Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет информационных технологий и управления

Кафедра экономической информатики

Дисциплина «Технологии разработки программного обеспечения»

|  |  |
| --- | --- |
|  | «К ЗАЩИТЕ ДОПУСТИТЬ» |
|  | Руководитель курсового проекта  Ассистент кафедры ЭИ  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Н. А. Шилов |
|  | \_\_\_.\_\_\_\_.2023 |

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе

на тему:

**«Разработка автоматизированной системы организации деятельности кинотеатра»**

БГУИР КР 1-40 05 01-12 003 ПЗ

|  |  |
| --- | --- |
|  | Выполнил студент группы 224402  Бурая Александра Константиновна  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |
|  | Курсовая работа представлена на проверку \_\_\_.\_\_\_\_.2023  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |

Минск 2023

**РЕФЕРАТ**

БГУИР КР 1-40 05 01-12 003 ПЗ

**Бурая, А. К.** Разработка автоматизированной системы организации деятельности кинотеатра/А.К. Бурая. – Минск: БГУИР, 2023. – 46 с.

Пояснительная записка 46 с., 37 рис., 10 источников, 1 приложение.

Разработка автоматизированной системы организации деятельности кинотеатра, модели *UML*, *IDEF*0, схемы алгоритмов, программное средство

*Цель* *курсовой работы*: уменьшение временных затрат на покупку билетов в кинотеатре. Программа позволяет вносить корректировки в уже имеющиеся данные, создавать новые записи, удалять старые и сохранять отредактированные записи.

*Методология проведения работы*: в процессе решения поставленных задач использованы принципы системного подхода, аналитические методы, методы компьютерной обработки экспериментальных данных и компьютерного моделирования.

*Результаты работы*: выполнена постановка задачи и определены основные методы ее решения; в ходе объектного моделирования системы построен ряд *UML*-диаграмм; разработаны модели бизнес-процессов предметной области на основе нотации IDEF0; описаны основные алгоритмы работы программного средства; разработано руководство пользователя; выполнено тестирование программного средства, показавшее его соответствие функциональным требованиям, поставленным в задании на разработку.

Программный продукт разработан на языке C++ с применением MS Visual Studio 2022.

*Область применения результатов*: c помощью разработанного программного средства можно купить билеты в кинотеатре.

Разработанное программное средство полностью отвечает всем функциональным требованиям, необходимым при учете и анализе данных кинотеатра.

**СОДЕРЖАНИЕ**

[Введение 6](#_Toc154059578)

[1 Анализ и моделирование предметной области программного средства 7](#_Toc154059579)

[1.1 Описание предметной области 7](#_Toc154059580)

[1.2 Разработка функциональной модели предметной области 8](#_Toc154059581)

[1.3 Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований 11](#_Toc154059582)

[1.4 UML-модели представления программного обеспечения и их описание 12](#_Toc154059583)

[2 Проектирование и конструирование программного средства 15](#_Toc154059584)

[2.1 Постановка задачи и обзор методов ее решения 15](#_Toc154059585)

[2.2 Разработка модульной структуры 15](#_Toc154059586)

[2.3 Разработка перечня пользовательских функций программы 16](#_Toc154059587)

[2.4 Разработка блок-схем алгоритмов программы 18](#_Toc154059588)

[3 Тестирование и проверка работоспособности программного средства 25](#_Toc154059589)

[4 Инструкция по развертыванию приложения и сквозной тестовый пример 29](#_Toc154059590)

[4.1 Авторизация 29](#_Toc154059591)

[4.2 Модуль пользователя 30](#_Toc154059592)

[4.3 Модуль администратора 33](#_Toc154059593)

[Заключение 36](#_Toc154059594)

[Список использованных источников 37](#_Toc154059595)

[Приложение А 38](#_Toc154059596)

# **ВВЕДЕНИЕ**

Кинематограф играет огромную роль в современном обществе [1]. Человек получает целый спектр эмоций от просмотра фильмов. С каждым годом становится всё больше кинолент, которые по праву заслуживают звания шедевров мировой киноиндустрии. Благодаря созданию фильмов люди самовыражаются, передают свои мысли, идеи и чувства другим.

В современном мире всё больше людей выбирают просмотр кино онлайн, отдавая предпочтение онлайн-кинотеатрам или различным сайтам, предлагающим огромное количество подборок самых разных жанров. Однако домашний просмотр не может подделать эмоции, получаемые от похода в кинотеатр со своей семьей или друзьями на ожидаемую премьеру. Привычные кинотеатры предлагают большой экран, качественный звук, шуршащий попкорн и особенную атмосферу, которую невозможно воссоздать дома [2].

Благодаря развитию технологий посетителям кинотеатров больше не нужно стоять в больших очередях на премьеры или приходить заранее, чтобы успеть купить билеты на понравившийся фильм. Оплатить билет, выбрав удобное место и время, можно онлайн. Информационные системы, которые получают, сортируют и хранят данные об актуальной афише, названиях фильмов и ценах на сеансы, позволяют посетителям экономить время на покупке билетов [3].

Целью данного курсового проекта является уменьшение временных затрат на покупку билетов в кинотеатре. Программа предназначена для просмотра актуальной афиши и информации о премьерах, поиска определенного фильма, выбора времени сеанса и места в кинозале.

Для достижения поставленной цели необходимо решить следующие задачи:

* ознакомиться с предметной областью;
* проанализировать особенности создания расписания сеансов и оформления билетов для посетителей;
* разработать алгоритмы работы программного средства;
* разработать модули структуры программного средства;
* протестировать работу программного средства.

Объектом исследования является деятельность кинотеатра. Предмет исследования: автоматизация деятельности кинотеатра с помощью консольного приложения. Таким образом, разработка данной программы является актуальной и важной задачей, направленной на повышение эффективности и удобства посещения кинотеатра.

# **1 АНАЛИЗ И МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ ПРОГРАММНОГО СРЕДСТВА**

## **1.1 Описание предметной области**

Эффективное управление процессом составления расписания кинопоказа требует тщательного планирования, аналитики и понимания динамики рынка для предоставления зрителям разнообразного контента [4].

Этот процесс обычно проходит следующим образом. Сначала администрация кинотеатра выбирает фильмы, которые будут представлены в расписании, учитывая интересы целевой аудитории и анализ предыдущих успешных прокатов. Затем заключаются соглашения с кинокомпаниями о предоставлении прав на показ фильмов. Далее рассчитывается бюджет на приобретение прав, определяются цены на билеты, оценивается потенциальный доход на основе предполагаемой посещаемости и прогнозируется будущая прибыль. Разнообразный кинорепертуар должен включать жанры фильмов, удовлетворяющих различным вкусам зрителей, а также свежие премьеры. Затем составляется расписание сеансов: устанавливается день и время начала каждого сеанса, учитывается длительность фильмов для расчета интервалов между показами. Сбалансированное расписание учитывает время суток, день недели и сезонные и временные факторы, например, каникулы школьников. В итоге расписание становится доступным посетителям кинотеатра.

Для покупки билета в кино посетителю нужно прийти в кассы кинотеатра, выбрать фильм и время сеанса. Затем из предложенных вариантов зритель выбирает место в зале. Посетитель оплачивает билет, а после успешной оплаты получает его в печатном виде. После этих этапов посетитель может прийти на просмотр фильма в выбранный день и время и наслаждаться показом кино.

Существует несколько программных аналогов, предназначенных для автоматизации деятельности кинотеатров. Например, silverscreen.by, представляет сайт сети кинотеатров «Silver Screen Cinemas». Функционал сайта включает показ актуальной афиши, списка адресов кинотеатров, скидки и информацию для посетителей и сотрудничества. На этом сайте можно приобрести электронные билеты в кино только в данной сети заведений [5].

Сайты bycard.by [6] и afisha.me [7] предоставляют доступ к афишам фильмов разных кинотеатров. Функционал сайтов позволяет выбрать подходящий кинотеатр, стоимость билетов, дату и время сеанса, а также места в зале. После чего можно приобрести билет онлайн.

## **1.2 Разработка функциональной модели предметной области**

Рассмотрим информационную систему работы кинотеатра, выполненную с помощью средств моделирования нотации IDEF0 [8] (см. рисунок 1.1). Для начала необходимо создать контекстную модель информационной системы. На контекстной диаграмме представлен один функциональный блок «Составить расписание кинопоказов». Были определены данные входа, механизма, контроля и выхода.

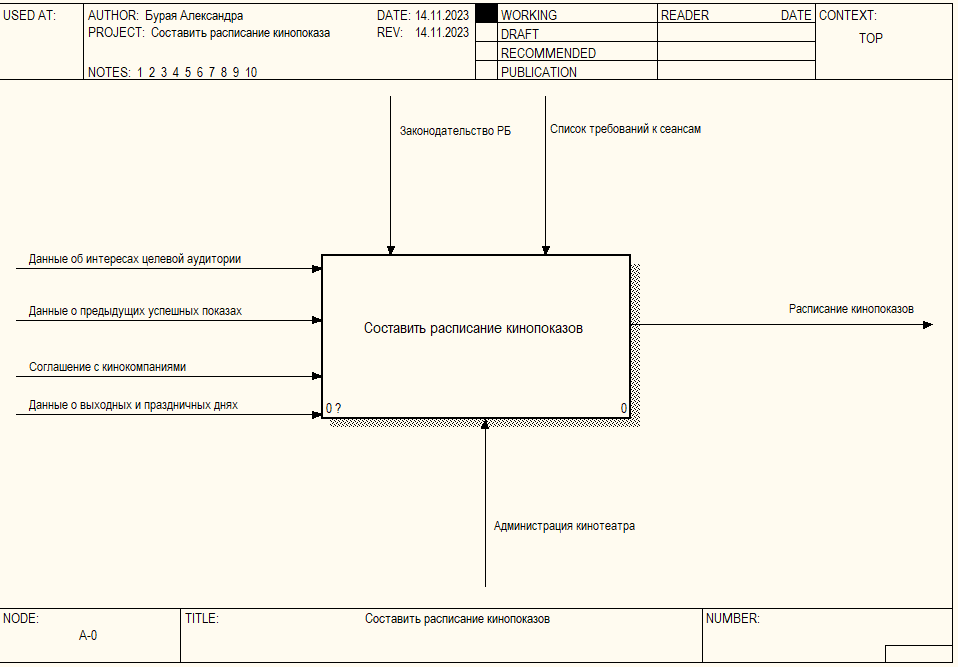


Рисунок 1.1 – Контекстная диаграмма

Контекстная диаграмма содержит следующие параметры:

* входные параметры: данные об интересах целевой аудитории, данные о предыдущих успешных показах, соглашение с кинокомпаниями, данные о выходных и праздничных днях;
* выходной параметр: расписание кинопоказов;
* управляющие параметры: законодательство РБ, список требований к сеансам;
* исполнительные параметры: администрация кинотеатра.

Нотация IDEF0 поддерживает последовательную декомпозициюпроцесса до требуемого уровня детализации. Декомпозиция позволяет представлять модель системы в виде иерархической структуры отдельных диаграмм, что делает ее менее перегруженной и легко усваиваемой. Диаграмма декомпозиции процесса «Составить расписание кинопоказов» представлена на рисунке 1.2.

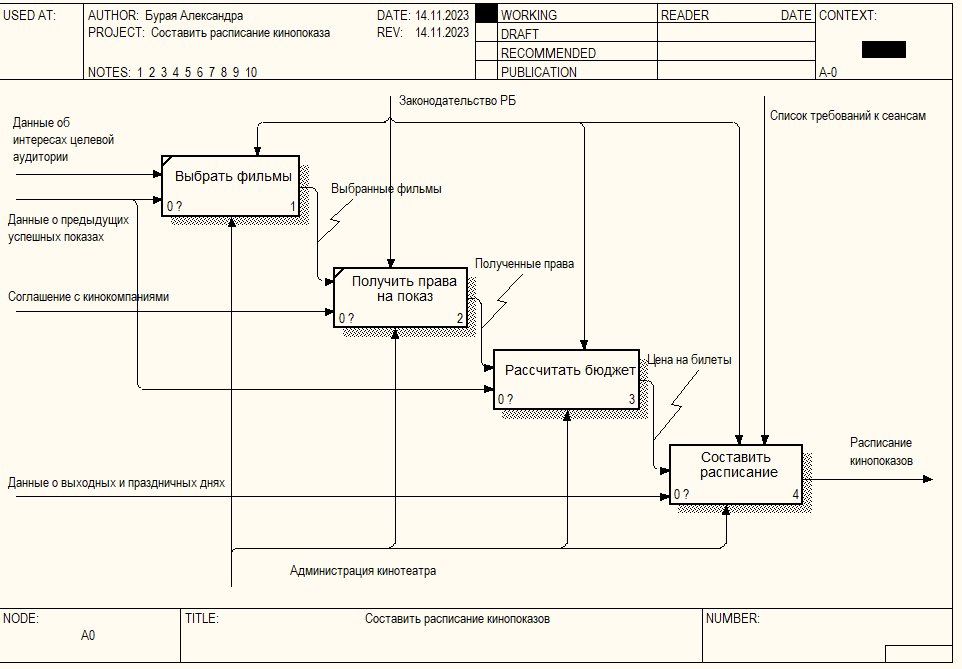


Рисунок 1.2 – Диаграмма декомпозиции

Данная диаграмма состоит из четырех функциональных блоков: «Выбрать фильмы», «Получить права на показ», «Рассчитать бюджет», «Составить расписание».

Чтобы более детально описать процесс «Рассчитать бюджет», происходит декомпозиция второго уровня (см. рисунок 1.3).

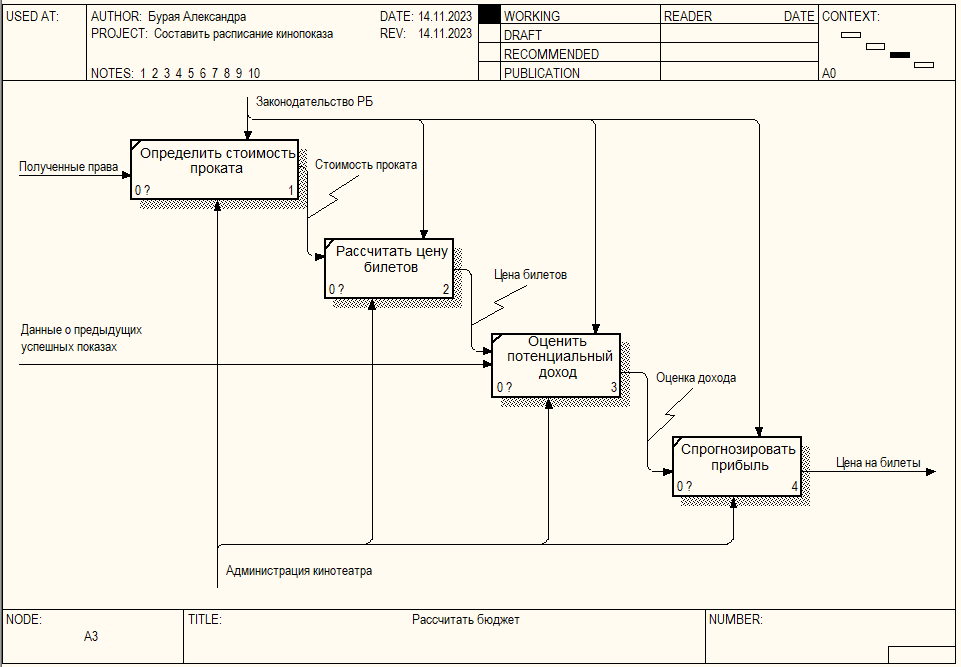


Рисунок 1.3 – Диаграмма декомпозиции процесса «Рассчитать бюджет»

Данная диаграмма состоит из четырех функциональных блоков: «Определить стоимость проката», «Рассчитать цену билетов», «Оценить потенциальный доход», «Спрогнозировать прибыль». Для более детального представления процесса «Составить расписание», происходит декомпозиция второго уровня (см. рисунок 1.4).

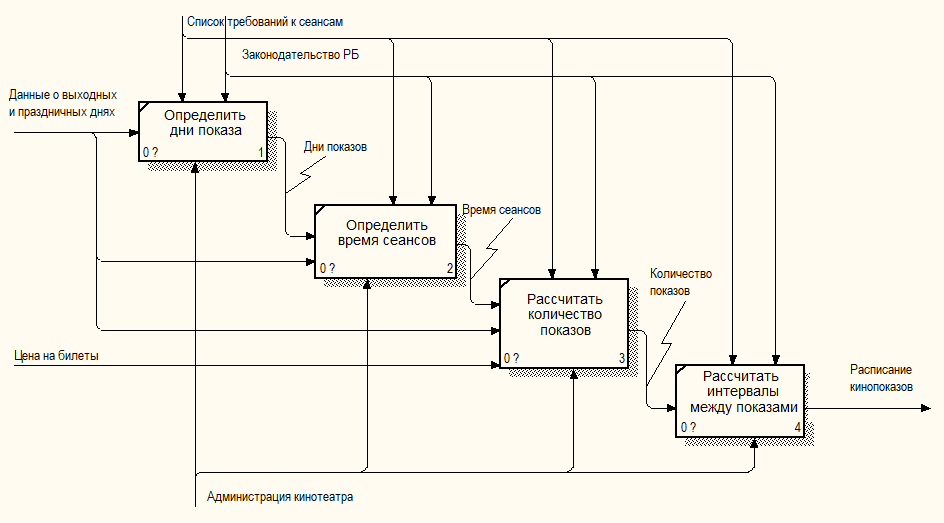


Рисунок 1.4 – Диаграмма декомпозиции процесса «Составить расписание»

Данная диаграмма состоит из четырех функциональных блоков: «Определить дни показа», «Определить время сеансов», «Рассчитать количество показов», «Рассчитать интервалы между показами».

## **1.3 Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований**

Анализ требований к разрабатываемому программному средству является важной частью процесса разработки, поскольку он помогает определить функциональные требования к программному продукту.

Разрабатываемое в ходе курсовой работы программное средство должно соответствовать следующим требованиям:

* наличие пользовательского меню, включающего разделение ролей, авторизацию пользователя, хранение пароля в зашифрованном виде;
* добавление, редактирование и удаление, поиск, сортировка и фильтрация записей;
* реализация заказа (вывод данных в виде отчета);
* хранение исходной и итоговой информации в текстовых или/и бинарных файлах;
* использование контейнеров библиотеки STL (Standard Template Library) для хранения данных в оперативной памяти;
* реализация базовых принципов объектно-ориентированного программирования;
* использование абстрактных классов;
* использование передачи параметров по ссылке и по значению;
* использование встроенных, пользовательских, дружественных, виртуальных функции;
* использование пространств имен (встроенных и собственных);
* использование перегрузки методов и операторов, переопределение методов;
* использование шаблонных классов и методов;
* использование статических методов и полей;
* использование динамического выделения памяти и умных указателей (smart pointers);
* использование потоков С++, перегрузки операторов ввода/вывода;
* обработка возможных ошибок, таких как неверный синтаксис выражения, деление на ноль и другие некорректные операции. В случае возникновения ошибки, приложение должно информировать пользователя об этом.

## **1.4 UML-модели представления программного обеспечения и их описание**

UMLявляется графическим языком для визуализации, конструирования и документирования систем, в которых большая роль принадлежит программному обеспечению. С его помощью разрабатывается детальный план создаваемой системы, содержащей не только ее концептуальные элементы, такие как системные функции и бизнес-процессы, но и конкретные особенности, например классы, написанные на специальных языках программирования, схемы баз данных и программные компоненты многократного использования. Диаграмма вариантов использования представляет совокупность прецедентов и актеров. Они описывают систему с точки зрения возможностей ее использования [9].

Система содержит следующие роли пользователей: администратор, пользователь. Администратор имеет следующие возможности использования системы:

* вход в систему;
* просмотр и изменение личных данных;
* регистрация нового администратора;
* просмотр данных пользователей;
* изменение статуса пользователя;
* просмотр и редактирование афиши.

Пользователь имеет следующие возможности использования системы:

* регистрация;
* вход в систему;
* просмотр и изменение личных данных;
* просмотр афиши;
* сортировка фильмов по жанрам и поиск фильма;
* покупка билетов;
* просмотр купленных билетов;
* просмотр афиши;
* сортировка фильмов по жанрам;
* поиск фильма.

На рисунке 1.5 представлена диаграмма вариантов использования системы.

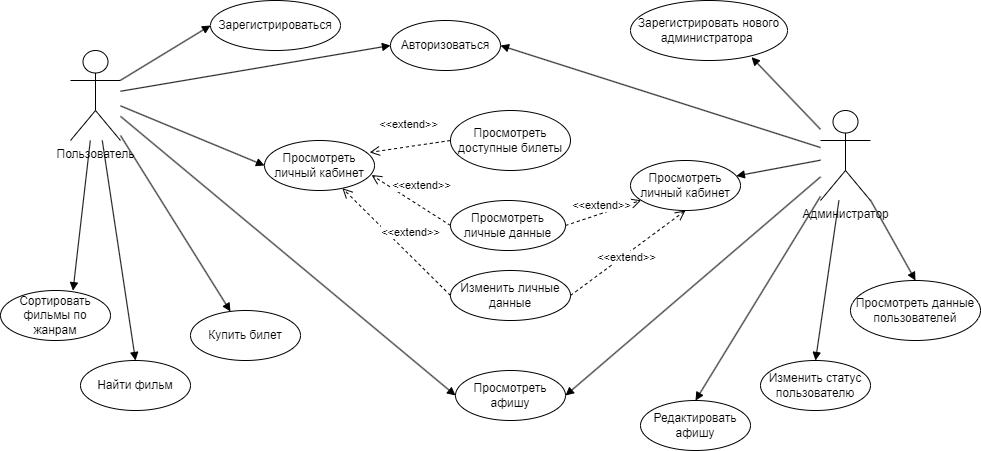


Рисунок 1.5 – Диаграмма вариантов использования

Диаграмма классов является основным логическим представлением модели и содержит детальную информацию об архитектуре программы. На рисунке 1.6 представлена диаграмма классов.

Классы и объекты системы описываются следующим образом.

Существует список пользователей (User). Каждый пользователь имеет свой статус (status). Администратор (User со статусом 1)имеет возможность создавать новых администраторов и менять статусы существующим. Также администратор может редактировать и создавать новые фильмы (Film). Пользователь (User со статусом 0) может покупать билеты (Session) и просматривать их в личном кабинете.

Абстрактный класс BaseTable и наследуемые классы Table и ColoredTable отвечают за вывод данных в табличном виде.

Шаблонный класс VectorSorte со статическим методом Sort сортирует переданный вектор по заданному полю.

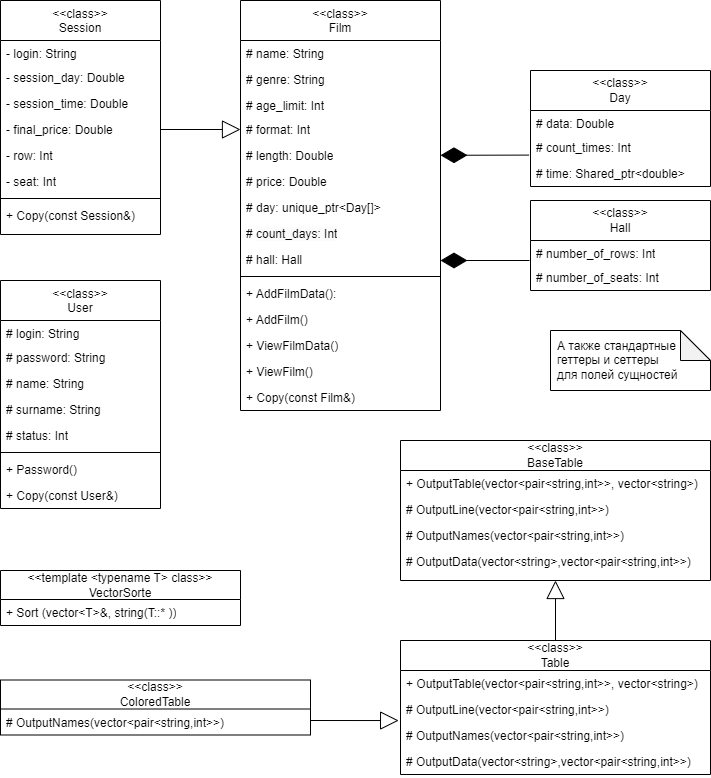


Рисунок 1.6 – Диаграмма классов

# **ПРОЕКТИРОВАНИЕ И КОНСТРУИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА**

## **2.1 Постановка задачи и обзор методов ее решения**

Главной задачей была разработка консольного приложения с функционалом, необходимым для эффективной организации системы кинотеатра, на языке C++. При разработке объектной модели требуется определить ее составные части, установить и спроектировать корректные связи между ними [10].

Для решения поставленной задачи существуют следующие требования к системе:

* операционная система Windows 7 и выше;
* интерфейс программы и данные должны быть представлены на русском или белорусском языках;
* приложение должно запускаться без использования интегрированных средств разработки.

После анализа поставленных задач были выбраны следующие средства их решения. В качестве среды для разработки была выбрана MS Visual Studio 2022. Для построения диаграмм IDEF0 была выбрана программа AllFusion Process Modeler 7. Данный инструмент содержит удобную структуру декомпозиций, которая помогает быстро создать нужные диаграммы.

## **2.2 Разработка модульной структуры**

На рисунке 2.1 представлена схема работы всей программы.

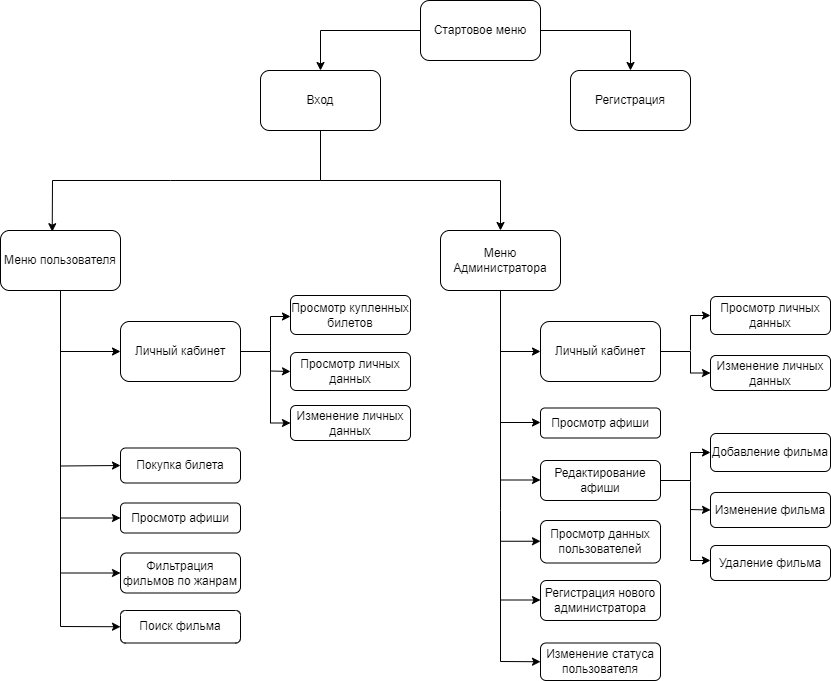


Рисунок 2.1 – Модульная структура программы

## **Разработка перечня пользовательских функций программы**

//Improcom.h

Функции, создающие меню разных уровней:

* void UserLoginMenu();
* void LoginMenu(int status, short x, short y);
* void AdminLoginMenu();
* void UserMenu(User account);
* void AdminMenu(User account);
* void PersonalAccount(User& account);
* void PersonalAccountAdmin(User& account).

//User.h

Функции для работы с пользовательскими данными:

* void Encrypt(string& pass); // Зашифровка
* void Decrypt(string& pass); // Расшифровка
* int LengthFile(string name); // Получение размера файла для "user.txt"
* void ViewUsers();
* void ViewUser(User account);
* void ChangeStatus();
* void ChangeUser(User& account);
* void SorteUsers(vector<User> &vec\_users).

//Cinema.h

Функции для работы с фильмами и билетами:

* void Calendar(int month);
* void ViewFilms();
* void EditFilms();
* void ChangeFilm();
* int FilmFileLength(string name);
* bool UniqueFilm(string film);
* void ChoiceFilm(string login);
* void GenreSorte();
* template <typename T> void Find(vector<T> vec);
* template void Find<Film>(vector<Film> vec);
* void ViewTickets(string account);
* void OrderedPlaces(vector<pair<int, int>>& ordered\_seats, string name, double data, double time);
* bool OrderedFilms(string film).

//Output.h

Функции для работы с консолью:

* void ConsoleCursorVisible(bool show, short size); // Видимость и размеры курсора
* void GoToXY(short x, short y); // Курсор в точку x,y
* void SetColor(ConsoleColor text, ConsoleColor background); // Установка цвета текста.

Функции для вывода меню:

* int DrawMenu(string start, string menu[], int size, int X, int Y);
* template <typename T>
* T DrawMenuChoice(string start, const vector <T>& films);
* template string DrawMenuChoice(string start, const vector <string>& films);
* template double DrawMenuChoice(string start, const vector <double>& films);
* void DrawHall(string start, int hall\_rows, int hall\_seats, int &row, int &seat, vector<pair<int, int>> ordered\_seats);

Функции для проверки на ввод:

* int CinIntErrorCheck(int min, int max);
* double CinDoubleErrorCheck(double min, double max);
* int CinIntErrorCheck();
* string CinStringSymRusErrorCheck();
* string CinStringSymAngErrorCheck(short& x, short& y);
* string CinStringSymRusErrorCheck(short& x, short& y);
* string GetlineErrorCheck();
* void RemoveSpace(string& a);

## **2.4 Разработка блок-схем алгоритмов программы**

На рисунке 2.2 представлена блок-схема алгоритма всей программы.

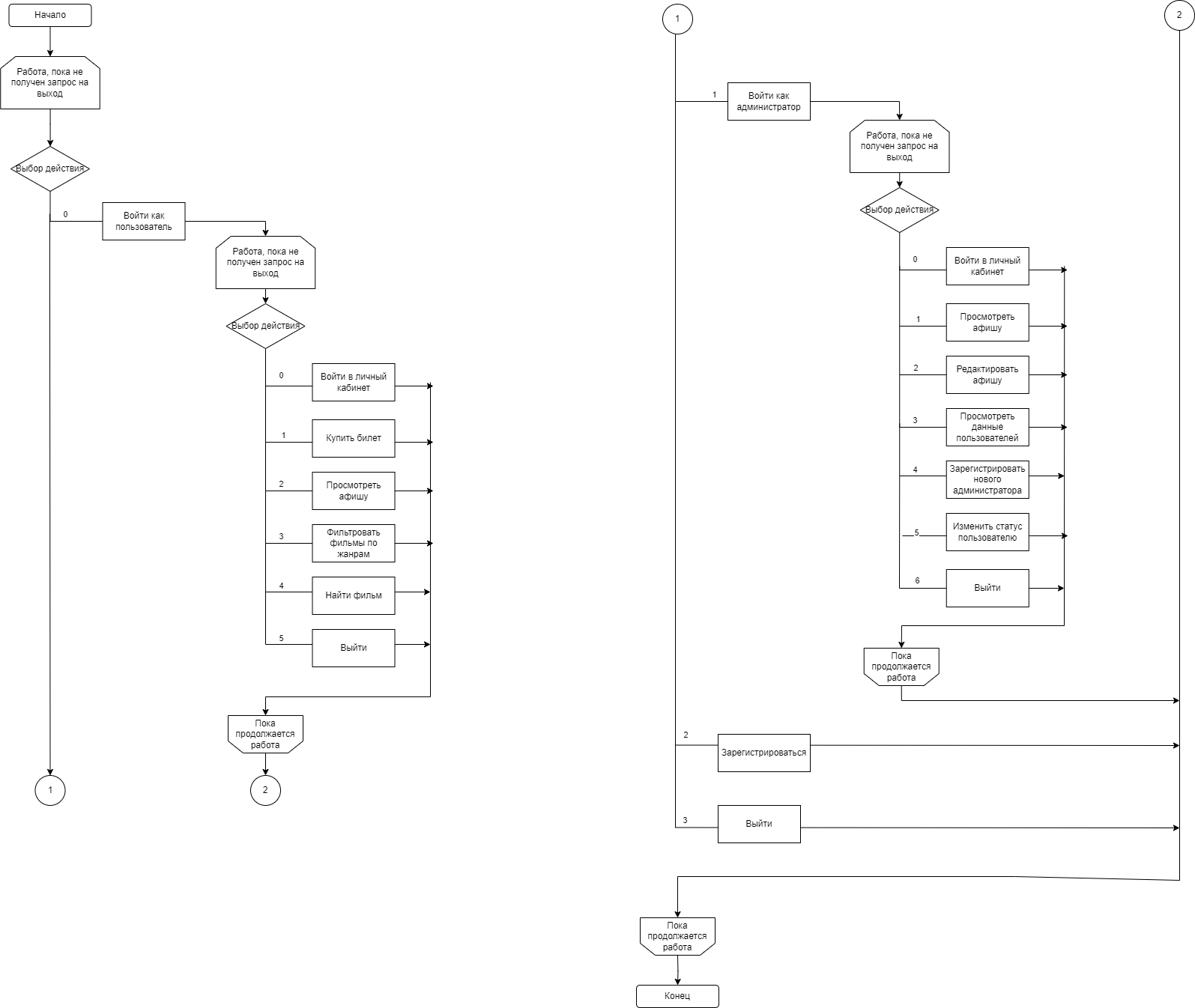


Рисунок 2.2 – Блок-схема алгоритма всей программы

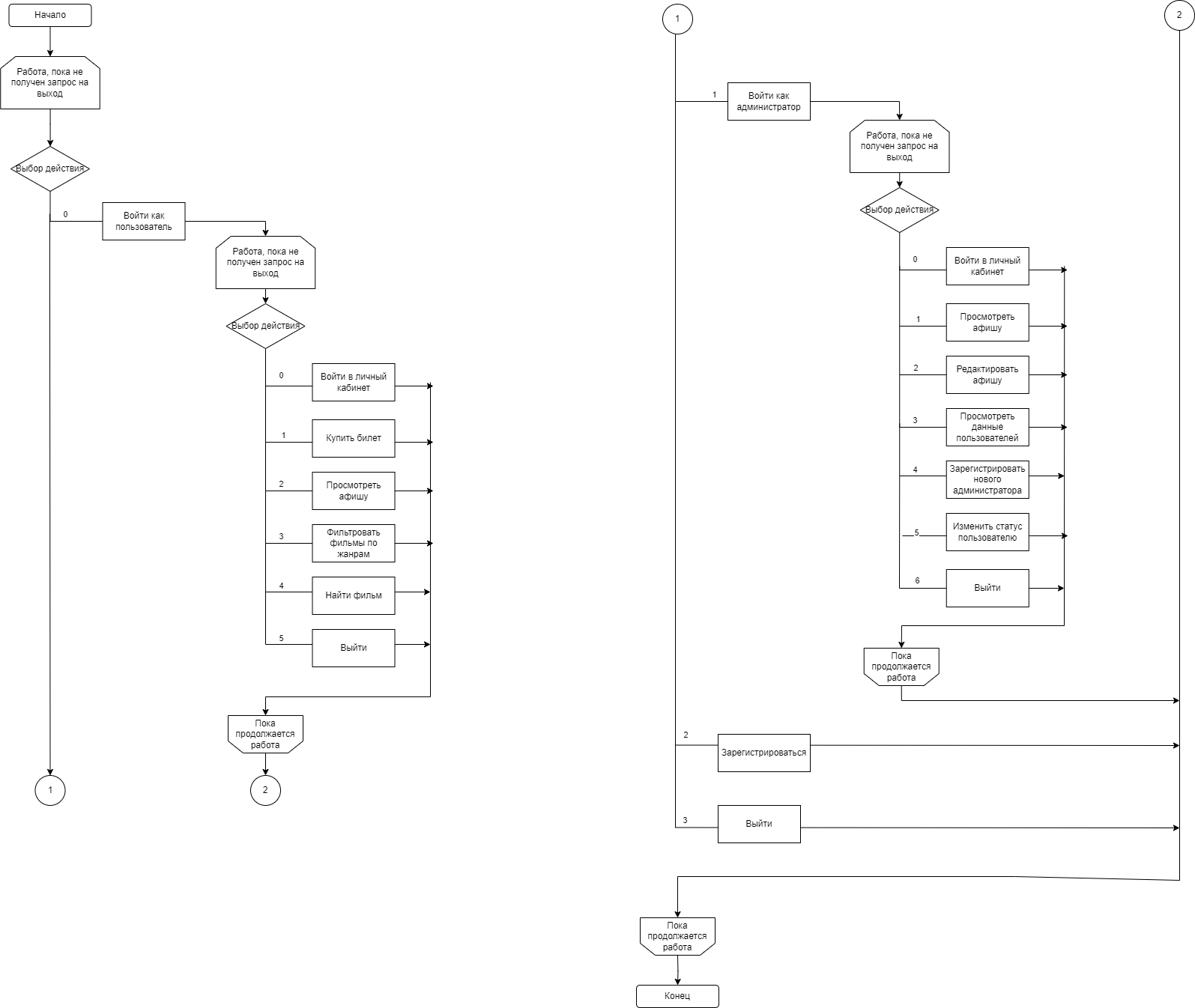


Рисунок 2.2 – Блок-схема алгоритма всей программы (продолжение)

Реализация функции DrawMenu(string start, string menu[], int size, int X, int Y):

int DrawMenu(string start, string menu[], int size, int X, int Y)

{

ConsoleCursorVisible(false, 100);

cout << fixed << setprecision(2);

int active\_menu = 0;

char ch;

while (true){

int y = Y, x = X;

system("cls");

int start\_size = start.size();

if (x > 3)

GoToXY(x - (start\_size / 2) + int(23 / 2), y);

else

GoToXY(x - 3, y);

SetColor(LightMagenta, Black);

cout << start;

SetColor(LightCyan, Black);

y += 3;

for (int i = 0; i < size; i++){

if (i == active\_menu)

{

SetColor(LightMagenta, Black);

GoToXY(x - 3, y);

cout << "> ";

}

else SetColor(LightCyan, Black);

GoToXY(x, y++);

cout << " " << menu[i] << endl;

}

ch = \_getch();

if (ch == -32) ch = \_getch();

switch (ch)

{

case ESC:

return ESC;

case UP:

if (active\_menu > 0)

active\_menu--;

else

active\_menu = size - 1;

break;

case DOWN:

if (active\_menu < size - 1)

active\_menu++;

else

active\_menu = 0;

break;

case ENTER:

return active\_menu;

default:

y++;

GoToXY(x, y);

SetColor(LightMagenta, Black);

cout << "Используйте стрелки" << endl;

\_getch();

system("CLS");

}

}

}

На рисунке 2.3 представлена схема алгоритма вывода меню.

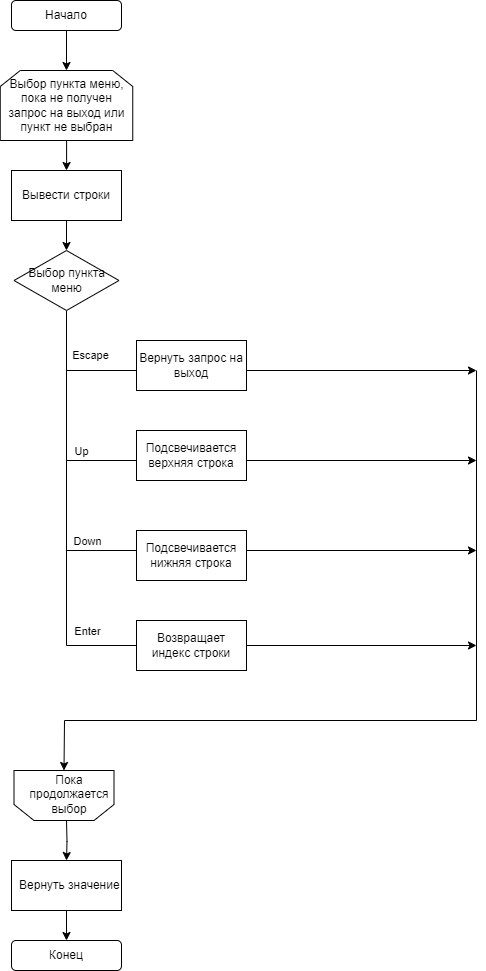


Рисунок 2.3 – Алгоритм функции DrawMenu

Реализация функции ViewTickets(string account) для просмотра билетов в личном кабинете:

void ViewTickets(string account)

{

int size = FilmFileLength("tickets.txt");

vector <Session> t;

ifstream f;

f.open("tickets.txt");

if (!f.is\_open())

{

cout << "Ошибка открытия файла" << endl;

return;

}

else

{

Session temp1;

for (int i = 0; i < size; i++)

{

f >> temp1;

t.push\_back(temp1);

}

}

f.close();

VectorSorte<Session>::Sort(t, &Session::GetName);

bool a = 0;

cout << "Ваши билеты: " << endl;

for (int i = 0; i < t.size(); i++)

{

if (t[i] == account)

{

a = 1;

cout << endl << t[i] << endl;

}

}

if (a == 0)

cout << "Нет доступных билетов" << endl;

ofstream file;

file.open("tickets.txt");

if (!file.is\_open())

{

cout << "Ошибка открытия файла" << endl;

return;

}

else

{

for (int i = 0; i < t.size(); i++)

{

file << t[i];

}

}

file.close();

}

На рисунке 2.4 представлена схема алгоритма просмотра билетов.

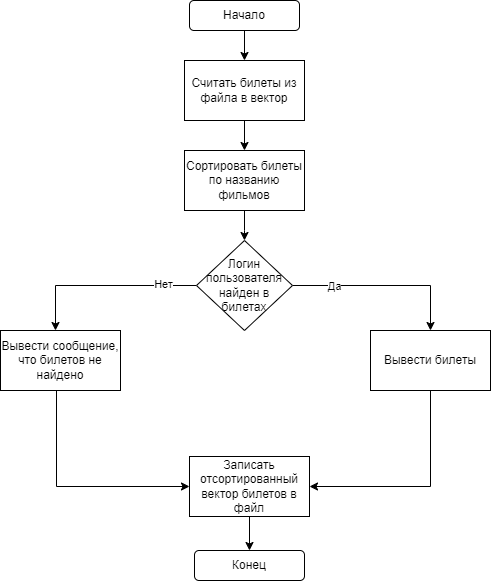


Рисунок 2.4 – Алгоритм функции ViewTickets

# **3 ТЕСТИРОВАНИЕ И ПРОВЕРКА РАБОТОСПОСОБНОСТИ ПРОГРАММНОГО СРЕДСТВА**

При написании кода программы необходимо предусмотреть обработку исключительных ситуаций, таких как:

* введенные пользователем данные не соответствуют формату поля (например, символы в числовом поле);
* введенные пользователем данные нелогичны (например, количество дней больше 31);
* ничего не найдено по результатам поиска;
* проверки на уникальность введенных данных (если необходимо).

Выбор в меню осуществляется только при помощи стрелок вверх-вниз. В противном случае появляется сообщение о необходимости использовать стрелки (см. рисунок 3.1).

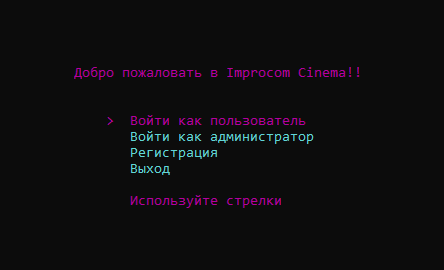


Рисунок 3.1 – Проверка выбора пункта меню

Проверка на уникальность логина при регистрации (см. рисунок 3.2). Если учетная запись с таким именем уже существует, то появляется сообщение об ошибке.

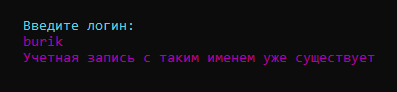


Рисунок 3.2 – Проверка ввода логина при регистрации

Проверка пароля при входе и регистрации (см. рисунок 3.3). При входе в роли администратора и регистрации у пользователя 3 попытки ввода пароля (подтверждения пароля), при входе в роли обычного пользователя – 5 попыток ввода.

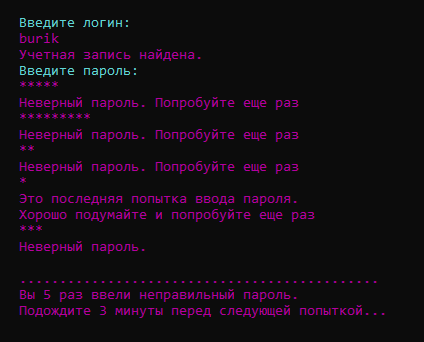


Рисунок 3.3 – Подтверждение пароля

Проверка на ввод нечисловых данных (принимаются только целые и дробные значения >0) представлена на рисунке 3.4.

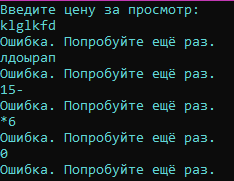


Рисунок 3.4 – Проверка на ввод целых чисел

Проверка на ввод названия фильма (принимается только кириллица и цифры) представлена на рисунке 3.5.

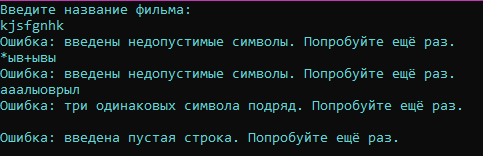


Рисунок 3.5 – Проверка на ввод строк

Проверка на ввод несуществующего возрастного ограничения представлена на рисунке 3.6 (можно ввести цифры 0, 3, 6, 12, 15, 16, 18, 21).

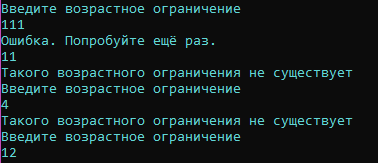


Рисунок 3.6 – Проверка на ввод возрастного ограничения

Проверка на ввод валидной даты: можно ввести только будущие дни (см. рисунок 3.7).

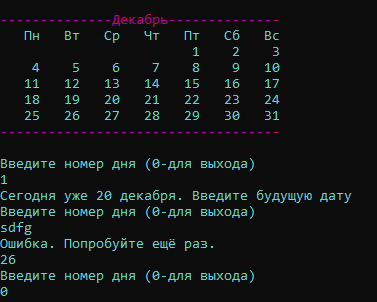


Рисунок 3.7 – Проверка на ввод даты

Ввод некорректного времени и количества сеансов (см. рисунок 3.8). Программа рассчитывает, сколько сеансов можно провести, учитывая интервалы >20 минут между сеансами, длину фильма и работу кинотеатра.

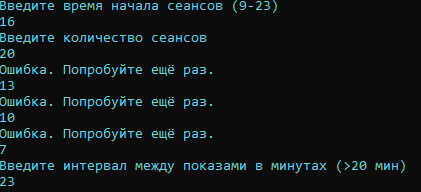


Рисунок 3.8 – Проверка на ввод времени и количества сеансов

Проверка на существования фильма (см. рисунок 3.9). Если фильма с таким названием не существует, то появляется сообщение об ошибке.



Рисунок 3.9 – Проверка на существование фильма

Проверка на необходимость изменений статуса пользователя представлена на рисунке 3.10. Если статус пользователя и так равен введенному, то появляется сообщение об ошибке.

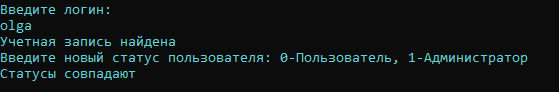


Рисунок 3.10 – Проверка на изменение статуса

# **4 ИНСТРУКЦИЯ ПО РАЗВЕРТЫВАНИЮ ПРИЛОЖЕНИЯ И СКВОЗНОЙ ТЕСТОВЫЙ ПРИМЕР**

## **4.1 Авторизация**

В данной части работы представлен пример работы программы при авторизации и регистрации пользователя. После загрузки программы появляется меню, с помощью которого производится вход в систему. При выборе варианта «Войти как пользователь» осуществляется вход от имени пользователя, при выборе варианта «Войти как администратор» – от имени администратора, если выбрать поле «Регистрация», начнется создание учетной записи пользователя. При выборе «Выхода» программа завершает свою работу (см. рисунок 4.1).

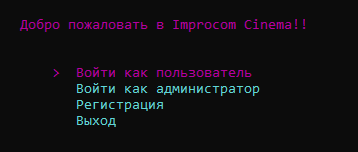


Рисунок 4.1 – Меню авторизации

Непосредственный вход в аккаунт представлен на рисунке 4.2. В целях безопасности пароль замаскирован с помощью символа «\*».

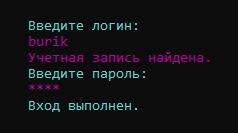


Рисунок 4.2 – Авторизация

## **4.2 Модуль пользователя**

При входе в систему под именем пользователя, появляется главное меню пользователя, которое включает функции, представленные на рисунке 4.3.

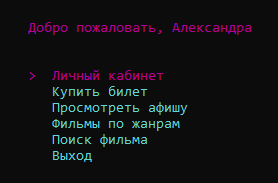


Рисунок 4.3 – Меню пользователя

Далее пользователь может перейти в личный кабинет, где можно посмотреть и редактировать данные своей учетной записи, а также посмотреть купленные билеты (см. рисунок 4.4).

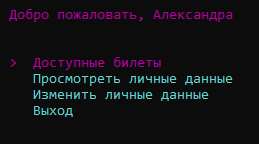


Рисунок 4.4 – Личный кабинет

Информация о доступных билетах представляется в виде чеков (см. рисунок 4.5).

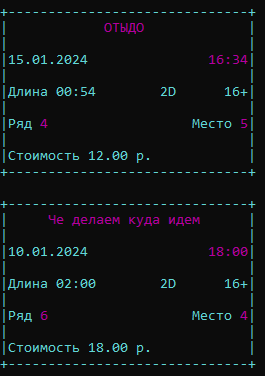


Рисунок 4.5 – Вывод информации о билетах

Функционал функции «Купить билет» включает в себя выбор фильма, затем выбор даты и времени, а потом мест в зале. При выборе мест в зале можно пользоваться стрелочками вверх-вниз, влево-вправо, для выбора места нужно нажать на enter, после чего на выбранном месте отобразится «+», обозначающий занятое место, за раз можно заказать несколько билетов, которые затем отобразятся в личном кабинете (см. рисунок 4.6-4.9).

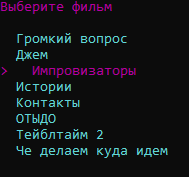


Рисунок 4.6 – Выбор фильма

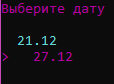


Рисунок 4.7 – Выбор даты

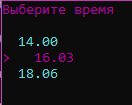


Рисунок 4.8 – Выбор времени

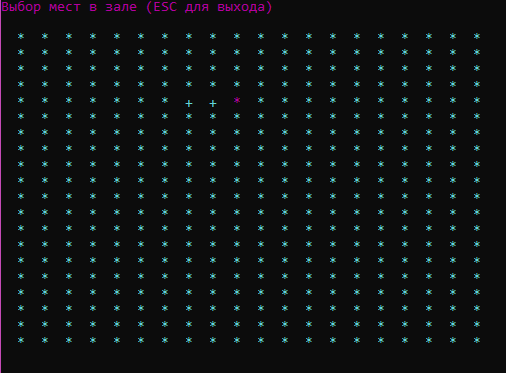


Рисунок 4.9 – Выбор места в зале

При выборе пункта меню «Просмотреть афишу», на экран выводится информация о фильмах в табличном виде (см. рисунок 4.10).



Рисунок 4.10 – Афиша

## **4.3 Модуль администратора**

При входе в систему под именем администратора, появляется главное меню администратора, которое включает функции, представленные на рисунке 4.11.

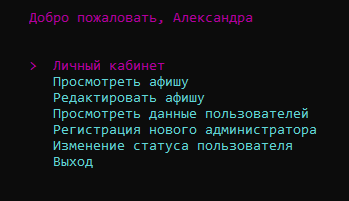


Рисунок 4.11 – Меню администратора

Меню «Личный кабинет» включает функции, представленные на рисунке 4.12.

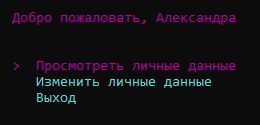


Рисунок 4.12 – Личный кабинет

При выборе пункта «Редактирование афиши» администратор получает доступ к функциям добавления, редактирования и удаления фильмов (см. рисунок 4.13).

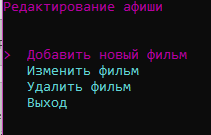


Рисунок 4.13 – Редактирование афиши

При выборе пункта «Изменить фильм» и вводе названия нужного фильма администратор переходит к выбору изменяемого параметра (см. рисунок 4.15-4.16).

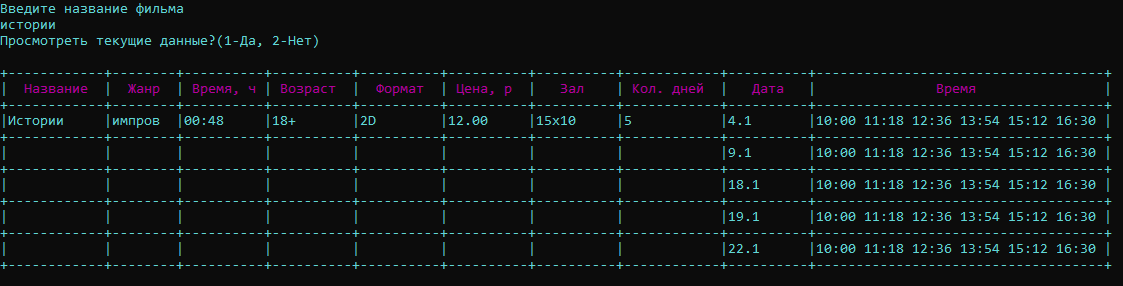


Рисунок 4.15 – Просмотр информации об изменяемом фильме

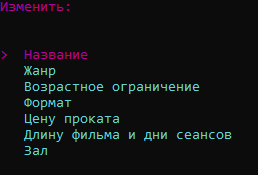


Рисунок 4.16 – Редактирование

При выборе пункта меню «Просмотреть данные пользователей», на экран выводится информация об учетных записях в табличном виде (см. рисунок 4.17).

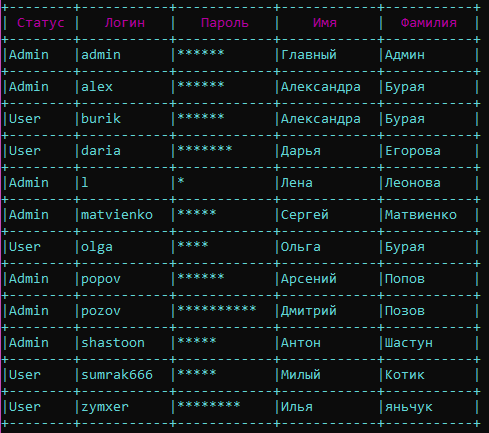


Рисунок 4.17 – Данные пользователей

# **ЗАКЛЮЧЕНИЕ**

При выполнении курсовой работы был изучен процесс составления расписания кинопоказов. Было установлено, что правильная организация и хранение данных позволяет уменьшить трудоемкость выполнения этой операции вручную.

В ходе объектного моделирования системы был построен ряд UML-диаграмм; разработаны модели бизнес-процессов предметной области на основе нотаций IDEF0; описаны основные алгоритмы работы программного средства; разработано руководство пользователя; выполнено тестирование программного средства, показавшее его соответствие функциональным требованиям, поставленным в задании на разработку.

В ходе работы были созданы пользовательские функции приложений: функции поиска, сортировки, получения данных из файла, авторизации, редактирования, проверки данных и другие. Были созданы следующие блок схемы алгоритмов функций: вывод меню, вывод купленных билетов, а также блок-схема всей программы. Были рассмотрены исключительные ситуации и их обработка с помощью функций проверок.

Цель курсовой работы (уменьшение временных затрат на покупку билетов в кинотеатре) была успешно достигнута.

Итог данной курсовой работы – консольное приложение, которое помогает администратору быстро редактировать и просматривать афишу, работать с учетными записями пользователей; а пользователю ускоряет процесс покупки билетов, просмотра афиши и поиска фильмов.

В этом приложении присутствуют конструкции объектно-ориентированного программирования, стандартные и пользовательские функции, абстрактные классы, шаблонные функции и классы, использование умных указателей и контейнеров библиотеки STL.

В заключении стоит отметить, что с данной программой может комфортно работать как обычный посетитель с правами пользователя, так и работник кинотеатра с правами администратора. Разработанное программное средство полностью отвечает всем функциональным требованиям, необходимым при учете и анализе данных деятельности кинотеатра.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Статья «Роль кинематографа в жизни современного человека». [Электронный ресурс]. – Режим доступа: https://edu-time.ru/pub/101767
2. Статья «Взгляд со стороны на онлайн-кинотеатры: что дальше» [Электронный ресурс]. – Режим доступа :

<https://adindex.ru/publication/opinion/offtop/2020/12/9/287489.phtml>

1. Статья «Комплексная информационная система (КИС) "Кинотеатр"» [Электронный ресурс]. – Режим доступа: <http://digitaldevice.by/kinoteatr.html>
2. Статья «Наука кинотеатра: проектирование и оснащение. Теория, правила, стандарты, рекомендации» [Электронный ресурс]. – Режим доступа : <https://magna-tech.org/blog/nauka-kinoteatra-proektirovanie-i-osnashhenie-teoriya-pravila-standarty-rekomendatsii-obnovlyaemyj-material/>
3. Официальный сайт «Silver Screen» [Электронный ресурс]. – Режим доступа : <https://silverscreen.by/afisha/>
4. Официальный сайт «Bycard» [Электронный ресурс]. – Режим доступа : https://bycard.by/afisha/minsk
5. Официальный сайт «Afisha.me» [Электронный ресурс]. – Режим доступа : https://afisha.me/film/
6. Методология IDEF0 [Электронный ресурс]. – Режим доступа: [..\трпо\ТРПО Лабораторная работа №2.docx](file:///C:\\Users\\Анюта\\трпо\\ТРПО%20Лабораторная%20работа%20№2.docx).
7. Методическое пособие для выполнения лабораторной роботы «Язык Uml. Диаграммы UML» [Электронный ресурс]. – Режим доступа: [..\трпо\ТРПО - Лабораторные работы №7-8.doc](file:///C:\\Users\\Анюта\\трпо\\ТРПО%20-%20Лабораторные%20работы%20№7-8.doc)x.
8. Статья «Программное обеспечение для кинотеатров: с чего начать выбор?» [Электронный ресурс]. – Режим доступа :

<https://www.kinometro.ru/materials/show/name/kak_vybrat_programmnoe_obespechenie_270117>

# **ПРИЛОЖЕНИЕ А**

**Листинг кода**

//Improcom.h

#pragma once

#include <iostream>

#include <string>

#include <vector>

#include <windows.h>

#include <conio.h>

#include <fstream>

#include "User.h"

#include "Cinema.h"

using namespace std;

void UserLoginMenu();

void LoginMenu(int status, short x, short y);

void AdminLoginMenu();

void UserMenu(User account);

void AdminMenu(User account);

void PersonalAccount(User& account);

void PersonalAccountAdmin(User& account);

//Improcom.cpp

#include "Improcom.h"

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

setlocale(LC\_ALL, "Ru");

SetConsoleTitle(L"Improcom Cinema");

ConsoleCursorVisible(false, 100);

short x = 55, y = 12, k=5;

cout << fixed << setprecision(2);

SetColor(LightMagenta, Black);

GoToXY(x-2, y);

cout << "Improcom Cinema"; GoToXY(x, y + k);

SetColor(LightCyan, Black);

cout << "Загрузка."; GoToXY(x, y + k);

Sleep(500);

cout << "Загрузка.."; GoToXY(x, y + k);

Sleep(500);

cout << "Загрузка..."; GoToXY(x - 13, y + 2 + k);

Sleep(1000);

cout << "Нажмите любую клавишу для продолжения"; GoToXY(x, y);

\_getch();

system("CLS");

while (TRUE) {

string m[] = { "Войти как пользователь", "Войти как администратор", "Регистрация", "Выход" };

int a = DrawMenu("Добро пожаловать в Improcom Cinema!!", m, size(m), 48, 10);

Продолжение приложения А

system("CLS");

switch (a)

{

case 0:

UserLoginMenu();

break;

case 1:

AdminLoginMenu();

break;

case 2:

LoginMenu(0, 48, 12);

\_getch();

system("CLS");

break;

case 3:

GoToXY(48, 13);

SetColor(LightCyan, Black);

cout << "Выход из программы...";

\_getch();

system("CLS");

GoToXY(0, 0);

exit(0);

break;

case ESC:

GoToXY(48, 13);

SetColor(LightCyan, Black);

cout << "Выход из программы...";

\_getch();

system("CLS");

GoToXY(0, 0);

exit(0);

break;

}

}

}

void UserLoginMenu()

{

ConsoleCursorVisible(false, 100);

User temp;

string log;

bool st = 0, l = 0, p = 0;

short y = 12, x = 48;

int k = 0;

temp.SetStatus(0);

GoToXY(x, y);

SetColor(LightCyan, Black);

cout << "Введите логин: ";

SetColor(LightMagenta, Black);

y++;

log = CinStringSymAngErrorCheck(x, y);

temp.SetLogin(log);

int size = LengthFile("user.txt");

vector<User> vec\_users;

ifstream f;

try

Продолжение приложения А

{

f.open("user.txt");

if (f.is\_open())

{

User temp1;

for (int i = 0; i < size; i++)

{

f >> temp1;

vec\_users.push\_back(temp1);

}

f.close();

}

else

throw("Файл не найден");

}

catch (const char\* message) {

cout << "Вызвано исключение." << message << endl;

exit(0);

}

for (int i = 0; i < vec\_users.size(); i++)

{

if (vec\_users[i].GetStatus() == temp.GetStatus())

{

st = 1;

if (vec\_users[i].GetLogin() == temp.GetLogin())

{

l = 1;

GoToXY(x, y++);

SetColor(LightMagenta, Black);

cout << "Учетная запись найдена.";

GoToXY(x, y++);

SetColor(LightCyan, Black);

cout << "Введите пароль: ";

SetColor(LightMagenta, Black);

while (k != 5)

{

GoToXY(x, y++);

temp.Password();

//SetColor(LightMagenta, Black);

while (temp.GetPassword() ==

"NOOOOOOOPASSWORDERRORERRORERROR")

{

cout << "Вы не ввели пароль, попробуйте еще раз" << endl;

GoToXY(x, y++);

temp.Password();

}

if (vec\_users[i].GetPassword() == temp.GetPassword())

{

temp.SetName(vec\_users[i].GetName());

temp.SetSurname(vec\_users[i].GetSurname());

p = 1;

GoToXY(x, y++);

SetColor(LightCyan, Black);

Продолжение приложения А

cout << "Вход выполнен.";

SetColor(LightMagenta, Black);

\_getch();

system("CLS");

UserMenu(temp);

return;

}

else if (k == 3)

{

GoToXY(x, y++);

cout << "Это последняя попытка ввода пароля.";

GoToXY(x, y++);

cout << "Хорошо подумайте и попробуйте еще раз";

k++;

}

else if (k == 4)

{

GoToXY(x, y++);

cout << "Неверный пароль.";

k++;

break;

}

else

{

GoToXY(x, y++);

cout << "Неверный пароль. Попробуйте еще раз";

k++;

}

}

}

}

}

if (st == 0 || l == 0)

{

GoToXY(x, y++);

cout << "Учетной записи с таким именем не найдено";

\_getch();

system("CLS");

}

else if (p == 0)

{

y++;

GoToXY(x, y++);

cout << ".............................................";

GoToXY(x, y++);

cout << "Вы 5 раз ввели неправильный пароль.";

GoToXY(x, y++);

cout << "Подождите 3 минуты перед следующей попыткой...";

Sleep(7000);

\_getch();

system("CLS");

}

Продолжение приложения А

return;

}

void LoginMenu(int status, short x, short y)

{

ConsoleCursorVisible(false, 100);

User temp, temp\_pass;

string log, name, surname;

bool st = 1, p = 0;

int k = 0;

temp.SetStatus(status);

GoToXY(x, y++);

SetColor(LightCyan, Black);

cout << "Введите логин: ";

SetColor(LightMagenta, Black);

rewind(stdin);

log = CinStringSymAngErrorCheck(x, y);

int size = LengthFile("user.txt");

vector<User> vec\_users;

ifstream f;

try

{

f.open("user.txt");

if (f.is\_open())

{

User temp1;

for (int i = 0; i < size; i++)

{

f >> temp1;

vec\_users.push\_back(temp1);

}

f.close();

}

else

throw("Файл не найден");

}

catch (const char\* message) {

cout << "Вызвано исключение." << message << endl;

exit(0);

}

for (int i = 0; i < size; i++)

{

if (vec\_users[i].GetLogin() == log)

st = 0;

}

if (st == 1)

{

st = 1;

temp.SetLogin(log);

GoToXY(x, y++);

SetColor(LightCyan, Black);

cout << "Введите ваше имя: ";

SetColor(LightMagenta, Black);

name = CinStringSymRusErrorCheck(x, y);

name[0] = toupper(name[0]);

SetColor(LightCyan, Black);

Продолжение приложения А

GoToXY(x, y++);

cout << "Введите вашу фамилию: ";

SetColor(LightMagenta, Black);

surname = CinStringSymRusErrorCheck(x, y);

surname[0] = toupper(surname[0]);

SetColor(LightCyan, Black);

GoToXY(x, y++);

cout << "Введите пароль: ";

SetColor(LightMagenta, Black);

GoToXY(x, y++);

temp.SetName(name);

temp.SetSurname(surname);

temp.Password();

while (temp.GetPassword() ==

"NOOOOOOOPASSWORDERRORERRORERROR")

{

cout << "Вы не ввели пароль, попробуйте еще раз" << endl;

GoToXY(x, y++);

temp.Password();

}

GoToXY(x, y++);

SetColor(LightCyan, Black);

cout << "Подтвердите пароль: ";

SetColor(LightMagenta, Black);

while (k != 3)

{

GoToXY(x, y++);

temp\_pass.Password();

while (temp\_pass.GetPassword() ==

"NOOOOOOOPASSWORDERRORERRORERROR")

{

cout << "Вы не ввели пароль, попробуйте еще раз" << endl;

GoToXY(x, y++);

temp\_pass.Password();

}

if (temp.GetPassword() == temp\_pass.GetPassword())

{

p = 1;

GoToXY(x, y++);

SetColor(LightCyan, Black);

cout << "Учетная запись создана";

SetColor(LightMagenta, Black);

GoToXY(x, y++);

ofstream f2;

f2.open("user.txt", ios::app);

if (!f2.is\_open())

{

cout << "Ошибка открытия файла" << endl;

return;

}

else

{

f2 << temp;

}

f2.close();

Продолжение приложения А

return;

}

else if (k == 1)

{

GoToXY(x, y++);

cout << "Это последняя попытка ввода пароля.";

k++;

}

else if (k == 2)

{

GoToXY(x, y++);

cout << "Неверный пароль.";

k++;

}

else

{

GoToXY(x, y++);

cout << "Неверный пароль. Попробуйте еще раз";

k++;

}

}

}

else

{

if(x!=0&&y!=0)

GoToXY(x, y++);

cout << "Учетная запись с таким именем уже существует";

p = 1;

}

if (p == 0)

{

y++;

GoToXY(x, y++);

cout << ".............................................";

GoToXY(x, y++);

cout << "Вы 3 раза ввели неправильный пароль.";

GoToXY(x, y++);

cout << "Начните регистрацию заново.";

\_getch;

}

return;

}

void AdminLoginMenu()

{

ConsoleCursorVisible(false, 100);

User temp;

string log;

bool st = 0, l = 0, p = 0;

short y = 12, x = 48;

int k = 0;

temp.SetStatus(1);

GoToXY(x, y++);

SetColor(LightCyan, Black);

cout << "Введите логин: ";

SetColor(LightMagenta, Black);

Продолжение приложения А

log = CinStringSymAngErrorCheck(x, y);

temp.SetLogin(log);

int size = LengthFile("user.txt");

vector<User> vec\_users;

ifstream f;

try

{

f.open("user.txt");

if (f.is\_open())

{

User temp1;

for (int i = 0; i < size; i++)

{

f >> temp1;

vec\_users.push\_back(temp1);

}

f.close();

}

else

throw("Файл не найден");

}

catch (const char\* message) {

cout << "Вызвано исключение." << message << endl;

exit(0);

}

for (int i = 0; i < size; i++)

{

if (vec\_users[i].GetStatus() == temp.GetStatus())

{

st = 1;

if (vec\_users[i].GetLogin() == temp.GetLogin())

{

l = 1;

GoToXY(x, y++);

cout << "Учетная запись найдена.";

GoToXY(x, y++);

SetColor(LightCyan, Black);

cout << "Введите пароль: ";

SetColor(LightMagenta, Black);

while (k != 3)

{

GoToXY(x, y++);

temp.Password();

while (temp.GetPassword() ==

"NOOOOOOOPASSWORDERRORERRORERROR")

{

cout << "Вы не ввели пароль, попробуйте еще раз" << endl;

GoToXY(x, y++);

temp.Password();

}

if (vec\_users[i].GetPassword() == temp.GetPassword())

{

temp.SetName(vec\_users[i].GetName());

Продолжение приложения А

temp.SetSurname(vec\_users[i].GetSurname());

p = 1;

GoToXY(x, y++);

SetColor(LightCyan, Black);

cout << "Вход выполнен.";

SetColor(LightMagenta, Black);

\_getch();

system("CLS");

AdminMenu(temp);

return;

}

else if (k == 1)

{

GoToXY(x, y++);

cout << "Это последняя попытка ввода пароля.";

k++;

}

else if (k == 2)

{

GoToXY(x, y++);

cout << "Неверный пароль.";

k++;

}

else

{

GoToXY(x, y++);

cout << "Неверный пароль. Попробуйте еще раз";

k++;

}

}

}

}

}

if (st == 0 || l == 0){

GoToXY(x, y++);

cout << "Учетной записи с таким именем не найдено";

\_getch();

system("CLS");

}

else if (p == 0){

y++;

GoToXY(x, y++);

cout << ".............................................";

GoToXY(x, y++);

cout << "Вы 3 раза ввели неправильный пароль.";

GoToXY(x, y++);

cout << "Подождите 3 минуты перед следующей попыткой...";

Sleep(7000);

\_getch();

system("CLS");

}

return;

}