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

[Условие задания 3](#_Toc168252647)

[ВВЕДЕНИЕ 4](#_Toc168252648)

[1 ПОСТАНОВКА ЗАДАЧИ 5](#_Toc168252649)

[2 ОБЗОР ЛИТЕРАТУРЫ 6](#_Toc168252650)

[3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ 7](#_Toc168252651)

[3.1 Структура данных 7](#_Toc168252652)

[4 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ 8](#_Toc168252653)

[4.1 Разработка схем алгоритмов 8](#_Toc168252654)

[4.2 Разработка алгоритмов 8](#_Toc168252655)

[5 РЕЗУЛЬТАТЫ РАБОТЫ 12](#_Toc168252656)

[ЗАКЛЮЧЕНИЕ 15](#_Toc168252657)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 16](#_Toc168252658)

[ПРИЛОЖЕНИЕ А 17](#_Toc168252659)

[ПРИЛОЖЕНИЕ Б 18](#_Toc168252660)

[ПРИЛОЖЕНИЕ В 19](#_Toc168252661)

[ПРИЛОЖЕНИЕ Г 20](#_Toc168252662)

[ПРИЛОЖЕНИЕ Д 22](#_Toc168252663)

[ПРИЛОЖЕНИЕ Е 23](#_Toc168252664)

# Условие задания

**Трекер финансов**

Написать программу, позволяющая отслеживать транзакции и поступление пользователя. В отдельный файл записывать текущий баланс пользователя для возможного редактирования прошлого баланса.

# ВВЕДЕНИЕ

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.

Целью данной работы является написание программы, которая позволяет пользователю отслеживать различные операции со своим счётом

Для создания этой программы был использован язык “C”, т.к. он является наиболее быстрым и функциональным. Язык “C” позволяет работать с 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.

# 2 ОБЗОР ЛИТЕРАТУРЫ

Для решения задачи был выбран язык программирования С и методология структурного программирования.

В процессе разработки программы были использованы различные возможности языка С, которые будут описаны ниже.

Для удобства работы с данными был использован такой инструмент языка С, как структуры данных. Структуры данных в языке С позволяют объединить несколько объектов в один и обращаться к каждому из них как к элементу общего объекта. Такой подход существенно облегчает процесс написания программы, а также увеличивает читабельность кода.

Для хранения информации были использованы стандартные функции для работы с файлами из библиотеке stdio.h. Для работы с данной программой создано четыре файла:

1. balance.txt – для хранения информации о балансе;
2. moneybox.txt – для хранения информации о балансе копилки;
3. transactions.txt – для хранения информации о транзакцияx;
4. receipt.txt – для хранения информации о поступлениях.

# 3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ

В данном разделе описываются входные и выходные данные программы, хранящиеся в файлах, а также описывается структура этих данных

## **3.1 Структура данных**

Для удобного хранения данных и их использования была создана общая структура данных.

struct moneyTracker

{

struct MB box;

union money balanceSettings;

union trans transactionSettings;

};

Рисунок 3.1.1 – содержание структуры moneyTracker

struct MB

{

float MB\_balance;

};

Рисунок 3.1.2 – содержание структуры MB

union money

{

float balance;

float moneyBox;

};

Рисунок 3.1.3 – содержание объединения money

union trans

{

float plus;

float minus;

};

Рисунок 3.1.4 – содержание объединения trans

# 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 inputBalance() позволяет пользователю изменить значение своего баланса и перезаписать его в соответствующий файл.

Передаваемые параметры:

float balance – переменная, хранящая данные о балансе.

FILE\* f – указатель на текстовый файл.  
 Шаг 1. Начало.  
 Шаг 2. Вывести значение переменной balance.  
 Шаг 3. Запросить у пользователя ввод нового значения.  
 Шаг 4. Считать введенное значение в переменную balance.  
 Шаг 5. Проверить, если введенный баланс меньше 0.01 или больше 1000000000.00. Если условие выполняется, очистить экран и вывести сообщение об ошибке ("Invalid value. Please, try again."). Продолжить цикл с шага 3.  
 Шаг 6. Очистить консоль.  
 Шаг 7. Записать введенный баланс в файл balance.txt, используя функцию rewriteFloatToFile.  
 Шаг 8. Функция возвращает изменённое значение balance.  
 Шаг 9. Конец.

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 moneyConverter() позволяет пользователю перевести всю и введённую сумму со своего баланса в одну из популярных валют.

Передаваемые параметры:

float balance – переменная, хранящая данные о балансе.  
 Шаг 1. Начало.  
 Шаг 2. Объявить переменные:  
 int choiceCon - числовое значение выбора желаемой валюты.  
 float helpAm - числовое значение вспомогательной переменной для суммы перевода.  
 Шаг 3. Вывести список доступных валют и запросить у пользователя выбор.  
 Шаг 4. Считать выбор пользователя в переменную choiceCon.  
 Шаг 5. Вывести приглашение к вводу суммы для перевода и проверить введенное значение. Если введенная сумма больше текущего баланса или меньше нуля, вывести сообщение об ошибке и запросить ввод снова.  
 Шаг 6. Используя оператор switch, выполнить соответствующие действия в зависимости от выбора пользователя. Если выбор соответствует одной из валют (1 - USD, 2 - RUB, 3 - CYN), вывести текущий баланс и введенную сумму в выбранной валюте, используя соответствующие курсы конвертации. Очистить ввод и ожидать ввода ENTER для продолжения программы.  
 Шаг 7. Конец.

Функция moneyBox() позволяет пользователю сохранить часть своего баланса на время, 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 РЕЗУЛЬТАТЫ РАБОТЫ

****

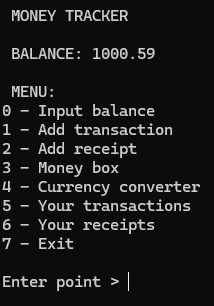
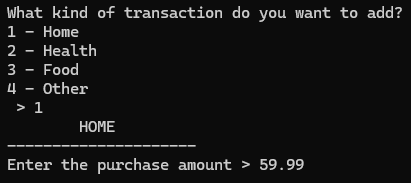
****

Рисунок 5.1 – Изменение баланса



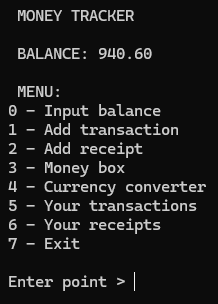
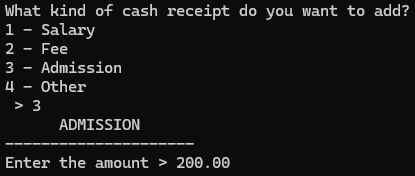


Рисунок 5.2 – Добавление транзакции



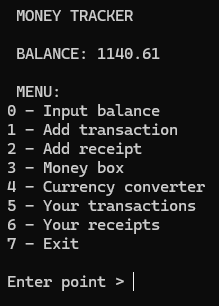
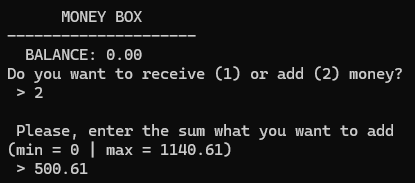


Рисунок 5.3 – Добавление поступления



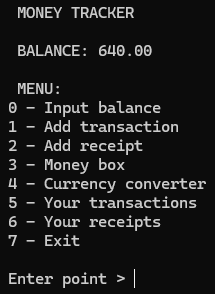


Рисунок 5.4 – Добавление денег в копилку

**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.

Выполнение задания позволило не только разработать алгоритмы, но и улучшить навыки работы со структурами данных и файлами. Полученные знания и опыт могут быть использованы в дальнейшей работе над улучшением программы и расширением её функциональности.

В заключение можно отметить, что разработанный в рамках данной работы алгоритм и его программная реализация для отслеживания операций над балансом пользователя показали свою эффективность и точность.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Луцик, Ю. А. Основы алгоритмизации и программирования: язык Си: учебно-метод. пособие / Ю. А. Луцик, А. М. Ковальчук, Е. А. Сасин. – Минск: БГУИР, 2015. – 170с. : ил.

2. 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.

3. Демидович, Е. М. Основы алгоритмизации и программирования. Язык Си / Е. М. Демидович. – СПб.: БХВ-Петербург, 2008. – 440 с.

4. 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;

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

tracker->balanceSettings.balance = 0;

tracker->box.MB\_balance = 0;

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 - Exit\n\n");

printf("Enter point > ");

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

return 0;

default:

system("cls");

printf(" Invalid value!\n\n");

rewind(stdin);

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

continue;

}

} while (1);

}

**ПРИЛОЖЕНИЕ Д**

*(обязательное)*

Код файла header.h

#pragma once

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

union money

{

float balance;

float moneyBox;

};

union trans

{

float plus;

float minus;

};

struct MB

{

float MB\_balance;

};

struct moneyTracker

{

struct MB box;

union money balanceSettings;

union trans transactionSettings;

};

// 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);

**ПРИЛОЖЕНИЕ Е**

*(обязательное)*

Ведомость документов