

МИНОБРАЗОВАНИЯ РОССИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт компьютерных технологий и информационной безопасности  
Кафедра физико-математических основ инженерного образования

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ №1  
по дисциплине «Алгоритмизация и Программирование»  
ВАРИАНТ 3

Выполнил  
студент группы КТб02-1 \_\_\_\_\_ Д.В. Манькус

Принял  
доцент кафедры САиТ \_\_\_\_\_ А.С. Свиридов

# Содержание

<b>1</b>	<b>Техническое задание</b>	<b>3</b>
1.1	Задание . . . . .	3
1.2	Описание задания . . . . .	3
1.3	Требования к представлению результатов работы программы . .	3
<b>2</b>	<b>Выполнение работы</b>	<b>5</b>
<b>3</b>	<b>Исходный код программы</b>	<b>8</b>
<b>4</b>	<b>Тексты файлов данных</b>	<b>14</b>
<b>5</b>	<b>Контрольный пример</b>	<b>15</b>

# 1 Техническое задание

## 1.1 Задание

Программирование на языке Си/C++. 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.

## 1.2 Описание задания

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.

Вся указанная информация представлена текстовыми файлами, структура которых выглядит следующим образом.

Файл информации о клиентах содержит фамилию, имя, отчество клиента, номер телефона, дату заключения договора, дату окончания договора, размер задолженности, допустимый кредит. Каждое поле отделяется запятой, запись – это строка текста.

Файл информации об услугах состоит также из записей, состоящих из полей, разделенных запятыми. Каждая запись включает наименование услуги, ее код, тариф (в рублях), временной интервал измерения ( мин., сутки, месяц. Если временной привязки нет, ставится символ ).

Файл информации об услугах, оказанных клиентам, включает в себя записи, состоящие из полей номера телефона, кода услуги, даты и времени ее использования в секундах. Знак используется в случае, если время не определяется (например, отправляется СМС).

## 1.3 Требования к представлению результатов работы программы

Каждый файл хранится в том же каталоге, что и разработанная программа. Формат файлов текстовый, каждый из них создан и редактируется текстовым редактором.

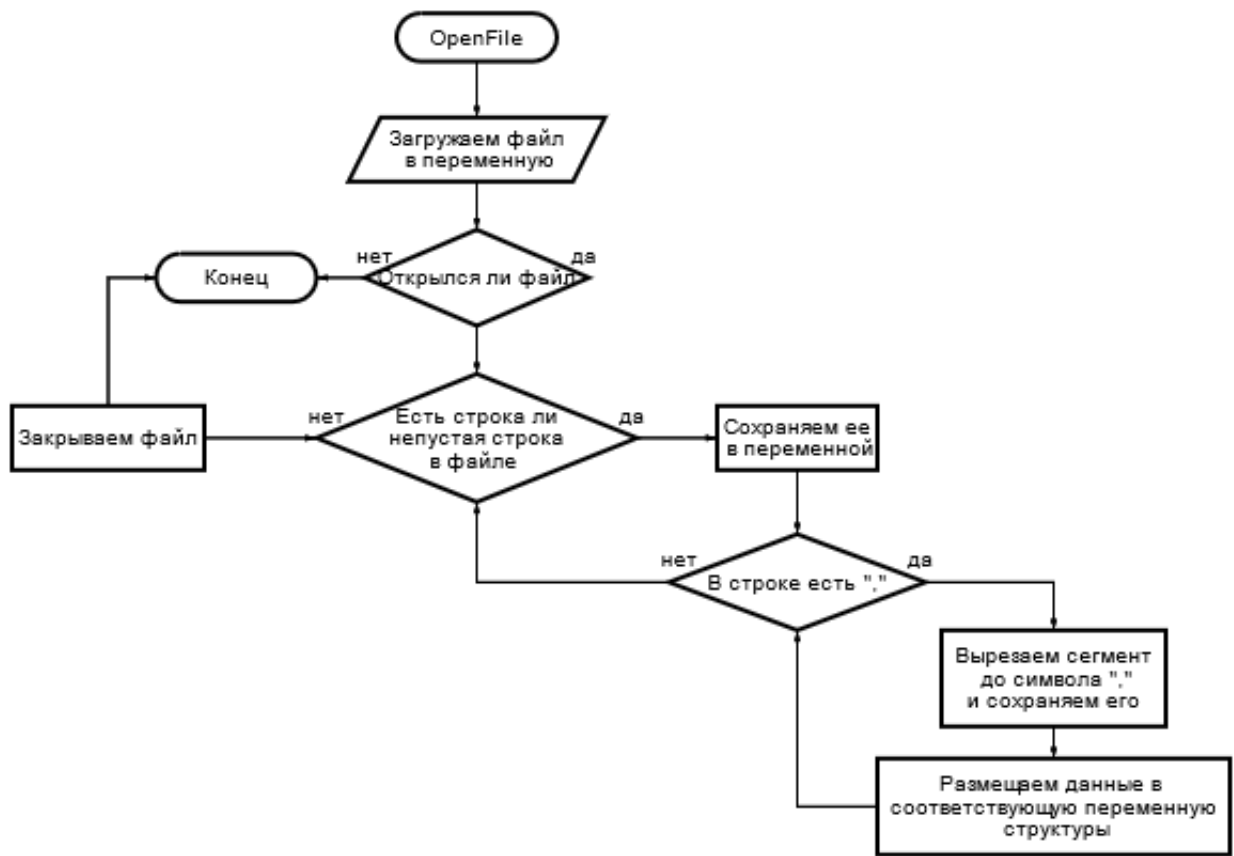
Параметры настройки записываются в текстовый файл с именем *Param.ini*. Каждый параметр – в отдельной строке. Даны наименования двух услуг, дата начала диапазона, дата конца диапазона.

Построить список клиентов, использовавших услуги двух наименований в указанный временной промежуток ( с ... по ....). Результирующий отчет (результат обработки) записывается в файл с именем *Report.txt*. Если информация отсутствует, вывести в файл строку «Нет данных».

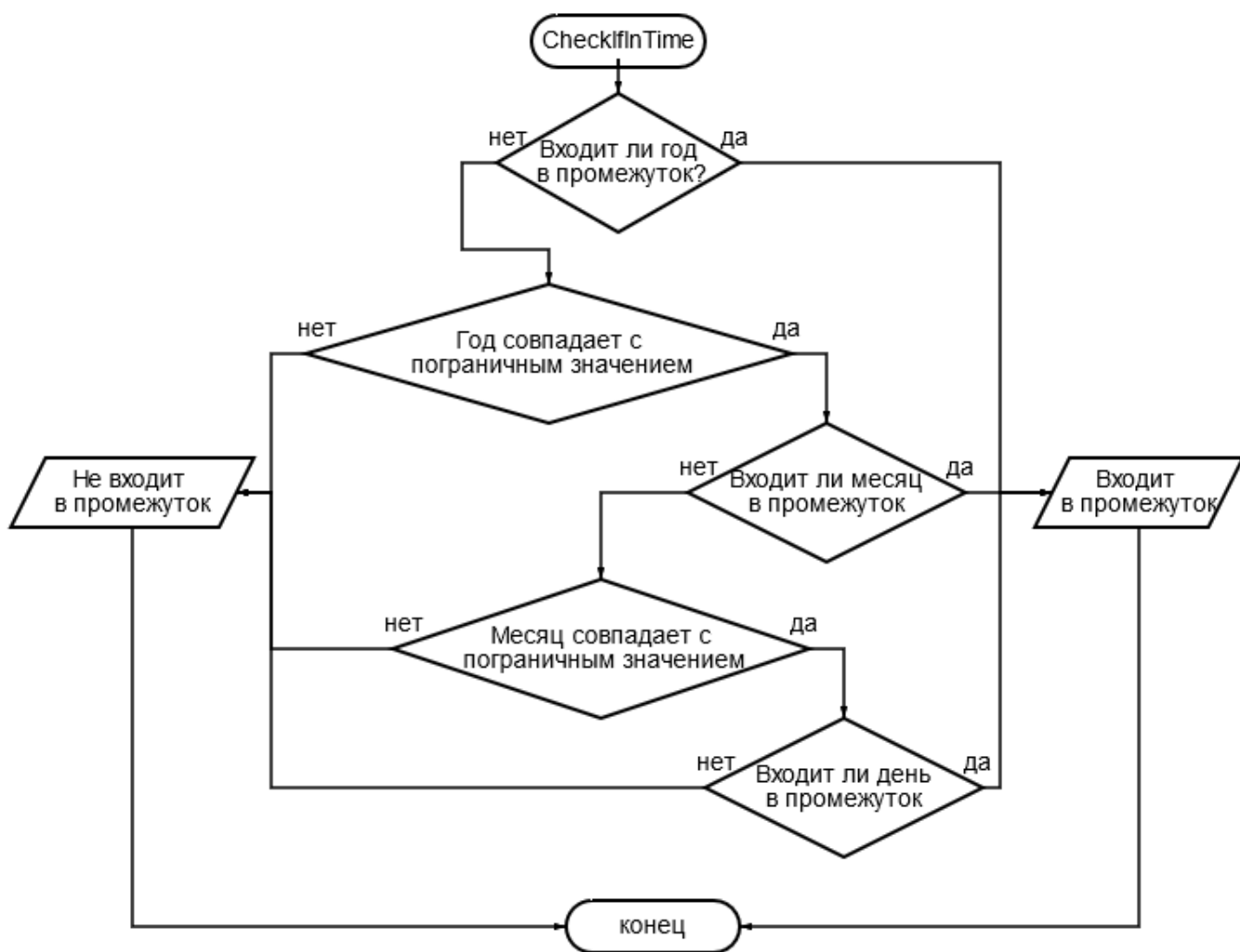
Вывести сообщение в окно программы о результатах её выполнения – есть или отсутствуют данные в результирующем файле. Язык сообщения английский либо русский на транслите.

Формат данных в файле *Report.txt*: каждая строка списка отделяется символами перевода строки. Если в строке несколько полей, они разделяются символами «,».

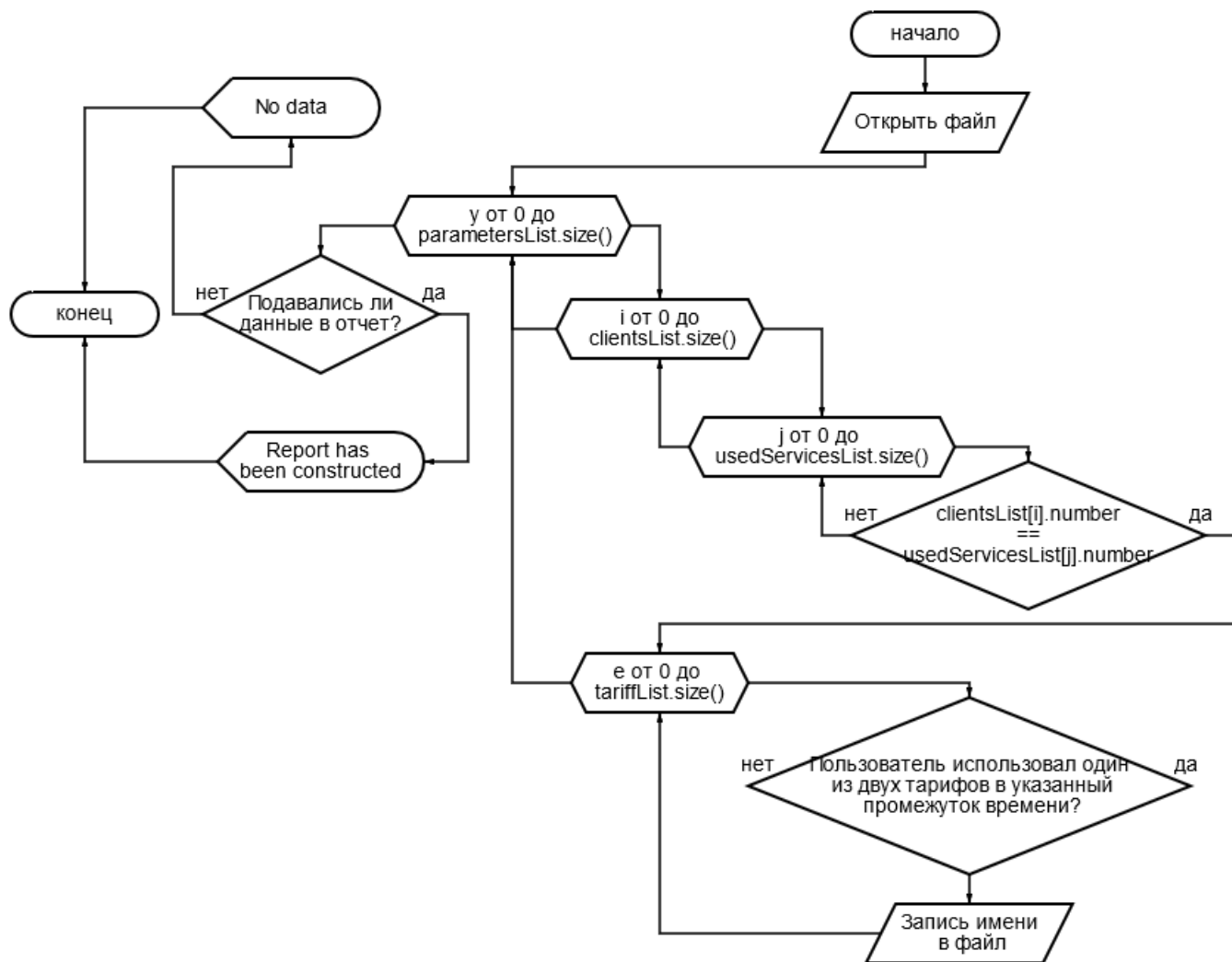
## 2 Выполнение работы



Блок-Схема подпрограммы открытия файлов



Блок-Схема подпрограммы выявления соответствия по времени



Блок-Схема основной части программы

### 3 Исходный код программы

Листинг 1: PTask.cpp

```
1 // , 3
2
3 #include <iostream>
4 #include <fstream>
5 #include <sstream>
6 #include <string>
7 #include <vector>
8 #include "PTask.h"
9
10 #define DEBUG
11
12 void openClients(std::vector<client>& clientsList) {
13     std::ifstream file;
14     unsigned int cnt = 0;
15     std::string token;
16     file.open("Clients.txt");
17     if (file.is_open()) {
18         std::string line;
19         while (std::getline(file, line)) {
20             size_t pos = 0;
21             int index = 0;
22             clientsList.push_back({});
23             while ((pos = line.find(',', pos)) != std::string::npos) {
24                 token = line.substr(0, pos);
25                 switch (index) {
26                     case 0:
27                         clientsList[cnt].name = token;
28                         break;
29                     case 1:
30                         clientsList[cnt].number = token;
31                         break;
32                     case 2:
33                         clientsList[cnt].sdate = token;
34                         break;
35                     case 3:
36                         clientsList[cnt].edate = token;
37                         break;
38                     case 4:
39                         clientsList[cnt].debt = stod(token);
40                         break;
41                     case 5:
42                         clientsList[cnt].credit = stod(token);
43                         break;
44                 }
45                 line.erase(0, pos + 1);
46                 index++;
47             }
48
49             cnt++;
50         }
51         file.close();
52     }
53     else {
54         std::cout << "Unable to open file";
55     }
56 }
57
58
```



```

59 void openTariffs(std::vector<tariff>& tariffList) {
60     std::ifstream file;
61     unsigned int cnt = 0;
62     std::string token;
63     file.open("Tariffs.txt");
64     if (file.is_open()) {
65         std::string line;
66         while (std::getline(file, line)) {
67             size_t pos = 0;
68             int index = 0;
69             tariffList.push_back({});
70             while ((pos = line.find(',', pos)) != std::string::npos) {
71                 token = line.substr(0, pos);
72                 switch (index) {
73                     case 0:
74                         tariffList[cnt].name = token;
75                         break;
76                     case 1:
77                         tariffList[cnt].id = stod(token);
78                         break;
79                     case 2:
80                         tariffList[cnt].rate = stod(token);
81                         break;
82                     case 3:
83                         tariffList[cnt].type = token;
84                         break;
85                 }
86                 line.erase(0, pos + 1);
87                 index++;
88             }
89             tariffList[cnt].type = line;
90             cnt++;
91         }
92         file.close();
93     }
94     else {
95         std::cout << "Unable to open file";
96     }
97 }
98
99 void openUsedServices(std::vector<uses>& usedServicesList) {
100     std::ifstream file;
101     unsigned int cnt = 0;
102     std::string token;
103     file.open("UsedServices.txt");
104     if (file.is_open()) {
105         std::string line;
106         while (std::getline(file, line)) {
107             size_t pos = 0;
108             int index = 0;
109             usedServicesList.push_back({});
110             while ((pos = line.find(',', pos)) != std::string::npos) {
111                 token = line.substr(0, pos);
112                 switch (index) {
113                     case 0:
114                         usedServicesList[cnt].number = token;
115                         break;
116                     case 1:
117                         usedServicesList[cnt].id = stod(token);
118                         break;
119                     case 2:
120                         usedServicesList[cnt].date = token;
121                         break;

```

```

122         case 3:
123             usedServicesList[cnt].timeused = stod(token);
124             break;
125         }
126         line.erase(0, pos + 1);
127         index++;
128     }
129     cnt++;
130 }
131 file.close();
132 }
133 else {
134     std::cout << "Unable to open file";
135 }
136 }
137
138 void openParam(std::vector<parameter>& parametersList) {
139     std::ifstream file;
140     unsigned int cnt = 0;
141     std::string token;
142     file.open("Param.ini");
143     if (file.is_open()) {
144         std::string line;
145         while (std::getline(file, line)) {
146             size_t pos = 0;
147             int index = 0;
148             parametersList.push_back({});
149             while ((pos = line.find(',', pos)) != std::string::npos) {
150                 token = line.substr(0, pos);
151                 switch (index) {
152                     case 0:
153                         parametersList[cnt].nameone = token;
154                         break;
155                     case 1:
156                         parametersList[cnt].nametwo = token;
157                         break;
158                     case 2:
159                         parametersList[cnt].sdate = token;
160                         break;
161                 }
162                 line.erase(0, pos + 1);
163                 index++;
164             }
165             parametersList[cnt].edate = line;
166             cnt++;
167         }
168         file.close();
169     }
170     else {
171         std::cout << "Unable to open file";
172     }
173 }
174
175 bool checkIfInTime(std::string sdate, std::string edate, std::string date) {
176     char discard;
177     int day, month, year;
178     int startDay, startMonth, startYear;
179     int endDay, endMonth, endYear;
180
181     std::stringstream ss1(sdate);
182     ss1 >> startDay >> discard >> startMonth >> discard >> startYear;
183     std::stringstream ss2(edate);
184     ss2 >> endDay >> discard >> endMonth >> discard >> endYear;

```

```

185     std::stringstream ss3(date);
186     ss3 >> day >> discard >> month >> discard >> year;
187     if (year > startYear && year < endYear) return true;
188     else if (year == startYear || year == endYear) {
189         if (month > startMonth && month < endMonth) return true;
190         else if (month == startMonth || month == endMonth) {
191             if (day >= startDay && day <= endDay) return true;
192             else return false;
193         }
194     }
195     return false;
196 }
197
198 int main() {
199     std::setlocale(LC_ALL, "Russian");
200
201     std::vector<client> clientsList;
202     std::vector<tariff> tariffList;
203     std::vector<uses> usedServicesList;
204     std::vector<parameter> parametersList;
205
206     openClients(clientsList);
207     openTariffs(tariffList);
208     openUsedServices(usedServicesList);
209     openParam(parametersList);
210
211     #ifdef DEBUG
212
213     for (int i = 0; i != 4; i++) {
214         std::cout << "Name: " << clientsList[i].name << std::endl;
215         std::cout << "Number: " << clientsList[i].number << std::endl;
216         std::cout << "Start Date: " << clientsList[i].sdate << std::endl;
217         std::cout << "End Date: " << clientsList[i].edate << std::endl;
218         std::cout << "Debt: " << clientsList[i].debt << std::endl;
219         std::cout << "Credit: " << clientsList[i].credit << std::endl;
220     }
221
222     std::cout << "-----" << std::endl;
223     for (int i = 0; i != 5; i++) {
224         std::cout << "Name: " << tariffList[i].name << std::endl;
225         std::cout << "ID: " << tariffList[i].id << std::endl;
226         std::cout << "Rate: " << tariffList[i].rate << std::endl;
227         std::cout << "Type: " << tariffList[i].type << std::endl;
228     }
229
230     std::cout << "-----" << std::endl;
231     for (int i = 0; i != 3; i++) {
232         std::cout << "Number: " << usedServicesList[i].number << std::endl;
233         std::cout << "ID: " << usedServicesList[i].id << std::endl;
234         std::cout << "Date of use: " << usedServicesList[i].date << std::endl;
235         std::cout << "Time used: " << usedServicesList[i].timeused << std::endl;
236     }
237
238     std::cout << "-----" << std::endl;
239     for (int i = 0; i != 2; i++) {
240         std::cout << "name one: " << parametersList[i].nameone << std::endl;
241         std::cout << "name two: " << parametersList[i].nametwo << std::endl;
242         std::cout << "Start Date: " << parametersList[i].sdate << std::endl;
243         std::cout << "End Date: " << parametersList[i].edate << std::endl;
244     }
245
246     std::cout << "-----" << std::endl;
247     for (int i = 0; i != 3; i++) {

```

```

248     std::cout << parametersList[i].sdate << std::endl;
249     std::cout << parametersList[i].edate << std::endl;
250     std::cout << usedServicesList[i].date << std::endl;
251     if (checkIfInTime(parametersList[i].sdate, parametersList[i].edate,
252         usedServicesList[i].date)) std::cout << "True" << std::endl;
253     else std::cout << "False" << std::endl;
254 }
255 #endif
256
257 bool nodata = true;
258 std::ofstream file("Report.txt");
259 if (file.is_open()) {
260     for (int y = 0; y < parametersList.size(); y++) {
261         for (int i = 0; i < clientsList.size(); i++) {
262             for (int j = 0; j < usedServicesList.size(); j++) {
263                 if (clientsList[i].number == usedServicesList[j].number) {
264                     for (int e = 0; e < tariffList.size(); e++) {
265                         if (usedServicesList[j].id == tariffList[e].id &&
266                             (tariffList[e].name==parametersList[y].nameone ||
267                                 tariffList[e].name == parametersList[y].nametwo) &&
268                             checkIfInTime(parametersList[y].sdate,
269                                 parametersList[y].edate,
270                                 usedServicesList[j].date)) {
271                             file << clientsList[i].name << std::endl;
272                             nodata = false;
273                         }
274                     }
275                 }
276             }
277         }
278     }
279     if (nodata) { file << "No Data"; std::cout << "No data to report"; }
280     else { std::cout << "Report has been constructed"; }
281     file.close();
282 }
283 else {
284     std::cout << "Unable to open file";
285 }
286
287 return 0;
288 }

```

## Листинг 2: PTask.h

```

1  // header file
2
3  #pragma once
4
5  struct client {                // Struct for clients
6      std::string name;         // Name of the client
7      std::string number;       // Phone number of the client
8      std::string sdate;        // Starting date of their subscription
9      std::string edate;        // Ending date of their subscription
10     double debt;               // Debt of the client
11     double credit;             // Available credit
12 };
13
14 struct tariff {                // Struct for tariffs
15     std::string name;          // Name of the tariff
16     unsigned int id;           // ID of specific tariff
17     double rate;               // Price per type
18     std::string type;          // The frequency of payment for the tariff

```

```

19 };
20
21 struct uses {           // Struct for used tariffs
22     std::string number;  // Phone number of the user
23     unsigned int id;     // ID of the used tariff
24     std::string date;    // Date of use (includes time of use)
25     unsigned int timeused; // How long the tariff was active
26 };
27
28 struct parameter {      // Struct for parameters of the programm
29     std::string nameone; // Name of the first tariff
30     std::string nametwo; // Name of the second tariff
31     std::string sdate;   // Start of the timeframe
32     std::string edate;   // End of the timeframe
33 };

```

## 4 Тексты файлов данных

### **Clients.txt**

Иванов Иван Иванович,9773672365,12.10.2012,12.10.2014,0,0  
Кузнецов Александр Игоревич,79338889911,25.11.2007,25.11.2021,2500,70000  
Петров Иван Васильевич,9734672311,22.01.2008,01.11.2011,210,200  
Васильев Илья Васильевич,9714679805,05.09.2010,01.12.2012,0,2000  
Иванов Иван Иванович,79991234567,01.05.2020,01.05.2022,5000,100000  
Петрова Елена Сергеевна,79876543210,15.09.2019,15.09.2021,3000,80000  
Смирнов Алексей Владимирович,79871112233,30.12.2020,30.12.2022,7000,120000  
Козлова Ольга Петровна,79995554433,20.06.2020,20.06.2022,2000,60000  
Морозов Игорь Дмитриевич,79117778899,10.03.2019,10.03.2021,4000,90000  
Николаева Анастасия Васильевна,79223332211,05.08.2020,05.08.2022,6000,110000

### **Tariffs.txt**

Связь внутри сети,1,0.30,мин  
СМС,2,0.15,  
Тариф безграничных возможностей,3,3,мин  
Экстремальная подключенность,4,12,час  
Связь с другими мобильными сетями,5,0.50,мин  
Роуминг,6,10,мин  
Международный тариф,7,50,мин  
Безграничная связь,8,5,мин  
Ультра-скоростная сеть,9,10,сек  
Гиперкоммуникация,10,20,

### **UsedServices.txt**

9734672311,1,13.02.2008 13:01:55,300  
9734672311,2,28.11.2012 01:32:30,  
9757282392,5,23.09.2013 19:14:00,54  
79876543210,7,08.10.2023 14:30:45,3600  
79871112233,9,15.06.2022 09:20:10,1800  
79338889911,3,15.06.2022 09:20:10,1800  
79661234567,2,20.12.2024 17:45:30,  
79338889911,2,24.04.2010 13:37:20,  
9773672365,1,25.09.2021 12:15:25,1500  
79117778899,7,03.07.2022 19:00:55,2200

## 5 Контрольный пример

### Param.ini

Связь внутри сети,1,0.30,мин

СМС,2,0.15,#

Тариф безграничных возможностей,3,3,мин

Экстремальная подключенность,4,12,час

Связь с другими мобильными сетями,5,0.50,мин

Роуминг,6,10,мин

Международный тариф,7,50,мин

Безграничная связь,8,5,мин

Ультра-скоростная сеть,9,10,сек

Гиперкоммуникация,10,20,#

### Report.txt

Кузнецов Александр Игоревич

Кузнецов Александр Игоревич

Петров Иван Васильевич

Кузнецов Александр Игоревич

Петров Иван Васильевич