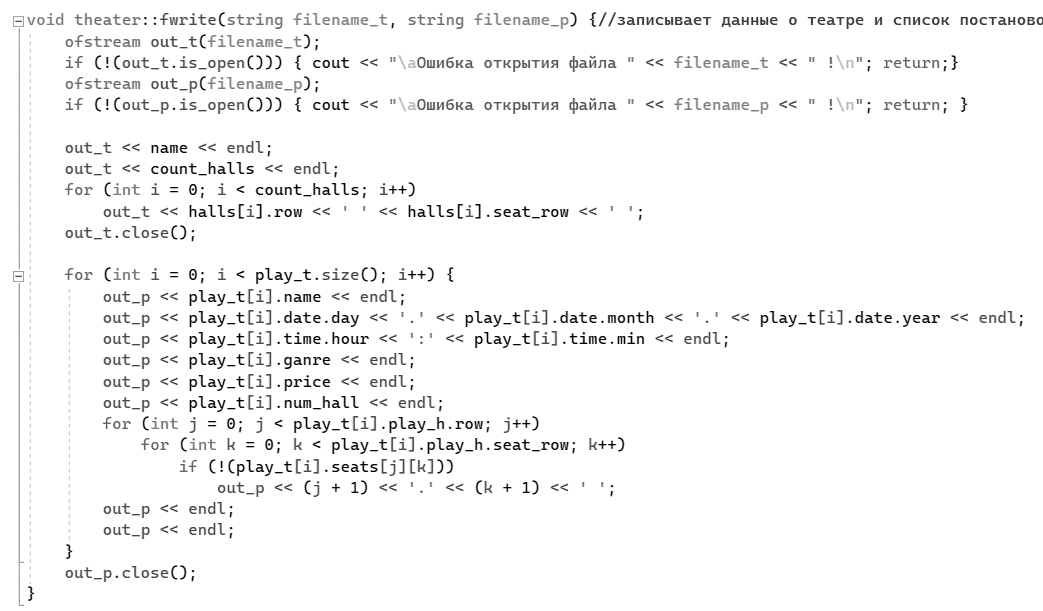


Рисунок . Описание функции записи данных в файлы

Функция print\_play имеет две реализации: одну без аргументов, тогда она будет выводить весь список постановок театра, а другую с аргументами, среди которых некоторое подмножество постановок и вектор с порядковыми номерами этих постановок в общем списке. Вторая форма нужна при поиске записи о постановке и получении его порядкового номера.

В классе реализован целый ряд поисков по разным полям: по наименованию постановки, по её жанру, по дню в месяце, месяцу, году её проведения, залу, дате и времени проведения в целом. Реализация всех этих функций идентична (см. рисунок 2.15) и различается лишь в поле поиска (для даты и времени комбинируются поиски по дню, месяцу и году для даты, и по часу и минутам для времени). Их основная идея заключается в использовании в сравнении строк функций lowercase для регистронезависимого поиска и функции find библиотеки <string> для поиска по неполным данным. Они возвращают подмножество удовлетворяющих условию поиска постановок для его передачи функции print\_play.

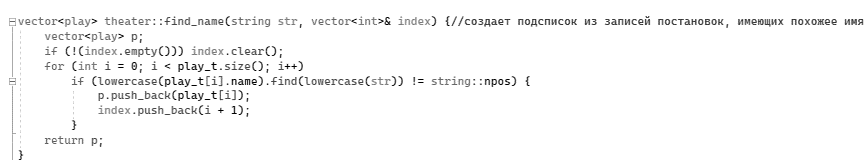


Рисунок . — Описание функции поиска по имени

Отдельно следует рассмотреть функции, реализующие систему бронирования места на постановку по порядковому номеру постановки в списке. Каждая из них реализует своё мини-меню. Например, функция book\_place позволяет пользователю как узнать о свободных местах, переводя его в функцию print\_hall, так и непосредственно забронировать место, или выйти из метода. Если введённое место существует или свободно, то при бронировании оно приобретёт значение false в векторе seats. Функция print\_hall тоже позволяет пользователю выбирать, каким способом он хочет узнать о свободных местах: графическим, то есть выводом в консоль с помощью условных обозначений, или тестовым, то есть в виде номеров свободных мест выбранного ряда.

Оператор присваивания будет стирать все данные операнда, стоящего слева, и присваивать освободившемся полям значения операнда, стоящего справа. Оператор же сложения двух театров будет работать так, словно создаётся новый театр, который является объединением двух складываемых театров. Функция перегрузки оператора «+» будет предлагать ввести наименование театра, затем она выделит память под массив залов, складываемый из залов правого и левого операнда, после чего номера залов левого операнда увеличатся на количество залов правого операнда, как в непосредственном определении, так и в определении залов. Затем в список постановок образующегося объекта перейдут все записи объединяемых театров, которой после будет отсортирован по времени с помощью функции sort\_time.

## Демонстрация работы программы

После того, как весь планируемый к разработке функционал класса был реализован, пришло время протестировать работу ключевых для этого проекта функций с некоторым набором входных данных. Проверим работу конструктора, считывающего данные с файла Data\_t.txt (см. рисунок 2.16).

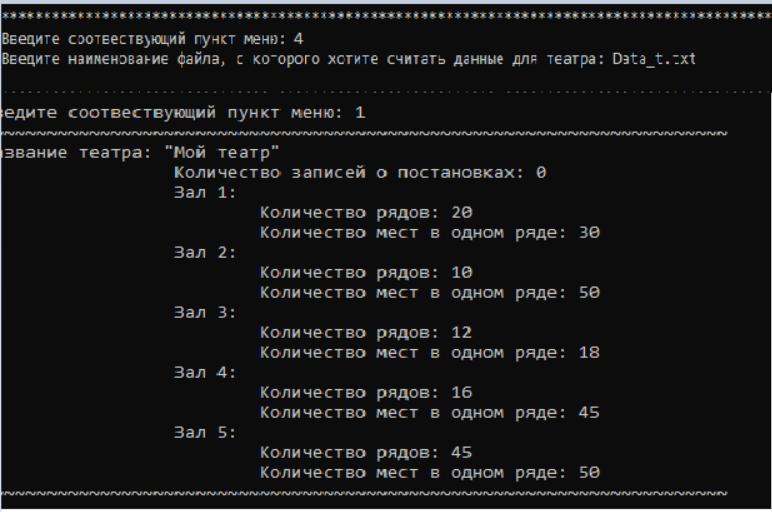


Рисунок . — Считывание данных о театре с файла Data\_t.txt

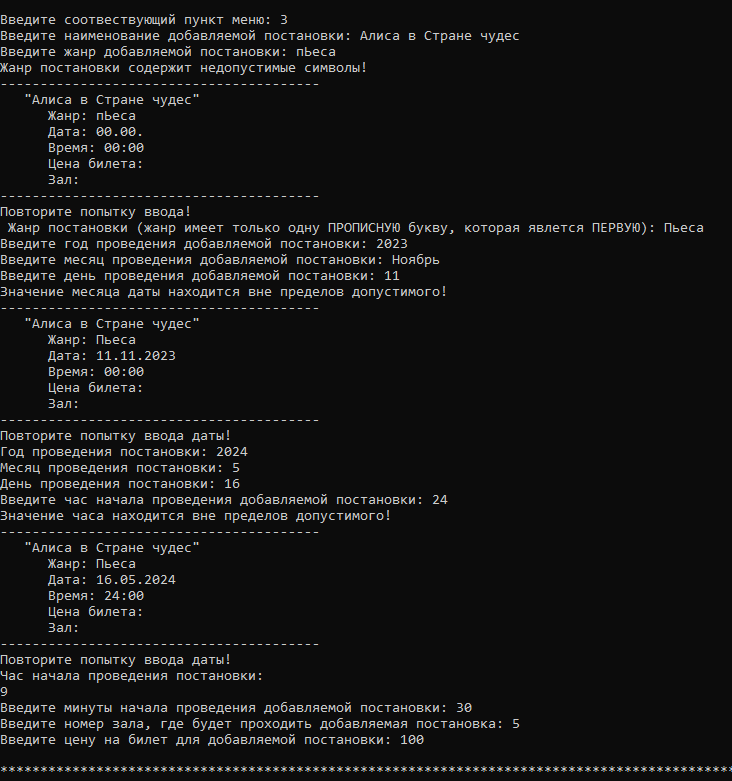


Рисунок . — Добавление записи постановки в массив

Далее проверим работу функции добавления записи постановки в массив постановок, а также её реакцию на некорректно введённые данные (см. рисунок 2.17).

Следующим шагом попробуем считать записи о постановках из вложенного файла Data\_p.txt с помощью функции fread и проверим корректность их ввода через вывод на консоль посредством использования функции print\_play (см. рисунок 2.18).

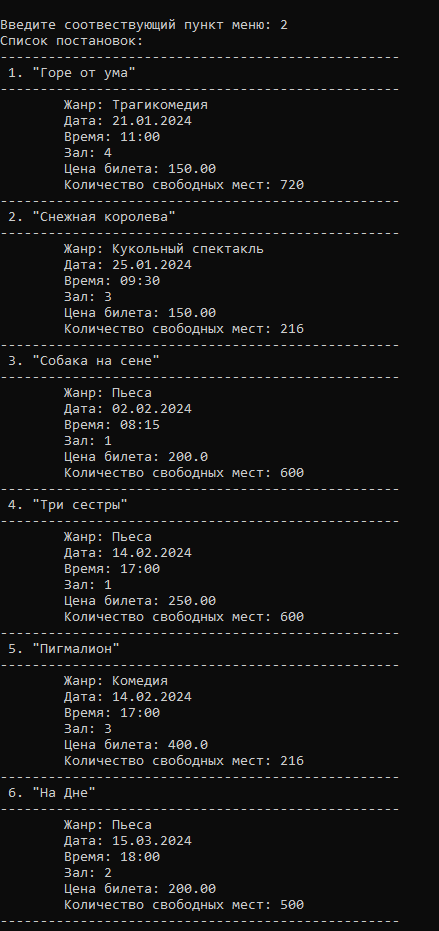


Рисунок . — Вывод списка постановок, считанного с файла, в консоль. Теперь протестируем одну из функций поиска по наименованию постановки. В качестве искомой информации введём в консоль значение «оР», поскольку функция должна искать по неполным данным и независимо от регистара в результате получим вывод в консоль, представленный на изображении ниже (см. рисунок 2.19). Аналогично будут работать функции поиска и по другим полям.

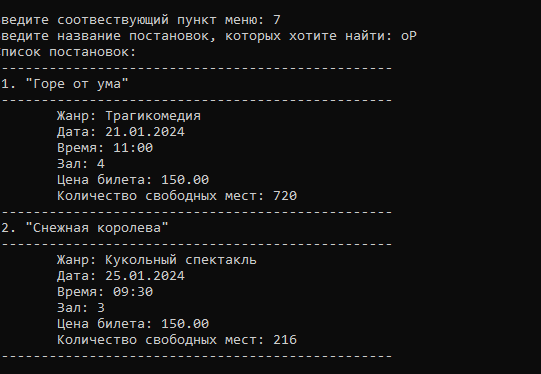


Рисунок . — Вывод записей постановок, найденных по наименованию

Протетсируем методы поиска свободных мест на выбранную по порядковому номеру постановку и их бронирования, путём ввода данных о бранируемом ряде и месте в консоль. Как видно из изображения (см. рисунок 2.20), вывести информацию о свободных местах можно как графическим способом, так и текстовым способом.

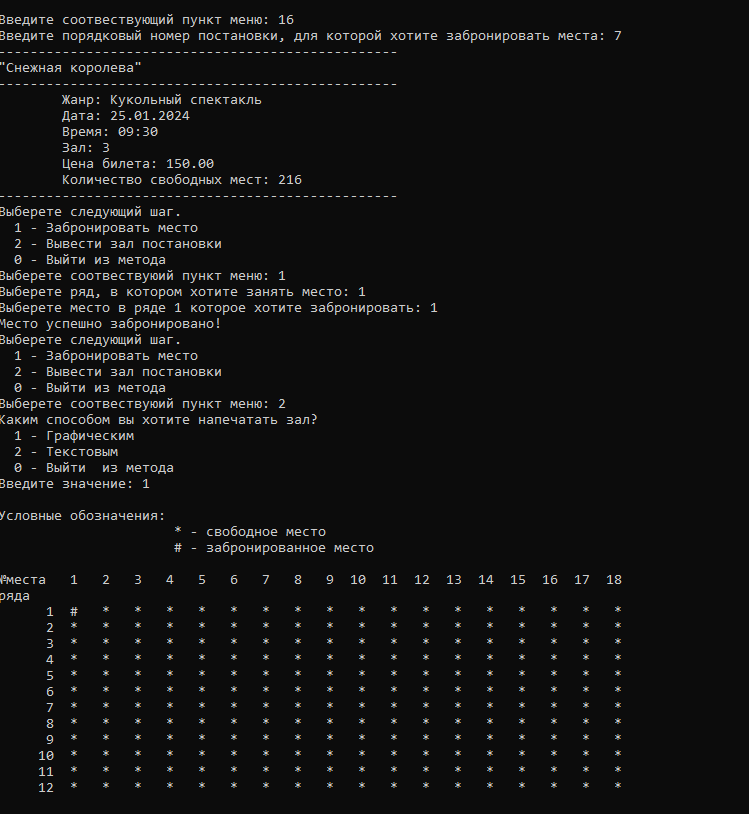


Рисунок . — Процесс бронирования места в зале

Завершающим шагом будет запись результатов, получившихся в процессе работы над объектом класса theater, в файл, который будет их хранить в табличном виде (см. рисунок 2.21).

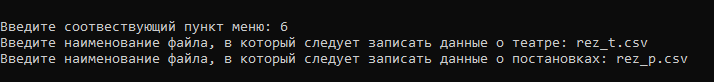


Рисунок . — Запись данных о театре и постановках в файлы

# ЗАКЛЮЧЕНИЕ

С использованием языка C++, был создан спроектирован и реализован класс "Театр", который предоставляет удобные интерфейсные методы для работы с информацией о постановках, залах, билетах и других аспектах театральной деятельности.

Особое внимание уделялось построению систем связи между структурами данных. Для этого проводились анализ предметной области и разработка ER-диаграммы. Проектирование ER-диаграммы позволило создать четкую картину структуры данных, определить ключевые атрибуты каждой сущности и логические связи между ними. Это, в свою очередь, способствовало созданию хорошо структурированного класса "Театр».

Разработка программного продукта предоставила возможность не только применить теоретические знания о языке программирования и ООП на практике , но научиться построению модульных и гибких программ, эффективно управляющих данными. Консольное приложение, реализованное в ходе работы, взаимодействует с пользователем, предоставляя возможность ввода и вывода информации о представлениях, расписании, билетах и других важных аспектах театральной деятельности. Работа была оформлена в соответствии с требованиями следующих стандартов: [1-5].

Таким образом, курсовая работа позволила не только решить конкретную проблему в области внедрения информационных технологий в сценический промысел, но и значительно расширить практические навыки в программировании, архитектурном проектировании и построении сложных систем. Полученные знания о структурах данных, ER-диаграммах и ООП предоставляют прочную основу для дальнейшего развития в области программной инженерии.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ 7.0.100-2018 «Библиографическая запись. Библиографическое описание. Общие требования и правила составления». : дата введения 2019-07-01. – Москва : Стандартинформ, 2018. – 70 с.
2. ГОСТ Р 2.105-2019 «Единая система конструкторской документации. Общие требования к текстовым документам» : дата введения 2023-03-01. – Москва : Стандартинформ, 2019. – 44 с.
3. ГОСТ 7.32-2017 «Отчет о научно-исследовательской работе. Структура и правила оформления» : дата введения 2017-10-24. – Москва : Стандартинформ, 2017. – 32 с.
4. ГОСТ 7.80-2000 "Библиографическая запись" : дата введения 2000-06-22. – Минск : Межгосударственный Совет по стандартизации, метрологии и сертификации Минск, 2000. – 11 с.
5. ГОСТ Р 7.0.5-2008 "Общие требования и правила составления" : дата введения 2008-04-28. – Москва : Стандартинформ, 2008. – 22 с.
6. Лафоре, Р. Объектно-ориентированное программирование в С++ / Р. Лафоре. – Санкт-Петербург : Питер, 2022. – 928 с.;
7. Стенли, Б.Л. C++ Базовый курс / Б.Л. Стенли. – Пятое издание. – Санкт-Петербугрг : ООО "Диалектика", 2019. – 1120 с. – ISBN 978-5-6041394-9-3;
8. c-news: сайт. – URL: <https://www.cnews.ru/news/top/2023-06-07_rossijskie_teatry_obedinyayutsya> (дата обращения: 08.12.2023);
9. Центр информационных технологий : сайт. – URL: <https://itvolga.com/269-crm-theatre> (дата обращения: 08.12.2023);
10. Microsoft: сайт. – URL: <https://learn.microsoft.com/ru-ru/visualstudio/get-started/visual-studio-ide?view=vs-2022> (дата обращения: 08.12.2023);
11. Строки в языке C++ (класс string) : сайт. – URL: <https://ejudge.179.ru/tasks/cpp/total/161.html> (дата обращения: 06.12.2023);
12. cppreference.com: сайт – URL: <https://en.cppreference.com/w/cpp/container/vector> (дата обращения: 10.12.2023);
13. METANIT.COM: сайт. – URL: <https://metanit.com/cpp/tutorial/8.2.php> (дата обращения: 06.12.2023);
14. Всё о Windows простым языком: сайт. – URL: <https://mambakabinet.ru/include-windows-h-chto-eto-takoe/>;
15. Записки информационщика : сайт. – URL: <https://nicknixer.ru/programmirovanie/russkie-simvolybukvy-pri-vvodevyvode-v-konsol-na-c/?ysclid=lq0ybpqqew973020192> (дата обращения: 06.12.2023);
16. Блог Я Практикума: сайт. – URL: <https://practicum.yandex.ru/blog/chto-takoe-er-diagramma> (дата обращения: 06.12.2023);

# ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ

A

add\_hall, 28

add\_play, 27

B

book\_place, 30

C

cin, 6

cout, 6

CRM-систему, 3

cur\_date, 24

D

Date, 15, 16, 17, 24

del\_hall, 28

del\_play, 28

dig, 23

E

ER-диаграмма, 11

F

fill, 6, 18

fread, 29, 32

free\_place, 19

fstream, 7

fwrite, 29

G

getline, 6

H

hall, 15, 16, 17, 19, 28, 30

I

IDE, 5

ifstream, 7, 22

iostream, 6

L

lowercase, 25

M

my\_stoi, 22, 24

O

ofstream, 7

only\_let, 25

P

play, 15, 17, 19, 26, 27, 28, 29, 30, 32

print\_hall, 30

print\_play, 30

S

sort\_name, 24

sort\_time, 24, 27, 31

string, 6, 22

T

Time, 15, 16, 17

V

vector, 7, 19, 37

W

width, 6, 18

Windows API, 7

Windows-1251, 8

А

Атрибуты, 10

И

Инфологическая модель, 9

ИС, 8, 9, 20

М

многие-к-одному, 10

О

один-ко-многим, 10

ООП, 3, 5, 35

П

Перегрузка вывода, 18

ПО, 3, 4

Постановка, 10

С

Структура, 15, 16, 36

сущность, 10

Т

Театр, 1, 3, 4, 8, 9, 10, 11, 35