**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра САПР**

**ОТЧЁТ**

**По лабораторной работе №1**

**По дисциплине «Программирование»**

Студент гр. 3351 \_\_\_\_\_\_\_\_\_\_\_\_ Морозов А.А.

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_ Рыжов Н.Г.

Санкт-Петербург

2024

цель работы

В банкомате имеется ограниченное количество купюр разного достоинства 100, 500, 1000 и 5000 руб. Составить программу выдачи денег банкоматом при условии, что банковская карта клиента прошла авторизацию (pin-код введен), вся необходимая информация о клиенте, и состоянии банкомата доступна программе.

*Требования по выполнению задания*

1. За один сеанс клиент должен иметь возможность выполнить несколько операций снятия наличных.
2. Помимо возможности снять любую допустимую сумму клиенту должны предлагаться на выбор стандартные суммы, определяемые банком, которые могут отличаться от номиналов купюр. Значения стандартных сумм и их количество могут меняться.
3. Номиналы купюр, находящихся в обращении, могут измениться, так же как и количество номиналов.
4. Предусмотреть по выбору клиента выдачу денег крупными купюрами или с разменом (разными купюрами).
5. Ограничения на выдачу наличных:
   * количество средств на счете клиента;
   * дневной лимит выдачи наличных;
   * разовый лимит выдачи наличных;
   * максимальное количество купюр, выдаваемых банкоматом за один раз.
6. Все параметры обслуживания, состояния счета клиента и банкомата должны храниться в конфигурационном файле, структура которого разрабатывается самостоятельно и приводится в отчете, в том числе: сумма средств на счете клиента, номиналы, стандартные суммы, разовый лимит, дневной лимит, максимальное выдаваемое за один раз количество купюр, количество купюр каждого номинала в банкомате и др.
7. Предусмотреть несколько уровней меню для выбора операций клиентом. Меню верхнего уровня должно содержать команду выдачи наличных, команду прекращения работы, а также возможность добавления других команд (проверки баланса карты, внесение наличных и т. п.), реализовывать которые не требуется.
8. В меню любого уровня, если команда не может быть реализована, то она должна быть для клиента недоступна (например, не выводиться на экран). Если в меню (кроме верхнего уровня) доступна только одна команда, то меню не выводится на экран, но команда выполняется.
9. Клиент должен иметь возможность отказаться от любой команды до ее выполнения.
10. Каждая схема алгоритма в программе должна быть оформлена отдельной функцией.
11. Схемы алгоритмов должны иметь не более 15 блоков, включая блоки «начало» и «конец». Каждая схема располагается на отдельном листе.
12. Привести примеры запуска программы для различных запросов клиентов, включая случаи недостаточного количества купюр некоторых номиналов в банкомате. Перед каждой операцией выдачи денег выводите на экран количество имеющихся в банкомате купюр каждого номинала. Выводите результат каждой операции в виде количества выдаваемых купюр каждого номинала.
13. Продемонстрируйте на скриншотах выполнение п. 8 задания: отсутствие в выводимых меню пунктов со стандартными суммами, которые не могут быть выданы, а также отсутствие вывода меню крупными/разными в случае, если выбранная сумма может быть выдана единственным способом.
14. Снабдите программу комментариями, соответствующими схеме алгоритма.

Для проверки работы приложите к отчету файл конфигурации и добавьте перед каждой операцией вывод всех имеющихся в банкомате купюр: номиналы и количество.

**Текст программы**

#include <iostream>

#include <fstream>

#include <vector>

#include <map>

#include <sstream>

using namespace std;

// Структура для хранения конфигурации банкомата и счета клиента

struct Config

{

int account\_balance; // Баланс счета клиента

int daily\_limit; // Дневной лимит выдачи наличных

int single\_limit; // Разовый лимит выдачи наличных

int max\_bills; // Максимальное количество купюр, выдаваемых за один раз

map<int, int> bills\_count; // Количество купюр каждого номинала в банкомате

vector<int> standard\_sums; // Стандартные суммы, определяемые банком

};

void load\_config(const string& filename, Config& config)

{

ifstream file(filename);

if (!file.is\_open())

{

cerr << "Ошибка в открытии файла." << endl;

exit(1);

}

string line;

// Считываем баланс

getline(file, line);

if (line == "Баланс:")

{

getline(file, line);

// Класс в стандартной библиотеке C++, который позволяет работать со строками как с потоками ввода-вывода.

// Также он позволяет выполнять преобразования строк в числа и другие типы данных с помощью >>. Подключил

stringstream ss(line);

ss >> config.account\_balance;

}

// Считываем дневной лимит выдачи наличных

getline(file, line);

if (line == "Дневной лимит выдачи наличных:")

{

getline(file, line);

stringstream ss(line);

ss >> config.daily\_limit;

}

// Считываем разовый лимит выдачи наличных

getline(file, line);

if (line == "Разовый лимит выдачи наличных:")

{

getline(file, line);

stringstream ss(line);

ss >> config.single\_limit;

}

// Считываем максимальное количество купюр, выдаваемых за один раз

getline(file, line);

if (line == "Максимальное количество купюр, выдаваемых за один раз:")

{

getline(file, line);

stringstream ss(line);

ss >> config.max\_bills;

}

cout << "Макс. купюр: " << config.max\_bills << endl;

// Считываем количество купюр каждого номинала в банкомате

getline(file, line);

while (line != "Стандартные суммы, определяемые банком:")

{

// Создаём переменную из считанной ранее строки

stringstream ss(line);

int nominal, count;

// Преобразовываем строку в числа

if (ss >> nominal >> count)

{

config.bills\_count[nominal] = count;

}

getline(file, line);

}

int standard\_sum;

// Считываем стандартные суммы

while (file >> standard\_sum)

{

config.standard\_sums.push\_back(standard\_sum);

}

file.close();

}

// Функция для выдачи денег банкоматом

// Принимает конфигурацию банкомата в виде ссылки на объект Config, сумму для выдачи и флаг large\_bills

// Возвращает вектор пар, где первый элемент пары - номинал купюры, а второй - количество купюр

vector<pair<int, int>> withdraw\_money(Config& config, int amount, bool large\_bills)

{

// Создаем вектор result для хранения результатов выдачи денег

vector<pair<int, int>> result;

// Создаем копию ассоциативного массива bills\_count из объекта config

map<int, int> bills\_count = config.bills\_count;

// Если флаг large\_bills установлен, обходим купюры в обратном порядке

if (large\_bills)

{

// Используем итератор it для обхода купюр в обратном порядке с помощью rbegin() и rend()

for (auto it = bills\_count.rbegin(); it != bills\_count.rend() && amount > 0; ++it)

{

// Получаем номинал и количество купюр из текущей пары

int nominal = it->first;

int count = min(amount / nominal, it->second);

// Если количество купюр больше нуля

if (count > 0)

{

// Добавляем пару (номинал, количество) в вектор result

result.push\_back(make\_pair(nominal, count));

// Вычитаем выданную сумму из общей суммы для выдачи

amount -= nominal \* count;

// Вычитаем выданное количество купюр из текущего номинала

it->second -= count;

}

}

}

// Если флаг large\_bills не установлен, обходим купюры в прямом порядке

else

{

// Используем итератор it для обхода купюр в прямом порядке с помощью begin() и end()

for (auto it = bills\_count.begin(); it != bills\_count.end() && amount > 0; ++it)

{

// Получаем номинал и количество купюр из текущей пары

int nominal = it->first;

int count = min(amount / nominal, it->second);

// Если количество купюр больше нуля

if (count > 0)

{

// Добавляем пару (номинал, количество) в вектор result

result.push\_back(make\_pair(nominal, count));

// Вычитаем выданную сумму из общей суммы для выдачи

amount -= nominal \* count;

// Вычитаем выданное количество купюр из текущего номинала

it->second -= count;

}

}

}

// Возвращаем вектор result с результатами выдачи денег

return result;

}

void withdraw\_money\_from\_account(Config& config, int amount, const vector<pair<int, int>>& result)

{

// Вычитаем сумму с банковского счета

config.account\_balance -= amount;

// Уменьшаем количество банкнот в конфигурации

for (const auto& p : result)

{

config.bills\_count[p.first] -= p.second;

}

}

// Функция для вывода меню выдачи денег

// Принимает конфигурацию банкомата в виде ссылки на объект Config и ссылку на вектор available\_standard\_sums

void print\_menu(Config& config, vector<int>& available\_standard\_sums)

{

// Выводим пункт меню "Withdraw custom amount"

cout << "\n1. Ввести вручную" << endl;

// Обходим все стандартные суммы из конфигурации банкомата

for (size\_t i = 0; i != config.standard\_sums.size(); ++i)

{

// Получаем текущую стандартную сумму

int standard\_sum = config.standard\_sums[i];

// Проверяем, можно ли выдать эту сумму банкоматом

if (standard\_sum <= config.account\_balance &&

((withdraw\_money(config, standard\_sum, false).size() > 0) ||

(withdraw\_money(config, standard\_sum, true).size() > 0)))

{

// Если да, добавляем ее в вектор available\_standard\_sums

available\_standard\_sums.push\_back(standard\_sum);

// Выводим пункт меню с номером i + 2 и суммой в рублях

cout << i + 2 << ". Выдать " << standard\_sum << " рублей" << endl;

}

}

// Выводим пункт меню "Cancel" с номером, равным количеству доступных стандартных сумм + 2

cout << available\_standard\_sums.size() + 2 << ". Назад" << endl;

}

// Создаем объект Config для хранения конфигурации банкомата

Config config;

// Главная функция программы

int main()

{

setlocale(LC\_ALL, "RU");

// Загружаем конфигурацию из файла config.txt

load\_config("config.txt", config);

// Запускаем главный цикл программы

while (true)

{

// Выводим приветственное сообщение и меню выбора действий

cout << "\n-=-=-=-=-=-=- Главное меню -=-=-=-=-=-=-" << endl;

cout << "1. Проверка баланса" << endl;

cout << "2. Снятие наличных" << endl;

cout << "3. Выйти\n"

<< endl;

// Считываем выбор пользователя

int choice;

cin >> choice;

// Если пользователь выбрал пункт "Check balance"

if (choice == 1)

{

cout << "\nВаш баланс: " << config.account\_balance << endl;

}

// Если пользователь выбрал пункт "Withdraw money"

else if (choice == 2)

{

// Создаем вектор для хранения доступных стандартных сумм

vector<int> available\_standard\_sums;

// Выводим меню выдачи денег

print\_menu(config, available\_standard\_sums);

// Считываем выбор пользователя

int choice;

cin >> choice;

// Если пользователь выбрал пункт "Withdraw custom amount"

if (choice == 1)

{

// Считываем сумму для выдачи

cout << "\nВведите сумму для выдачи: ";

int amount;

cin >> amount;

// Проверяем, что сумма не превышает баланс счета, дневной и разовый лимиты

if (amount <= config.account\_balance &&

amount <= config.daily\_limit &&

amount <= config.single\_limit)

{

// Выдаем деньги с разменом

vector<pair<int, int>> result = withdraw\_money(config, amount, false);

// Если выдача удачна

if (result.size() > 0)

{

// Спрашиваем у пользователя, хочет ли он получить деньги крупными купюрами

cout << "\nВыдача крупными или мелкими (1 - крупными, 2 - мелкими): ";

int large\_bills\_choice;

cin >> large\_bills\_choice;

// Если пользователь выбрал вариант "large" или "change"

if (large\_bills\_choice == 1 || large\_bills\_choice == 2)

{

// Выдаем деньги в соответствии с выбором пользователя

result = withdraw\_money(config, amount, large\_bills\_choice == 1);

withdraw\_money\_from\_account(config, amount, result);

cout << "\nСнятие средств:" << endl;

// Выводим результат выдачи

for (const auto& pair : result)

{

cout << pair.first << " рублей: " << pair.second << " купюр(-а, -ы)" << endl;

}

// Выводим количество купюр в банкомате

cout << "Количество оставшихся купюр в банкомате:" << endl;

for (const auto& pair : config.bills\_count)

{

cout << pair.first << " рублей: " << pair.second << " купюр(-а, -ы)" << endl;

}

}

else

{

// Если пользователь ввел неверный выбор, сообщаем об ошибке

cout << "\nНеверный выбор. Операция отменена.\n"

<< endl;

}

}

else

{

// Если выдача не удачна, сообщаем об ошибке

cout << "\nОшибка! Выдача не удачна.\n" << endl;

}

}

else

{

// Если сумма превышает баланс счета или лимиты, сообщаем об ошибке

cout << "\nInsufficient funds or exceeded limits.\n" << endl;

}

}

// Если пользователь выбрал один из пунктов "Withdraw standard sum"

else if (choice <= available\_standard\_sums.size() + 1)

{

// Вычисляем сумму для выдачи

int amount = available\_standard\_sums[choice - 2];

// Проверяем, что сумма не превышает баланс счета

if (amount <= config.account\_balance)

{

// Выдаем деньги

vector<pair<int, int>> result = withdraw\_money(config, amount, true);

// Если выдача удачна

if (result.size() > 0)

{

withdraw\_money\_from\_account(config, amount, result);

// Выводим результат выдачи

cout << "\nСнятие средств:" << endl;

for (const auto& pair : result)

{

cout << pair.first << " рублей: " << pair.second << " купюр(-а, -ы)" << endl;

}

// Выводим количество купюр в банкомате

cout << "Количество купюр в банкомате:" << endl;

for (const auto& pair : config.bills\_count)

{

cout << pair.first << " рублей: " << pair.second << " купюр(-а, -ы)" << endl;

}

}

else

{

// Если выдача не удачна, сообщаем об ошибке

cout << "Ошибка! Выдача не удачна.\n"

<< endl;

}

}

else

{

cout << "\nНедостаточный баланс.\n" << endl;

}

}

// Если пользователь выбрал пункт "Назад" или ввел неверный выбор

else

{

cout << "\nОперация отменена.\n" << endl;

}

}

// Если пользователь выбрал пункт "Выйти"

else if (choice == 3)

{

// Выходим из цикла

break;

}

// Если пользователь ввел неверный выбор

else

{

cout << "\nНеверный выбор. Повторите ввод." << endl;

}

}

// Возвращаем 0, чтобы указать на успешное завершение программы

return 0;

}

**Конфигурационный файл**

Баланс:

20000

Дневной лимит выдачи наличных:

20000

Разовый лимит выдачи наличных:

10000

Максимальное количество купюр, выдаваемых за один раз:

50

Количество купюр каждого номинала в банкомате:

100 50

500 50

1000 50

5000 50

Стандартные суммы, определяемые банком:

1000

2000

5000

10000

**Описание программы**

Программное обеспечение – Microsoft Visual Studio 2022. Язык программирования – C++. Программа имитирует работу банкомата и имеет конфигурационный файл с данными. При запуске сразу открывается главное меню с пунктами: проверка баланса, снятие средств, выйти. Пункт снятие средств имеет подменю с фиксированными суммами и ручным вводом для снятия средств с баланса клиента. Пункт проверка баланса выводит в консоль баланс счёта клиента. Любую операцию можно прервать в любом уровне меню.

**Руководство пользователя**

В начале работы программы выводится главное меню банкомата. В нём есть 3 пункта: проверка баланса, снятие средств, выйти. Пункт проверка баланса выводит баланс счёта клиента из конфигурационного файла банкомата. С помощью пункта снятие средств можно снять деньги со счёта клиента. Если денег на счёте клиента недостаточно, то некоторые фиксированные суммы могут не выводиться. Если денег на счёте клиента нет, то пункт снятие средств в главном меню не выводится. После каждого снятия средств выводятся системные данные, показывающие количество выданных купюр каждого номинала, а также количество оставшихся в банкомате купюр разного номинала. В банкомате присутствует дневной и месячный лимит на снятие средств, в случае превышения которого некоторые пункты меню могут не выводиться. Привожу примеры работы программы

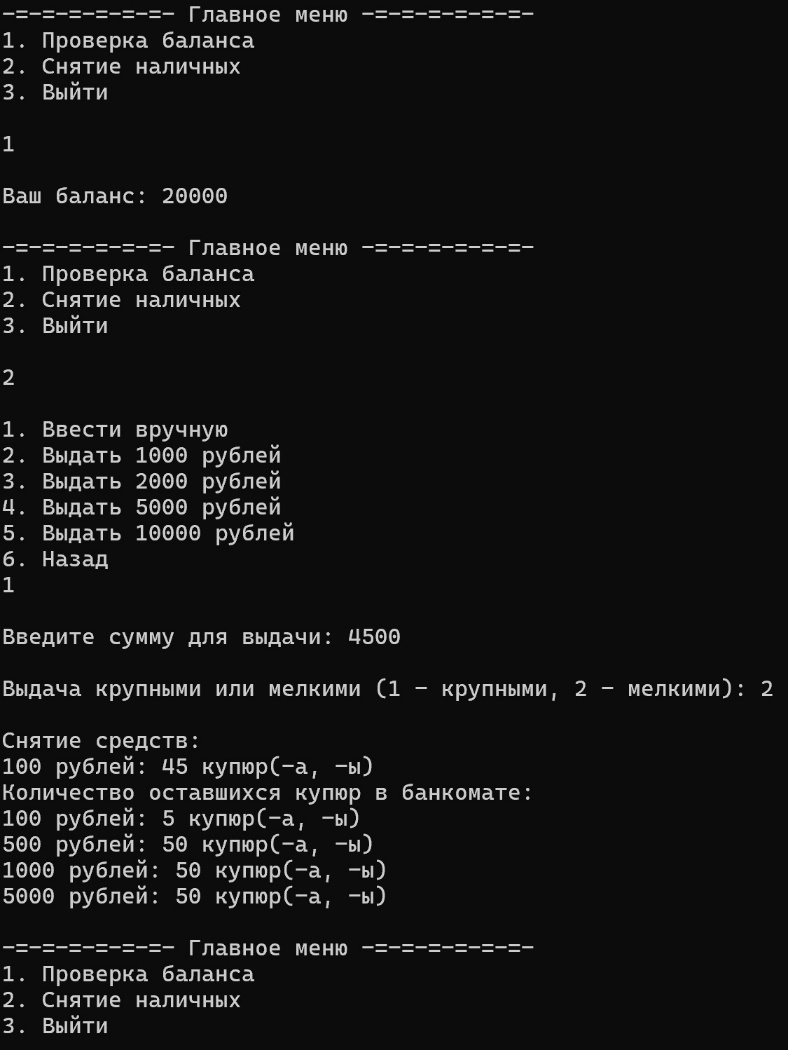


Рис. 1 – Пример снятие средств

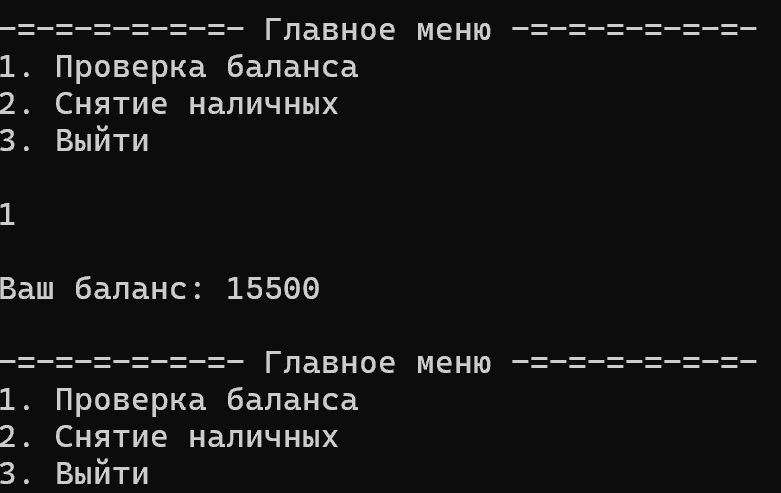


Рис. 2 – Повторная проверка баланса

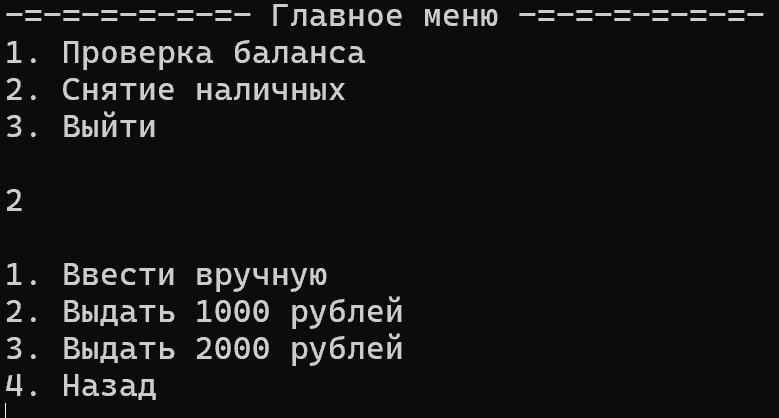


Рис. 3 – Пункты подменю не выводятся, т.к. средств недостаточно

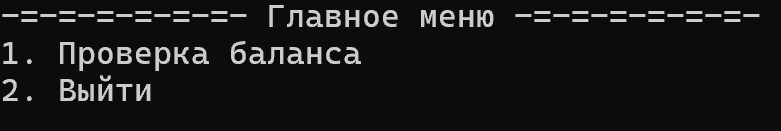
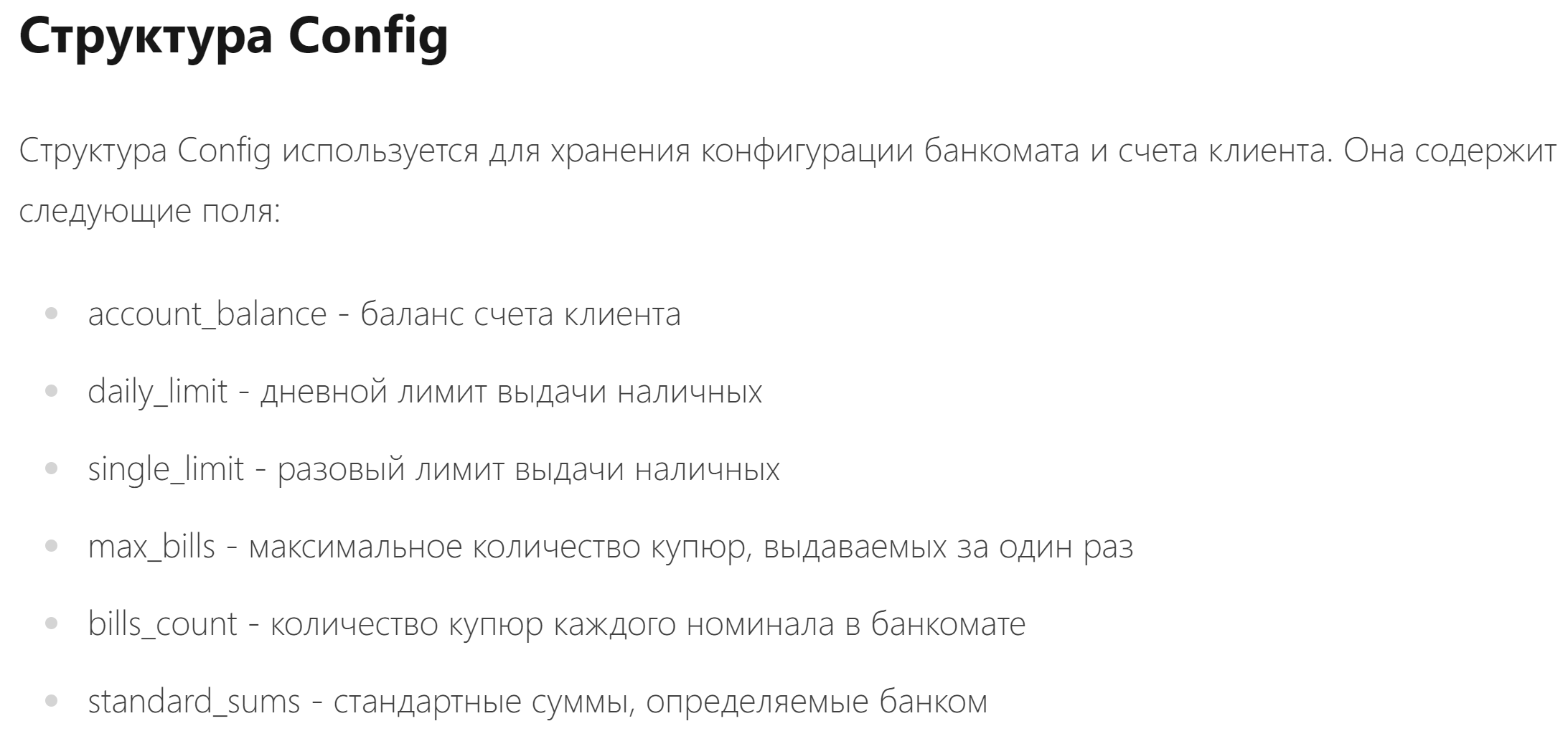
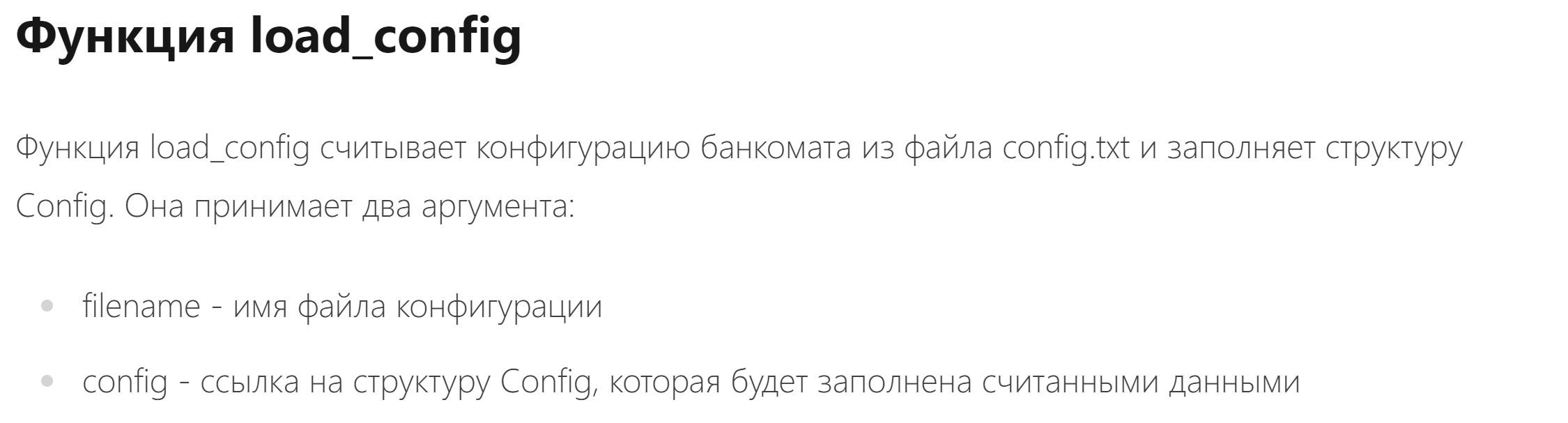
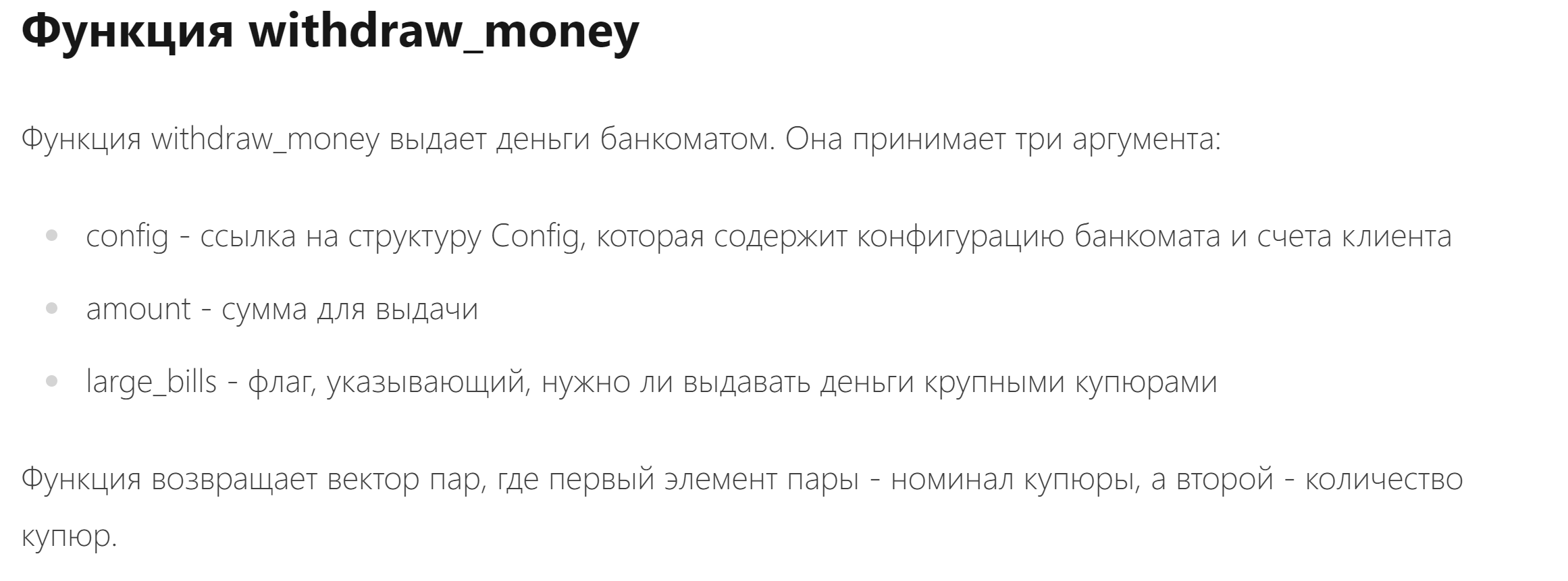
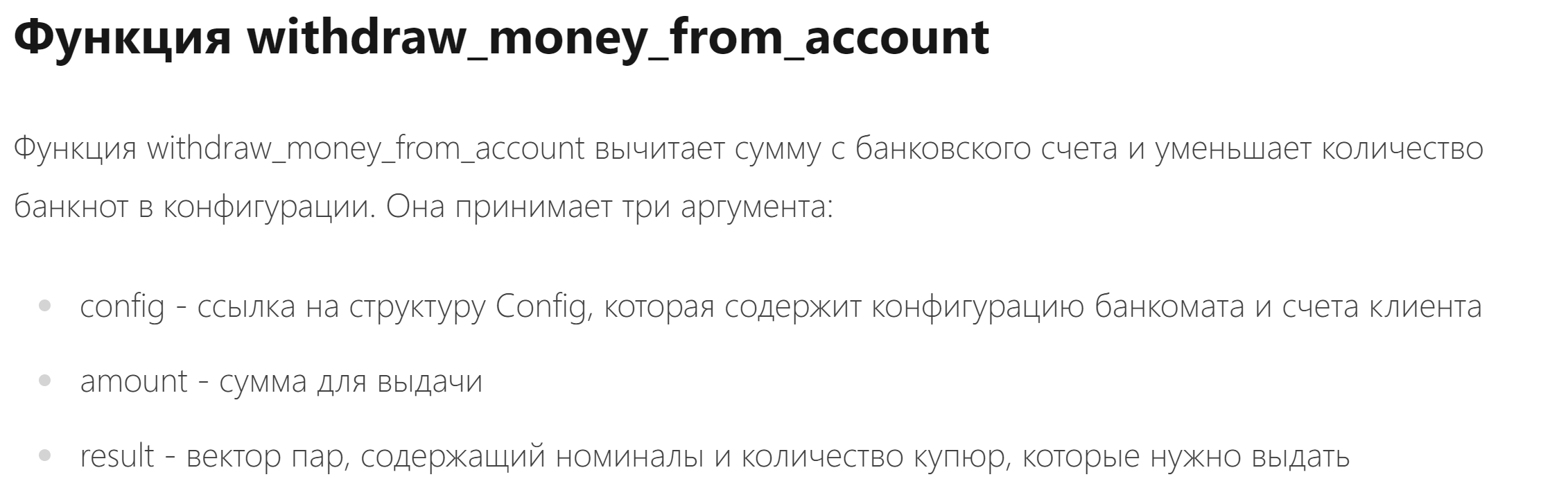


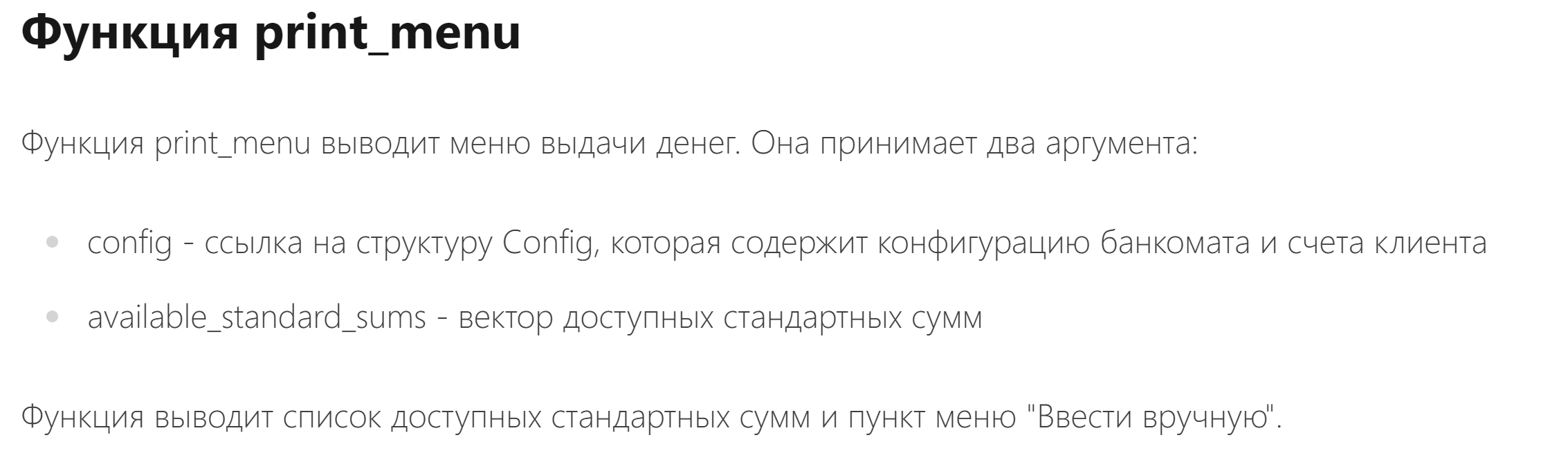
Рис. 4 – Пункт не выводится, т.к. баланс счёта клиента 0

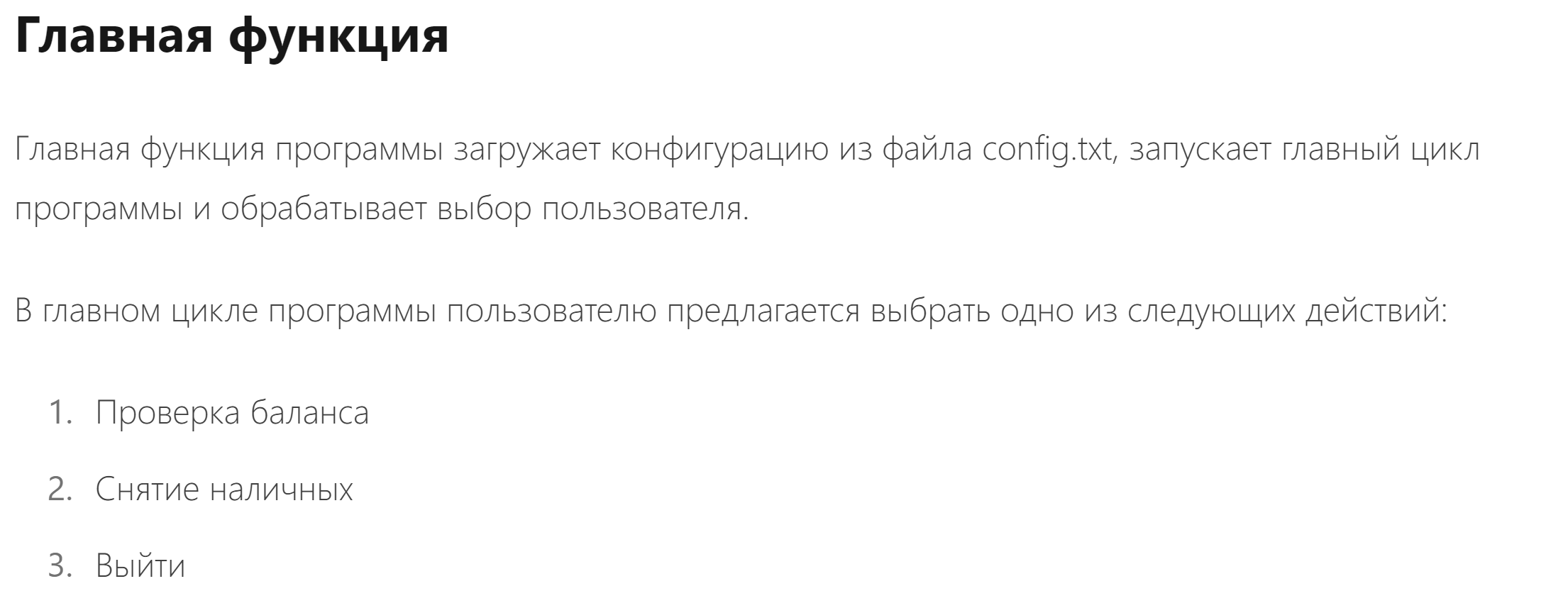


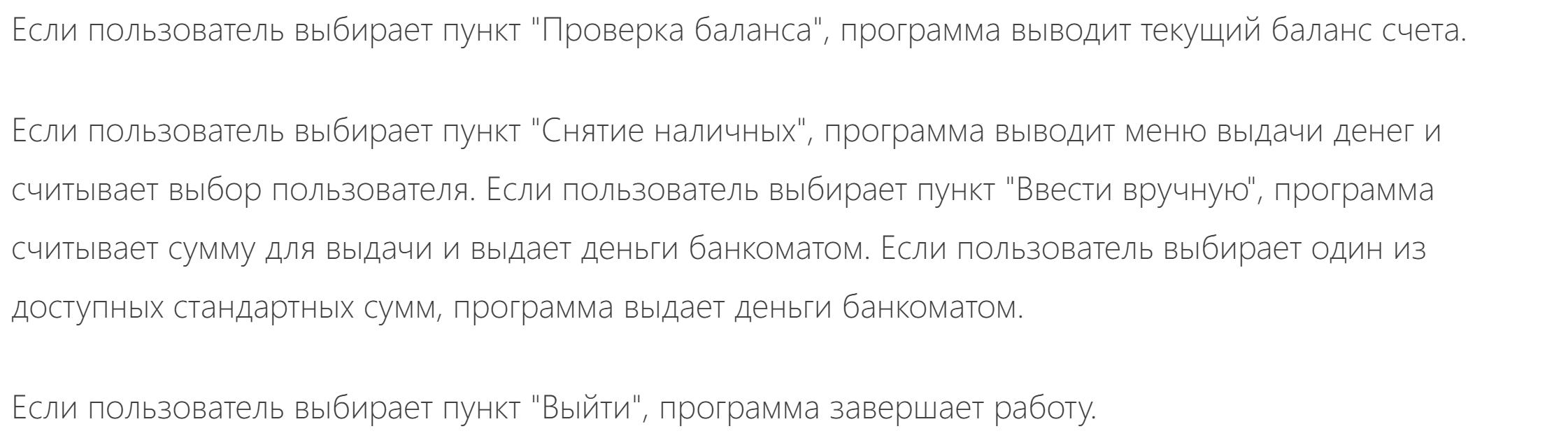












**Пути дальнейшего улучшения программ**

**Переворот**

1. Добавление новых функций банкомата, например внесение средств, перевод.

2. Улучшение визуальной части программы.