

**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**

**СОДЕРЖАНИЕ**

**Стр.**

**ВВЕДЕНИЕ** 3

**1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

1.1. Информационное обеспечение задачи

1.2. Обзор и анализ существующих программных решений

1.3. Постановка задачи. Структура входной и выходной информации

**2. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА МОДУЛЯ**

2.1. Построение модели системы

2.2. Реализация интерфейса приложения

**3. ОТЛАДКА И ТЕСТИРОВАНИЕ МОДУЛЯ**

3.1. Перехват исключительных ситуаций

3.4. Технология работы с приложением

**ЗАКЛЮЧЕНИЕ**

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

**ПРИЛОЖЕНИЯ**

# Глава 1. Теоретическая часть

# 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.

**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.

//Дописать

* 1. Постановка задачи

Основные задачи и функции.

1. Ввод и редактирование данных о проведенных операциях обмена;
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.

//Дописать

**Глава 2. Проектная часть**

**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. Диаграмма структур

В диаграмме структур (см. рисунок 1) представлены три структуры date (для хранения даты), manager (для хранения информации о менеджере), sale (для представления продажи).

После блока с атрибутами, расположен блок с операциями (функциями структуры).



Рисунок 1. Диаграмма структур

* 1. Описание главного модуля
  2. Описание спецификаций к модулям
  3. Описание модулей

**Глава 3. Тестирование и отладка 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 о продаже (позитивный). | Пользователь находится в меню ввода продажи. | 1. Ввести цифру 2.  2. Ввести данные в поля «id продажи», «дата продажи», «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 о продаже (негативный). | Пользователь находится в меню ввода продажи. | 1. Ввести цифру 2.  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». |
| **3** | Ввод данных о продаже (негативный). | Пользователь находится в меню ввода продажи. | 1. Ввести цифру 2.  2. Ввести id продажи, которая уже существует в файле.  3. Нажать кнопку «Enter» на клавиатуре. | Вывод на экран:  «продажа с таким ID уже существует», «введите ID заново». | Вывод на экран:  «продажа с таким ID уже существует», «введите ID заново». |
| **4** | Ввод данных о продаже (позитивный). | Пользователь находится в меню ввода продажи. | 1. Ввести цифру 2.  2. Ввести корректный id менеджера.  3. Нажать кнопку «Enter» на клавиатуре. | Продолжение ввода данных о продаже. | Продолжение ввода данных о продаже. |
| **5** | Ввод данных о продаже (негативный). | Пользователь находится в меню ввода продажи. | 1. Ввести цифру 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 о продаже (негативный). | Пользователь находится в меню ввода продажи. | 1. Ввести цифру 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 продаж (позитивный). | Пользователь находится в основном меню. | 1. Ввести цифру 4.  2. Нажать кнопку «Enter» на клавиатуре. | Вывод на экран: «файл продаж загружен в память». | Вывод на экран: «файл продаж загружен в память». |
| **8** | Загрузка данных их файла менеджеров (позитивный). | Пользователь находится в основном меню. | 1. Ввести цифру 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 продажи (негативный) | Пользователь находится в меню редактирования продажи. | 1. Ввести цифру 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** | Редактирование продажи (негативный) | Пользователь находится в меню редактирования продажи. | 1. Ввести цифру 8.  2. Ввести некорректный id менеджера.  3. Нажать кнопку «Enter» на клавиатуре. | Вывод на экран:  «неверный ввод валюты»;  «введите заново». | Вывод на экран:  «неверный ввод валюты»;  «введите заново». |
| **11** | Редактирование продажи (позитивный) | Пользователь находится в меню редактирования продажи. | 1. Ввести цифру 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** | Редактирование продажи (позитивный) | Пользователь находится в меню редактирования продажи. | 1. Ввести цифру 8.  2. Ввести корректные данные даты обмена.  3. Нажать кнопку «Enter» на клавиатуре. | Продолжение ввода данных о продаже. | Продолжение ввода данных о продаже. |
| **13** | Печать списка менеджеров (позитивный) | Пользователь находится в основном меню. | 1. Ввести цифру 5.  2. Нажать кнопку «Enter» на клавиатуре. | Вывод на экран: id и ФИО менеджеров. | Вывод на экран: id и ФИО менеджеров. |
| **14** | Печать списка продаж (позитивный) | Пользователь находится в основном меню. | 1. Ввести цифру 6.  2. Нажать кнопку «Enter» на клавиатуре. | Вывод на экран: «id продажи», «дата продажи», «id менеджера», «валюта»,  «сумма обмена». | Вывод на экран: «id продажи»,  «дата продажи»,  «id менеджера», «валюта»,  «сумма обмена». |
| **15** | Выполнение запроса по продажам | Пользователь находится в основном меню. | 1. Ввести цифру 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. Обновление курса валют в режиме реального времени
3. Создание и последующая доработка интерфейса.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

Приложение 1. Код главного модуля main.c

// main.c

#include <stdio.h>

#include <locale.h>

#include <stdlib.h>

#include "func.h"

#include "sale.h"

#include "manager.h"

#define AR\_MAN\_LEN 100

#define AR\_SALES\_LEN 1000

#define TRUE 1

//Прототипы функций

void save\_new\_sale(char \*fname);

int main() {

setlocale(LC\_ALL, "");

int man\_count = 0; //Количество менеджеров

int sales\_count = 0; //Количество продаж

Sale sales[AR\_SALES\_LEN]; //Массив для хранения продаж

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

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

while (item != 0) {

show\_menu(); //Вывод меню

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

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\_sale("sales.txt"); //Сохранить новую продажу

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:

sales\_count = load\_sales("sales.txt", sales, AR\_SALES\_LEN); //Загрузить продажи

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

break;

case 5:

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

break;

case 6:

sales\_print(sales, sales\_count); //Печать списка продаж

break;

case 7:

do\_query1(sales, sales\_count, "query1.txt"); //Выполнение запроса по продажам

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

break;

case 8:

edit\_sale("sales.txt"); //Редактировать продажу

break;

default:

printf("Ошибка. Пункты от 0 до 8\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 "sale.h"

#include "manager.h"

#define MONTH\_NAMES\_COUNT 12

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 продажи в строку

void sale\_to\_s(char s[100], Sale sale) {

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

sprintf(s, format, sale.sale\_id, sale.date.day, sale.date.month, sale.date.year, sale.manager\_id, sale.currency, sale.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 продаж\n"

"3. Загрузка из файла менеджеров\n"

"4. Загрузка из файла продаж\n"

"5. Печать списка менеджеров\n"

"6. Печать списка продаж\n"

"7. Выполнение запроса по продажам\n"

"8. Редактировать продажу\n\n"); //Добавлено новое меню

}

//Проверка существования ID продажи в файле

int check\_sale\_id\_exists(char \*fname, int sale\_id) {

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

if (in == NULL) {

perror("Ошибка открытия файла для чтения");

return 0;

}

Sale rec; //Запись продажи

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

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

fclose(in);

return 1; //ID продажи существует

}

}

fclose(in);

return 0; //ID продажи не существует

}

//Ввод данных о продаже

void input\_sale(Sale \*sale, char \*fname) {

printf("Введите ID продажи: ");

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

if (check\_sale\_id\_exists(fname, sale->sale\_id)) {

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

} else {

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

}

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

}

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

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

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

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

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

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

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

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

(sale->date.month == 11 && sale->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", &sale->manager\_id) != 1 || (sale->manager\_id < 1 || sale->manager\_id > 2)) {

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

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

}

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

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

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

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

}

printf("Введите цену: ");

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

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

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

}

}

//Сохранение новой продажи в файл

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

Sale sale; //Объект продажи

input\_sale(&sale, fname); //Ввод данных о продаже

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", sale.sale\_id, sale.date.day, sale.date.month, sale.date.year, sale.manager\_id, sale.currency, sale.price);

fclose(out);

printf("Продажа добавлена.\n\n");

}

//Функция для редактирования существующей продажи

void edit\_sale(char \*fname) {

Sale sale; //Объект продажи

int sale\_id;

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

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

//Проверка существования ID продажи

if (!check\_sale\_id\_exists(fname, sale\_id)) {

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

return;

}

//Загрузка существующей продажи

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", &sale.sale\_id, &sale.date.day, &sale.date.month, &sale.date.year, &sale.manager\_id, sale.currency, &sale.price) == 7) {

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

printf("Редактирование продажи с ID %d:\n", sale\_id);

//Ввод новых данных, но ID остается прежним

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

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

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

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

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

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

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

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

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

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

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

}

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

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

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

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

}

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

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

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

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

}

printf("Введите новую цену: ");

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

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

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

}

}

//Запись данных в временный файл

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

}

fclose(in);

fclose(temp);

remove(fname); //Удаление старого файла

rename("temp.txt", fname); //Переименование временного файла в оригинальный

printf("Продажа с ID %d успешно отредактирована.\n", sale\_id);

}

//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

}

//Загрузка продаж из файла

int load\_sales(char \*fname, Sale sales[], int limit) {

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

if (in == NULL) {

perror("Ошибка открытия файла");

return 0;

}

int count = 0; //Счетчик загруженных продаж

Sale rec; //Запись продажи

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

sales[count] = rec; //Сохранение продажи в массив

count++;

}

fclose(in);

return count; //Возврат количества загруженных продаж

}

//Печать списка продаж

void sales\_print(Sale sales[], int limit) {

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

char s[150] = "";

sale\_to\_s(s, sales[i]); //Преобразование продажи в строку

puts(s); //Вывод строки

puts("");

}

}

//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 продажам

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

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

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

return;

}

int month\_sales[N\_MONTHS + 1] = {0}; //Массив для хранения количества продаж 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 продаж и подсчет по месяцам и годам

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

int month = sales[i].date.month; //Месяц продажи

int year = sales[i].date.year; //Год продажи

//Считаем только продажи текущего года

if (year == current\_year) {

month\_sales[month]++; //Увеличиваем количество продаж за месяц

month\_totals[month] += sales[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 продажам\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 продаж по месяцам

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

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

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

}

Приложение 3. Код заголовочного файла прототипов функций func.h

// func.h

#ifndef FUNC\_H\_INCLUDED

#define FUNC\_H\_INCLUDED

#define N\_MONTHS 12

#include "sale.h"

#include "manager.h"

//Прототипы функций

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

void sale\_to\_s(char s[100], Sale sale); //Преобразование продажи в строку

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

void show\_menu(); //Показать меню

void save\_new\_sale(char \*fname); //Сохранить новую продажу

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

int load\_sales(char \*fname, Sale sales[], int limit); //Загрузить продажи

void sales\_print(Sale sales[], int limit); //Печать списка продаж

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

void do\_query1(Sale sales[], int limit, const char \*fname); //Выполнение запроса по продажам

void edit\_sale(char \*fname); //Редактировать продажу

#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 Sale sale.h

// sale.h

#ifndef SALE\_H\_INCLUDED

#define SALE\_H\_INCLUDED

#include "date.h"

//Структура для представления продажи

typedef struct \_sale {

Date date; //Дата продажи

int sale\_id; //ID продажи

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

char currency[20]; //Модель автомобиля, который был продан

int price; //Цена продажи

} Sale;

#endif //SALE\_H\_INCLUDED