Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Отчёт

по учебной практике(ознакомительной)

на тему

ТРЕКЕР ФИНАНСОВ

Студент:

гр.358301 Корховой М.Б.

Руководитель: старший преподаватель

каф. ЭВМ Ковальчук А. М.

МИНСК 2024

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc168474814)

[1 ПОСТАНОВКА ЗАДАЧИ 4](#_Toc168474815)

[2 ОБЗОР ЛИТЕРАТУРЫ 5](#_Toc168474816)

[3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВНИЕ 6](#_Toc168474817)

[3.1 Структура данных 6](#_Toc168474818)

[4 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ 7](#_Toc168474819)

[4.1 Разработка схем алгоритмов 7](#_Toc168474820)

[4.2 Разработка алгоритмов 7](#_Toc168474821)

[5 РЕЗУЛЬТАТ РАБОТЫ 10](#_Toc168474822)

[ЗАКЛЮЧЕНИЕ 13](#_Toc168474823)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 14](#_Toc168474824)

[ПРИЛОЖЕНИЕ А 15](#_Toc168474825)

[ПРИЛОЖЕНИЕ Б 16](#_Toc168474826)

[ПРИЛОЖЕНИЕ В 17](#_Toc168474827)

[ПРИЛОЖЕНИЕ Г 18](#_Toc168474828)

[ПРИЛОЖЕНИЕ Д 20](#_Toc168474829)

[ПРИЛОЖЕНИЕ Е 22](#_Toc168474830)

[ПРИЛОЖЕНИЕ Ж 30](#_Toc168474831)

# ВВЕДЕНИЕ

Современная жизнь требует от нас не только умения зарабатывать, но и умения управлять своими доходами и расходами. В жизни человека редко случаются дни, когда он не покупает себе что-либо, ну а главный ресурс для этого – деньги, которые стремительно заканчиваются, если не отслеживать свои транзакции. Для этого и была создана программа, которая помогает пользователю отслеживать свои траты, поступления и даже сохранять деньги на будущее.

Разработка приложений трекера финансов становится необходимой задачей для тех, кто стремится к финансовой грамотности и уверенному управлению своими финансами. Такие приложения предлагают пользователю не только контроль над своими тратами и доходами, но и аналитику, помогающую понять, куда уходят деньги, и планировать свои действия.

Для создания этой программы был использован язык “C”, т.к. он является наиболее быстрым и функциональным. Язык “C” позволяет работать со структурами для более удобной организации хранимой информации и возможности реализации стека для отмены некоторых действий, файлами, поэтому в программе можно настроить хранение информации о балансе, транзакциях, поступлениях и копилки.

# 1 ПОСТАНОВКА ЗАДАЧИ

Программа для отслеживания финансов должна иметь удобный интерфейс взаимодействия с пунктами меню.

Реализовать функции для работы с файлом (изменение баланса, добавление транзакций и поступлений), ввод информации, функции для добавления транзакций и поступлений, копилки, конвертера валют на разные валюты по выбору, вывод всех транзакций и поступлений и удаление последних транзакций.

# 2 ОБЗОР ЛИТЕРАТУРЫ

Разработка программы трекер финансов на языке Си является актуальной задачей, поскольку существует большой спрос на подобные приложения среди людей, которые хотят обучиться финансовой грамотности.

Для удобства работы с данными был использован такой инструмент языка С, как структуры данных [1, 2]. Структуры данных в языке С позволяют объединить несколько объектов в один и обращаться к каждому из них как к элементу общего объекта. Такой подход позволяет использовать такой тип динамических структур данных как стек, что позволяет мне проводить отмену последних операций.

Для хранения контактов были использованы стандартные функции для работы с файлами из библиотеки stdio.h[1].

Кроме того, при разработке программы следует обратить внимание на статьи и примеры кода, посвященные работе с текстовыми файлами. Примеры таких статей можно найти на различных программистских порталах [3].

# 3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВНИЕ

В данном разделе описывается структура данных, используемых в данном приложении.

## **3.1 Структура данных**

struct moneyTracker

{

struct money balanceSettings; // Настройки баланса

union trans transactionSettings; // Настройки транзакций

struct Node\* stack; // Стек для хранения пополнений

};

Рисунок 3.1.1 – содержание структуры moneyTracker

typedef struct money

{

float balance; // Поле для хранения баланса

float MB\_balance; // Поле для хранения баланса копилки

};

Рисунок 3.1.3 – содержание объединения money

union trans

{

float plus; // Поле для операции пополнения

float minus; // Поле для операции списания

};

Рисунок 3.1.4 – содержание объединения trans

typedef struct Node

{

float amount; // Поле для хранения последнего списания

struct Node\* next; // Указатель на следующий элемент стека

} Node;

Рисунок 3.1.5 – содержание структуры Node

# 4 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ

## **4.1 Разработка схем алгоритмов**

Схема алгоритма программы main.cpp представлена в приложении А.

Функции printFileContents() и rewriteFloatToFile позволяют работать с файлами (вывод на экран, обновление информации). Схемы алгоритмов представлены в приложениях Б и В.

## **4.2 Разработка алгоритмов**

Функция addTrans() позволяет пользователю добавлять транзакции, записывая сумму самой транзакции и изменённый баланс в файл.

Передаваемые параметры:

moneyTracker\* tracker – указатель на структуру moneyTracker.

FILE\* f – указатель на текстовый файл.

FILE\* t – указатель на текстовый файл.   
 Шаг 1. Начало.  
 Шаг 2. Объявить переменные:  
 int choice - числовое значение выбора типа транзакции.  
 Шаг 3. Вывести список доступных типов транзакций и запросить у пользователя выбор.  
 Шаг 4. Считать выбор пользователя в переменную choice.  
 Шаг 5. Используя оператор switch, выполнить соответствующие действия. Если выбор соответствует одному из типов транзакций (1 - Home, 2 - Health, 3 - Food, 4 - Other), вывести заголовок для этого типа транзакции. Если выбор не соответствует ни одному из доступных типов транзакций, вывести сообщение об ошибке ("Invalid choice") и завершить выполнение функции.  
 Шаг 6. Вывести приглашение к вводу суммы транзакции и считать введенное значение в переменную tracker->transactionSettings.minus.  
 Шаг 7. Вычесть введенную сумму из баланса, хранящегося в структуре tracker.  
 Шаг 8. Записать сумму транзакции в файл t, используя функцию rewriteToFile.  
 Шаг 9. Записать новый баланс в файл f, используя функцию rewriteFloatToFile.  
 Шаг 10. Вернуть обновленный баланс.  
 Шаг 11. Конец.

Функция addReceipt() позволяет пользователю добавлять поступления, записывая сумму самого поступления и изменённый баланс в файл.

Передаваемые параметры:

moneyTracker\* tracker – указатель на структуру moneyTracker.

FILE\* f – указатель на текстовый файл.

FILE\* r – указатель на текстовый файл.   
 Шаг 1. Начало.  
 Шаг 2. Объявить переменные:  
 int choice - числовое значение выбора типа квитанции.  
 Шаг 3. Вывести список доступных типов квитанций и запросить у пользователя выбор.  
 Шаг 4. Считать выбор пользователя в переменную choice.  
 Шаг 5. Используя оператор switch, выполнить соответствующие действия в зависимости от выбора пользователя. Если выбор соответствует одному из типов квитанций (1 - Salary, 2 - Fee, 3 - Admission, 4 - Other), вывести заголовок для этого типа квитанции. Если выбор не соответствует ни одному из доступных типов квитанций, вывести сообщение об ошибке ("Invalid choice") и завершить выполнение функции.  
 Шаг 6. Вывести приглашение к вводу суммы квитанции и считать введенное значение в переменную tracker->transactionSettings.plus.  
 Шаг 7. Прибавить введенную сумму к балансу, хранящемуся в структуре tracker.  
 Шаг 8. Записать сумму поступления в файл r, используя функцию rewriteToFile.  
 Шаг 9. Записать новый баланс в файл f, используя функцию rewriteFloatToFile.  
 Шаг 10. Вернуть обновленный баланс.  
 Шаг 11. Конец.

Функция removeLastTransFromStack() позволяет пользователю удалить последнее проведённое списание.

Передаваемые параметры:

moneyTracker\* tracker – указатель на структуру moneyTracker.

FILE\* f – указатель на текстовый файл.

FILE\* r – указатель на текстовый файл.  
 Шаг 1. Начало.  
 Шаг 2. Объявить переменные:  
float lastTrans - для хранения значения последней транзакции.  
 Шаг 3. Если стек пуст, вывести сообщение об ошибке "ERROR! You have no transactions", очистить входной буфер (rewind(stdin)) и дождаться нажатия клавиши Enter, затем вернуть 0.0.  
 Шаг 4. Вызвать функцию pop для удаления и возврата последнего элемента из стека, передавая ей указатель на стек в трекере.  
 Шаг 5. Прибавить значение последней транзакции к общему балансу в трекере. Добавить значение lastTrans к полю balance.  
 Шаг 6. Вызвать функцию rewriteFloatToFile, передавая ей указатель на файл f и обновленное значение общего баланса из tracker.  
 Шаг 7. Вызвать функцию rewriteToFile, передавая ей указатель на файл r и значение lastTrans.  
 Шаг 8. Вернуть значение lastTrans.  
 Шаг 9. Конец.

Функция moneyBox() позволяет пользователю сохранить часть своего баланса, после чего он может получить его обратно по собственному желанию.

Передаваемые параметры:

moneyTracker\* tracker – указатель на структуру moneyTracker.

FILE\* f – указатель на текстовый файл.

FILE\* а – указатель на текстовый файл.   
 Шаг 1. Начало.  
 Шаг 2. Объявить переменные:  
 int MB\_choice - числовое значение выбора операции с деньгами в копилке.  
 float MB\_operation - числовое значение суммы операции с деньгами.  
 Шаг 3. Вывести информацию о текущем состоянии копилки, включая текущий баланс.  
 Шаг 4. Запросить у пользователя желаемую операцию с деньгами в копилке (получение или добавление).  
 Шаг 5. Считать выбор пользователя в переменную MB\_choice.  
 Шаг 6. Если пользователь выбрал получение денег из копилки (MB\_choice == 1):  
 Шаг 7. Проверить, если баланс в копилке равен нулю, вывести сообщение об ошибке и завершить операцию.  
 Шаг 8. Запросить у пользователя сумму, которую он хочет получить из копилки.  
 Шаг 9. Проверить сумму на корректность с помощью функции checkMB\_op.  
 Шаг 10. Прибавить полученную сумму к общему балансу.  
 Шаг 11. Если пользователь выбрал добавление денег в копилку (MB\_choice == 2):  
 Шаг 12. Запросить у пользователя сумму, которую он хочет добавить в копилку.  
 Шаг 13. Проверить сумму на корректность с помощью функции checkMB\_op\_add.  
 Шаг 14. Прибавить добавленную сумму к балансу копилки и вычесть эту сумму из общего баланса.  
 Шаг 15. Если выбор пользователя не соответствует ни одному из вариантов, вывести сообщение об ошибке и запросить ввод снова.  
 Шаг 16. Записать баланс копилки в файл а, используя функцию rewriteFloatToFile.  
 Шаг 17. Записать новый баланс в файл f, используя функцию rewriteFloatToFile.  
 Шаг 18. Конец.

# 5 РЕЗУЛЬТАТ РАБОТЫ

****

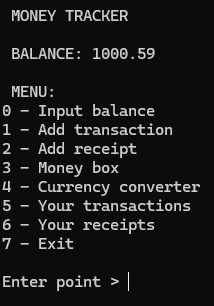
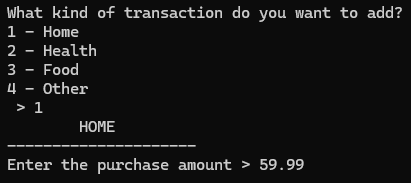
****

Рисунок 5.1 – Изменение баланса



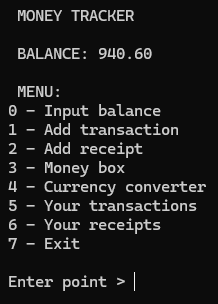
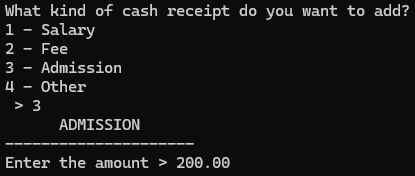


Рисунок 5.2 – Добавление транзакции



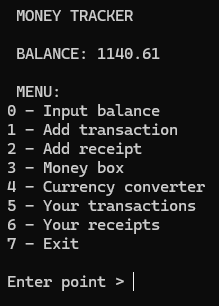


Рисунок 5.3 – Добавление поступления

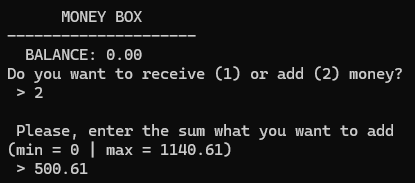


Рисунок 5.4.1 – Добавление денег в копилку

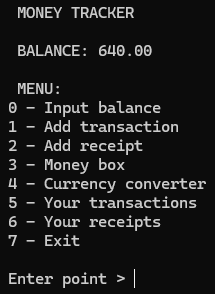


Рисунок 5.4.2 – Добавление денег в копилку

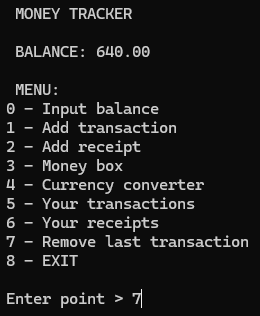




Рисунок 5.4 – Отмена последней транзакции

# ЗАКЛЮЧЕНИЕ

В рамках данной работы были успешно реализованы заданные цели, а именно успешно разработана программа для отслеживания своего баланса. Эта программа обеспечивает эффективную и точную обработку данных, что позволяет обеспечить удобное взаимодействие с пользователем.

Разработанная программа демонстрирует удобный и понятный интерфейс, что делает её простой в использовании для конечных пользователей. Интуитивно понятное меню, наглядная визуализация проведения операций, их результатов и простота управления основными функциями программы – всё это способствует быстрому освоению и комфортному применению разработанного проекта.

Реализованный проект был протестирован сторонними лицами и в результате пользователь успешно освоил интерфейса проекта и успешно пользовался им не обнаружив ошибок.

Особое внимание уделено оптимизации производительности приложения. Благодаря грамотному использованию динамической памяти и эффективным алгоритмам, время отклика и скорость работы приложения остаются на высоком уровне. Это делает его пригодным для использования даже на устройствах с ограниченными ресурсами.

Подводя итог, данный проект позволил не только закрепить знания языка С, но и приобрести опыт разработки практически значимого программного обеспечения. Полученные навыки будут полезны в дальнейшей работе над подобными задачами в области обработки данных для приложений отслеживания операций.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1]. Основы алгоритмизации и программирования : лаборатор. практикум для студентов специальности 1-40 02 01 «Вычисл. машины, системы и сети» всех форм обучения. В 2 ч. Ч. 2 / сост. Ю. А. Луцик [и др.]. –Минск : БГУИР, 2010. – 36 с. : ил.

[2]. Петцольд Ч. Windows программирование на Си. – М.: Русская Редакция, 2002. – 1312 с.

[3]. StackOverflow [Электронный ресурс] – Вопросы – режим доступа: https://ru.stackoverflow.com/questions – дата доступа – 12.04.2024.

**ПРИЛОЖЕНИЕ А**

*(обязательное)*

Схема алгоритма программы main.cpp

**ПРИЛОЖЕНИЕ Б**

*(обязательное)*

Схема алгоритма функции printFileContents()

**ПРИЛОЖЕНИЕ В**

*(обязательное)*

Схема алгоритма функции rewriteFloatToFile()

**ПРИЛОЖЕНИЕ Г**

*(обязательное)*

Код файла main.cpp

#include "header.h"

int main()

{

struct moneyTracker\* tracker; // инициализация структуры

int menuChoice = 0;

tracker = (struct moneyTracker\*)malloc(1 \* sizeof(struct moneyTracker));

tracker->balanceSettings.balance = 0;

tracker->balanceSettings.MB\_balance = 0;

initializeStack(&(tracker->stack));

int files;

FILE\* mt = NULL;

FILE\* ut = NULL;

FILE\* tt = NULL;

FILE\* rt = NULL;

printf("Do you want to delete old data, or use old data?\n 1 - DELETE || Any Key - OLD DATA\n > ");

scanf\_s("%d", &files);

openFiles(tracker, files, &mt, &ut, &tt, &rt); // функция открытия файлов

do

{

system("cls");

rewind(stdin);

printf(" MONEY TRACKER ");

printf("\n\n");

printf(" BALANCE: %.2f\n\n", tracker->balanceSettings.balance);

printf(" MENU: \n");

printf("0 - Input balance\n1 - Add transaction\n2 - Add receipt\n3 - Money box\n4 - Currency converter\n5 - Your transactions\n6 - Your receipts\n7 - Remove last transaction\n8 - EXIT\n\n");

printf("Enter point > ");

menuChoice = isInt(menuChoice);

/\* меню приложения \*/

switch (menuChoice)

{

case 0:

system("cls");

tracker->balanceSettings.balance = inputBalance(tracker->balanceSettings.balance, mt); // вызов функции ввода баланса

continue;

case 1:

system("cls");

addTrans(tracker, mt, tt); // вызов функции добавления транзакций

continue;

case 2:

system("cls");

addReceipt(tracker, mt, rt); // вызов функции добавления поступлений

continue;

case 3:

system("cls");

moneyBox(tracker, mt, ut); // выхов функции копилки

continue;

case 4:

system("cls");

// вызов функции конвертера валют

moneyConverter(tracker->balanceSettings.balance);

continue;

case 5:

system("cls");

// вызов функции вывода транзакций

printFileContents(tt, menuChoice);

continue;

case 6:

system("cls");

// вызов функции вывода поступлений

printFileContents(rt, menuChoice);

continue;

case 7:

system("cls");

// вызов функции удаления последней транзакции

removeLastTransFromStack(tracker, mt, rt);

continue;

case 8:

system("cls");

return 0;

default:

continue;

}

} while (1);

}

**ПРИЛОЖЕНИЕ Д**

*(обязательное)*

Код файла header.h

#pragma once

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

// Объявление объединения money

union money {

float balance;

float MB\_balance;

};

// Объявление объединения trans

union trans {

float plus;

float minus;

};

// Структура узла стека

typedef struct Node {

float amount; // Сумма транзакции

struct Node\* next; // Указатель на следующий узел

} Node;

// Структура moneyTracker

struct moneyTracker {

union money balanceSettings;

union trans transactionSettings;

struct Node\* stack; // Указатель на вершину стека

};

// функция инициализаци стека

void initializeStack(Node\*\* top);

// функция проверки стека на пустоту

int isEmpty(Node\* top);

// функция добавления элемента в стек

void push(Node\*\* top, float newData);

// функция удаления элемента со стека

float pop(Node\*\* top);

// функция удаления последней транзакции

float removeLastTransFromStack(struct moneyTracker\* tracker, FILE\* f, FILE\* r);

// функция открытия файлов

void openFiles(moneyTracker\* tracker, int files, FILE\*\* mt, FILE\*\* ut, FILE\*\* tt, FILE\*\* rt);

// функция записи в файл

void printFileContents(FILE\* file, int a);

// функция чтения с файла

float readFloatFromFile(FILE\* file);

// функция перезаписи в файл

void rewriteFloatToFile(FILE\* file, float value);

// функция добавления записи в файл

void rewriteToFile(FILE\* file, float value);

// функция ввода баланса

float inputBalance(float a, FILE\* f);

// функция добавления транзакции

float addTrans(moneyTracker\* tracker, FILE\* f, FILE\* t);

// функция добавления поступелния

float addReceipt(moneyTracker\* tracker, FILE\* f, FILE\* r);

// функция конвертера валют

void moneyConverter(float balance);

// функция проверки значения

float checkMB\_op(float a, moneyTracker\* tracker);

// функция проверки значения

float checkMB\_op\_add(float a, moneyTracker\* tracker);

// функция копилки

void moneyBox(moneyTracker\* tracker, FILE\* f, FILE\* a);

// функция проверки корректности ввода

float isInt(int a);

**ПРИЛОЖЕНИЕ Е**

*(обязательное)*

Код файла functions.cpp

#include "header.h"

// Инициализация стека

void initializeStack(Node\*\* top)

{

\*top = NULL;

}

// Проверка, пуст ли стек

int isEmpty(Node\* top)

{

return (top == NULL);

}

// Добавление элемента в стек

void push(Node\*\* top, float newData)

{

Node\* newNode = (Node\*)malloc(sizeof(Node));

if (newNode == NULL)

{

printf("Ошибка выделения памяти\n");

exit(EXIT\_FAILURE);

}

newNode->amount = newData;

newNode->next = \*top;

\*top = newNode;

}

// Удаление элемента из стека

float pop(Node\*\* top)

{

if (isEmpty(\*top))

{

printf("Стек пуст\n");

exit(EXIT\_FAILURE);

}

Node\* temp = \*top;

float data = temp->amount;

\*top = (\*top)->next;

free(temp);

return data;

}

// Функция добавления транзакции в стек

void addTransToStack(struct moneyTracker\* tracker, float amount)

{

push(&(tracker->stack), amount);

}

int isBalanceZero(float a)

{

if (a == 0)

{

printf("ERROR!\nYour balance is NULL.\n");

rewind(stdin);

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

return 0;

}

else

{

return 1;

}

}

float isFloat(float a)

{

int result = 0;

do

{

rewind(stdin);

result = scanf\_s("%f", &a);

if (result == true) // проверка типа

{

return a;

}

else

{

printf("INVALID VALUE! Try again.\n > ");

rewind(stdin);

}

} while (1);

}

float isInt(int a)

{

int result = 0;

do

{

rewind(stdin);

result = scanf\_s("%d", &a);

if (result == true) // проверка типа

{

return a;

}

else

{

printf("INVALID VALUE! Try again.\n > ");

rewind(stdin);

}

} while (1);

}

void openFiles(moneyTracker\* tracker, int files, FILE\*\* mt, FILE\*\* ut, FILE\*\* tt, FILE\*\* rt)

{

/\* W+ - очистка фалйа + доступ к записи и вывода

R+ - открытие файла + доступ к записи и вывода \*/

errno\_t mrr, urr, trr, rrr;

if (files == 1)

{

mrr = fopen\_s(mt, "balance.txt", "w+");

if (mrr != 0) // проверка на открытие файла

{

printf("Ошибка открытия файла.");

return;

}

urr = fopen\_s(ut, "moneybox.txt", "w+");

if (urr != 0)

{

printf("Ошибка открытия файла.");

return;

}

trr = fopen\_s(tt, "transactions.txt", "w+");

if (trr != 0)

{

printf("Ошибка открытия файла.");

return;

}

rrr = fopen\_s(rt, "recieves.txt", "w+");

if (rrr != 0)

{

printf("Ошибка открытия файла.");

return;

}

}

else

{

mrr = fopen\_s(mt, "balance.txt", "r+");

if (mrr != 0) // проверка на открытие файла

{

printf("Ошибка открытия файла.");

return;

}

urr = fopen\_s(ut, "moneybox.txt", "r+");

if (urr != 0)

{

printf("Ошибка открытия файла.");

return;

}

trr = fopen\_s(tt, "transactions.txt", "r+");

if (trr != 0)

{

printf("Ошибка открытия файла.");

return;

}

rrr = fopen\_s(rt, "recieves.txt", "r+");

if (rrr != 0)

{

printf("Ошибка открытия файла.");

return;

}

// запись значения из файла

fscanf\_s(\*mt, "%f", &tracker->balanceSettings.balance);

// запись значения из файла

fscanf\_s(\*ut, "%f", &tracker->balanceSettings.MB\_balance);

}

}

void printFileContents(FILE\* file, int a)

{

char line[100] = {};

if (a == 5)

{

printf("Transactions:\n");

}

else

{

printf("Receipts:\n");

}

rewind(file); // перемещаем позицию в файле на начало

while (fgets(line, sizeof(line), file))

{

printf("%s\n", line);

}

rewind(stdin);

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

}

void rewriteFloatToFile(FILE\* file, float value)

{

// проверяем, открылся ли файл успешно

if (file != NULL)

{

// перемещаем указатель в начало файла

fseek(file, 0, SEEK\_SET);

// перезаписываем значение

fprintf(file, "%f", value);

// очищаем буфер вывода

fflush(file);

}

}

void rewriteToFile(FILE\* file, float value)

{

// проверяем, открылся ли файл успешно

if (file != NULL)

{

// перезаписываем значение

fprintf(file, "%f\n", value);

// очищаем буфер вывода

fflush(file);

}

}

float readFloatFromFile(FILE\* file)

{

float value;

// Проверяем, что файл открыт

if (file == NULL)

{

fprintf(stderr, "Ошибка: Файл не открыт!\n");

return 0.0f;

}

// Читаем значение типа float из файла

if (fscanf\_s(file, "%f", &value) != 1)

{

fprintf(stderr, "Ошибка чтения значения типа float из файла!\n");

return 0.0f;

}

return value;

}

float inputBalance(float balance, FILE\* f)

{

do

{

printf("Current balance: %.2f", balance);

rewind(stdin);

printf("\n\nEnter balance (min = 0.01 | max = 1 000 000 000.00) > ");

balance = isFloat(balance); // проверка на ввод корректного значения

if (balance < 0.01 || balance > 1000000000.00)

{

system("cls");

printf("Invalid value.\nPlease, try again.");

continue;

}

system("cls");

} while (balance < 0.01 || balance > 1000000000.00);

rewriteFloatToFile(f, balance); // перезапись баланса в файл

return balance;

}

float addTrans(struct moneyTracker\* tracker, FILE\* f, FILE\* t)

{

int choice;

float help = 0.00;

int rezult = 0;

rezult = isBalanceZero(tracker->balanceSettings.balance); // проверка на значение баланса

if (rezult == 0)

{

return 0.0;

}

printf("What kind of transaction do you want to add?\n");

printf("1 - Home\n2 - Health\n3 - Food\n4 - Other\n");

scanf\_s("%d", &choice);

/\* Тип транзакций \*/

switch (choice)

{

case 1:

printf(" HOME\n");

printf("---------------------\n");

break;

case 2:

printf(" HEALTH\n");

printf("---------------------\n");

break;

case 3:

printf(" FOOD\n");

printf("---------------------\n");

break;

case 4:

printf(" OTHER\n");

printf("---------------------\n");

break;

default:

printf("Invalid choice\n");

rewind(stdin);

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

return 0.0;

}

printf("Enter the purchase amount > ");

scanf\_s("%f", &(tracker->transactionSettings.minus));

tracker->balanceSettings.balance -= tracker->transactionSettings.minus;

// Добавление транзакции в стек

addTransToStack(tracker, tracker->transactionSettings.minus);

// Запись транзакции в файл

rewriteToFile(t, tracker->transactionSettings.minus);

// Перезапись баланса

rewriteFloatToFile(f, tracker->balanceSettings.balance);

return tracker->balanceSettings.balance;

}

float removeLastTransFromStack(struct moneyTracker\* tracker, FILE\* f, FILE\* r)

{

if (isEmpty(tracker->stack))

{

printf("ERROR! You have no transactions\n");

rewind(stdin);

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

return 0.0;

}

float lastTrans = pop(&(tracker->stack));

tracker->balanceSettings.balance += lastTrans;

rewriteFloatToFile(f, tracker->balanceSettings.balance);

rewriteToFile(r, lastTrans);

return lastTrans;

}

float addReceipt(moneyTracker\* tracker, FILE\* f, FILE\* r)

{

int choice;

printf("What kind of cash receipt do you want to add?\n");

printf("1 - Salary\n2 - Fee\n3 - Admission\n4 - Other\n");

scanf\_s("%d", &choice);

/\* Тип поступления \*/

switch (choice)

{

case 1:

printf(" SALARY\n");

printf("---------------------\n");

break;

case 2:

printf(" FEE\n");

printf("---------------------\n");

break;

case 3:

printf(" ADMISSION\n");

printf("---------------------\n");

break;

case 4:

printf(" OTHER\n");

printf("---------------------\n");

break;

default:

printf("Invalid choice\n");

rewind(stdin);

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

return 0.0;

}

printf("Enter the amount > ");

scanf\_s("%f", &(tracker->transactionSettings.plus));

tracker->balanceSettings.balance += tracker->transactionSettings.plus;

rewriteToFile(r, tracker->transactionSettings.plus); // запись поступлений в фалй

rewriteFloatToFile(f, tracker->balanceSettings.balance); // перезапись баланса

return tracker->balanceSettings.balance;

}

void moneyConverter(float balance)

{

int rezult = 0;

rezult = isBalanceZero(balance);

if (rezult == 0)

{

return;

}

int choiceCon = 0;

float helpAm = 0.0;

printf("Choose desired currency\n\n 1 - USD\n 2 - RUB\n 3 - CNY\n > ");

scanf\_s("%d", &choiceCon);

printf("Enter the amount you want you transfer. (from 0.00 to %.2f)\n > ", balance);

while (1)

{

helpAm = isFloat(helpAm); // проверка на ввод корректного значения

if (helpAm >= balance || helpAm < 0)

{

printf("Invalid value!\nPlease, try again.");

}

else

{

break;

}

}

/\* Выбор валюты для конвертирования \*/

switch (choiceCon)

{

case 1:

printf("\nYour current balance in USD - %f\nThe entered amount in USD - %f", balance = balance / 3.23, helpAm = helpAm / 3.23);

rewind(stdin);

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

break;

case 2:

printf("\nYour current balance in RUB - %f\nThe entered amount in RUB - %f", balance = balance \* 3.56, helpAm = helpAm \* 3.56);

rewind(stdin);

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

break;

case 3:

printf("\nYour current balance in CYN - %f\nThe entered amount in CYN - %f", balance = balance \* 2.28, helpAm = helpAm \* 2.28);

rewind(stdin);

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

break;

}

}

float checkMB\_op(float a, moneyTracker\* tracker)

{

do

{

a = isFloat(a); // проверка на ввод корректного значения

if (a <= 0 || a > tracker->balanceSettings.MB\_balance)

{

printf("ERROR! Try again.\n > ");

}

else

{

return a;

}

} while (1);

}

float checkMB\_op\_add(float a, moneyTracker\* tracker)

{

do

{

a = isFloat(a); // проверка на ввод корректного значения

if (a <= 0 || a > tracker->balanceSettings.balance)

{

printf("ERROR! Try again.\n > ");

}

else

{

return a;

}

} while (1);

}

void moneyBox(moneyTracker\* tracker, FILE\* f, FILE\* a)

{

int rezult = 0, rez = 0;

int MB\_choice = 0;

float MB\_operation = 0;

do

{

MB\_operation = 0;

printf(" MONEY BOX\n---------------------\n");

printf(" BALANCE: %.2f\n", tracker->balanceSettings.MB\_balance);

printf("Do you want to receive (1) or add (2) money? (Any key to EXIT)\n > ");

scanf\_s("%d", &MB\_choice);

if (MB\_choice == 1)

{

if (tracker->balanceSettings.MB\_balance == 0.00)

{

printf("ERROR!\nYou habe no money in your money box.");

rewind(stdin);

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

break;

}

else

{

rez = isBalanceZero(tracker->balanceSettings.MB\_balance); // проверка на значение баланса

if (rez == 0)

{

return;

}

printf("\n Please, enter the sum what you want to recive\n(min = 0 | max = %.2f)\n > ", tracker->balanceSettings.MB\_balance);

MB\_operation = checkMB\_op(MB\_operation, tracker); // проверка ввода

tracker->balanceSettings.MB\_balance -= MB\_operation;

tracker->balanceSettings.balance += MB\_operation;

break;

}

}

else if (MB\_choice == 2)

{

rez = isBalanceZero(tracker->balanceSettings.balance); // проверка на значение баланса

if (rez == 0)

{

return;

}

printf("\n Please, enter the sum what you want to add\n(min = 0 | max = %.2f)\n > ", tracker->balanceSettings.balance);

MB\_operation = checkMB\_op\_add(MB\_operation, tracker); // проверка ввода

tracker->balanceSettings.MB\_balance += MB\_operation;

tracker->balanceSettings.balance -= MB\_operation;

break;

}

else

{

system("cls");

printf("INVALID VALUE!\nTry again.");

}

} while (MB\_choice == 1 || MB\_choice == 2);

rewriteFloatToFile(f, tracker->balanceSettings.balance); // перезапись баланса

rewriteFloatToFile(a, tracker->balanceSettings.MB\_balance); // перезапись баланса копилки

}

**ПРИЛОЖЕНИЕ Ж**

*(обязательное)*

Ведомость документов