Выполнила

студентка группы КТбо1-2 В. Е. Денисова

Принял

доцент кафедры САиТ А. С. Свиридов

ИНДИВИДУАЛЬНАЯ РАБОТА

по дисциплине «Алгоритмизация и программирование»

**«ИНДИВИДУАЛЬНАЯ РАБОТА №1»**

Вариант 10

Таганрог 2023

Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное

учреждение высшего образования

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

(ФГАОУ ВО «ЮФУ»)

Институт компьютерных технологий и информационной безопасности

Кафедра системного анализа и телекоммуникаций

**ОГЛАВЛЕНИЕ**

Техническое задание 3

Цель задания 3

Задача 3

Ход работы 4

Алоритм решения 4

Блок-схемы 5

Написание программы 7

Пример работы 11

Вывод 13

Приложение А – Листинг 14

ИЗ1\_вар\_10.cpp 14

users.cpp....................................................................................................15

clients.cpp..................................................................................................16

service.cpp..................................................................................................17

clients.h......................................................................................................18

users.h.......................................................................................................19

service.h....................................................................................................20

Приложение B - Листинг......................................................................................21

clients.txt...................................................................................................22

service.txt..................................................................................................23

serviceclients.txt........................................................................................24

report.txt....................................................................................................25

param.txt....................................................................................................26

# Техническое задание

## Цель задания

Цель данной индивидуальной работы проверить навыки студента работы с файлами, работы со строковыми переменными, работы с функциями.

## Задача

Задание состоит в разработке программы, которая считывает настроечные параметры и формирует отчет по имеющимся данным в файлах. Отчет записывается в текстовый файл. Число записей в каждом из описанных выше файлов произвольно. Вся указанная информация представлена текстовыми файлами, каждое поле отделяется запятой, запись – это строка текста.

Имеется информация о клиентах телефонной компании и предоставляемых им услугах. Каждая услуга имеет собственный тариф, а каждый клиент может пользоваться произвольным набором услуг в течение ограниченного интервала времени (соответственно срокам договора). Имеются данные о фактическом использовании услуг.

Вариант 10.

Сформировать список услуг, не использовавшихся в первом квартале текущего года заданный пятью клиентами.

# Ход работы

## Алгоритм решения

Суть алгоритма заключается в следующем:

Идем от обратного:

Начинаем с последнего этапа и двигаемся к началу.

Находим в списке использованных услуг записи, которые противоположны нашему условию:

Это означает, что мы ищем записи, где клиенты использовали услуги в первом квартале текущего года.

Копируем эти записи в промежуточный список:

После нахождения таких записей мы копируем их в отдельный список для дальнейшей обработки.

Проходим по списку услуг:

Далее мы проходим по списку всех услуг.

Если мы обнаруживаем, что код услуги не совпадает ни с одним из кодов услуг в промежуточном списке, то наименование этой услуги добавляется в конец списка с результатами.

Таким образом, процесс состоит из обратного перебора записей, создания промежуточного списка на основе определенных условий и дальнейшей обработки полного списка услуг для получения итоговых результатов.

## БЛОК-СХЕМЫ

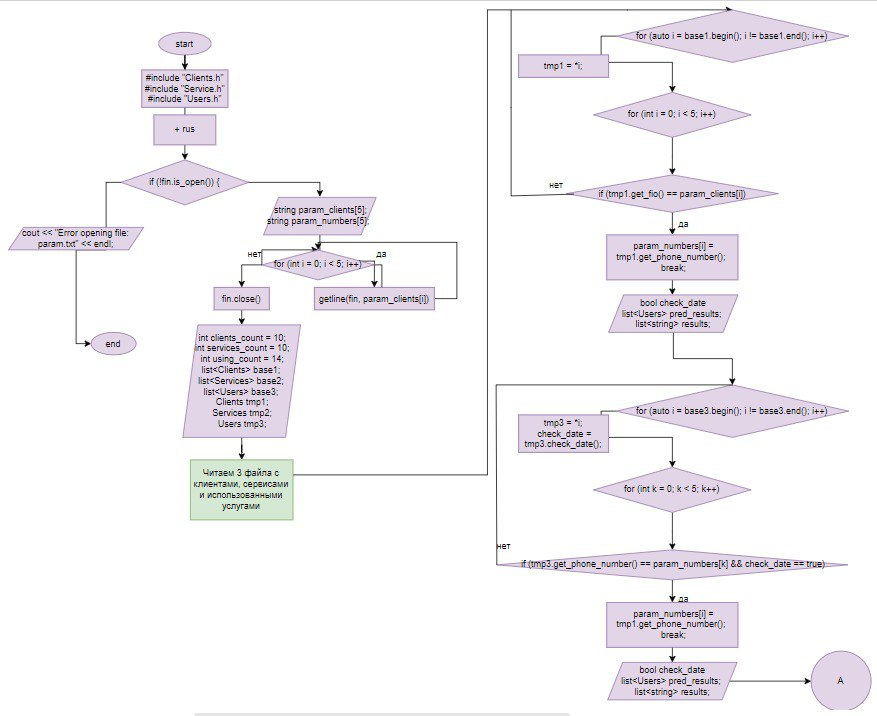
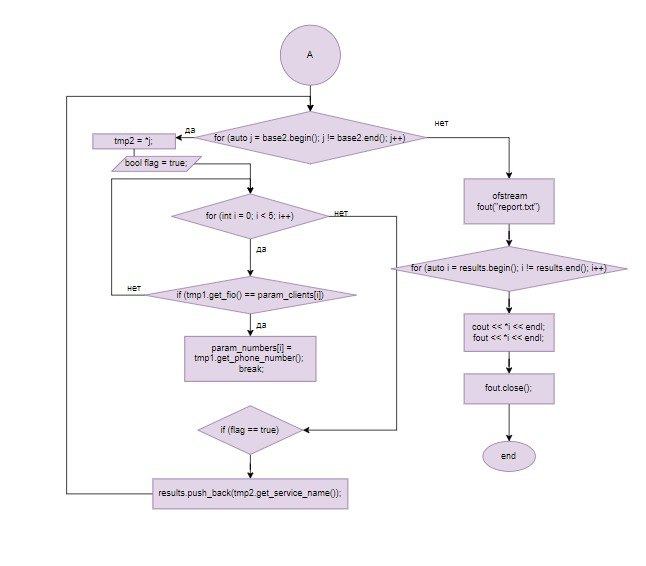


Рисунок 1

Рисунок 2

## Написание программы

Используемые библиотеки:

1. #include <iostream>
2. #include <fstream> //файловый ввод/вывод
3. #include <string>
4. #include <list> //для работы со списками

Создаем класс Clients в котором храним переменные string fio, string phone\_number, int date\_begin[3], int date\_end[3], int debt, string credit. Чтобы структурировать наш входной файл clients.

Так же создадим другие классы для других файлов:

Класс Service, где есть переменные string service\_name, int service\_code, float rate, string interval.

Класс User переменные: string phone\_number, int service\_code, int using\_date[3], int using\_time[3], string interval.

Подключаем заголовочные файлы классов #include "Clients.h", #include "Service.h", #include "Users.h".

Clients.cpp, сначала проходим по исходной строке от первого до последнего символа, если символ не является разделителем то добавляем его во временную строку если следующий символ после текущего это разделитель, или если дошли до конца строки. Чтение файла, открытие файла, проверка на открытие, временные строки для чтения даты начала и даты конца договора. Строки читаются через пробел, вместе с запятыми, удаляем запятые, переводим временные переменные даты в правильный формат и закрываем файл после чтения.

Service.cpp, функция реализуется иначе, потому что наименование услуги может состоять из любого количества слов поэтому читаем файл построчно, а потом разбиваем строку по запятым. Так же открываем файл и проверяем на открытие, если символ это не запятая, добавляем его tmp если это первая запятая, если это вторая запятая, то переводим в int и записываем в код услуги если это третья запятая, то переводим в float и записываем в тариф если запятых больше чем 3, то ошибка в формате данных в файле, когда дошли до конца строки записываем значение tmp в interval, закрываем файл.

Users.cpp, чтение из файла аналогично функции file\_reading, определение даты запуска программы, проверка, если услуга использована в этом году, если использована в первом квартале, возвращаем true, иначе - false.

Считываем ФИО из файла param.txt. Создаем списки обьетов из классов, читаем из файла список клиентов, читаем из файла список услуг, читаем из файла список использовавшихся услуг. Создаем массив, в котором будут номера соответствующих клиентов из файла с параметрами.

Сперва проходим по списку и копируем в промежуточный список записи, в которых номер клиента совпадает с одним из параметров и дата использования - 1 квартал текущего года. Проходим по списку услуг из файла service.txt проверяем, если код услуги совпадает с кодом услуги из промежуточного списка если запись нам подходит, в список результатов добавляем наименование этой услуги. Вывод результатов.

## Пример работы

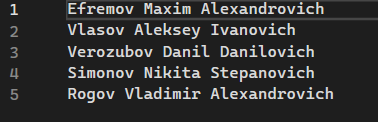


Рисунок 3 – Пример файла param.txt

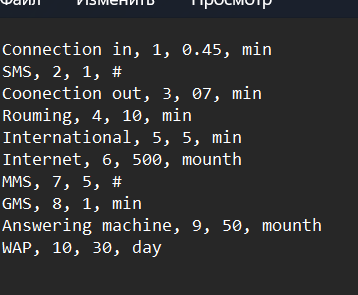


Рисунок 4 – Пример файла service.txt

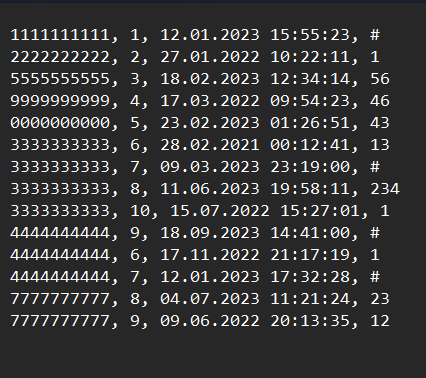


Рисунок 5 –Пример файла serviceclient.txt

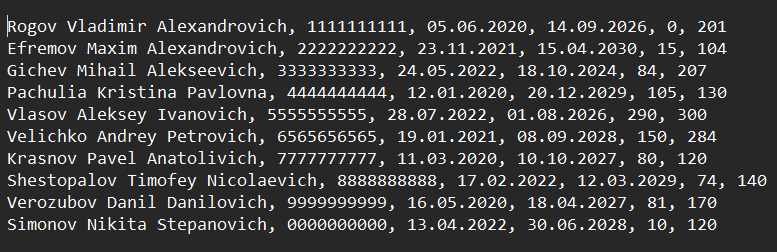


Рисунок 6 – Пример файла clients.txt

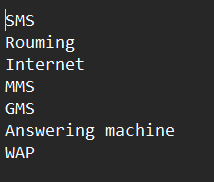


Рисунок 7 – Пример работы программы

# Вывод

Итогом работы стала программа, считывающая исходные данные из файлов и выводящая список не использовавшихся услуг за первый квартал текущего года. Были закреплены знания о работе с файлами, строками, классами, динамической памятью, временем.

# Приложение А – Листинг

**ИЗ1\_вар\_10.cpp**

/\*

ИЗ1

Задание состоит в разработке программы,

которая считывает настроечные параметры и формирует отчет по

имеющимся данным в файлах. Отчет записывается в текстовый файл.

Вариант 10:

Задание:

Сформировать список услуг, не использовавшихся в первом квартале текущего года заданный пятью переменными

\*/

//подключаем библиотеки

#include <iostream>

#include <fstream> //файловый ввод/вывод

#include <string>

#include <list> //для работы со списками

//подключаем заголовочные файлы классов

#include "Clients.h"

#include "Service.h"

#include "Users.h"

using namespace std;

int main()

{

//считываем ФИО из файла с параметрами

ifstream fin("param.txt"); //открываем файл для чтения

if (!fin.is\_open()) { //проверяем файл на открытие

cout << "Error opening file: param.txt" << endl; //если не открылася выводим ошибку

return 0; //и выходим из программы

}

string param\_clients[5]; //строковый массив с именами клиентов из файла с параметрами

for (int i = 0; i < 5; i++)

{

getline(fin, param\_clients[i]); //читаем массив параметров из файла построчно

}

fin.close(); //закрываем файл

//устанавливаем количество элементов в списках

//в зависимости от количества строк в каждом файле

int clients\_count = 10;

int services\_count = 10;

int using\_count = 14;

//создаем списки объектов классов

list<Clients> base1; //список клиентов из файла Clients.txt

list<Services> base2;

list<Users> base3; //список использованных услуг из файла Users.txt

//временные объекты классов для чтения из файла

Clients tmp1;

Services tmp2;

Users tmp3;

//читаем из файла список клиентов

for (int i = 0; i < clients\_count; i++)

{

tmp1.file\_reading("clients.txt", i); //читаем i-ую строку из файла во временную переменную

base1.push\_back(tmp1); //добавляем заполненную переменную в конец списка

}

//читаем из файла список клиентов

for (int i = 0; i < services\_count; i++)

{

tmp2.file\_reading("services.txt", i); //читаем i-ую строку из файла во временную переменную

base2.push\_back(tmp2); //добавляем заполненную переменную в конец списка

}

//читаем из файла список использованных услуг

for (int i = 0; i < using\_count; i++)

{

tmp3.file\_reading("servicesclient.txt", i);

base3.push\_back(tmp3);

}

string param\_numbers[5]; //создаем массив в котором будут номера соответствующих клиентов из файла с параметрами

for (auto i = base1.begin(); i != base1.end(); i++)//записываем эти номера в массив

{

tmp1 = \*i;

for (int i = 0; i < 5; i++)

{

if (tmp1.get\_fio() == param\_clients[i]) //если имя совпадает

{

param\_numbers[i] = tmp1.get\_phone\_number(); //запоминаем номер

break;

}

}

}

bool check\_date; //переменная, в которой будет результат проверки даты

list<Users> pred\_results; //промежуточный список из НЕПОДХОДЯЩИХ записей из списка использованных услуг

list<string> results; //список для записи окончательных результатов

//сперва проходим по списку и копируем в промежуточный список записи

//в которых номер клиента совпадает с одним из параметров и дата использования - 1 квартал текущего года:

for (auto i = base3.begin(); i != base3.end(); i++){ //проходим по списку использованных услуг

tmp3 = \*i; //копируем текущий элемент во временную переменную

check\_date = tmp3.check\_date(); //проверяем дату

for (int k = 0; k < 5; k++) //перебираем номера клиентов из параметров

{

//если номер совпадает и дата использования - 1 квартал текущего года:

if (tmp3.get\_phone\_number() == param\_numbers[k] && check\_date == true)

{

pred\_results.push\_back(tmp3); //копируем запись в промежуточный список

break; //переходим сразу к следующей записи, не перебирая остальные номера

}

}

}

//проходим по списку услуг из файла service.txt

for (auto j = base2.begin(); j != base2.end(); j++)

{

tmp2 = \*j; //копируем текущий элемент во временную переменную

bool flag = true; //флаг определяет, подходит ли нам запись или нет

//проверяем, если код услуги совпадает с кодом услуги из промежуточного списка

for (auto i = pred\_results.begin(); i != pred\_results.end(); i++)

{

tmp3 = \*i;

if (tmp2.get\_service\_code() == tmp3.get\_service\_code()) //если совпадает

{

flag = false; //то запись нам не подходит

}

}

if (flag == true) //если запись нам подходит, в список результатов добавляем наименование этой услуги

{

results.push\_back(tmp2.get\_service\_name());

}

}

//вывод результатов

ofstream fout("report.txt");

for (auto i = results.begin(); i != results.end(); i++)

{

cout << \*i << endl; //в консоль

fout << \*i << endl; //в файл

}

fout.close();

return 0; //конец программы

}

**Users.cpp**

#include "Users.h"

Users::Users()

{

//значения по умолчанию

phone\_number = "нет данных";

service\_code = -1;

using\_date[0] = 0;

using\_date[1] = 0;

using\_date[2] = 0;

using\_time[0] = 0;

using\_time[1] = 0;

using\_time[2] = 0;

interval = "нет данных";

}

void Users::file\_reading(string path, int index)

{

//чтение из файла аналогично функции file\_reading в файле ClintBase.cpp

ifstream fin(path);

if (!fin.is\_open())

{

cout << "Error opening file: Users.txt";

return;

}

string date, time, code;

for (int i = 0; i <= index; i++)

{

fin >> phone\_number;

fin >> code;

fin >> date;

fin >> time;

fin >> interval;

}

phone\_number.pop\_back();

code.pop\_back();

service\_code = stoi(code);

time.pop\_back();

StrToDateOrTime(date, using\_date);

StrToDateOrTime(time, using\_time);

fin.close();

return;

}

bool Users::check\_date() //проверка даты

{

//определение даты запуска программы

struct tm newtime;

time\_t now = time(0);

localtime\_s(&newtime, &now);

int year = newtime.tm\_year + 1900; //год

//проверка

if (using\_date[2] == year) { //если услуга использована в этом году

if (1 <= using\_date[1] <=3) {//если использована в первом квартале

return true; //возвращаем true

}

}

else return false; // иначе - false

}

**Clients.cpp**

#include "Clients.h"

Clients::Clients()

{

//в конструкторе указываем значения по умолчанию

date\_begin[0] = 0;

date\_begin[1] = 0;

date\_begin[2] = 0;

date\_end[0] = 0;

date\_end[1] = 0;

date\_end[2] = 0;

fio = "нет данных";

phone\_number = "нет данных";

debt = 0;

credit = "нет данных";

}

void Clients::StrToDateOrTime(string arg, int\* result)

{

//result[0] - day or hours

//result[1] - month or minutes

//result[2] - year or seconds

string tmp; //временная строка для копирования символов из исходной

int index = 0; //порядковый номер заполняемого элемента массива

// 0 - day or hours, 1 - month or minutes 2 - year or seconds

//проходим по исходной строке от первого до последнего символа

for (int i = 0; i < arg.length(); i++)

{

if (arg[i] != '.' && arg[i] != ':') tmp += arg[i]; //если символ не является разделителем то добавляем его во временную строку

//(разделитель для даты - '.', для времени - ':')

//если следующий символ после текущего это разделитель, или если дошли до конца строки,то

if (arg[i + 1] == '.' || arg[i + 1] == ':' || i == arg.length() - 1) {

result[index] = stoi(tmp); //переводим tmp в int и записываем в массив по соответствующему индексу

tmp = ""; //опустошаем строку

index++; //увеличиваем индекс на 1

}

}

}

//чтение файла

void Clients::file\_reading(string path, int index)

{

ifstream fin(path); //открываем файл для чтения

if (!fin.is\_open()) //проверка на открытие

{

cout << "Error opening file: Clients.txt";

return;

}

//временные строки для чтения даты начала и даты конца договора

string date\_1, date\_2;

string debt1;

string n1, n2, n3;

//чтобы считать N-ую строку из файла цикл выполняется N раз

for (int i = 0; i <= index; i++)

{

//строки читаются через пробел, вместе с запятыми

fin >> n1;

fin >> n2;

fin >> n3;

fin >> phone\_number;

fin >> date\_1;

fin >> date\_2;

fin >> debt1;

fin >> credit;

}

//удаляем запятые с концов строк

n3.pop\_back();

fio = n1 + ' ' + n2 + ' ' + n3;

phone\_number.pop\_back();

date\_1.pop\_back();

date\_2.pop\_back();

debt1.pop\_back();

debt = stoi(debt1);

//переводим временные переменные даты в правильный формат

StrToDateOrTime(date\_1, date\_begin);

StrToDateOrTime(date\_2, date\_end);

fin.close(); //закрываем файл после окончания чтения

return;

}

**Service.cpp**

#include "Service.h"

Services::Services()

{

//значения по умолчанию

service\_name = "нет данных";

service\_code = -1;

rate = -1;

interval = "нет данных";

}

void Services::file\_reading(string path, int index)

{

//функция реализуется иначе, потому что наименование услуги может состоять из любого количества слов

//поэтому читаем файл построчно, а потом разбиваем строку по запятым

ifstream fin(path); //открываем файл для чтения

if (!fin.is\_open()) //проверка на открытие

{

cout << "Error opening file: Service.txt";

return;

}

string line; //временная строка для построчного чтения файла

//чтобы считать N-ую строку из файла цикл выполняется N раз

for (int i = 0; i <= index; i++)

{

getline(fin, line);

}

string tmp; //временная строка для копирования символов из line

int points\_count = 0; //счетчик количества запятых

for (int i = 0; i < line.length(); i++) //проходим по строке line

{

if (line[i] != ',') tmp += line[i]; //если символ это не запятая, добавляем его tmp

if (line[i] == ',') //если символ это запятая,

{

switch (points\_count) { //то в зависимости от текущего количества запятых:

case 0: {

service\_name = tmp; //если это первая запятая, то записываем в наименование услуги

tmp = ""; //опустошаем временную строку

points\_count++; //увеличиваем количество запятых

}

break;

case 1: {

service\_code = stoi(tmp); //если это вторая запятая, то переводим в int и записываем в код услуги

tmp = ""; //опустошаем временную строку

points\_count++; //увеличиваем количество запятых

}break;

case 2: {

rate = stof(tmp); //если это третья запятая, то переводим в float и записываем в тариф

tmp = ""; //опустошаем временную строку

points\_count++; //увеличиваем количество запятых

}break;

default: { //если запятых больше чем 3, то ошибка в формате данных в файле

cout << "Ошибка формата данных: Service.txt";

return;

}

}

//когда дошли до конца строки

if (i == line.length() - 1)

{

interval = tmp; //записываем значение tmp в interval

}

}

}

fin.close(); //закрываем файл

return;

}

**Clients.h**

#pragma once

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

//основной класс Clients для работы со списком клиентов

class Clients

{

private:

//переменные-члены класса

string fio;

string phone\_number;

int date\_begin[3];

int date\_end[3];

int debt;

string credit;

public:

Clients(); //конструктор класса

//функции для доступа к соответствующим переменным-членам класса

string get\_fio() { return fio; };

string get\_phone\_number() { return phone\_number; };

int get\_debt() { return debt; }

/\*file\_reading - чтение строки из файла

параметры:

path - имя файла для чтения

index - порядковый номер строки, которую нужно прочитать

\*/

void file\_reading(string path, int index);

/\*

StrToDateOrTime - перевод строки в формат даты или времени

параметры:

string arg - исходная строка

int\* result - имя целочисленного массива для записи

\*/

void StrToDateOrTime(string arg, int\* result);

};

**Users.h**

#pragma once

#include "Clients.h"

//класс UsingService является наследником от класса ClientBase

class Users :

public Clients

{

private:

//переменные-члены класса

string phone\_number;

int service\_code;

int using\_date[3];

int using\_time[3];

string interval;

public:

Users(); //конструктор класса

//функции для доступа к соответствующим переменным-членам класса

string get\_phone\_number() { return phone\_number; }

int get\_service\_code() { return service\_code; }

int\* get\_using\_date() { return using\_date; }

int\* get\_using\_time() { return using\_time; }

string get\_interval() { return interval; }

//функция проверяет, использовал ли клиент услугу в предыдущем месяце

//если НЕ использовал возвращает - true

//иначе - false

bool check\_date();

//чтение файла и вывод на консоль

void file\_reading(string path, int index);

};

**Service.h**

#pragma once

#include "Clients.h"

//класс ServiceBase является наследником от класса ClientBase

class Services :

public Clients

{

private:

//переменные-члены класса

string service\_name;

int service\_code;

float rate;

string interval;

public:

Services(); //конструктор класса

//функции для доступа к соответствующим переменным-членам класса

string get\_service\_name() { return service\_name; }

int get\_service\_code() { return service\_code; }

float get\_rate() { return rate; }

string get\_interval() { return interval; }

//чтение файла и вывод на консоль

void file\_reading(string path, int index);

};

**Приложение B - Листинг:**

**Clients.txt**

Rogov Vladimir Alexandrovich, 1111111111, 05.06.2020, 14.09.2026, 0, 201

Efremov Maxim Alexandrovich, 2222222222, 23.11.2021, 15.04.2030, 15, 104

Gichev Mihail Alekseevich, 3333333333, 24.05.2022, 18.10.2024, 84, 207

Pachulia Kristina Pavlovna, 4444444444, 12.01.2020, 20.12.2029, 105, 130

Vlasov Aleksey Ivanovich, 5555555555, 28.07.2022, 01.08.2026, 290, 300

Velichko Andrey Petrovich, 6565656565, 19.01.2021, 08.09.2028, 150, 284

Krasnov Pavel Anatolivich, 7777777777, 11.03.2020, 10.10.2027, 80, 120

Shestopalov Timofey Nicolaevich, 8888888888, 17.02.2022, 12.03.2029, 74, 140

Verozubov Danil Danilovich, 9999999999, 16.05.2020, 18.04.2027, 81, 170

Simonov Nikita Stepanovich, 0000000000, 13.04.2022, 30.06.2028, 10, 120

**Service.txt**

Connection in, 1, 0.45, min

SMS, 2, 1, #

Coonection out, 3, 07, min

Rouming, 4, 10, min

International, 5, 5, min

Internet, 6, 500, mounth

MMS, 7, 5, #

GMS, 8, 1, min

Answering machine, 9, 50, mounth

WAP, 10, 30, day

**Serviceclients.txt**

1111111111, 1, 12.01.2023 15:55:23, #

2222222222, 2, 27.01.2022 10:22:11, 1

5555555555, 3, 18.02.2023 12:34:14, 56

9999999999, 4, 17.03.2022 09:54:23, 46

0000000000, 5, 23.02.2023 01:26:51, 43

3333333333, 6, 28.02.2021 00:12:41, 13

3333333333, 7, 09.03.2023 23:19:00, #

3333333333, 8, 11.06.2023 19:58:11, 234

3333333333, 10, 15.07.2022 15:27:01, 1

4444444444, 9, 18.09.2023 14:41:00, #

4444444444, 6, 17.11.2022 21:17:19, 1

4444444444, 7, 12.01.2023 17:32:28, #

7777777777, 8, 04.07.2023 11:21:24, 23

7777777777, 9, 09.06.2022 20:13:35, 12

**Param.txt**

Efremov Maxim Alexandrovich

Vlasov Aleksey Ivanovich

Verozubov Danil Danilovich

Simonov Nikita Stepanovich

Rogov Vladimir Alexandrovich