Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Программирование на языках высокого уровня

К ЗАЩИТЕ ДОПУСТИТЬ

\_\_\_\_\_\_\_\_\_\_ Марзалюк А.В.

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе  
на тему

МОБИЛЬНЫЙ ОПЕРАТОР

БГУИР КР 1-40 02 01 325 ПЗ

Студент:

Руководитель:

Ходосевич М. А.

Марзалюк А.В.

МИНСК 2022

Учреждение образования

«Белорусский государственный университет информатики   
и радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ

Заведующий кафедрой ЭВМ

\_\_\_\_\_\_\_\_\_ Б. В. Никульшин

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_2022 г.

ЗАДАНИЕ

по курсовому проектированию

Студенту *Ходосевичу Матвею Александровичу*

1. Тема проекта *Мобильный оператор*

2. Срок сдачи студентом законченного проекта *15 декабря 2022 г.*

3. Исходные данные к проекту *Файл: Client.txt – содержит информацию о клиентах компании, ConstantsFile – содержит информацию о константах используемых в программе. Контейнер: свой контейнер – OwnContainer(List), STL – vector.*

4. Содержание расчетно-пояснительной записки (перечень вопросов, которые подлежат разработке)

*Введение. Содержание. 1. Задача данной курсовой работы 2. Структура используемых данных. 3. Диаграмма классов. 4. Описание классов созданных программой. 5. Разработка алгоритмов. 6. Тест работы программы. Заключение. Список использованной литературы.*

5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)

*1. Диаграмма классов*

*2. Схема алгоритма метода*

*3. Схема алгоритма метода*

6. Консультант по проекту *Марзалюк А. В*.

7. Дата выдачи задания  *сентября 2022 г.*

8. Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и трудоемкости отдельных этапов):

*разделы 1, 2 к 10 октября 2022 г. – 20 %;*

*разделы 3, 4 к 10 ноября 2022 г. – 30 %;*

*разделы 5, 6 ,7 к 10 декабря 2022 г. – 30 %;*

*оформление пояснительной записки и графического материала к 15 декабря 2022 г. – 20 %*

*Защита курсового проекта с 19 декабря 2022 г. по 30 декабря 2022 г.*

РУКОВОДИТЕЛЬ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *Марзалюк А. В*.

(подпись)

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  *Ходосевич М.А.*

(дата и подпись студента)

Оглавление

[ВВЕДЕНИЕ 5](#_Toc121760015)

[1 ЗАДАЧА ДАННОЙ КУРСОВОЙ РАБОТЫ 6](#_Toc121760016)

[1.1 Постановка задачи 6](#_Toc121760017)

[1.2 Функционирование данной задачи 6](#_Toc121760018)

[2 СТРУКТУРА ИСПОЛЬЗУЕМЫХ ДАННЫХ 7](#_Toc121760019)

[3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ 9](#_Toc121760020)

[3.1 Диаграмма классов 9](#_Toc121760021)

[3.1 Описание классов 9](#_Toc121760022)

[4 ОПИСАНИЕ КЛАССОВ СОЗДАННЫХ ПРОГРАММОЙ 14](#_Toc121760023)

[5 РАЗРАБОТКА АЛГОРИТМОВ 15](#_Toc121760024)

[6 ТЕСТ РАБОТЫ ПРОГРАММЫ 15](#_Toc121760025)

[ЗАКЛЮЧАНИЕ 15](#_Toc121760026)

[СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 15](#_Toc121760027)

# **ВВЕДЕНИЕ**

Данная курсовая работа посвящена созданию информационной системы «Мобильный оператор».

Я выбрал данную тему в связи с тем, что на сегодняшний день мобильные операторы играют важную роль в нашей жизни. Каждый день мы делаем десятки, сотни звонков, пользуемся мобильным интернетом.

В Республике Беларусь есть несколько операторов мобильной связи. Одним из самых больших и самых хорошим является А1. Данная компания занимаю большую долю рынка Беларуси и предоставляется качественные услуги не первый десяток лет.

Мне очень нравится их деятельность, поэтому в данной курсовой работе, я попытался реализовать информационную систему, похожую на А1.

В данной курсовой работе, я буду использовать один из самых быстрых и один из самых популярных языков программирования – это язык C++.

Язык C++ является базой для информационных технологии. Именно он поможет понять основы программирования. Возможно, для начинающих он сложен для понимания, но этот язык поможет сделать такой фундамент знаний, что другие языки буду казаться простыми.

Язык С++ является одним из самых популярных на сегодняшний день.

Уже не один год занимает хорошие места в рейтинге языков программирования. Например: недавно занял 3 место среди всех языков, обогнав Java (по рейтингу TIOBE). Уступив лишь Python и языку Си.

# 

# **1 ЗАДАЧА ДАННОЙ КУРСОВОЙ РАБОТЫ**

## 1.1 Постановка задачи

Задача курсовой работы заключается в следующем: создать информационную систему с удобным пользовательским интерфейсом с пунктами меню. Система должна хранить информацию, хорошо обрабатывать информацию. Также должна быть организована работа с файлами. Должны присутствовать основные признаки ООП. А также работа с контейнерами (свой контейнер и встроенный STL). Предусмотреть обработку исключительных ситуаций.

Основные задачи:

* Создание пользователя
* Работа существующего пользователя
* Админ, который имеет ряд своих функций
* Удаление и добавление в файл
* Поиск пользователей по параметрам

## 1.2 Функционирование данной задачи

После того, как пользователь заведет свою учетную запись, ему становится доступен ряд функций: смена тарифа, смена номера, пополнение баланса, сохранение изменений, удаление аккаунта.

Админ может просмотреть всех абонентов, производить поиск по некоторым параметрам(фамилия, паспорт, номер, тариф), также может удалять пользователей.

# **2 СТРУКТУРА ИСПОЛЬЗУЕМЫХ ДАННЫХ**

Для хранения данных в данной работе используются файлы с расширением txt. Они наиболее используемые файлы по умолчанию текстовые файлы, которые обеспечивают всем, чем требуется для работы с программой.

В программе можно выделить несколько основных элементов: блок взаимодействия пользователя с программой, блок ввода, вывода, чтения и записи в файл, блок исключений, блок констант.

Блок взаимодействия пользователя с программой отвечает за выбор пользователем определённых действий, например, смена тарифа ,пополнение баланса.

В блоке исключений происходит обработка исключительных ситуаций, например, проверка на ввод чисел, проверка на ввод другого типа данных(string,char) или проверка успешного открытия файла для записи или чтения.

Блок констант используется для хранения всех констант, которые используются в программе.

В таблице 2.1 представлена структура данных, записи объекта в файл, хранящий информацию о клиентах.

Таблица 2.1 – Структура данных файла «Clients.txt»

|  |  |
| --- | --- |
| Имя | Ivan |
| Фамилия | Ivanov |
| Возраст | 25 |
| Паспорт | MC7777777 |
| Пароль | qwertyui |
| Место проживания | Minsk |
| Тариф | Plus |
| Номер телефона | +375447778880 |
| Баланс | 12.5 |

На рисунке 2.1 представлен файл, который хранит в себе данные о клиентах компании.

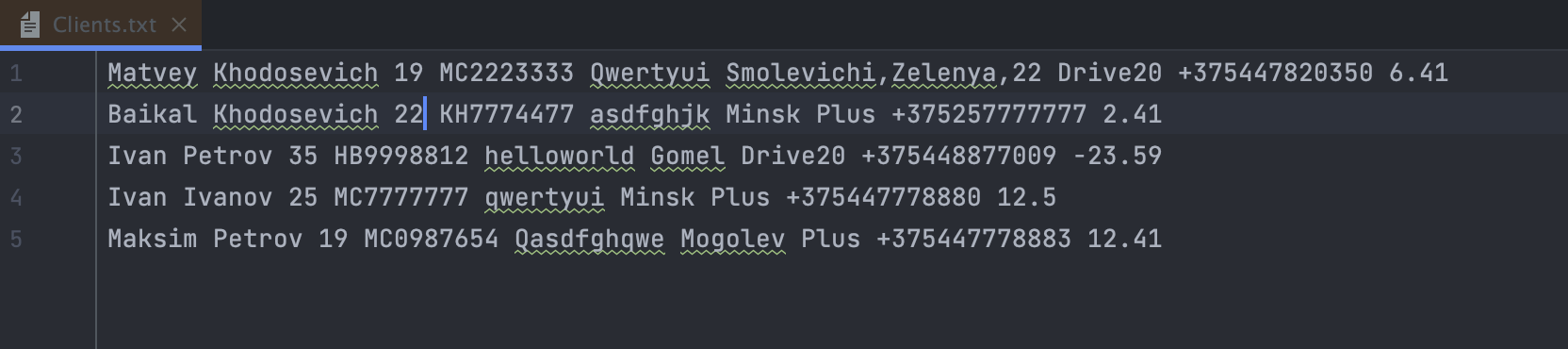


Рисунок 2.1 – Файл с клиентами компании

# **3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ**

## **3.1 Диаграмма классов**

Диаграмма классов – структурная диаграмма языка моделирования UML, демонстрирующая общую структуру иерархии классов системы, их коопераций, атрибутов, методов, интерфейсов и взаимосвязей (отношений) между ними. Диаграммы позволяют быстро разобраться в сложных системах, а также служить справочными картами, позволяющими в них ориентироваться. Она отражает связи между классами и структурами.

Программная реализация представлена такими классами, как

«Client», «User», «Admin», «Tarif», «PlusTarif», «Drive5Tarif», «Drive20Tarif», «StudentTarif», «PensionerTarif», «OwnContainer»,

«MyException», «InCorrectIntInput», «InCorrectStringInput»,

«InCorrectOpenFiles» , «InCorrectLengthPassword».

Диаграммы классов информационной системы «Мобильный оператор» представлены в приложении А.

## **3.1 Описание классов**

1. Класс Client – описывает объект клиента компании

*class Client {*

*private:*

*string FirstName;* - имяклиента

*string LastName;* - фамилияклиента

*int Age;*- возрастклиента

*string Passport;*- паспорт

*string Residence;*- местожительство

*string Number;*- номер телефонаклиента

*double BalanceAccount;*- баланс

*string NameTarif;* - название тарифа

*string PasswordAccount;*- пароль

*public:*

*Client() {}*- конструктор по умолчанию

*~Client();*- деструктор по умолчанию

*void CreateClient(Client &person1){}*- создание клиента

*void SetName(){}*- сеттер имени

*void SetLastName(){}*- сеттер фамилии

*void SetAge(){}*- сеттер возраста

*void SetPassport(){}*- сеттер паспорта

*void SetResidence(){}*- сеттер местажительства

*void SetNumber() {}* - сеттер номера

*void SetPassword(){}*- сеттер пароля

*bool CheckNumberInBase(string CheckNumber){ }*- проверка , есть ли номер в базе

*vector<Client> LoadPeopleInVector(){}*- загрузка всех клиентов в вектор

*void SetClient() {}* - сеттер клиента

*void AllTarif(){}*- информация о тарифах

*void setTarif() {}*- сеттер тарифа

*string GetPassport() {}*- геттер паспорта

*string GetPassword(){}*- геттер пароля

*void SetBalance(){}*- геттер баланса

*string GetName(){}*- геттер имении

*string GetLastName(){}* - геттер фамилии

*string GetResidence(){}*- геттер местажительсва

*int GetAge(){}* - геттер возраста

*void DepositBalance() {}* - пополнение баланса

*double GetBalance() {}*- геттер баланса

*void GetClient() {}*- геттер клиента

*string GetTarif() {}*- геттер тарифа

*string GetNumber() {}*- геттер номера

*void WriteFile(Client person1) {}*- запись клиента в файл

*friend ostream& operator << (std::ostream &os, Client &p);*- перегрузка оператора <<

*friend istream& operator >> (std::istream& in, Client& p);  };* - перегрузка оператора >>

1. Класс User

*class User: public Client{*

*public:*

*User(){}* - конструктор по умолчанию

*~User(){}* - деструктор по умолчанию

*void ForNewUser(){}* – функция для нового пользователя

*void ForBasicUser(){}* – функция для существующего пользователя

*void DeleteThisPerson(Client personDelete){}* –удаление клиента

*void WtiteFilesPersons(vector<Client> persons){}* – запись клиента в файл

*bool CheckIsEmptyClient(Client person){}* – проверка на пустого клиента, чтобы не записать в файл клиента с пустыми полями

*void CheckOnSameTarif(Client &person){}* – функция для проверки единакого тарифа, то есть, когда клиент меняет тариф, он не может выбрать тот же тариф, что и был у него

*};*

*3.*Класс Admin

*class Admin:public User{*

*private:*

*string Password = "admin1";* – пароль для входа админа

*public:*

*Admin(){}*- конструктор по умолчанию

*~Admin(){}*

*void CheckAdmin(){}* – проверка на правильность ввода пароля

*void ForAdmin(){}* – функционал админа

*void ReadFileAdmin(){}* – чтение файла

*void ReadFileAdminPasport(){}* – чтение файла по паспорту

*void ReadFileAdminNumber(){}* – чтение файла по номеру

*void ReadFileAdminLastName(){}* – чтение файла по фамилии

*void ReadFileAdminTarif() {}* – чтение файла по тарифу

*void DeleteFileAdmin(){}* – удаление клиента

*};*

1. Класс Tarif

*class Tarif*

*{*

*protected:*

*string NameTarif;* – имя тарифа

*double Price;* – цена тарифа

*int minutes;* – количество минут тарифа

*int internet;* – количество интернета тарифа

*public:*

*string virtual NameOfTarif() = 0;* – виртуальный метод для возврата имени тарифа

*double virtual Balance() = 0;* – виртуальный метод для возврата баланса

*int virtual MinBalance() = 0;* – виртуальный метод для возврата минут тарифа

*int virtual MbBalance() = 0;* – виртуальный метод для возврата интернета тарифа

*};*

4. Классы «PlusTarif», «Drive5Tarif», «Drive20Tarif», «StudentTarif», «PensionerTarif»

«PlusTarif», «Drive5Tarif», «Drive20Tarif», «StudentTarif», «PensionerTarif» эти тарифы имеют одинаковую структуру отличаются лишь своими индивидуальным характеристиками.

Рассмотрим на примере «PlusTarif»:

*class PlusTarif : public Tarif*

*{*

*public:*

*PlusTarif(){*

*NameTarif = "Plus";*

*Price = 17.59;*

*minutes = 100;*

*internet = 10000;*

*}* – конструктор

*~PlusTarif(){}* – деструктор по умолчанию

*string NameOfTarif() override{}* – переопределённый метод возращения тарифа

*double Balance() override{}* – переопределённый метод возращения стоимости

*int MinBalance() override}{}*– переопределённый метод возращения количества минут

*int MbBalance() override{ }*– переопределённый метод возращения количества интернета

*};*

5.Класс OwnContainer

Свой контейнер List.

*template <typename T>*

*class List {*

*public:*

*List();* – конструктор по умолчанию

*~List();* – деструктор по умолчанию

*void push\_back(T data);* – занесения обьекта в контейнер

*int getSize();* – геттер размера контейнера

*T& operator[](const int index);* – перегрузка оператора [ ]

*void clear();* – чистка листа

*bool isEmpty();* – проверка на пустой контейнер

*private :*

*template<typename T>*

*class Node {*

*public:*

*Node\* next;* – указатель на следующий элемент

*T data;*

*Node(T data = T(), Node\* next = nullptr) {}* – конструктор с параметрами

*};*

*int size;* - переменная размера контейнера

*Node<T>\* first;* – первый элемент контейнера

*Node<T>\* last;* – последний элемент контейнера

*};*

6.Класс MyException

*class MyException{*

*public:*

*string message;* –хранит ошибку

*public:*

*MyException(){}* – конструктор по умолчанию

*MyException(string \_msg){}* – конструктор с параметром

*~MyException() {}* – деструктор по умолчанию

*void show(){}* – вывод ошибки

*};*

7. Классы «InCorrectIntInput», «InCorrectStringInput»,

«InCorrectOpenFiles»,«InCorrectLengthPassword».

Эти классы являются наследниками *MyException* и отличаются лишь названием своих конструкторов. Также рассмотрим на примере одного класса.

Класс «InCorrectIntInput»:

*class InCorrectStringInput:public MyException {*

*public:*

*InCorrectStringInput(string message){}* – конструктор с параметрами

*};*

«InCorrectStringInput» – обработка исключительных ситуация на строки, в моем случае это ввод цифр от 0 до 9.

«InCorrectOpenFiles» – обработка исключительных ситуация на открытие файлов.

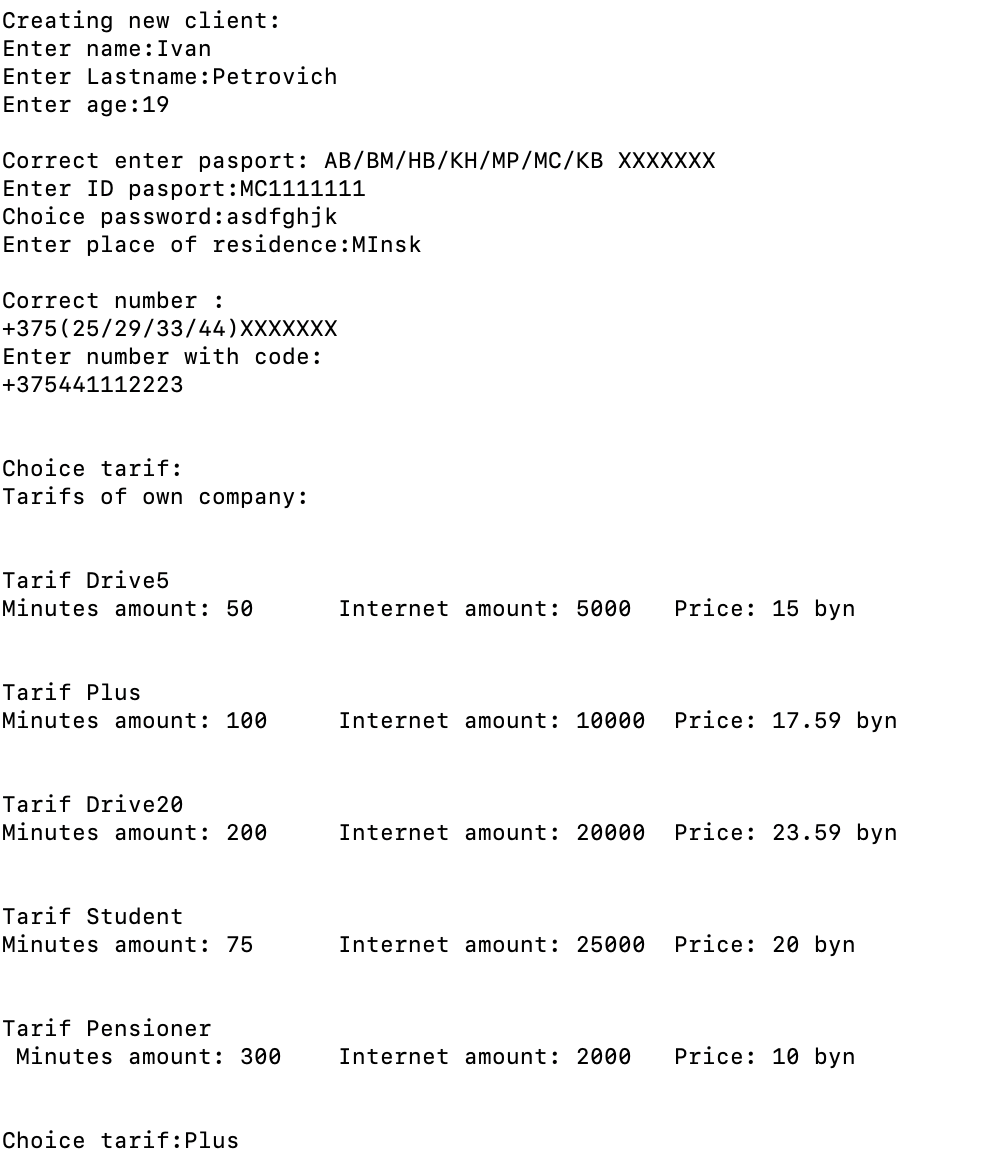
«InCorrectLengthPassword» – обработка исключительных ситуация на превышенную длину пароля.

# **4 РАЗРАБОТКА АЛГОРИТМОВ**

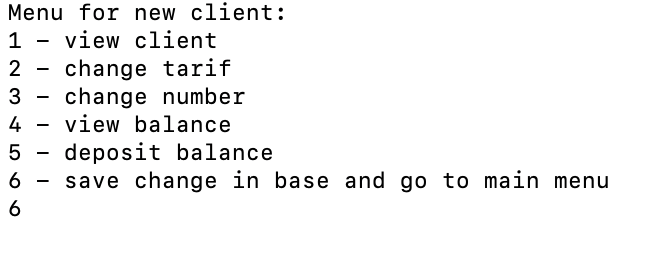
## **4.1 Алгоритмы по шагам**

Алгоритм по шагам метода User::ForNewUser(person1) – создание пользователя.

1. Начало.
2. Вызывается метод из Client::CreateClient(person1).
3. После чего вызывается метод Client::SetClient().
4. Требуется заполнить все поля.



1. После чего мы попадаем в главное меню метода User::ForNewUser()



1. Нажав «6», мы вызываем метод из Client::WtiteFile(Client person1).
2. Происходит открытие файла «Clients.txt».
3. После чего записывается обьект в файл «Clients.txt».
4. Закрытие файла «Clients.txt».
5. Выход в главное меню.

Алгоритм по шагам метода void Admin::ReadFileAdminPasport() – поиск клиента по паспорту

1. Начало.
2. Входные данные: паспорт клиента для поиска(Indificator).
3. Открытие файла fin.open(“Client.txt”).
4. Проверка на открытие файла.
5. Создаем объект Client “x” для считывания из файла.
6. Создаем vector<Client> persons.
7. Запускаем цикл while(!(fin.eof())
8. Помещаем в объект “x” клиента из файла
9. У обьекта “x” вызывает геттер паспорта(x.GetPassport)
10. Сравниваем с Indificator(if(x.GetPassport == Indificator)
11. Если совпадает заносим в persons.push\_back(x).
12. После того, как цикл закончился, закрываем файл fin.close().
13. Проверяем пустой ли наш persons.
14. Если не пуст, то вызывается метод persons.GetClient.
15. Иначе , вывод сообщения о том, что такого клиента нет.
16. Очистка vector<Client> persons.
17. Конец.

## **4.2 Схемы алгоритмов**

Схема алгоритма метода void User::ForNewUser(person1) приведена в приложении Б – создание пользователя.

Схема алгоритма метода void Admin::ReadFileAdminPasport() приведена в приложении В – поиск клиента по паспорту.

# **5 ТЕСТ РАБОТЫ ПРОГРАММЫ**

При запуске программы , мы попадаем в главное меню. Так как мы является пользователем, то выбираем 2.(рис 5.1)

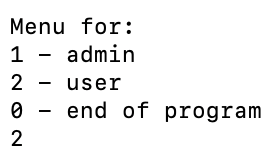
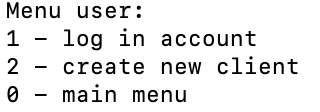


Рисунок 5.1

После чего попадаем в меню для пользователя.( рис 5.2)

**** Рисунок 5.2

# Где мы можем зайти в существующий аккаунт(log in account) – 1

Или можем создать новый аккаунт – 2

Также можно выйти в главное меню – 0

Мы являемся новый пользователем, выбираем – 2 (рис 5.3)

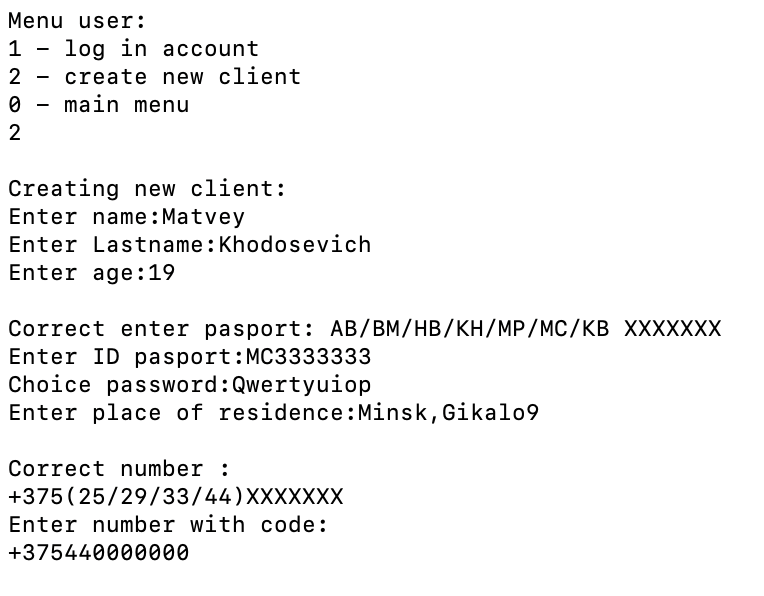


Рисунок 5.3 (продолжение на стр. 19)

# 

Рисунок 5.3(продолжение)

# Заполнив все поля, мы попадаем в меню нового пользователя.(рис 5.4)

# 

Рисунок 5.4

# Мы можем просмотреть данные – 1 (рис 5.5)

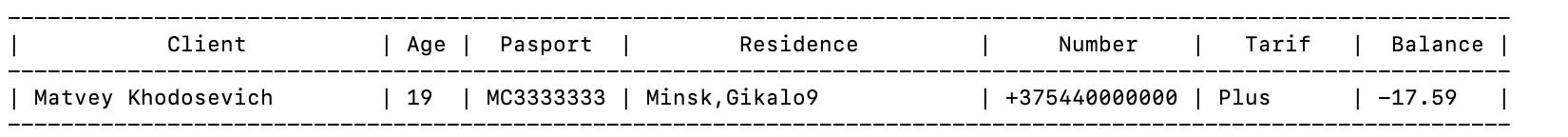


Рисунок 5.5

# 2 – смена тарифа

3 – смена номера

4 – просмотр баланса

5 – пополнение баланса

6 – сохранить клиента в базу и выйти

Пополним баланс – 5 и просмотрим его – 4 (рис 5.6 и рис 5.7)

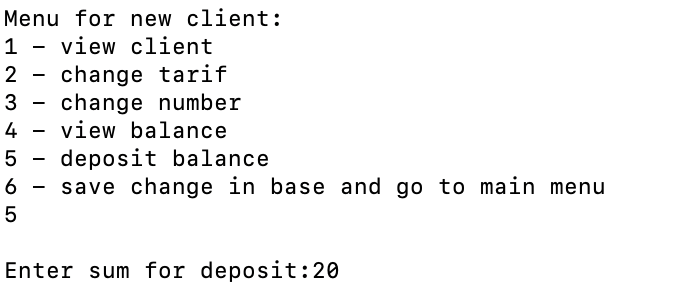


Рисунок 5.6

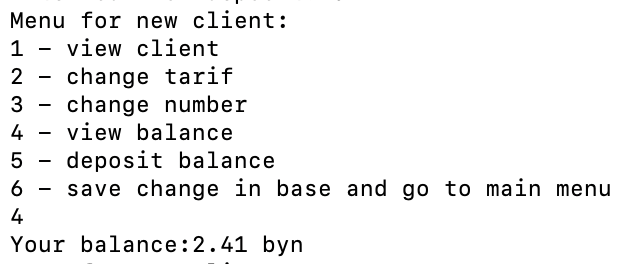


Рисунок 5.7

После чего выберем занесен в файл и выйдем – 6(рис 5.8)

# 

Рисунок 5.8

# 

Рисунок 5.9

На рисунке 5.9 мы видим, что наш клиент записался в файл.

# Теперь мы можем зайти в свой аккаунт.(рис 5.10)

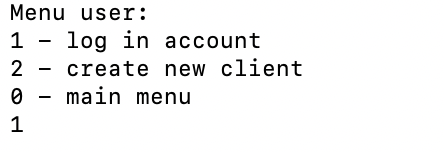


Рисунок 5.10

Нас просят ввести свой паспорт и пароль (рис 5.11)

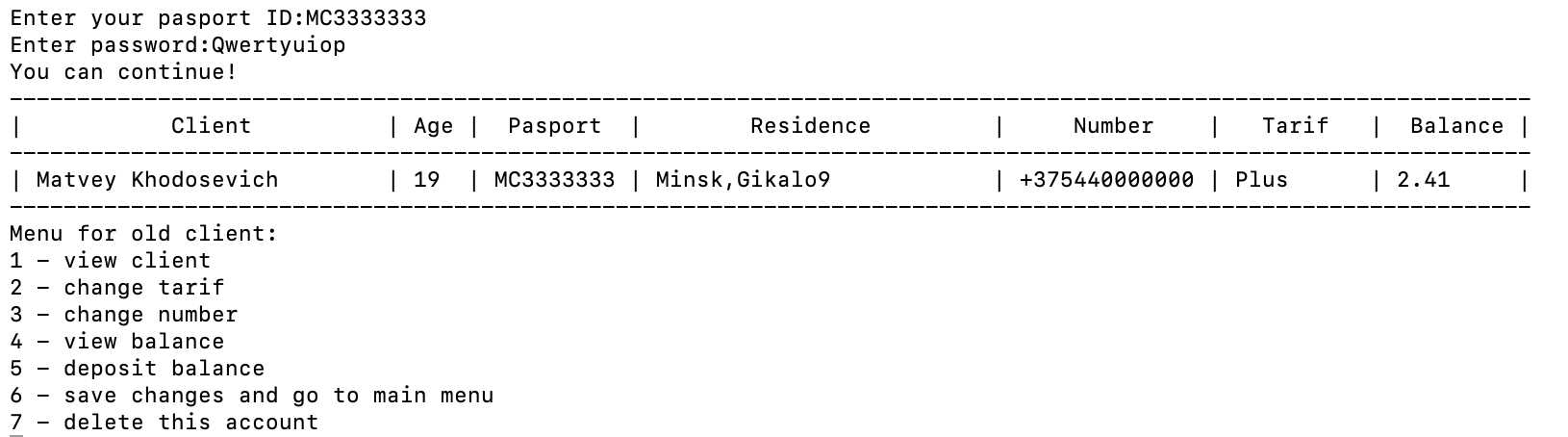


Рисунок 5.11

Введя свои данные, мы смогли зайти в наш аккаунт, функции похожие , но добавилась функция удаления аккаунта(клиента) , например , клиент решил перейти к другому оператору.

Чтобы выйти в главное меню нажимаем – 6(рис 5.12)

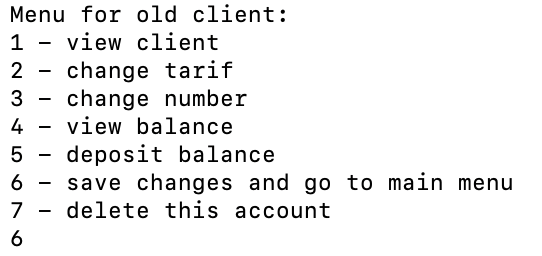
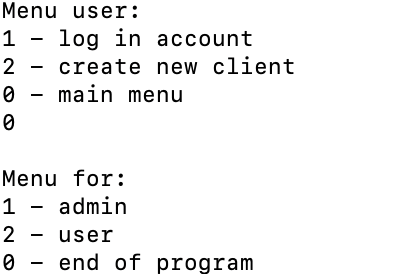


Рисунок 5.12



Теперь рассмотрим функционал админа. Чтобы мы получили меню для админа, нужно ввести пароль (рис 5.13)

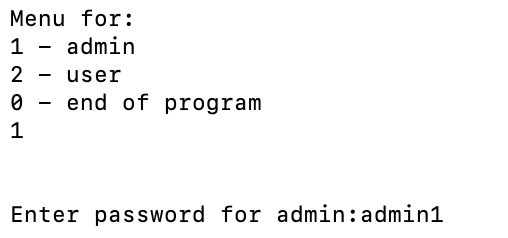


Рисунок 5.13

# После того, как ввели правильный пароль, мы видим меню админа. (рис 5.14)

# 

Рисунок 5.14

# 1 – просмотр всех клиентов

2 – удаление клиента

3 – найти клиента по паспорту

4 – найти клиента по номеру

5 – найти клиента фамилии

6 – просмотр клиентов по тарифу

Например, выберем просмотр всех клиентов – 1 ( рис 5.15)

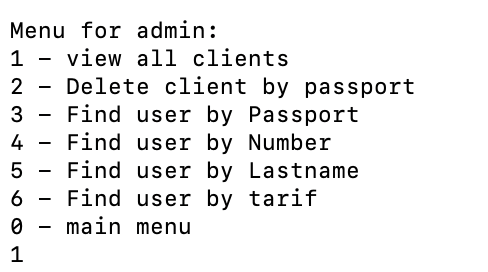


Рисунок 5.15(продолжение на стр 23 )

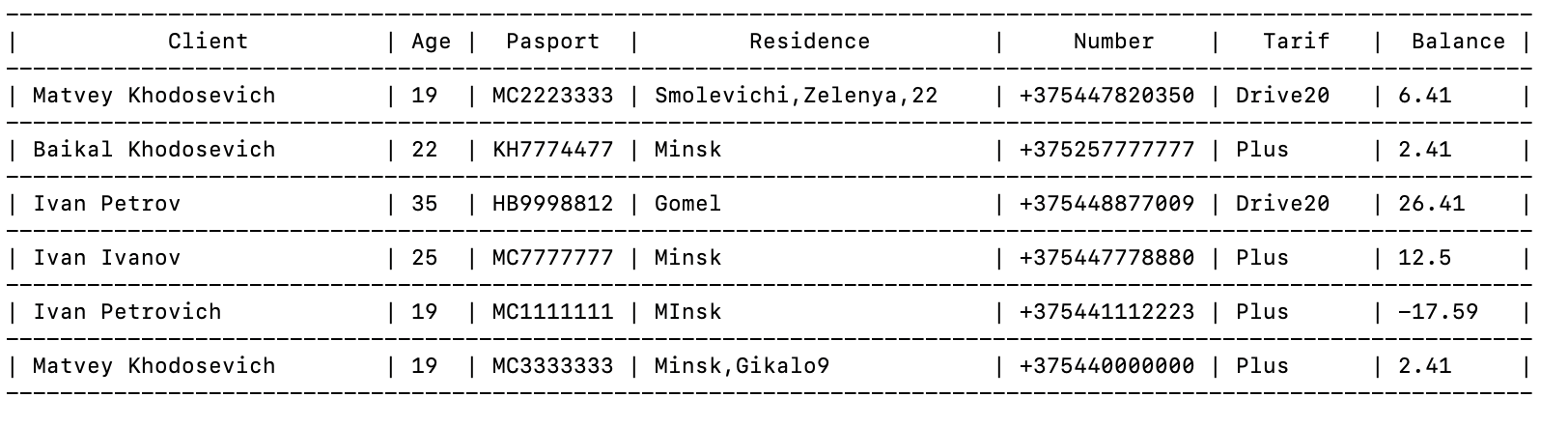


Рисунок 5.15(продолжение)

# Дальше попробуем функцию найти по фамилии – 5(рис 5.16)

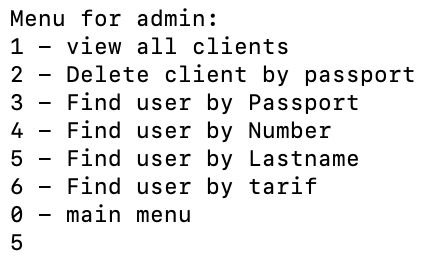
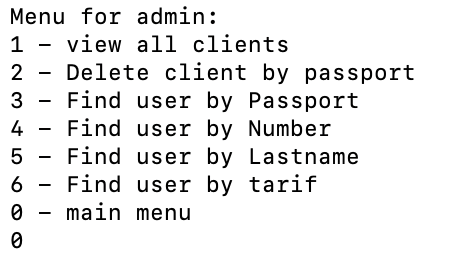




Рисунок 5.16

Дальше выходим в главное меню – 0 (рис 5.17)



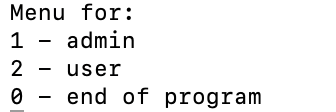


Рисунок 5.17

# **ЗАКЛЮЧАНИЕ**

# **СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ**