

## 1. Вариант задания на курсовой проект

Вариант задания на курсовой проект формируется из нескольких компонентов:

1. предметная область (табл. 1);
2. метод хеширования (табл. 2);
3. метод сортировки (табл. 3);
4. вид списка (табл. 4);
5. метод обхода дерева (табл. 5);
6. алгоритм поиска слова в тексте (табл. 6).

Номер варианта для конкретного компонента определяется студентом как три последние цифры номера его студенческого билета, взятые по модулю количества вариантов для конкретного компонента, то есть

$$N_{\text{вар}} = nnn \bmod K,$$

где

$N_{\text{вар}}$  – номер варианта;

$nnn$  – три последние цифры номера студенческого билета;

$K$  – количество вариантов заданий для конкретного компонента.

Таблица 1

Номер п/п	Предметная область
0	Обслуживание читателей в библиотеке (см. п. 2.2.1)
1	Обслуживание клиентов в бюро проката автомобилей (см. п. 2.2.2)
2	Регистрация постояльцев в гостинице (см. п. 2.2.3)
3	Регистрация больных в поликлинике (см. п. 2.2.4)
4	Продажа авиабилетов (см. п. 2.2.5)
5	Обслуживание клиентов оператора сотовой связи (см. п. 2.2.6)

Таблица 2

Номер п/п	Метод хеширования
0	Открытое хеширование
1	Закрытое хеширование с линейным опробованием
2	Закрытое хеширование с квадратичным опробованием
3	Закрытое хеширование с двойным хешированием

Таблица 3

Номер п/п	Метод сортировки
0	Подсчетом
1	Включением
2	Извлечением
3	Пузырьковый
4	Быстрый (Хоара)
5	Слиянием
6	Распределением

Таблица 4

Номер п/п	Вид списка
0	Линейный однонаправленный
1	Линейный двунаправленный
2	Циклический однонаправленный
3	Циклический двунаправленный
4	Слоеный

Таблица 5

Номер п/п	Метод обхода дерева
0	Симметричный
1	Обратный
2	Прямой

Таблица 6

Номер п/п	Алгоритм поиска слова в тексте
0	Боуера и Мура (БМ)
1	Прямой

По итогам вычислений (номер студ 741) мой вариант курсовой состоит из следующих компонентов:

1. Обслуживание клиентов оператора сотовой связи (таб.1 №5)
2. Закрытое хеширование с квадратичным апробированием (таб.2 № 3)
3. Метод сортировки распределением (был оговорён с преподавателем т.к считается не эффективным, поэтому использовался метод слияния таб.3 №5)
4. Линейный однонаправленный список (таб.4 №0)
5. Метод обхода дерева: симметричный (таб.5 №0)
6. Алгоритм поиска слова в тексте: Прямой (таб.6 №1)

## 2. Описание программы

В данном курсовом проекте были использованы следующие структуры и алгоритмы данных:

1. Линейный однонаправленный список используется для хранения данных о выдаче сим-карт.

Каждый узел списка состоит из

```
string passport; // номер паспорта клиента
string number; // номер сим-карты
string start_date; // дата выдачи
string finish_date; // дата окончания обслуживания
bool is_returned; // false - выдали, true - вернули
Record* next; // указатель на следующий узел
```

Действия над списком:

1. Сортировка слиянием `void MergeSort(Record** headRef)`
2. Загрузка списка из файла `void List::load(string filename)`
3. Вставка в список `void List::insert(Record n)`
4. Ввод новой строки выдачи `void Record::input()`

2. Клиенты хранятся в виде сбалансированного дерева

Для отображения узлов дерева была создана структура, состоящая из

```
string passport; // номер паспорта
string info; // кем и когда выдан
string fio; // фιο владельца
int year; // год рождения
string address; // адрес
```

Действия над деревом:

1. Загрузка структуры в дерево `void load(istream& in);`
2. Ввод новых клиентов `void input();`
3. Печать дерева `void print();`
4. Правый поворот вокруг p `Client* rotateright(Client* p);`
5. Левый поворот вокруг q `Client* rotateleft(Client* q);`
6. Балансировка узла p `Client* balance(Client* p);`
7. Вставка в дерево `Client* insert(Client* p, Client k);`
8. Удаление клиента по номеру паспорта `Client* remove(Client* p, string passport);`
9. Поиск в дереве по паспорту `Client* find(Client* p, string passport);`

10. Поиск по части ФИО или адреса `void find_by_info(Client* p, string info);`

### 3. Сим-карты хранятся в виде хеш-таблицы

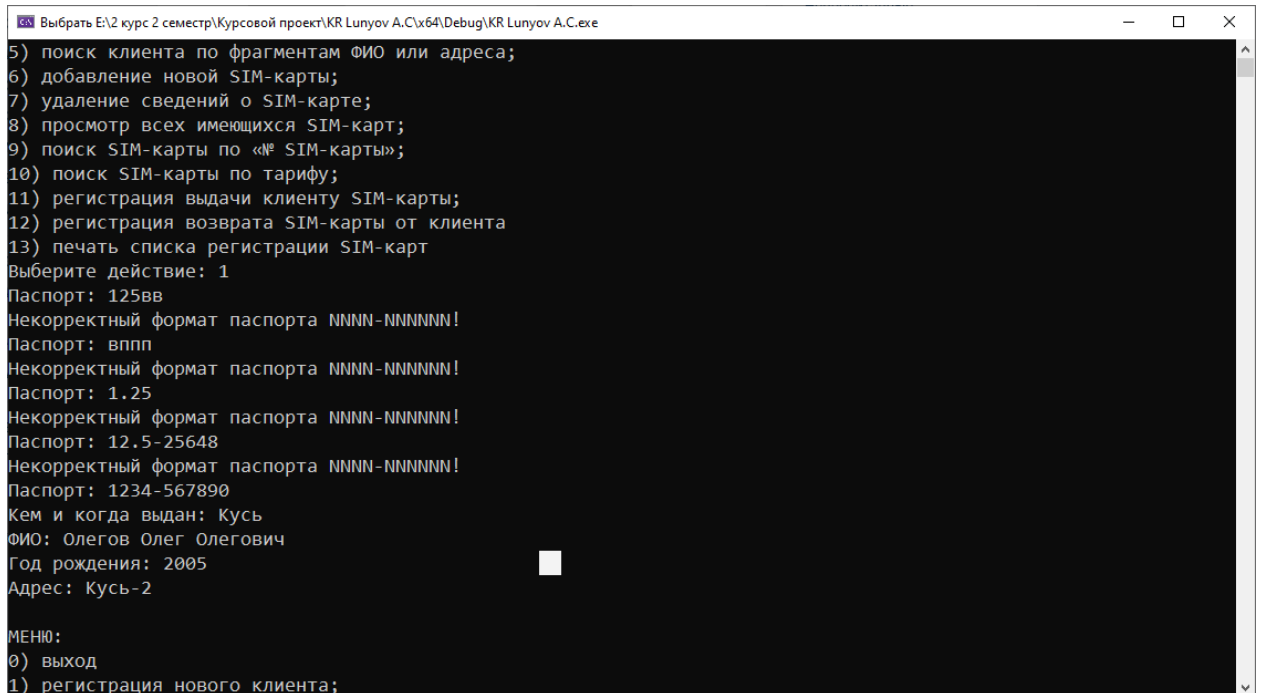
Каждый элемент хэш-таблицы — это структура, состоящая из

```
string sim; // номер сим-карты
string tarif; // тариф сим-карты
int year; // год выдачи
bool exist; признак наличия
```

Действия над хеш-таблицей:

1. Загрузка `void load(istream& in);`
2. Печать в консоль `void input();`
3. Регистрация новой сим-карты `void input();`
4. Удаление элементов из хеш-таблицы `bool remove(K key)`
5. Поиск в хеш-таблице по номеру сим-карты `Sim* find(string sim)`
6. Поиск в хеш-таблице по тарифу `void findByTarif(string tarif)`

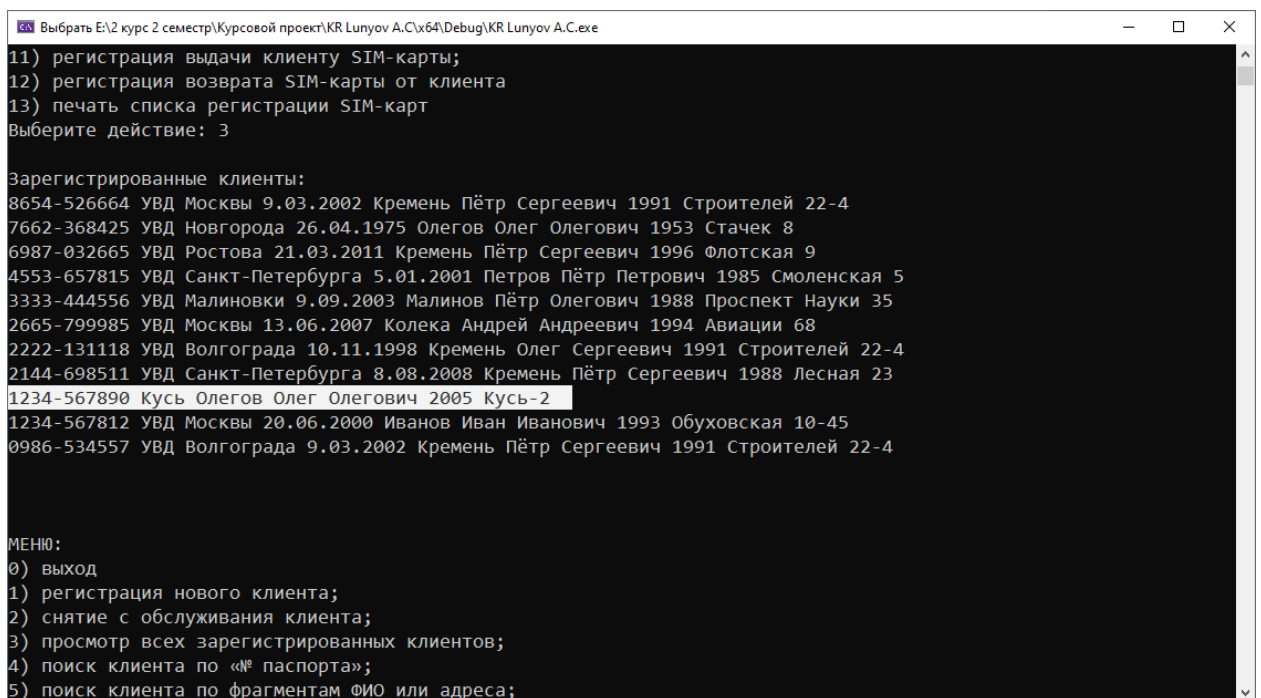
### 3. Тестирование программы



```
Выбрать E:\2 курс 2 семестр\Курсовой проект\KR Lunyov A.C\64\Debug\KR Lunyov A.C.exe
5) поиск клиента по фрагментам ФИО или адреса;
6) добавление новой SIM-карты;
7) удаление сведений о SIM-карте;
8) просмотр всех имеющихся SIM-карт;
9) поиск SIM-карты по «№ SIM-карты»;
10) поиск SIM-карты по тарифу;
11) регистрация выдачи клиенту SIM-карты;
12) регистрация возврата SIM-карты от клиента
13) печать списка регистрации SIM-карт
Выберите действие: 1
Паспорт: 125vv
Некорректный формат паспорта NNNN-NNNNNN!
Паспорт: vppp
Некорректный формат паспорта NNNN-NNNNNN!
Паспорт: 1.25
Некорректный формат паспорта NNNN-NNNNNN!
Паспорт: 12.5-25648
Некорректный формат паспорта NNNN-NNNNNN!
Паспорт: 1234-567890
Кем и когда выдан: Кусь
ФИО: Олегов Олег Олегович
Год рождения: 2005
Адрес: Кусь-2

МЕНЮ:
0) выход
1) регистрация нового клиента;
```

Рисунок 1 – Тестирование регистрации новых клиентов



```
Выбрать E:\2 курс 2 семестр\Курсовой проект\KR Lunyov A.C\64\Debug\KR Lunyov A.C.exe
11) регистрация выдачи клиенту SIM-карты;
12) регистрация возврата SIM-карты от клиента
13) печать списка регистрации SIM-карт
Выберите действие: 3

Зарегистрированные клиенты:
8654-526664 УВД Москвы 9.03.2002 Кремень Пётр Сергеевич 1991 Строителей 22-4
7662-368425 УВД Новгорода 26.04.1975 Олегов Олег Олегович 1953 Стачек 8
6987-032665 УВД Ростова 21.03.2011 Кремень Пётр Сергеевич 1996 Флотская 9
4553-657815 УВД Санкт-Петербурга 5.01.2001 Петров Пётр Петрович 1985 Смоленская 5
3333-444556 УВД Малиновки 9.09.2003 Малинов Пётр Олегович 1988 Проспект Науки 35
2665-799985 УВД Москвы 13.06.2007 Колека Андрей Андреевич 1994 Aviации 68
2222-131118 УВД Волгограда 10.11.1998 Кремень Олег Сергеевич 1991 Строителей 22-4
2144-698511 УВД Санкт-Петербурга 8.08.2008 Кремень Пётр Сергеевич 1988 Лесная 23
1234-567890 Кусь Олегов Олег Олегович 2005 Кусь-2
1234-567812 УВД Москвы 20.06.2000 Иванов Иван Иванович 1993 Обуховская 10-45
0986-534557 УВД Волгограда 9.03.2002 Кремень Пётр Сергеевич 1991 Строителей 22-4

МЕНЮ:
0) выход
1) регистрация нового клиента;
2) снятие с обслуживания клиента;
3) просмотр всех зарегистрированных клиентов;
4) поиск клиента по «№ паспорта»;
5) поиск клиента по фрагментам ФИО или адреса;
```

Рисунок 2 – Тестирование просмотра всех зарегистрированных клиентов

```
E:\2 курс 2 семестр\Курсовой проект\KR Lunyov A.C\64\Debug\KR Lunyov A.C.exe
4) поиск клиента по «№ паспорта»;
5) поиск клиента по фрагментам ФИО или адреса;
6) добавление новой SIM-карты;
7) удаление сведений о SIM-карте;
8) просмотр всех имеющихся SIM-карт;
9) поиск SIM-карты по «№ SIM-карты»;
10) поиск SIM-карты по тарифу;
11) регистрация выдачи клиенту SIM-карты;
12) регистрация возврата SIM-карты от клиента
13) печать списка регистрации SIM-карт
Выберите действие: 2
Введите паспорт клиента для снятия с обслуживания: 1153f
Нет клиента с таким паспортом

МЕНЮ:
0) выход
1) регистрация нового клиента;
2) снятие с обслуживания клиента;
3) просмотр всех зарегистрированных клиентов;
4) поиск клиента по «№ паспорта»;
5) поиск клиента по фрагментам ФИО или адреса;
6) добавление новой SIM-карты;
7) удаление сведений о SIM-карте;
8) просмотр всех имеющихся SIM-карт;
9) поиск SIM-карты по «№ SIM-карты»;
10) поиск SIM-карты по тарифу;
11) регистрация выдачи клиенту SIM-карты;
```

Рисунок 3 – Тестирование снятия с обслуживания клиента

```
E:\2 курс 2 семестр\Курсовой проект\KR Lunyov A.C\64\Debug\KR Lunyov A.C.exe
7) удаление сведений о SIM-карте;
8) просмотр всех имеющихся SIM-карт;
9) поиск SIM-карты по «№ SIM-карты»;
10) поиск SIM-карты по тарифу;
11) регистрация выдачи клиенту SIM-карты;
12) регистрация возврата SIM-карты от клиента
13) печать списка регистрации SIM-карт
Выберите действие: 4
Введите паспорт клиента: 4155
Нет клиента с таким паспортом

МЕНЮ:
0) выход
1) регистрация нового клиента;
2) снятие с обслуживания клиента;
3) просмотр всех зарегистрированных клиентов;
4) поиск клиента по «№ паспорта»;
5) поиск клиента по фрагментам ФИО или адреса;
6) добавление новой SIM-карты;
7) удаление сведений о SIM-карте;
8) просмотр всех имеющихся SIM-карт;
9) поиск SIM-карты по «№ SIM-карты»;
10) поиск SIM-карты по тарифу;
11) регистрация выдачи клиенту SIM-карты;
12) регистрация возврата SIM-карты от клиента
13) печать списка регистрации SIM-карт
Выберите действие: _
```

Рисунок 4 – Тестирование поиска клиенту по номеру паспорта

```
Выбрать E:\2 курс 2 семестр\Курсовой проект\KR Lunyov A.C\64\Debug\KR Lunyov A.C.exe
4) поиск клиента по «№ паспорта»;
5) поиск клиента по фрагментам ФИО или адреса;
6) добавление новой SIM-карты;
7) удаление сведений о SIM-карте;
8) просмотр всех имеющихся SIM-карт;
9) поиск SIM-карты по «№ SIM-карты»;
10) поиск SIM-карты по тарифу;
11) регистрация выдачи клиенту SIM-карты;
12) регистрация возврата SIM-карты от клиента
13) печать списка регистрации SIM-карт
Выберите действие: 4
Введите паспорт клиента: 4553-657815
4553-657815 УВД Санкт-Петербурга 5.01.2001 Петров Пётр Петрович 1985 Смоленская 5

МЕНЮ:
0) выход
1) регистрация нового клиента;
2) снятие с обслуживания клиента;
3) просмотр всех зарегистрированных клиентов;
4) поиск клиента по «№ паспорта»;
5) поиск клиента по фрагментам ФИО или адреса;
6) добавление новой SIM-карты;
7) удаление сведений о SIM-карте;
8) просмотр всех имеющихся SIM-карт;
9) поиск SIM-карты по «№ SIM-карты»;
10) поиск SIM-карты по тарифу;
11) регистрация выдачи клиенту SIM-карты;
```

Рисунок 5 – Тестирование поиска клиенту по номеру паспорта

```
Выбрать E:\2 курс 2 семестр\Курсовой проект\KR Lunyov A.C\64\Debug\KR Lunyov A.C.exe
6) добавление новой SIM-карты;
7) удаление сведений о SIM-карте;
8) просмотр всех имеющихся SIM-карт;
9) поиск SIM-карты по «№ SIM-карты»;
10) поиск SIM-карты по тарифу;
11) регистрация выдачи клиенту SIM-карты;
12) регистрация возврата SIM-карты от клиента
13) печать списка регистрации SIM-карт
Выберите действие: 5
Введите фрагмент ФИО или адреса для поиска: ё
3333-444556 УВД Малиновки 9.09.2003 Малинов Пётр Олегович 1988 Проспект Науки 35
0986-534557 УВД Волгограда 9.03.2002 Кремень Пётр Сергеевич 1991 Строителей 22-4
2144-698511 УВД Санкт-Петербурга 8.08.2008 Кремень Пётр Сергеевич 1988 Лесная 23
6987-032665 УВД Ростова 21.03.2011 Кремень Пётр Сергеевич 1996 Флотская 9
4553-657815 УВД Санкт-Петербурга 5.01.2001 Петров Пётр Петрович 1985 Смоленская 5
8654-526664 УВД Москвы 9.03.2002 Кремень Пётр Сергеевич 1991 Строителей 22-4

МЕНЮ:
0) выход
1) регистрация нового клиента;
2) снятие с обслуживания клиента;
3) просмотр всех зарегистрированных клиентов;
4) поиск клиента по «№ паспорта»;
5) поиск клиента по фрагментам ФИО или адреса;
6) добавление новой SIM-карты;
7) удаление сведений о SIM-карте;
8) просмотр всех имеющихся SIM-карт;
```

Рисунок 6 – Тестирование поиска клиенту по фрагменту ФИО или адреса



```
E:\2 курс 2 семестр\Курсовой проект\KR Lunyov A.C\64\Debug\KR Lunyov A.C.exe
11) регистрация выдачи клиенту SIM-карты;
12) регистрация возврата SIM-карты от клиента
13) печать списка регистрации SIM-карт
Выберите действие: 8

Список сим-карт:
759-1552443 Вместе 2004 Нет в наличии
111-8963447 Домашний 2000 В наличии
165-6665458 Всё включено 2005 В наличии
166-2345985 Вне сети 2007 Нет в наличии
222-2226440 Всё включено 2005 В наличии
956-6254895 Домашний 2010 В наличии
488-4888520 Домашний 2010 В наличии
333-1554781 Корпоративный 1998 Нет в наличии
669-8965529 Вместе 2009 Нет в наличии
336-3365140 Роуминг 2004 Нет в наличии

МЕНЮ:
0) выход
1) регистрация нового клиента;
2) снятие с обслуживания клиента;
3) просмотр всех зарегистрированных клиентов;
4) поиск клиента по «№ паспорта»;
5) поиск клиента по фрагментам ФИО или адреса;
6) добавление новой SIM-карты;
7) удаление сведений о SIM-карте;
8) просмотр всех имеющихся SIM-карт;
```

Рисунок 7 – Тестирование просмотра всех сим-карт

```
E:\2 курс 2 семестр\Курсовой проект\KR Lunyov A.C\64\Debug\KR Lunyov A.C.exe
Выберите действие: 7
Введите номер сим-карты для удаления:
333-1554781

МЕНЮ:
0) выход
1) регистрация нового клиента;
2) снятие с обслуживания клиента;
3) просмотр всех зарегистрированных клиентов;
4) поиск клиента по «№ паспорта»;
5) поиск клиента по фрагментам ФИО или адреса;
6) добавление новой SIM-карты;
7) удаление сведений о SIM-карте;
8) просмотр всех имеющихся SIM-карт;
9) поиск SIM-карты по «№ SIM-карты»;
10) поиск SIM-карты по тарифу;
11) регистрация выдачи клиенту SIM-карты;
12) регистрация возврата SIM-карты от клиента
13) печать списка регистрации SIM-карт
Выберите действие: 8

Список сим-карт:
759-1552443 Вместе 2004 Нет в наличии
111-8963447 Домашний 2000 В наличии
165-6665458 Всё включено 2005 В наличии
166-2345985 Вне сети 2007 Нет в наличии
222-2226440 Всё включено 2005 В наличии
```

Рисунок 8 – Тестирование удаления сведений о сим-карте

```
E:\2 курс 2 семестр\Курсовой проект\KR Lunyov A.C\Debug\KR Lunyov A.C.exe
5) поиск клиента по фрагментам ФИО или адреса;
6) добавление новой SIM-карты;
7) удаление сведений о SIM-карте;
8) просмотр всех имеющихся SIM-карт;
9) поиск SIM-карты по «№ SIM-карты»;
10) поиск SIM-карты по тарифу;
11) регистрация выдачи клиенту SIM-карты;
12) регистрация возврата SIM-карты от клиента
13) печать списка регистрации SIM-карт
Выберите действие: 9
Введите номер сим-карты для удаления: 222-2226440
222-2226440 Всё включено 2005 В наличии

МЕНЮ:
0) выход
1) регистрация нового клиента;
2) снятие с обслуживания клиента;
3) просмотр всех зарегистрированных клиентов;
4) поиск клиента по «№ паспорта»;
5) поиск клиента по фрагментам ФИО или адреса;
6) добавление новой SIM-карты;
7) удаление сведений о SIM-карте;
8) просмотр всех имеющихся SIM-карт;
9) поиск SIM-карты по «№ SIM-карты»;
10) поиск SIM-карты по тарифу;
11) регистрация выдачи клиенту SIM-карты;
```

Рисунок 9 – Тестирование поиска сим-карты по № сим-карты

```
E:\2 курс 2 семестр\Курсовой проект\KR Lunyov A.C\Debug\KR Lunyov A.C.exe
Введите тариф для поиска: домашний

МЕНЮ:
0) выход
1) регистрация нового клиента;
2) снятие с обслуживания клиента;
3) просмотр всех зарегистрированных клиентов;
4) поиск клиента по «№ паспорта»;
5) поиск клиента по фрагментам ФИО или адреса;
6) добавление новой SIM-карты;
7) удаление сведений о SIM-карте;
8) просмотр всех имеющихся SIM-карт;
9) поиск SIM-карты по «№ SIM-карты»;
10) поиск SIM-карты по тарифу;
11) регистрация выдачи клиенту SIM-карты;
12) регистрация возврата SIM-карты от клиента
13) печать списка регистрации SIM-карт
Выберите действие: 10
Введите тариф для поиска: Домашний
111-8963447 Домашний 2000 В наличии
956-6254895 Домашний 2010 В наличии
488-4888520 Домашний 2010 В наличии

МЕНЮ:
0) выход
1) регистрация нового клиента;
2) снятие с обслуживания клиента;
```

Рисунок 10 – Тестирование поиска сим-карты по тарифу

```
E:\2 курс 2 семестр\Курсовой проект\KR Lunyov A\Cy44\Debug\KR Lunyov A.C.exe
7662-3684254 669-8965529 2009 2023 Выдана
0986-5345578 759-1552443 2002 2018 Выдана
4553-6578153 956-6254895 2001 2015 Вернули

МЕНЮ:
0) выход
1) регистрация нового клиента;
2) снятие с обслуживания клиента;
3) просмотр всех зарегистрированных клиентов;
4) поиск клиента по «№ паспорта»;
5) поиск клиента по фрагментам ФИО или адреса;
6) добавление новой SIM-карты;
7) удаление сведений о SIM-карте;
8) просмотр всех имеющихся SIM-карт;
9) поиск SIM-карты по «№ SIM-карты»;
10) поиск SIM-карты по тарифу;
11) регистрация выдачи клиенту SIM-карты;
12) регистрация возврата SIM-карты от клиента
13) печать списка регистрации SIM-карт
Выберите действие: 11
Паспорт: 1234-123456
Номер симки: 956-6254895
Дата выпуска: 2001
Дата окончания: 2015

МЕНЮ:
```

Рисунок 11 – Тестирование выдачи сим-карты

```
Выбрать E:\2 курс 2 семестр\Курсовой проект\KR Lunyov A\Cy44\Debug\KR Lunyov A.C.exe
10) поиск SIM-карты по тарифу;
11) регистрация выдачи клиенту SIM-карты;
12) регистрация возврата SIM-карты от клиента
13) печать списка регистрации SIM-карт
Выберите действие: 13

Регистрация Сим-карт:
6987-0326657 111-8963447 1996 2013 Вернули
1234-5678123 165-6665458 2005 2021 Вернули
8654-5266644 166-2345985 2007 2025 Выдана
2144-6985114 222-2226440 1994 2016 Вернули
2222-1311184 333-1554781 1998 2019 Выдана
2665-7999855 336-3365140 1994 2019 Выдана
3333-4445565 488-4888520 1998 2006 Вернули
7662-3684254 669-8965529 2009 2023 Выдана
0986-5345578 759-1552443 2002 2018 Выдана
1234-123456 956-6254895 2001 2015 Выдана
4553-6578153 956-6254895 2001 2015 Вернули

МЕНЮ:
0) выход
1) регистрация нового клиента;
2) снятие с обслуживания клиента;
3) просмотр всех зарегистрированных клиентов;
4) поиск клиента по «№ паспорта»;
5) поиск клиента по фрагментам ФИО или адреса;
```

Рисунок 12 – Тестирование выдачи сим-карты

```
E:\2 курс 2 семестр\Курсовой проект\KR Lunyov A\Cy64\Debug\KR Lunyov A.C.exe
1) регистрация нового клиента;
2) снятие с обслуживания клиента;
3) просмотр всех зарегистрированных клиентов;
4) поиск клиента по «№ паспорта»;
5) поиск клиента по фрагментам ФИО или адреса;
6) добавление новой SIM-карты;
7) удаление сведений о SIM-карте;
8) просмотр всех имеющихся SIM-карт;
9) поиск SIM-карты по «№ SIM-карты»;
10) поиск SIM-карты по тарифу;
11) регистрация выдачи клиенту SIM-карты;
12) регистрация возврата SIM-карты от клиента
13) печать списка регистрации SIM-карт
Выберите действие: 12
Введите номер сим-карты для возврата: 759-1552443

МЕНЮ:
0) выход
1) регистрация нового клиента;
2) снятие с обслуживания клиента;
3) просмотр всех зарегистрированных клиентов;
4) поиск клиента по «№ паспорта»;
5) поиск клиента по фрагментам ФИО или адреса;
6) добавление новой SIM-карты;
7) удаление сведений о SIM-карте;
8) просмотр всех имеющихся SIM-карт;
```

Рисунок 13 – Тестирование возврата сим-карты

```
Выбрать E:\2 курс 2 семестр\Курсовой проект\KR Lunyov A\Cy64\Debug\KR Lunyov A.C.exe
7) удаление сведений о SIM-карте;
8) просмотр всех имеющихся SIM-карт;
9) поиск SIM-карты по «№ SIM-карты»;
10) поиск SIM-карты по тарифу;
11) регистрация выдачи клиенту SIM-карты;
12) регистрация возврата SIM-карты от клиента
13) печать списка регистрации SIM-карт
Выберите действие: 13

Регистрация Сим-карт:
6987-0326657 111-8963447 1996 2013 Вернули
1234-5678123 165-6665458 2005 2021 Вернули
8654-5266644 166-2345985 2007 2025 Выдана
2144-6985114 222-2226440 1994 2016 Вернули
2222-1311184 333-1554781 1998 2019 Выдана
2665-7999855 336-3365140 1994 2019 Выдана
3333-4445565 488-4888520 1998 2006 Вернули
7662-3684254 669-8965529 2009 2023 Выдана
0986-5345578 759-1552443 2002 2018 Вернули
1234-123456 956-6254895 2001 2015 Выдана
4553-6578153 956-6254895 2001 2015 Вернули

МЕНЮ:
0) выход
1) регистрация нового клиента;
2) снятие с обслуживания клиента;
```

Рисунок 14 – Тестирование возврата сим-карты

## 6. Приложение

### Код clients.h

---

```
#pragma once

#include <iostream>
#include <fstream>
#include <cstdlib>
#include <stdexcept>
#include <string>

using namespace std;

struct Client // структура для представления узлов дерева
{
    string passport;
    string info;
    string fio;
    int year;
    string address;

    unsigned char height;
    Client* left;
    Client* right;
    Client() : left(nullptr), right(nullptr), height(1)
    {
    }

    Client(string passport, string info, string fio, int year, string address) :
        passport(passport), info(info), fio(fio), year(year), address(address),
        left(nullptr), right(nullptr), height(1)
    {
    }

    void load(istream& in);
    void input();
    void print();
};

unsigned char height(Client* p);
void fixheight(Client* p);
int bfactor(Client* p);
Client* rotateright(Client* p); // правый поворот вокруг p
Client* rotateleft(Client* q); // левый поворот вокруг q
Client* balance(Client* p); // балансировка узла p
Client* insert(Client* p, Client k); // вставка ключа k в дерево с корнем p
void print(Client* root, int level); // печать дерева по уровням
Client* findmin(Client* p); // поиск узла с минимальным ключом в дереве p
Client* removemin(Client* p); // удаление узла с минимальным ключом из дерева p
Client* remove(Client* p, string passport); // удаление ключа k из дерева p
Client* find(Client* p, string passport); // поиск в дереве по паспорту
void find_by_info(Client* p, string info); // поиск в дереве по части фио или адреса

// AVL-дерево клиентов
class AvlTree
{
    Client* root;

public:
    AvlTree() : root(nullptr)
    {
    }
};
```

```

    }

    // печать всех клиентов
    void print()
    {
        cout << endl;
        cout << "Зарегистрированные клиенты: " << endl;
        ::print(root, 0);
        cout << endl << endl;
    }

    // вставка нового клиента в дерево
    void insert(Client n)
    {
        root = ::insert(root, n);
    }

    // удаление клиента по паспорту
    void remove(string passport)
    {
        root = ::remove(root, passport);
    }

    // загрузка клиентов из файла
    void load(string filename)
    {
        ifstream fin(filename);

        int n;
        fin >> n;
        fin.ignore();

        for (int i = 0; i < n; i++)
        {
            Client client;
            client.load(fin);
            fin.ignore();

            insert(client);
        }

        fin.close();
    }

    // поиск клиента по паспорту
    Client* find(string passport)
    {
        return ::find(root, passport);
    }

    // поиск клиента по части адреса или ФИО
    void find_by_info(string info)
    {
        ::find_by_info(root, info);
    }
};

```

---

## Код clients.cpp

---

```

#include <regex>

#include "clients.h"

```

```

// считывание клиента из потока ввода (из файла)
void Client::load(istream& in)
{
    getline(in, passport);
    getline(in, info);
    getline(in, fio);
    string year_str;
    in >> year;
    in.ignore();
    getline(in, address);
}

// ввод данных нового клиента
void Client::input()
{
    istream& in = cin;
    in.ignore();

    while (true)
    {
        cout << "Паспорт: "; getline(in, passport);
        if (regex_match(passport, regex("\\[0-9]{4}-\\[0-9]{6}")))
            break;
        else
            cout << "Некорректный формат паспорта NNNN-NNNNNN!" << endl;
    }
    cout << "Кем и когда выдан: "; getline(in, info);
    cout << "ФИО: "; getline(in, fio);
    cout << "Год рождения: "; string year_str;

    in >> year;
    in.ignore();
    cout << "Адрес: "; getline(in, address);
}

// печать информации о клиенте
void Client::print()
{
    cout << passport << " " << info << " " << fio << " " << year << " " << address <<
endl;
}

// высота дерева
unsigned char height(Client* p)
{
    return p ? p->height : 0;
}

// пересчет высоты дерева
void fixheight(Client* p)
{
    unsigned char hl = height(p->left);
    unsigned char hr = height(p->right);
    p->height = (hl > hr ? hl : hr) + 1;
}

// b-фактор узла (разность высот правого и левого поддеревя)
int bfactor(Client* p)
{
    return height(p->right) - height(p->left);
}

Client* rotateright(Client* p) // правый поворот вокруг p

```

```

{
    Client* q = p->left;
    p->left = q->right;
    q->right = p;
    fixheight(p);
    fixheight(q);
    return q;
}

Client* rotateleft(Client* q) // левый поворот вокруг q
{
    Client* p = q->right;
    q->right = p->left;
    p->left = q;
    fixheight(q);
    fixheight(p);
    return p;
}

Client* balance(Client* p) // балансировка узла p
{
    fixheight(p);
    if (bfactor(p) == 2)
    {
        if (bfactor(p->right) < 0)
            p->right = rotateright(p->right);
        return rotateleft(p);
    }
    if (bfactor(p) == -2)
    {
        if (bfactor(p->left) > 0)
            p->left = rotateleft(p->left);
        return rotateright(p);
    }
    return p; // балансировка не нужна
}

Client* insert(Client* p, Client k) // вставка ключа k в дерево с корнем p
{
    if (!p) return new Client(k);
    if (k.passport < p->passport)
        p->left = insert(p->left, k);
    else
        p->right = insert(p->right, k);
    return balance(p);
}

void print(Client* root, int level = 0)
{
    if (root == nullptr)
        return;

    print(root->right, level + 1);
    // for(int i=0; i<level; i++)
    //     cout<<" ";

    root->print(); // cout << endl;
    print(root->left, level + 1);
}

Client* findmin(Client* p) // поиск узла с минимальным ключом в дереве p
{
    return p->left ? findmin(p->left) : p;
}

```



```

}

Client* removemin(Client* p) // удаление узла с минимальным ключом из дерева p
{
    if (p->left == 0)
        return p->right;
    p->left = removemin(p->left);
    return balance(p);
}

Client* remove(Client* p, string passport) // удаление ключа k из дерева p
{
    if (!p) return 0;
    if (passport < p->passport)
        p->left = remove(p->left, passport);
    else if (passport > p->passport)
        p->right = remove(p->right, passport);
    else
    {
        Client* q = p->left;
        Client* r = p->right;
        delete p;
        if (!r) return q;
        Client* min = findmin(r);
        min->right = removemin(r);
        min->left = q;
        return balance(min);
    }
    return balance(p);
}

// поиск клиента по номеру паспорта (вернет указатель на узел дерева)
Client* find(Client* p, string passport)
{
    if (!p) return 0;
    if (passport == p->passport) return p;
    if (passport < p->passport)
        return find(p->left, passport);
    return find(p->right, passport);
}

// поиск клиента по части адреса или ФИО
void find_by_info(Client* p, string info)
{
    if (!p) return;
    if (p->fio.find(info) != string::npos) {
        p->print();
    }
    if (p->address.find(info) != string::npos) {
        p->print();
    }
    find_by_info(p->left, info);
    find_by_info(p->right, info);
}

```

---

## Код delivery.h

---

```

#pragma once

#include <cstdlib>
#include <fstream>
#include <iostream>

```

```

using namespace std;

// Узел списка записей о выдаче симок
struct Record
{
    string passport;
    string number;
    string start_date;
    string finish_date;
    bool is_returned; // false - выдали, true - вернули
    Record* next;

    Record() : next(nullptr), is_returned(false)
    {
    }

    Record(string pass, string num, string start, string finish) :
        passport(pass), number(num), start_date(start), finish_date(finish),
        is_returned(false), next(nullptr)
    {
    }

    void load(istream& in);
    void input();

    friend ostream& operator<<(ostream& out, Record n);
};
//-----

// Прототипы функций для СОРТИРОВКИ СПИСКА СЛИЯНИЕМ
int Length(Record* head);
void FrontBackSplit(Record* source, Record** frontRef, Record** backRef);
Record* SortedMerge(Record* a, Record* b);
void MergeSort(Record** headRef);

// Список записей о выдаче сим-карт клиентам
class List
{
    Record* head;
    int count;

public:
    List() : head(nullptr), count(0) {}

    Record** get_head()
    {
        return &head;
    }

    void insert(Record n);
    void returnSim(string sim);
    void load(string filename);
    friend ostream& operator<<(ostream& out, List list);

    void sort()
    {
        MergeSort(&head);
    }
};
//-----

```

## Код delivery.cpp

---

```
#include "delivery.h"

#include <regex>

// Загрузка записей о выдаче симок из файла
void Record::load(istream& in)
{
    getline(in, passport);
    getline(in, number);
    getline(in, start_date);
    getline(in, finish_date);

    string return_str;
    getline(in, return_str);
    is_returned = return_str == "true" ? true : false;
}

// Добавление новой записи о выдаче сим-ки (ввод данных пользователем)
void Record::input()
{
    istream& in = cin;

    while (true)
    {
        cout << "Паспорт: "; getline(in, passport);
        if (regex_match(passport, regex("\\[0-9\\]{4}-\\[0-9\\]{6}")))
            break;
        else
            cout << "Некорректный формат паспорта!" << endl;
    }

    while (true)
    {
        cout << "Номер симки: "; getline(in, number);
        if (regex_match(number, regex("\\[0-9\\]{3}-\\[0-9\\]{6}")))
            break;
        else
            cout << "Некорректный формат сим-карты!" << endl;
    }

    cout << "Дата выпуска: "; getline(in, start_date);
    cout << "Дата окончания: "; getline(in, finish_date);

    is_returned = false;
}

// печать записи о выдаче симки
ostream& operator<<(ostream& out, Record n) // перегрузка потока вывода
{
    out << n.passport << " " << n.number << " " << n.start_date << " " <<
n.finish_date << " ";
    if (n.is_returned == true)
        out << "Вернули";
    else
        out << "Выдана";
    out << endl;
}
```

```

        return out;
    }

    // Список выдачи и возврата SIM-карт пользователям
    void List::insert(Record n)
    {
        Record* tnode;
        tnode = new Record(n);
        if (tnode != nullptr)
        {
            tnode->next = head;
            count++;
            head = tnode;
        }
    }

    // возврат сим-карты
    void List::returnSim(string sim)
    {
        Record* current = head;
        while (current != nullptr)
        {
            if (current->number == sim)
                current->is_returned = true;
            current = current->next;
        }
        cout << endl;
    }

    // Загрузка из файла списка выдачи и возврата SIM-карт пользователям
    void List::load(string filename)
    {
        ifstream fin(filename);

        int n;
        fin >> n;
        fin.ignore();

        for (int i = 0; i < n; i++)
        {
            Record record;
            record.load(fin);

            fin.ignore();
            insert(record);
        }

        fin.close();
    }

    // Печать списка выдачи и возврата SIM-карт пользователям
    ostream& operator<<(ostream& out, List list)
    {
        out << "Регистрация Сим-карт: " << endl;
        Record* current = list.head;
        while (current != nullptr)
        {
            cout << *current;
            current = current->next;
        }
        cout << endl;
        return out;
    }
    //-----

```

```

// Функции для сортировки односвязного списка слиянием
int Length(Record* head)
{
    int count = 0;
    Record* current = head;
    while (current != nullptr)
    {
        count++;
        current = current->next;
    }
    return(count);
}

void FrontBackSplit(Record* source, Record** frontRef, Record** backRef)
{
    int len = Length(source);
    int i;
    Record* current = source;
    if (len < 2)
    {
        *frontRef = source;
        *backRef = nullptr;
    }
    else
    {
        int hopCount = (len - 1) / 2;
        for (i = 0; i < hopCount; i++)
        {
            current = current->next;
        }

        // исходный список разбивается на два подсписка
        *frontRef = source;
        *backRef = current->next;
        current->next = nullptr;
    }
}

Record* SortedMerge(Record* a, Record* b)
{
    Record* result = nullptr;
    if (a == nullptr) return(b);
    else if (b == nullptr) return(a);
    if (a->number <= b->number)
    {
        result = a;
        result->next = SortedMerge(a->next, b);
    }
    else
    {
        result = b;
        result->next = SortedMerge(a, b->next);
    }
    return(result);
}

void MergeSort(Record** headRef)
{
    Record* head = *headRef;
    Record* a;
    Record* b;
    // вырожденный случай – длина списка равно 0 или 1

```

```

        if ((head == nullptr) || (head->next == nullptr))
        {
            return;
        }
        FrontBackSplit(head, &a, &b);
        MergeSort(&a); // рекурсивная сортировка подписков
        MergeSort(&b);
        *headRef = SortedMerge(a, b);
    }
    //-----

```

---

## Код sim.h

---

```

#pragma once

#include <iostream>
#include <fstream>
#include <stdexcept>
#include <functional>
#include <limits>

using namespace std;

unsigned int str_hash(string s);
int try_func(int i);

struct Sim
{
    string sim;
    string tarif;
    int year;
    bool exist;

    Sim()
    {
    }

    Sim(string s, string t, int y, bool e)
        : sim(s), tarif(t), year(y), exist(e)
    {
    }

    void load(istream& in);
    void input();
    friend ostream& operator<<(ostream& out, Sim s);
};

template<typename K, typename V>
class HashNode
{
public:
    K key;
    V value;
    HashNode() {}
    HashNode(K key, V value)
    {
        this->value = value;
        this->key = key;
    }
};

template<typename K, typename V>
class HashMap

```

```

{
    HashNode<K, V>*& arr;
    int capacity;
    int size;
    HashNode<K, V>* dummy; // для того, чтобы помечать элементы удаленными

    function<unsigned int(K)> hashFunc;
    function<int(int)> tryFunc;

public:
    HashMap(function<unsigned int(K)> myHashFunc, function<int(int)> myTryFunc)
    {
        capacity = 100; // ЕМКОСТЬ ХЭШ ТАБЛИЦЫ!!!
        size = 0;
        arr = new HashNode<K, V> * [capacity];
        for (int i = 0; i < capacity; i++)
            arr[i] = NULL;

        dummy = new HashNode<K, V>;

        hashFunc = myHashFunc;
        tryFunc = myTryFunc;
    }

    int hashCode(K key)
    {
        return hashFunc(key) % capacity;
    }

    HashNode<K, V>*& find(K key, bool breakOnDummy = false)
    {
        int index;
        int tryCount = 0;
        int hashIndex = hashCode(key);
        do
        {
            index = (hashIndex + tryFunc(tryCount)) % capacity;
            tryCount++;

            if (breakOnDummy && arr[index] == dummy)
                break;

            if (index == hashIndex) // зацикливание поиска
            {
                hashIndex++; // смещаемся на одну позицию
                hashIndex %= capacity;
            }
        } while (arr[index] != NULL && arr[index]->key != key);

        // cout << "find() -> index = " << index << endl;
        return arr[index];
    }

    void put(K key, V value)
    {
        if (size == capacity)
        {
            throw length_error("Hash::put() size==capacity, no empty slots!");
        }

        HashNode<K, V>*& p = find(key, true);

        if (p == NULL || p == dummy)
            size++;
    }
}

```

```

        p = new HashNode<K, V>(key, value);
    }

    bool remove(K key)
    {
        HashNode<K, V>* p = find(key);
        if (!p)
            return false;

        p = dummy;
        size--;
        return true;
    }

    V* get(K key)
    {
        HashNode<K, V>* p = find(key);
        if (!p)
            throw invalid_argument("Hash::get() no such key!");

        return &p->value;
    }

    int sizeofMap()
    {
        return size;
    }

    bool isEmpty()
    {
        return size == 0;
    }

    void display()
    {
        cout << "Hash Table:" << endl;
        for (int i = 0; i < capacity; i++)
        {
            cout << i << "\t";
            if (arr[i] != NULL && arr[i] != dummy)
                cout << "key = " << arr[i]->key
                    << ", value = " << arr[i]->value << endl;
            else if (arr[i] == NULL)
                cout << "NULL" << endl;
            else
                cout << "DEL" << endl;
        }
        cout << "-----" << endl << endl;
    }

    void print()
    {
        for (int i = 0; i < capacity; i++)
        {
            if (arr[i] != NULL && arr[i] != dummy)
                cout << arr[i]->value;
        }
    }

    friend class SimHash;
};

```



```

class SimHash
{
    HashMap<string, Sim>* hash;

public:
    SimHash()
    {
        hash = new HashMap<string, Sim>(str_hash, try_func);
    }

    void load(string filename)
    {
        ifstream fin(filename);

        int n;
        fin >> n;
        fin.ignore();

        for (int i = 0; i < n; i++)
        {
            Sim sim;
            sim.load(fin);
            fin.ignore();
            hash->put(sim.sim, sim);
        }

        fin.close();
    }

    void put(Sim s)
    {
        hash->put(s.sim, s);
    }

    Sim* find(string sim)
    {
        try
        {
            Sim* s = hash->get(sim);
            return s;
        }
        catch (invalid_argument e)
        {
            cout << "Нет сим карты с таким номером!" << endl;
            return nullptr;
        }
    }

    void findByTarif(string tarif)
    {
        for (int i = 0; i < hash->capacity; i++)
        {
            if (hash->arr[i] != NULL && hash->arr[i] != hash->dummy)
            {
                Sim s = hash->arr[i]->value;
                if (s.tarif == tarif) {
                    cout << s;
                }
            }
        }
    }
}

```

```

void remove(string sim)
{
    hash->remove(sim);
}

void print()
{
    cout << endl;
    cout << "Список сим-карт: " << endl;
    hash->print();
}

~SimHash()
{
    delete hash;
}
};

```

---

## Код sim.cpp

---

```

#include <iostream>
#include <fstream>
#include <stdexcept>
#include <functional>
#include <limits>
#include <regex>

#include "sim.h"

void Sim::load(istream& in)
{
    getline(in, sim);
    getline(in, tarif);
    in >> year;
    in.ignore();

    string exists_str;
    getline(in, exists_str);
    exist = exists_str == "true" ? true : false;
}

void Sim::input()
{
    istream& in = cin;

    while (true)
    {
        cout << "Номер симки: "; getline(in, sim);
        if (regex_match(sim, regex("[0-9]{3}-[0-9]{7}")))
            break;
        else
            cout << "Некорректный формат сим-карты NNN-NNNNNNN!" << endl;
    }
    cout << "Тариф: "; getline(in, tarif);
    cout << "Год выпуска: "; in >> year;
    in.ignore();

    exist = true;
}

ostream& operator<<(ostream& out, Sim s)

```

```

{
    out << s.sim << " " << s.tarif << " " << s.year << " " << " ";
    if (s.exist)
        out << "В наличии";
    else
        out << "Нет в наличии";
    out << endl;
    return out;
}

unsigned int str_hash(string s)
{
    const int p = 31;
    long long hash = 0, p_pow = 1;
    for (size_t i = 0; i < s.length(); ++i)
    {
        hash += (s[i] - 'a' + 1) * p_pow;
        p_pow *= p;
    }
    return hash;
}

int try_func(int i)
{
    return i + i * i;
}

```

---

## Код main.cpp

---

```

#include <cstdlib>
#include <fstream>
#include <iostream>
#include <locale>
#include <string>
#include <vector>

#include <windows.h>

#include "clients.h"
#include "sim.h"
#include "delivery.h"

using namespace std;

int menu()
{
    vector<string> vars =
    {
        "выход",
        "регистрация нового клиента;",
        "снятие с обслуживания клиента;",
        "просмотр всех зарегистрированных клиентов;",
        "поиск клиента по «№ паспорта»;",
        "поиск клиента по фрагментам ФИО или адреса;",

        "добавление новой SIM-карты;",
        "удаление сведений о SIM-карте;",
        "просмотр всех имеющихся SIM-карт;",

        "поиск SIM-карты по «№ SIM-карты»;",
        "поиск SIM-карты по тарифу;",
        "регистрация выдачи клиенту SIM-карты;",
    }
}

```

```

        "регистрация возврата SIM-карты от клиента",
        "печать списка регистрации SIM-карт"
    };

    cout << endl;
    cout << "МЕНЮ:" << endl;
    for (int i = 0; i < vars.size(); i++)
    {
        cout << i << ") " << vars[i] << endl;
    }

    while (true)
    {
        cout << "Выберите действие: ";
        int v;
        cin >> v;

        if (v < 0 || v >= vars.size())
            cout << "Неправильный индекс действия" << endl;
        else
            return v;
    }
}

int main()
{
    setlocale(LC_CTYPE, "");
    SetConsoleCP(1251); // установка кодовой страницы win-ср 1251 в поток ввода
    SetConsoleOutputCP(1251); // установка кодовой страницы win-ср 1251 в поток вывода

    AVLTree clients;
    clients.load("clients.txt");

    SimHash sims;
    sims.load("sim.txt");

    List delivery;
    delivery.load("delivery.txt");
    delivery.sort();

    int action = menu();
    while (action)
    {
        switch (action)
        {
            case 1: // "регистрация нового клиента;"
            {
                Client c;
                c.input();
                clients.insert(c);
            }
            break;

            case 2: // "снятие с обслуживания клиента;"
            {
                string passport;
                cout << "Введите паспорт клиента для снятия с обслуживания: ";
                cin >> passport;
                Client* c = clients.find(passport);
                if (!c)
                {
                    cout << "Нет клиента с таким паспортом" << endl;
                }
                else

```

```

        clients.remove(passport);
    }
    break;

case 3: // "просмотр всех зарегистрированных клиентов;"
    clients.print();
    break;

case 4: // "поиск клиента по «№ паспорта»;"
{
    string passport;
    cout << "Введите паспорт клиента: ";
    cin >> passport;
    Client* c = clients.find(passport);
    if (!c)
    {
        cout << "Нет клиента с таким паспортом" << endl;
    }
    else
        c->print();
}
break;

case 5: // "поиск клиента по фрагментам ФИО или адреса;"
{
    string info;
    cout << "Введите фрагмент ФИО или адреса для поиска: ";
    cin >> info;

    clients.find_by_info(info);
}
break;

case 6: // "добавление новой SIM-карты;"
{
    Sim s;
    cin.ignore();
    s.input();
    sims.put(s);
}
break;

case 7: // "удаление сведений о SIM-карте;"
{
    string sim;
    cout << "Введите номер сим-карты для удаления: ";
    cin >> sim;
    sims.remove(sim);
}
break;

case 8: // "просмотр всех имеющихся SIM-карт;"
    sims.print();
    break;

case 9: // "поиск SIM-карты по «№ SIM-карты»;"
{
    string sim;
    cout << "Введите номер сим-карты для удаления: ";
    cin >> sim;
    Sim* s = sims.find(sim);
    cout << *s << endl;
}
break;

```

```

case 10: // "поиск SIM-карты по тарифу;"
{
    string tarif;
    cout << "Введите тариф для поиска: ";
    cin >> tarif;
    sims.findByTarif(tarif);
}
break;

case 11: // "регистрация выдачи клиенту SIM-карты;"
{
    Record r;
    cin.ignore();
    r.input();

    Sim* s = sims.find(r.number);
    if (!s) break;

    if (s->exist) // если sim с таким номером в наличии
    {
        delivery.insert(r); // регистрация выдачи сим-карты
        s->exist = false; // пометили, как отсутствующую
    }
    else
        cout << "У нас нет сим-карты с таким номером" << endl;
}
break;

case 12: // "регистрация возврата SIM-карты от клиента"
{
    string sim;
    cout << "Введите номер сим-карты для возврата: ";
    cin >> sim;
    delivery.returnSim(sim);

    // Отмечаем, что теперь эта симка в наличии
    Sim* s = sims.find(sim);
    s->exist = true;
}
break;

case 13: // "печать списка регистрации SIM-карт"
    delivery.sort();
    cout << endl << delivery;
    break;
}
action = menu();
}
cout << "Bye!" << endl;
}

```

---