

**КУРСОВОЙ ПРОЕКТ**

**по дисциплине: МДК 01.01 Разработка программных модулей**

**Тема: Разработка системы классов для базы данных обменного пункта.**

**Специальность 09.02.07 «Информационные системы и программирование»**

|  |  |  |
| --- | --- | --- |
| **Выполнил студент(ка) группы 309ИС-22** | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | **П.Д. Донских** |
| **Руководитель** | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | **Л.Б. Гусятинер** |

**Москва 2024**



**УТВЕРЖДАЮ**

**Зам. директора КМПО**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_С.Ф. Гасанов**

**«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.**

**ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ**

**по дисциплине: МДК 01.01 Разработка программных модулей**

**Специальность 09.02.07 «Информационные системы и программирование»**

**Студент группы 309ИС-22 Павел Донских**

**ТЕМА: «Разработка системы классов для базы данных обменного пункта»**

Дата выдачи задания «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_ 2024 г.

Срок сдачи проекта «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2024 г.

**Москва 2024**

**ОТЗЫВ**

**научного руководителя на курсовой проект**

**Студента(ки**)

(фамилия, имя и отчество)

**группа**  **курс**  3

**Дисциплина**  МДК 01.01 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ

**Тема работы**

Отмечаются следующие моменты: актуальность темы исследования; степень разработанности проблемы, наиболее интересно исследованные вопросы. Оценивается степень самостоятельности и творчества студента; уровень его теоретической подготовки; умение анализировать научные материалы, делать практические выводы. Знание основных концепций, научной и специальной литературы по избранной теме.

**Научный руководитель**

(Уч. степень, уч. звание, должность, ФИО)

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2024 г

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 5](#_Toc184345412)

[Глава 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 8](#_Toc184345413)

[1.1 Информационное обеспечение задачи 8](#_Toc184345414)

[1.2 Обзор и анализ существующих программных решений 9](#_Toc184345415)

[1.3 Постановка задачи 11](#_Toc184345416)

[Глава 2. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА МОДУЛЯ 12](#_Toc184345417)

[2.1 Выбор инструментов 12](#_Toc184345418)

[2.2 Проектирование сценария 13](#_Toc184345419)

[2.3 Таблицы структур 15](#_Toc184345420)

[2.4 Описание главного модуля 16](#_Toc184345421)

[2.5 Описание модулей 20](#_Toc184345422)

[Глава 3. ТЕСТИРОВАНИЕ ПРОГРАММНОГО МОДУЛЯ 22](#_Toc184345423)

[ЗАКЛЮЧЕНИЕ 25](#_Toc184345424)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 26](#_Toc184345425)

[Приложение 1. Код главного модуля main.c 27](#_Toc184345426)

[Приложение 2. Код модуля функций func.c 31](#_Toc184345427)

[Приложение 3. Код файла прототипов функций func.h 45](#_Toc184345428)

[Приложение 4. Код файла структуры Date date.h 47](#_Toc184345429)

[Приложение 5. Код файла структуры Manager manager.h 48](#_Toc184345430)

[Приложение 6. Код файла структуры Exchange exchange.h 49](#_Toc184345431)

# ВВЕДЕНИЕ

Первая признанная международная система валютного обмена — «Золотой стандарт», установленный в 1944 году на Бреттон-Вудской конференции. Этот стандарт действовал до начала 70-х годов и способствовал созданию Международного Валютного Фонда, который контролировал валютные курсы и предлагал консультации. На Ямайской конференции 1976 года произошли изменения, которые привели к переходу от фиксированных к плавающим обменным курсам, и валютный рынок получил название «Форекс».

Обмен валют играет важную роль в финансовой системе каждой страны, обеспечивая возможность конвертации фиатной валюты как для физических лиц, так и для юридических лиц. С каждым днем интерес к операциям конвертации только растет, что связано с увеличением международной торговли и развитием туризма. Кроме того, с ростом интернет-транзакций значение онлайн-обменников становится все более актуальным.

Использование автоматизированных систем не только повышает общую эффективность, но и обеспечивает высокую точность проводимых обменов. Автоматизация процессов помогает снизить ошибки, возникающие при ручной обработке данных. Внедрение таких систем позволяет обменным пунктам быстрее реагировать на изменения на рынке и предлагать клиентам безопасные способы проведения операций.

Объектом исследования является система, которая включает в себя все процессы, связанные с обменом валют, его анализом и отчетностью.

Предметом исследования является разработка программного модуля, который будет автоматизировать процессы учета обменов и формирования отчетов.

Актуальность разработки обоснована рядом факторов, касающихся современного бизнеса и управления трудозатратами.

1. Увеличение конкуренции на рынке

* В условиях растущей конкуренции предприятия стремятся оптимизировать свои процессы для повышения эффективности и снижения затрат.
* Необходимость адаптации: компании должны адаптироваться к изменениям, чтобы оставаться конкурентоспособными.

1. Автоматизация процессов

* Снижение ручного труда: ручной учет обменов часто приводит к ошибкам, задержкам и увеличению затрат времени.
* Упрощение работы: Автоматизированные системы упрощают работу менеджеров и сотрудников, позволяя им сосредоточиться на более важных задачах.

1. Повышение точности учета

* Точность данных: автоматизированные системы обеспечивают более точный учет обменов, что способствует правильному расчету финансовых показателей.
* Снижение ошибок: исключение человеческого фактора уменьшает вероятность ошибок в данных, что критически важно для финансовой отчетности.

1. Отчетность

* Автоматизированные системы могут генерировать отчеты, необходимые для соблюдения законодательных требований, что упрощает процесс отчетности.

Цель работы: создание функционального программного модуля, который будет обеспечивать:

Упрощение процесса записи истории обменов в файл

Представление инструментов для генерации отчетов и анализа обменов за текущий год.

Для достижения цели необходимо:

* Проанализировать предметную область;
* Проанализировать готовое решение;
* Подготовить техническое задание;
* Обосновать выбор инструментов и средств разработки;
* Подготовить план тестирования;
* Описать реализацию технического задания;
* Выполнить тестирование.

**Глава 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

* 1. Информационное обеспечение задачи

Обменный пункт — это учреждение предназначено для предоставления услуг по обмену валют. Основные клиенты обменного пункта — это туристы, бизнесмены и местные жители, нуждающиеся в наличной валюте для повседневных нужд. Назначение разрабатываемого программного продукта заключается в автоматизации процессов учета операций обмена валют, управления данными о проведенных сделках и формирования отчетности. Программное обеспечение должно обеспечить эффективное хранение данных о проведенных операциях, включая даты обмена, суммы и валютные пары, а также возможность генерации отчетов по месяцам и за текущий год.

Основные функции обменного пункты:

* Установление курсов валют: курсы должны обновляться в реальном времени на основе рыночных данных;
* Обработка заявок на обмен: система должна обеспечивать возможность ввода данных о сделках;
* Учет проведенных операций: все операции должны фиксироваться для дальнейшего анализа;
* Генерация отчетов: система должна формировать отчеты по сделкам за определенные периоды.

Основные термины:

1. Валюта — денежные знаки, используемые в разных странах (например, $ — доллар, ₽ — рубль);
2. Валютная пара – это финансовый инструмент, состоящий из 2-х разных валют. Обозначается краткими кодами валют через дробь, например EUR/USD – это валютная пара евро/американский доллар. Или USD/JPY – валютная пара американский доллар/японская йена;
3. Курс валюты — цена одной валюты, выраженная в другой валюте, устанавливаемая на основе рыночного спроса и предложения;
4. Обменный курс — курс, по которому происходит обмен валют (включает комиссию обменного пункта);
5. Базовая валюта — валюта, которую клиент хочет получить;
6. Котируемая валюта — валюта, за которую клиент получает базовую валюту;
7. Спред — разница цены покупки и продажи;
8. ASK — цена покупки;
9. BID — цена продажи;
10. Транзакция — любая сделка или операция, для совершения которой используется банковский счет, при этом баланс на нем меняется в меньшую или большую сторону.
    1. Обзор и анализ существующих программных решений

Система "Currency Tracker" предоставляет пользователям возможность отслеживать количество операций обмена валюты за заданный период. Функционал включает в себя создание отчетов, которые отображают объем сделок по каждой валютной.

Минусы системы:

* Ограниченная функциональность: может не поддерживать интеграцию с другими системами, что затрудняет обмен данными.
* Сложность в использовании: интерфейс, может быть, неинтуитивным для новых пользователей, что требует времени на обучение.
* Ограниченные возможности кастомизации отчетов, что может не удовлетворять специфические потребности бизнеса.

Программное обеспечение "Forex Report Generator" предлагает инструменты для создания детализированных отчетов о сделках на рынке Forex. Пользователи могут анализировать количество сделок и прибыль по каждой валютной паре за выбранные временные интервалы.

Минусы системы:

* Высокая стоимость лицензии, что может быть неприемлемо для небольших обменных пунктов.
* Возможные проблемы с производительностью при обработке больших объемов данных, что может замедлить генерацию отчетов.

Система "Trade Analyzer" предлагает функционал для создания отчетов о сделках за указанный период времени. Пользователи могут просматривать количество операций и анализировать прибыль по каждой валютной паре.

Минусы системы:

* Сложность в интеграции с другими финансовыми системами, что может затруднить работу пользователей.
* Отсутствие поддержки нескольких языков, что ограничивает использование программы в международной среде.
* Возможные проблемы с безопасностью данных, если система не имеет надежных мер защиты информации.

Учитывая все вышеперечисленные недостатки, становится очевидным, что для повышения эффективности работы обменных пунктов необходимо искать решения, которые обеспечат более высокую степень интеграции, удобство использования и гибкость в отчетности. Это позволит не только оптимизировать внутренние процессы, но и улучшить качество обслуживания клиентов.

* 1. Постановка задачи

Основные задачи и функции.

1. Ввод, удаление, редактирование данных о проведенных операциях обмена и менеджерах;
2. Учет количества сделок по каждой валютной паре;
3. Формирование отчетов о количестве операций и суммах обмененных денег за определенные периоды;

Возникает необходимость в компьютерной обработке вводимых данных о проведенных операциях с целью хранения этой информации на носителях и анализа эффективности работы обменного пункта.

Структура входной и выходной информации

Структура входной информации:

* Данные о менеджерах: номер менеджера, ФИО менеджера;
* Данные об обменах: номер обмена, дата обмена, номер менеджера, валюта, сумма в рублях.

Структура выходной информации:

* Готовый документ: текущая дата, количество обменов за каждый месяц, сумма обмененных денег по месяцам за текущий год, место для подписи.

# Глава 2. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА МОДУЛЯ

* 1. Выбор инструментов

При выборе инструмента и языка программирования было проведено сравнение по критериям, которые представлены в таблицах 1 и 3.

Степень важности критерия выбиралась из: низкая, ниже средней, средняя, ниже высокой, высокая.

|  |  |
| --- | --- |
| Таблица 1. Критерии выбора инструмента. | |
| Критерий | Важность критерия |
| Функционал | Ниже высокой |
| Удобство использования | Высокая |
| Скорость разработки | Ниже высокой |

Исходя из данных критериев, я сравнил 2 языка программирования от 0 до 10 баллов за критерий

|  |  |  |
| --- | --- | --- |
| Таблица 2. Оценка языков программирования. | | |
| Критерий/Язык программирования | Python | C |
| Функционал | 10 | 8 |
| Удобство использования | 7 | 9 |
| Скорость разработки | 7 | 7 |
| Итого | 24 | 24 |

|  |  |
| --- | --- |
| Таблица 3. Критерии выбора среды разработки. | |
| Критерий | Важность критерия |
| Простота | Средняя |
| Функционал | Высокая |
| Удобство использования | Высокая |
| Документация на русском языке | Ниже средней |

Исходя из данных критериев, я сравнил 2 среды разработки от 0 до 10 баллов за критерий.

|  |  |  |
| --- | --- | --- |
| Таблица 4. Оценка сред разработки | | |
| Критерий/среда разработки | CodeBlocks | Microsoft Visual Studio |
| Простота | 7 | 5 |
| Функционал | 10 | 10 |
| Удобство использования | 8 | 6 |
| Документация на русском языке | 6 | 9 |
| Итого | 31 | 30 |

По результатам сравнения была выбрана среда разработки CodeBlocks

CodeBlocks — это интегрированная среда разработки (IDE) с открытым исходным кодом, предназначенная для программирования на языках C, C++ и Fortran. Она предоставляет пользователям удобный интерфейс для написания, компиляции и отладки кода. CodeBlocks поддерживает различные компиляторы и может быть расширена с помощью плагинов, что делает её гибким инструментом для разработчиков.

* 1. Проектирование сценария

В данном разделе приведён пример сценария использования программы сотрудником (см. рисунок 1).

После запуска программы сотрудник должен загрузить файлы менеджеров, операций обменов. Затем он может редактировать данные в файлах. После этих действий программа выдаст отчет об обменах.

Меню функций

Запуск программы

Загрузка файлов (менеджеры, история операций)

Управление менеджерами

Удаление менеджеров

Записать данные в файл обменов

Генерация отчета

Завершение программы

Добавление менеджеров

Редактировать файл обменов

Рисунок 1. Сценарий использования

* 1. Таблицы структур

В таблицах структур (см. таблицы 5, 6) представлены две структуры, manager (для хранения информации о менеджере), exchange (для представления данных обменов).

В правом блоке, расположен блок с операциями (функциями, где используются структуры).

|  |  |  |
| --- | --- | --- |
| Таблица 5. Структуры Exchange | |  |
| Элементы | Функции | Описание функции |
| int exchange\_id — идентификатор продажи. | int load\_exchanges(char \*fname, Exchange exchanges[], int limit); | Загружает данные о обменах из файла и сохраняет их в массиве exchanges, возвращая количество загруженных обменов. |
| struct Date date — дата продажи (содержит поля day, month, year). | void exchanges\_print(Exchange exchanges[], int limit); | Печатает список обменов, используя данные из массива exchanges. |
| int manager\_id — идентификатор менеджера. | void save\_new\_exchange(char \*fname); | Сохраняет новый обмен в файл, запрашивая у пользователя необходимые данные. |
| char currency — валюта. | void edit\_exchange(char \*fname); | Позволяет редактировать существующий обмен, запрашивая у пользователя новый ввод данных. |
| int price — цена обмена. | void do\_query1(Exchange exchanges[], int limit, const char \*fname); | Выполняет запрос по обменам, анализируя данные и записывая результаты в файл. |
|  | void exchange\_to\_s(char s[100], Exchange exchange); | Преобразует объект Exchange в строку, включая дату, и записывает результат в переданную строку. |

|  |  |  |
| --- | --- | --- |
| Таблица 6. Структуры Manager | |  |
| Элементы | Функции | Описание функции |
| int manager\_id — идентификатор продажи. | int load\_managers(char \*fname, Manager managers[], int limit); | Загружает данные о менеджерах из файла и сохраняет их в массиве managers, возвращая количество загруженных менеджеров. |
| struct Date date — дата продажи (содержит поля day, month, year). | void managers\_print(Manager managers[], int limit); | Печатает список менеджеров, используя данные из массива managers. |
| int manager\_id — идентификатор менеджера. | void manage\_managers(Manager managers[], int \*count); | Позволяет управлять менеджерами, предлагая пользователю добавить или удалить менеджера. |
| char currency — валюта. | void add\_manager(Manager managers[], int \*count); | Добавляет нового менеджера в массив managers, запрашивая у пользователя необходимые данные. |
| int price — цена продажи. | void remove\_manager(Manager managers[], int \*count); | Удаляет менеджера из массива managers по указанному ID, сдвигая оставшиеся элементы. |

* 1. Описание главного модуля

Главный модуль представляет собой main.c

К главному модулю подключаются остальные модули, содержащие в себе структуры и функции.

Листинг 1. Код главного модуля

// main.c

#include <stdio.h>

#include <locale.h>

#include <stdlib.h>

#include <Windows.h>

#include "func.h"

#include "exchange.h"

#include "manager.h"

#define AR\_MAN\_LEN 100

#define AR\_EXCHANGE\_LEN 1000

#define TRUE 1

int main() {

setlocale(LC\_ALL, "");

int man\_count = 0; // Количество менеджеров

int exchange\_count = 0; // Количество обменов

Exchange exchanges[AR\_EXCHANGE\_LEN]; // Массив для хранения обменов

Manager managers[AR\_MAN\_LEN]; // Массив для хранения менеджеров

int item = -1; // Переменная для выбора пункта меню

while (item != 0) {

show\_menu(); // Вывод меню

printf("Выберите действие: "); // Добавлено сообщение после вывода меню

scanf("%d", &item); // Ввод выбора пользователя

system("cls"); // Очистка консоли после выбора

switch (item) {

case 0:

printf("Спасибо за использование нашей разработки\n\n");

break;

case 1:

printf("309ИС-22 Донских П. Д.\n");

printf("Тема курсового проекта: Обменный пункт валют\n\n");

break;

case 2:

save\_new\_exchange("exchanges.txt"); // Записать данные в файл обмены

break;

case 3:

man\_count = load\_managers("managers.txt", managers, AR\_MAN\_LEN); // Загрузить менеджеров

printf("Файл менеджеров загружен в память\n\n");

break;

case 4:

exchange\_count = load\_exchanges("exchanges.txt", exchanges, AR\_EXCHANGE\_LEN); // Загрузить обмены

printf("Файл обменов загружен в память\n\n");

break;

case 5:

managers\_print(managers, man\_count); // Печать списка менеджеров

break;

case 6:

exchanges\_print(exchanges, exchange\_count); // Печать списка обмена

break;

case 7:

do\_query1(exchanges, exchange\_count, "query1.txt"); // Отчет

printf("Результат сохранен в файле %s\n\n", "query1.txt");

break;

case 8:

edit\_exchange("exchanges.txt"); // Изменено на edit\_exchange

break;

case 9:

manage\_managers(managers, &man\_count); // Управление менеджерами

break;

default:

printf("Ошибка. Пункты от 0 до 9\n\n");

break;

}

}

return 0;

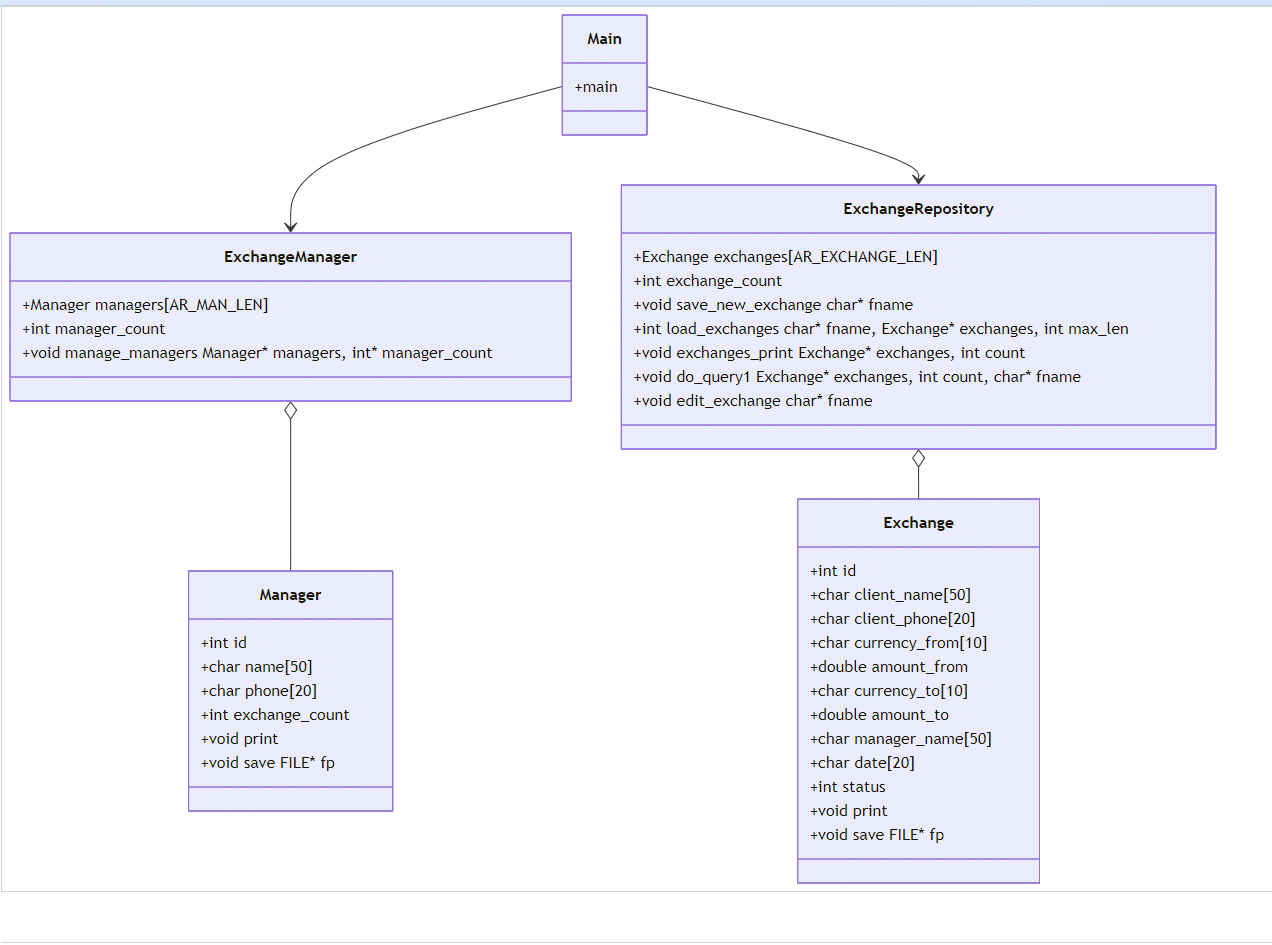
}

Рисунок 2. Блок-схема главного модуля

* 1. Описание модулей

В данной главе описаны используемые модули.

1. Главный модуль

Главный модуль представляет собой файл main.c. К главному модулю подключаются остальные модули, содержащие в себе структуры и функции.

1. Модуль функций

В модуле функций находятся функции, используемые в проекте:

1. void exchange\_to\_s(char s[100], Exchange exchange) — преобразует объект Exchange в строку, включая дату, и записывает результат в переданную строку.
2. void manager\_to\_s(char s[150], Manager manager) — преобразует объект Manager в строку и записывает результат в переданную строку.
3. void show\_menu() — отображает меню с доступными действиями для пользователя.
4. int check\_exchange\_id\_exists(char \*fname, int exchange\_id) —проверяет, существует ли ID обмена в указанном файле.
5. void input\_exchange(Exchange \*exchange, char \*fname) —запрашивает у пользователя данные для нового обмена и заполняет объект Exchange.
6. void save\_new\_exchange(char \*fname) — сохраняет новый обмен в файл, запрашивая у пользователя необходимые данные.
7. void edit\_exchange(char \*fname) — позволяет редактировать существующий обмен, запрашивая у пользователя новый ввод данных.
8. void exchanges\_print(Exchange exchanges[], int limit) — печатает список обменов, используя данные из массива exchanges.
9. int load\_exchanges(char \*fname, Exchange exchanges[], int limit) —загружает данные о обменах из файла и сохраняет их в массиве exchanges, возвращая количество загруженных обменов.
10. int load\_managers(char \*fname, Manager managers[], int limit) — загружает данные о менеджерах из файла и сохраняет их в массиве managers, возвращая количество загруженных менеджеров.
11. void managers\_print(Manager managers[], int limit) — печатает список менеджеров, используя данные из массива managers.
12. void do\_query1(Exchange exchanges[], int limit, const char \*fname) — выполняет запрос по обменам, анализируя данные и записывая результаты в файл.
13. void manage\_managers(Manager managers[], int \*count) —Позволяет управлять менеджерами, предлагая пользователю добавить или удалить менеджера.
14. void add\_manager(Manager managers[], int \*count) —добавляет нового менеджера в массив managers, запрашивая у пользователя необходимые данные.
15. void remove\_manager(Manager managers[], int \*count) — yдаляет менеджера из массива managers по указанному ID, сдвигая оставшиеся элементы.

# Глава 3. ТЕСТИРОВАНИЕ ПРОГРАММНОГО МОДУЛЯ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Таблица 5. План тестирования для программного модуля обменного-пункта | | | | | |
| **ID теста** | **Описание теста (тип)** | **Предусловия** | **Шаги для воспроизведения** | **Ожидаемый результат** | **Фактический результат** |
| **1** | Ввод данных об обмене (позитивный). | Пользователь находится в меню ввода данных обмена. | 1. Выбрать действие 2.  2. Ввести данные в поля «id обмена», «дата обмена», «id менеджера», «валюта», «сумма обмена».  3. Нажать кнопку «Enter» на клавиатуре. | Вывод на экран:  «данные успешно сохранены в файл». | Вывод на экран:  «данные успешно сохранены в файл». |
| **2** | Ввод данных об обмене (негативный). | Пользователь находится в меню ввода данных обмена. | 1. Выбрать действие 2.  2. Ввести некорректные данные в поле «дата обмена».  3. Нажать кнопку «Enter» на клавиатуре. | Вывод на экран:  «ошибка ввода даты»; «введите дату заново». | Вывод на экран:  «ошибка ввода даты»; «введите дату заново». |
| **3** | Ввод данных об обмене (негативный). | Пользователь находится в меню ввода данных обмена. | 1. Выбрать действие 2.  2. Ввести id обмена, который уже существует в файле.  3. Нажать кнопку «Enter» на клавиатуре. | Вывод на экран:  «обмен с таким ID уже существует», «введите ID заново». | Вывод на экран:  «обмен с таким ID уже существует», «введите ID заново». |
| **4** | Ввод данных об обмене (позитивный). | Пользователь находится в меню ввода данных обмена. | 1. Выбрать действие 2.  2. Ввести корректный id менеджера.  3. Нажать кнопку «Enter» на клавиатуре. | Продолжение ввода данных об обмене. | Продолжение ввода данных об обмене. |
| **5** | Ввод данных об обмене (негативный). | Пользователь находится в меню ввода данных обмена. | 1. Выбрать действие 2.  2. Ввести некорректное обозначение валюты.  3. Нажать кнопку «Enter» на клавиатуре. | Вывод на экран:  «неверный ввод валюты»;  «введите заново». | Вывод на экран:  «неверный ввод валюты»;  «введите заново». |
| **6** | Ввод данных об обмене (негативный). | Пользователь находится в меню ввода данных обмена. | 1. Выбрать действие 2.  2. Ввести некорректное значение суммы обмена  3. Нажать кнопку «Enter» на клавиатуре. | Вывод на экран:  «ошибка ввода»; «введите заново». | Вывод на экран:  «ошибка ввода»; «введите заново». |
| **7** | Загрузка данных из файла обмена (позитивный). | Пользователь находится в основном меню. | 1. Выбрать действие 4.  2. Нажать кнопку «Enter» на клавиатуре. | Вывод на экран: «файл обмена загружен в память». | Вывод на экран: «файл обмена загружен в память». |
| **8** | Загрузка данных их файла менеджеров (позитивный). | Пользователь находится в основном меню. | 1. Выбрать действие 3.  2. Нажать кнопку «Enter» на клавиатуре. | Вывод на экран: «файл менеджеров загружен в память». | Вывод на экран: «файл менеджеров загружен в память». |
| **9** | Редактирование данных об обмене (негативный) | Пользователь находится в основном меню. | 1. Выбрать действие 8.  2. Ввести некорректные данные в поле «дата продажи».  3. Нажать кнопку «Enter» на клавиатуре. | Вывод на экран:  «ошибка ввода даты»; «введите дату заново». | Вывод на экран:  «ошибка ввода даты»; «введите дату заново». |
| **10** | Удаление данных менеджеров (негативный) | Пользователь находится в меню редактирования данных о менеджере. | 1. Выбрать действие 9.  2. Выбрать 2 (удалить менеджера)  3. Ввести некорректный id менеджера.  3. Нажать кнопку «Enter» на клавиатуре. | Вывод на экран:  «менеджер не найден». | Вывод на экран:  «менеджер не найден». |
| **11** | Редактирование данных об обмене (позитивный) | Пользователь находится в основном меню. | 1. Выбрать действие 8.  2. Ввести данные в поля «id продажи», «дата продажи», «id менеджера», «валюта», «сумма обмена».  3. Нажать кнопку «Enter» на клавиатуре. | Вывод на экран:  «файл успешно отредактирован». | Вывод на экран:  «файл успешно отредактирован». |
| **12** | Добавление менеджеров (позитивный) | Пользователь находится в меню редактирования данных о менеджере. | 1. Выбрать действие 9.  2. Выбрать действие 1  2. Ввести корректное данные: id\_менеджера, ФИО.  3. Нажать кнопку «Enter» на клавиатуре. | Вывод на экран:  «менеджер успешно добавлен». | Вывод на экран:  «менеджер успешно добавлен». |
| **13** | Печать списка менеджеров (позитивный) | Пользователь находится в меню редактирования данных о менеджере. | 1. Выбрать действие 5.  2. Нажать кнопку «Enter» на клавиатуре. | Вывод на экран: id и ФИО менеджеров. | Вывод на экран: id и ФИО менеджеров. |
| **14** | Печать списка обмена (позитивный) | Пользователь находится в основном меню. | 1. Выбрать действие 6.  2. Нажать кнопку «Enter» на клавиатуре. | Вывод на экран: «id обмена», «дата обмена», «id менеджера», «валюта»,  «сумма обмена». | Вывод на экран: «id обмена»,  «дата обмена»,  «id менеджера», «валюта»,  «сумма обмена». |
| **15** | Выполнение запроса по обменам | Пользователь находится в основном меню. | 1. Выбрать действие 7.  2. Нажать кнопку «Enter» на клавиатуре. | 1. Вывод на экран:  «Результат сохранен в файле query1.txt» 2. Сохранение результата в файле query1.txt. | 1. Вывод на экран:  «Результат сохранен в файле query1.txt» 2. Сохранение результата в файле query1.txt. |

# ЗАКЛЮЧЕНИЕ

В результате выполнения курсового проекта была написана программа "CurrencyReport" для автоматизации процесса отчетности обменного пункта.

В ходе работы были проанализированы предметная область, существующие разработки, посвященные данному направлению, получены практические навыки с работой в Microsoft Visual Studio.

Так же планируется продолжить работу над данным проектом с целью расширения возможностей и удобства для пользователей. Планы по доработке представлены ниже.

1. Реализация методов и функций, учитывающих количество оставшихся в наличии валют;
2. Добавить реализацию функций для управления обменами;
3. Обновление курса валют в режиме реального времени;
4. Создание и последующая доработка интерфейса.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Баранов, И. В. (2020). Финансовые рынки и их инструменты. Москва: Издательство "Финансы и статистика".
2. Кузнецов, А. П. (2019). Обменные пункты: организация и управление. Санкт-Петербург: Издательство "Экономика".
3. Сидоров, В. Н. (2021). Анализ валютного рынка. Москва: Издательство "Наука".
4. Петрова, Е. А. (2022). Правовые аспекты деятельности обменных пунктов. Журнал "Финансовое право", 3(2), 45-50.
5. Официальный сайт Центрального банка Российской Федерации. (2023). Курсы валют.

# Приложение 1. Код главного модуля main.c

// main.c

#include <stdio.h>

#include <locale.h>

#include <stdlib.h>

#include <Windows.h>

#include "func.h"

#include "exchange.h"

#include "manager.h"

#define AR\_MAN\_LEN 100

#define AR\_EXCHANGE\_LEN 1000

#define TRUE 1

int main() {

setlocale(LC\_ALL, "");

int man\_count = 0; // Количество менеджеров

int exchange\_count = 0; // Количество обменов

Exchange exchanges[AR\_EXCHANGE\_LEN]; // Массив для хранения обменов

Manager managers[AR\_MAN\_LEN]; // Массив для хранения менеджеров

int item = -1; // Переменная для выбора пункта меню

while (item != 0) {

show\_menu(); // Вывод меню

printf("Выберите действие: "); // Добавлено сообщение после вывода меню

scanf("%d", &item); // Ввод выбора пользователя

system("cls"); // Очистка консоли после выбора

switch (item) {

case 0:

printf("Спасибо за использование нашей разработки\n\n");

break;

case 1:

printf("309ИС-22 Донских П. Д.\n");

printf("Тема курсового проекта: Обменный пункт валют\n\n");

break;

case 2:

save\_new\_exchange("exchanges.txt"); // Записать данные в файл обмены

break;

case 3:

man\_count = load\_managers("managers.txt", managers, AR\_MAN\_LEN); // Загрузить менеджеров

printf("Файл менеджеров загружен в память\n\n");

break;

case 4:

exchange\_count = load\_exchanges("exchanges.txt", exchanges, AR\_EXCHANGE\_LEN); // Загрузить обмены

printf("Файл обменов загружен в память\n\n");

break;

case 5:

managers\_print(managers, man\_count); // Печать списка менеджеров

break;

case 6:

exchanges\_print(exchanges, exchange\_count); // Печать списка обмена

break;

case 7:

do\_query1(exchanges, exchange\_count, "query1.txt"); // Отчет

printf("Результат сохранен в файле %s\n\n", "query1.txt");

break;

case 8:

edit\_exchange("exchanges.txt"); // Изменено на edit\_exchange

break;

case 9:

manage\_managers(managers, &man\_count); // Управление менеджерами

break;

default:

printf("Ошибка. Пункты от 0 до 9\n\n");

break;

}

}

return 0;

}

# Приложение 2. Код модуля функций func.c

// func.c

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <time.h>

#include <locale.h>

#include "func.h"

#include "exchange.h"

#include "manager.h"

#define MONTH\_NAMES\_COUNT 12

#define AR\_MAN\_LEN 100

const char \*month\_names[MONTH\_NAMES\_COUNT] = {

"января", "февраля", "марта", "апреля", "мая", "июня",

"июля", "августа", "сентября", "октября", "ноября", "декабря"

};

// Получение текущей даты в строковом формате

void get\_current\_date(char \*date\_str) {

time\_t t = time(NULL);

struct tm tm = \*localtime(&t);

sprintf(date\_str, "%d %s %d", tm.tm\_mday, month\_names[tm.tm\_mon], tm.tm\_year + 1900);

}

// Преобразование обмена в строку

void exchange\_to\_s(char s[100], Exchange exchange) {

char format[] = "%d. %02d.%02d.%04d %d %s %d";

sprintf(s, format, exchange.exchange\_id, exchange.date.day, exchange.date.month, exchange.date.year, exchange.manager\_id, exchange.currency, exchange.price);

}

// Преобразование менеджера в строку

void manager\_to\_s(char s[150], Manager manager) {

snprintf(s, 150, "%d. %s %s %s", manager.manager\_id, manager.manager\_surname, manager.manager\_name, manager.manager\_patronymic);

}

// Меню

void show\_menu() {

printf(

"0. Выход\n"

"1. О программе\n"

"2. Записать данные в файл обменов\n"

"3. Загрузка из файла менеджеров\n"

"4. Загрузка из файла обменов\n"

"5. Печать списка менеджеров\n"

"6. Печать списка обменов\n"

"7. Выполнение запроса по обменам\n"

"8. Редактировать обмен\n"

"9. Управление менеджерами\n\n");

}

// Проверка существования ID обмена в файле

int check\_exchange\_id\_exists(char \*fname, int exchange\_id) {

FILE \*in = fopen(fname, "r");

if (in == NULL) {

perror("Ошибка открытия файла для чтения");

return 0;

}

Exchange rec; // Запись обмена

while (fscanf(in, "%d %d.%d.%d %d %s %d", &rec.exchange\_id, &rec.date.day, &rec.date.month, &rec.date.year, &rec.manager\_id, rec.currency, &rec.price) == 7) {

if (rec.exchange\_id == exchange\_id) {

fclose(in);

return 1; // ID обмена существует

}

}

fclose(in);

return 0; // ID обмена не существует

}

// Ввод данных о обмене

void input\_exchange(Exchange \*exchange, char \*fname) {

printf("Введите ID обмена: ");

while (scanf("%d", &exchange->exchange\_id) != 1 || check\_exchange\_id\_exists(fname, exchange->exchange\_id)) {

if (check\_exchange\_id\_exists(fname, exchange->exchange\_id)) {

printf("Ошибка: ID обмена %d уже существует. Пожалуйста, введите уникальный ID: ", exchange->exchange\_id);

} else {

printf("Ошибка ввода. Пожалуйста, введите целое число для ID обмена: ");

}

while(getchar() != '\n');

}

printf("Введите дату обмена (дд.мм.гггг): ");

while (scanf("%d.%d.%d", &exchange->date.day, &exchange->date.month, &exchange->date.year) != 3 ||

exchange->date.day < 1 || exchange->date.day > 31 ||

exchange->date.month < 1 || exchange->date.month > 12 ||

(exchange->date.month == 2 && exchange->date.day > 29) || // Проверка на февраль

(exchange->date.month == 4 && exchange->date.day > 30) || // Проверка на апрель

(exchange->date.month == 6 && exchange->date.day > 30) || // Проверка на июнь

(exchange->date.month == 9 && exchange->date.day > 30) || // Проверка на сентябрь

(exchange->date.month == 11 && exchange->date.day > 30)) { // Проверка на ноябрь

printf("Ошибка ввода. Пожалуйста, введите дату в формате дд.мм.гггг: ");

while(getchar() != '\n');

}

printf("Введите ID менеджера (1-2): ");

while (scanf("%d", &exchange->manager\_id) != 1 || (exchange->manager\_id < 1 || exchange->manager\_id > 2)) {

printf("Ошибка ввода. Пожалуйста, введите ID менеджера (1-2): ");

while(getchar() != '\n');

}

printf("Введите валюту (USD, EURO): ");

while (scanf("%s", exchange->currency) != 1 || (strcmp(exchange->currency, "USD") != 0 && strcmp(exchange->currency, "EURO") != 0)) {

printf("Ошибка ввода. Пожалуйста, введите валюту (USD, EURO): ");

while(getchar() != '\n');

}

printf("Введите цену: ");

while (scanf("%d", &exchange->price) != 1 || exchange->price < 0) {

printf("Ошибка ввода. Пожалуйста, введите положительное целое число для цены: ");

while(getchar() != '\n');

}

}

// Сохранение нового обмена в файл

void save\_new\_exchange(char \*fname) {

Exchange exchange; // Объект обмена

input\_exchange(&exchange, fname); // Ввод данных о обмене

FILE \*out = fopen(fname, "a");

if (out == NULL) {

perror("Ошибка открытия файла для записи");

return;

}

fprintf(out, "%d %02d.%02d.%04d %d %s %d\n", exchange.exchange\_id, exchange.date.day, exchange.date.month, exchange.date.year, exchange.manager\_id, exchange.currency, exchange.price);

fclose(out);

printf("Обмен добавлен.\n\n");

}

// Функция для редактирования существующего обмена

void edit\_exchange(char \*fname) {

Exchange exchange; // Объект обмена

int exchange\_id;

printf("Введите ID обмена для редактирования: ");

scanf("%d", &exchange\_id);

// Проверка существования ID обмена

if (!check\_exchange\_id\_exists(fname, exchange\_id)) {

printf("Ошибка: ID обмена %d не существует.\n", exchange\_id);

return;

}

// Загрузка существующего обмена

FILE \*in = fopen(fname, "r");

FILE \*temp = fopen("temp.txt", "w"); // Временный файл для хранения обновленных данных

if (in == NULL || temp == NULL) {

perror("Ошибка открытия файла");

return;

}

// Чтение данных из файла и редактирование нужной записи

while (fscanf(in, "%d %d.%d.%d %d %s %d", &exchange.exchange\_id, &exchange.date.day, &exchange.date.month, &exchange.date.year, &exchange.manager\_id, exchange.currency, &exchange.price) == 7) {

if (exchange.exchange\_id == exchange\_id) {

printf("Редактирование обмена с ID %d:\n", exchange\_id);

// Ввод новых данных, но ID остается прежним

printf("Введите новую дату обмена (дд.мм.гггг): ");

while (scanf("%d.%d.%d", &exchange.date.day, &exchange.date.month, &exchange.date.year) != 3 ||

exchange.date.day < 1 || exchange.date.day > 31 ||

exchange.date.month < 1 || exchange.date.month > 12 ||

(exchange.date.month == 2 && exchange.date.day > 29) || // Проверка на февраль

(exchange.date.month == 4 && exchange.date.day > 30) || // Проверка на апрель

(exchange.date.month == 6 && exchange.date.day > 30) || // Проверка на июнь

(exchange.date.month == 9 && exchange.date.day > 30) || // Проверка на сентябрь

(exchange.date.month == 11 && exchange.date.day > 30)) { // Проверка на ноябрь

printf("Ошибка ввода. Пожалуйста, введите дату в формате дд.мм.гггг: ");

while(getchar() != '\n');

}

printf("Введите новую валюту (USD, EURO): ");

while (scanf("%s", exchange.currency) != 1 || (strcmp(exchange.currency, "USD") != 0 && strcmp(exchange.currency, "EURO") != 0)) {

printf("Ошибка ввода. Пожалуйста, введите валюту (USD, EURO): ");

while(getchar() != '\n');

}

printf("Введите новую цену: ");

while (scanf("%d", &exchange.price) != 1 || exchange.price < 0) {

printf("Ошибка ввода. Пожалуйста, введите положительное целое число для цены: ");

while(getchar() != '\n');

}

}

fprintf(temp, "%d %02d.%02d.%04d %d %s %d\n", exchange.exchange\_id, exchange.date.day, exchange.date.month, exchange.date.year, exchange.manager\_id, exchange.currency, exchange.price);

}

fclose(in);

fclose(temp);

remove(fname); // Удаляем старый файл

rename("temp.txt", fname); // Переименовываем временный файл

printf("Обмен с ID %d успешно отредактирован.\n\n", exchange\_id);

}

// Печать списка обменов

void exchanges\_print(Exchange exchanges[], int limit) {

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

char s[150] = "";

exchange\_to\_s(s, exchanges[i]);

puts(s); // Вывод строки

puts("");

}

}

// Загрузка обменов из файла

int load\_exchanges(char \*fname, Exchange exchanges[], int limit) {

FILE \*in = fopen(fname, "r");

if (in == NULL) {

perror("Ошибка открытия файла");

return 0;

}

int count = 0; // Счетчик загруженных обменов

Exchange rec; // Запись обмена

while (count < limit && fscanf(in, "%d %d.%d.%d %d %s %d", &rec.exchange\_id, &rec.date.day, &rec.date.month, &rec.date.year, &rec.manager\_id, rec.currency, &rec.price) == 7) {

exchanges[count] = rec; // Сохранение обмена в массив

count++;

}

fclose(in);

return count; // Возврат количества загруженных обменов

}

// Загрузка менеджеров из файла

int load\_managers(char \*fname, Manager managers[], int limit) {

FILE \*in = fopen(fname, "r");

if (in == NULL) {

perror("Ошибка открытия файла");

return 0;

}

int count = 0; // Счетчик загруженных менеджеров

Manager rec; // Запись менеджера

while (count < limit && fscanf(in, "%d %s %s %s", &rec.manager\_id, rec.manager\_surname, rec.manager\_name, rec.manager\_patronymic) == 4) {

managers[count] = rec; // Сохранение менеджера в массив

count++;

}

fclose(in);

return count; // Возврат количества загруженных менеджеров

}

// Печать списка менеджеров

void managers\_print(Manager managers[], int limit) {

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

char s[150] = "";

manager\_to\_s(s, managers[i]); // Преобразование менеджера в строку

puts(s); // Вывод строки

puts("");

}

}

// Выполнение запроса по обменам

void do\_query1(Exchange exchanges[], int limit, const char \*fname) {

if (limit == 0) { // Проверка на наличие обменов

printf("Нет обменов для обработки запроса.\n");

return;

}

int month\_exchanges[N\_MONTHS + 1] = {0}; // Массив для хранения количества обменов по месяцам

int month\_totals[N\_MONTHS + 1] = {0}; // Массив для хранения сумм по месяцам

// Получение текущего года

time\_t t = time(NULL);

struct tm tm = \*localtime(&t);

int current\_year = tm.tm\_year + 1900; // Текущий год

// Перебор всех обменов и подсчет по месяцам и годам

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

int month = exchanges[i].date.month; // Месяц обмена

int year = exchanges[i].date.year; // Год обмена

// Считаем только обмены текущего года

if (year == current\_year) {

month\_exchanges[month]++; // Увеличиваем количество обменов за месяц

month\_totals[month] += exchanges[i].price; // Суммируем цену по месяцам

}

}

FILE \*out = fopen(fname, "w");

if (out == NULL) {

perror("Ошибка открытия файла");

return;

}

// Шапка документа

fprintf(out, "Отчет по обменам\n");

fprintf(out, "Курс EURO: \n");

fprintf(out, "Курс USD: \n");

fprintf(out, "Дата: ");

char current\_date[100];

get\_current\_date(current\_date); // Получение текущей даты

fprintf(out, "%s\n", current\_date);

fprintf(out, "-----------------------------------\n");

fprintf(out, "Месяц/год Количество\n");

// Вывод количества обменов по месяцам

for (int i = 1; i <= N\_MONTHS; i++) {

if (month\_exchanges[i] != 0) {

fprintf(out, "%02d.%d %d\n", i, current\_year, month\_exchanges[i]); // Используем текущий год

}

}

// Сумма обмененных денег по месяцам для текущего года

fprintf(out, "-----------------------------------\n");

fprintf(out, "Сумма обмененных денег по месяцам в %d году:\n", current\_year);

for (int i = 1; i <= N\_MONTHS; i++) {

if (month\_totals[i] != 0) {

fprintf(out, "%02d.%d: %d\n", i, current\_year, month\_totals[i]);

}

}

// Подпись на последней странице

fprintf(out, "-----------------------------------\n");

fprintf(out, "Подпись: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

fclose(out);

}

// Функция для управления менеджерами

void manage\_managers(Manager managers[], int \*count) {

int action;

printf("1. Добавить менеджера\n");

printf("2. Удалить менеджера\n");

printf("Выберите действие: ");

scanf("%d", &action);

switch (action) {

case 1:

add\_manager(managers, count);

break;

case 2:

remove\_manager(managers, count);

break;

default:

printf("Ошибка: Неверный выбор.\n");

break;

}

}

// Функция для добавления менеджера

void add\_manager(Manager managers[], int \*count) {

if (\*count >= AR\_MAN\_LEN) {

printf("Ошибка: Достигнуто максимальное количество менеджеров.\n");

return;

}

Manager new\_manager;

printf("Введите ID менеджера: ");

scanf("%d", &new\_manager.manager\_id);

// Проверка на уникальность ID

for (int i = 0; i < \*count; i++) {

if (managers[i].manager\_id == new\_manager.manager\_id) {

printf("Ошибка: Менеджер с ID %d уже существует.\n", new\_manager.manager\_id);

return;

}

}

printf("Введите имя менеджера: ");

scanf("%s", new\_manager.manager\_name);

printf("Введите фамилию менеджера: ");

scanf("%s", new\_manager.manager\_surname);

printf("Введите отчество менеджера: ");

scanf("%s", new\_manager.manager\_patronymic);

managers[\*count] = new\_manager; // Сохранение нового менеджера в массив

(\*count)++; // Увеличение счетчика менеджеров

printf("Менеджер добавлен.\n\n");

}

// Функция для удаления менеджера

void remove\_manager(Manager managers[], int \*count) {

int id;

printf("Введите ID менеджера для удаления: ");

scanf("%d", &id);

int found = 0;

for (int i = 0; i < \*count; i++) {

if (managers[i].manager\_id == id) {

found = 1;

// Сдвигаем всех менеджеров после удаляемого на одну позицию влево

for (int j = i; j < \*count - 1; j++) {

managers[j] = managers[j + 1];

}

(\*count)--; // Уменьшаем счетчик менеджеров

printf("Менеджер с ID %d удален.\n\n", id);

break;

}

}

if (!found) {

printf("Ошибка: Менеджер с ID %d не найден.\n\n", id);

}

}

# Приложение 3. Код файла прототипов функций func.h

// func.h

#ifndef FUNC\_H\_INCLUDED

#define FUNC\_H\_INCLUDED

#define N\_MONTHS 12

#include "exchange.h"

#include "manager.h"

// Прототипы функций

void get\_current\_date(char \*date\_str); // Получение текущей даты

void exchange\_to\_s(char s[100], Exchange exchange); // Преобразование обмена в строку

void manager\_to\_s(char s[150], Manager manager); // Преобразование менеджера в строку

void show\_menu(); // Показать меню

void save\_new\_exchange(char \*fname); // Сохранить новый обмен

int load\_managers(char \*fname, Manager managers[], int limit); // Загрузить менеджеров

int load\_exchanges(char \*fname, Exchange exchanges[], int limit); // Загрузить обмены

void exchanges\_print(Exchange exchanges[], int limit); // Печать списка обменов

void managers\_print(Manager managers[], int limit); // Печать списка менеджеров

void do\_query1(Exchange exchanges[], int limit, const char \*fname); // Выполнение запроса по обменам

void edit\_exchange(char \*fname); // Редактировать обмен

void manage\_managers(Manager managers[], int \*count); // Управление менеджерами

void add\_manager(Manager managers[], int \*count); // Добавить менеджера

void remove\_manager(Manager managers[], int \*count); // Удалить менеджера

#endif // FUNC\_H\_INCLUDED

# Приложение 4. Код файла структуры Date date.h

//date.h

#ifndef DATE\_H\_INCLUDED

#define DATE\_H\_INCLUDED

//Структура для хранения даты

typedef struct \_date {

int year; //Год

int month; //Месяц

int day; //День

} Date;

#endif //DATE\_H\_INCLUDED

# Приложение 5. Код файла структуры Manager manager.h

// manager.h

#ifndef MANAGER\_H\_INCLUDED

#define MANAGER\_H\_INCLUDED

// Структура для хранения информации о менеджере

typedef struct \_manager {

int manager\_id; // ID менеджера

char manager\_name[50]; // Имя менеджера

char manager\_surname[50]; // Фамилия менеджера

char manager\_patronymic[50]; // Отчество менеджера

} Manager;

#endif // MANAGER\_H\_INCLUDED

# Приложение 6. Код файла структуры Exchange exchange.h

// exchange.h

#ifndef EXCHANGE\_H\_INCLUDED

#define EXCHANGE\_H\_INCLUDED

#include "date.h"

// Структура для представления обмена

typedef struct \_exchange {

Date date; // Дата обмена

int exchange\_id; // ID обмена

int manager\_id; // ID менеджера, ответственного за обмен

char currency[20]; // Валюта обмена

int price; // Цена обмена

} Exchange;

#endif // EXCHANGE\_H\_INCLUDED