**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра САПР**

Пояснительная записка

**к курсовой работе**

**по дисциплине «Программирование»**

Тема: **«Хранение и поиск необходимой информации»**

|  |  |  |
| --- | --- | --- |
| Студент гр. 2309 |  | Савин П.А. |
| Преподаватель |  | Калмычков В.А. |

Санкт-Петербург

2023

**Задание на курсовую работу**

Студент: Савин П.А.

Группа: 2309

Тема работы:

Хранение и поиск необходимой информации.

Исходные данные:

Организовать информационную систему страховой компании по договорам страхования частных лиц. Необходимо обеспечить эффективную выдачу конкретных договоров по ключевым фильтрам (например, заключенные определенным частным лицом или агентом и т. п.)

Содержание пояснительной записки:

Титульный лист, задание, аннотация, содержание, исходная формулировка, анализ задания, контрольный пример, описание реализации, алгоритм, текст программы, результаты работы, заключение.

Предполагаемый объем пояснительной записки:

Не менее 30 страниц.

Дата выдачи задания: 20.02.2023

Дата сдачи курсовой работы:

Дата защиты курсовой работы:

|  |  |  |
| --- | --- | --- |
| Студент гр. 2309 |  | Савин П.А. |
| Преподаватель |  | Калмычков В.А. |

**Аннотация.**

В данной работе используется комплексный подход к хранению данных. Все исходные данные хранятся в единственном экземпляре в соответствующих их группе односвязных списках (к примеру, список ФИО клиентов или список адресов филиалов). Сами договоры страхования реализованы как ссылки на исходные данные, а их группировка по параметрам — как массивы из 33 элементов (соответствует русскому алфавиту), где каждый элемент – список из значений группировки и им соответствующие списки исходных договоров, сгруппированные по заданному параметру. Такая структура организации называется мультисписок — несколько односвязных списков в программе, которые не хранят новые данные, а лишь отсылают на исходные в исходном списке, но при этом сами данные организованы в другом порядке с целью группировки.

Все исходные данные, в том числе и данные для запросов пользователя считываются с соответствующих им входных файлов, каждый этап выполнения программы (запись данных, группировка, обработка запросов и ошибок) фиксируется в файле протокола, также есть выходной файл, который выводит лишь результаты обработки запросов, либо критические ошибки, не позволяющие программе отработать корректно.

**Summary.**

This paper uses a comprehensive approach to data storage. All source data are stored in a single instance in the singly linked lists corresponding to their group (for example, the list of clients' names or the list of branch addresses). The insurance contracts themselves are implemented as references to the source data, and their grouping by parameters - as arrays of 33 elements (corresponds to the Russian alphabet), where each element is a list of grouping values and their corresponding lists of source contracts grouped by a given parameter. This organization structure is called multilist - several singly linked lists in the program, which do not store new data, but only refer to the original data in the original list, but the data themselves are organized in a different order for the purpose of grouping.

All source data, including data for user queries, are read from their corresponding input files; each step of the program execution (data recording, grouping, query processing and errors) is recorded in a log file; there is also an output file that displays only the results of query processing or critical errors that do not allow the program to work correctly.

**Формулировка задания.**

Организовать информационную систему страховой компании по договорам страхования частных лиц.

Должны храниться сведения о:

- филиалах кампании (название, местонахождение),

- страховых агентах в каждом филиале (ФИО),

- застрахованных лицах (ФИО),

- договорах страхования (дата заключения, срок, страховая сумма, филиал, агент) по виду страхования (например, страхование автотранспорта от угона, страхование домашнего имущества, добровольное медицинское страхование и т.п.).

Факт заключения нового договора страхования фиксируется.

Необходимо обеспечить эффективную выдачу сведений:

- о результатах работы определенного страхового агента,

- о результатах работы определенного филиала,

- о договорах страхования определенного частного лица,

- о договорах страхования определенного вида (по филиалам и компании в целом).

**Анализ задания.**

Формулировка не оговаривает дополнительное требование — все исходные данные должны быть в единственном экземпляре, то есть, все последующие структуры, организующие эффективное хранение и выдачу данных, должны хранить не копии исходных данных, а адреса памяти, в которых и размещены эти самые данные.

Также стоит уточнить, что подразумевается под эффективной выдачей сведений — сбор информации для сводки не должен представлять собой линейный поиск по исходным данным, так как это затратно по времени и производительности. Требуется организовать сложную структуру хранения данных, позволяющую, пусть и длинной цепочкой иерархии, но сгруппировать данные по ключевым параметрам и снизить время на поиск.

Реализация не ограничена исключительно одним подходом, можно как сразу вводить уникальные договоры, от которых будут пополняться списки исходных данных, а можно сделать для каждой категории исходных данных создать свой файл с уникальными исходными данными, а очередной договор будет строиться как набор индексов, отсылающих на соответствующие данные в списках.

В работе был применен второй способ.

На этот раз за корректность введенных данных ответственен сам пользователь, поскольку сами данные по себе не влияют на логику программы, а запросы пользователь все равно будет писать по этим же данным, точь-в-точь. Однако все же придется проверять корректность вводимых страховых сумм, а именно отсекать отрицательные значения.

**Контрольный пример.**

Названия:

Росгосстрах-альфа

Росгосстрах-бета

Адреса:

Белоостровская улица, 17

Набережная Черной речки, 41

ФИО агентов:

Григорьева Елена Сергеевна

Корнеев Андрей Александрович

Соколова Анастасия Александровна

Макаров Даниил Александрович

Николаев Алексей Сергеевич

Королева Анастасия Викторовна

Шестаков Дмитрий Александрович

Васильев Иван Иванович

Петров Сергей Дмитриевич

Гусева Анастасия Александровна

ФИО клиентов:

Баранов Андрей Михайлович

Андреева Екатерина Игоревна

Макаров Александр Александрович

Чернова Екатерина Сергеевна

Григорьев Денис Александрович

Петрова Дарья Сергеевна

Белова Анастасия Сергеевна

Соколов Алексей Александрович

Волков Андрей Сергеевич

Тихонов Алексей Александрович

Даты заключения договоров:

30.08.2022

17.04.2021

27.02.2022

11.07.2022

06.06.2021

03.01.2023

20.05.2022

24.12.2022

15.08.2021

07.11.2021

Сроки страхования:

150 дней

40 дней

17 лет

25 дней

9 лет

15 дней

270 дней

30 лет

10 лет

600 дней

35 лет

14 лет

6 лет

4 года

90 дней

Виды страхования:

КАСКО

ОСАГО

ДМС

НС

СМР

Страхование имущества

Страхование от последствий COVID-19

Страховые договоры:

3 12 3 0 1 3 4 4

2 6 2 0 1 7 4 5

1 12 7 0 1 5 3 5

5 9 5 1 0 3 7 0

4 3 6 1 0 2 6 1

6 11 8 0 1 3 2 2

0 8 1 1 0 1 9 1

7 13 8 0 1 1 1

1 1 4 0 1 8 0

2 0 1 1 0 5 3

Запросы:

1 Николаев Алексей Сергеевич

2 Росгосстрах-альфа; Набережная Черной речки, 41

3 Чернова Екатерина Сергеевна

4 ОСАГО; Росгосстрах-бета; Белоостровская улица, 17

5 ДМС

Ответы на запросы:

Запрос с кодом 1 - результаты работы агента:

Николаев Алексей Сергеевич

Договоры:

Дата заключения: 11.07.2022

Срок: 6 лет

Страховая сумма: 970513

Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41

ФИО клиента: Чернова Екатерина Сергеевна

ФИО агента: Николаев Алексей Сергеевич

Тип страхования: СМР

Дата заключения: 27.02.2022

Срок: 270 дней

Страховая сумма: 340832

Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41

ФИО клиента: Соколов Алексей Александрович

ФИО агента: Николаев Алексей Сергеевич

Тип страхования: Страхование имущества

Выручка агента: 1311345

Запрос с кодом 2 - результаты работы филиала:

"Росгосстрах-альфа", Набережная Черной речки, 41

Дата заключения: 11.07.2022

Срок: 6 лет

Страховая сумма: 970513

Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41

ФИО клиента: Чернова Екатерина Сергеевна

ФИО агента: Николаев Алексей Сергеевич

Тип страхования: СМР

Дата заключения: 27.02.2022

Срок: 270 дней

Страховая сумма: 340832

Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41

ФИО клиента: Соколов Алексей Александрович

ФИО агента: Николаев Алексей Сергеевич

Тип страхования: Страхование имущества

Дата заключения: 17.04.2021

Срок: 6 лет

Страховая сумма: 232581

Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41

ФИО клиента: Петрова Дарья Сергеевна

ФИО агента: Макаров Даниил Александрович

Тип страхования: Страхование имущества

Дата заключения: 20.05.2022

Срок: 14 лет

Страховая сумма: 600413

Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41

ФИО клиента: Чернова Екатерина Сергеевна

ФИО агента: Соколова Анастасия Александровна

Тип страхования: ДМС

Дата заключения: 24.12.2022

Срок: 4 года

Страховая сумма: 600413

Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41

ФИО клиента: Андреева Екатерина Игоревна

ФИО агента: Корнеев Андрей Александрович

Тип страхования: ОСАГО

Выручка филиала: 2744752

Запрос с кодом 3 - страховые договоры клиента:

Чернова Екатерина Сергеевна

Дата заключения: 11.07.2022

Срок: 6 лет

Страховая сумма: 970513

Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41

ФИО клиента: Чернова Екатерина Сергеевна

ФИО агента: Николаев Алексей Сергеевич

Тип страхования: СМР

Дата заключения: 03.01.2023

Срок: 600 дней

Страховая сумма: 832404

Филиал: "Росгосстрах-бета", Белоостровская улица, 17

ФИО клиента: Чернова Екатерина Сергеевна

ФИО агента: Васильев Иван Иванович

Тип страхования: КАСКО

Дата заключения: 20.05.2022

Срок: 14 лет

Страховая сумма: 600413

Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41

ФИО клиента: Чернова Екатерина Сергеевна

ФИО агента: Соколова Анастасия Александровна

Тип страхования: ДМС

Запрос с кодом 4 - база договоров филиала по типу:

Тип: ОСАГО

Филиал: "Росгосстрах-бета", Белоостровская улица, 17

Дата заключения: 06.06.2021

Срок: 25 дней

Страховая сумма: 926120

Филиал: "Росгосстрах-бета", Белоостровская улица, 17

ФИО клиента: Макаров Александр Александрович

ФИО агента: Шестаков Дмитрий Александрович

Тип страхования: ОСАГО

Дата заключения: 30.08.2022

Срок: 10 лет

Страховая сумма: 111347

Филиал: "Росгосстрах-бета", Белоостровская улица, 17

ФИО клиента: Андреева Екатерина Игоревна

ФИО агента: Гусева Анастасия Александровна

Тип страхования: ОСАГО

Запрос с кодом 5 - база договоров компании по типу:

ДМС

Дата заключения: 20.05.2022

Срок: 14 лет

Страховая сумма: 600413

Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41

ФИО клиента: Чернова Екатерина Сергеевна

ФИО агента: Соколова Анастасия Александровна

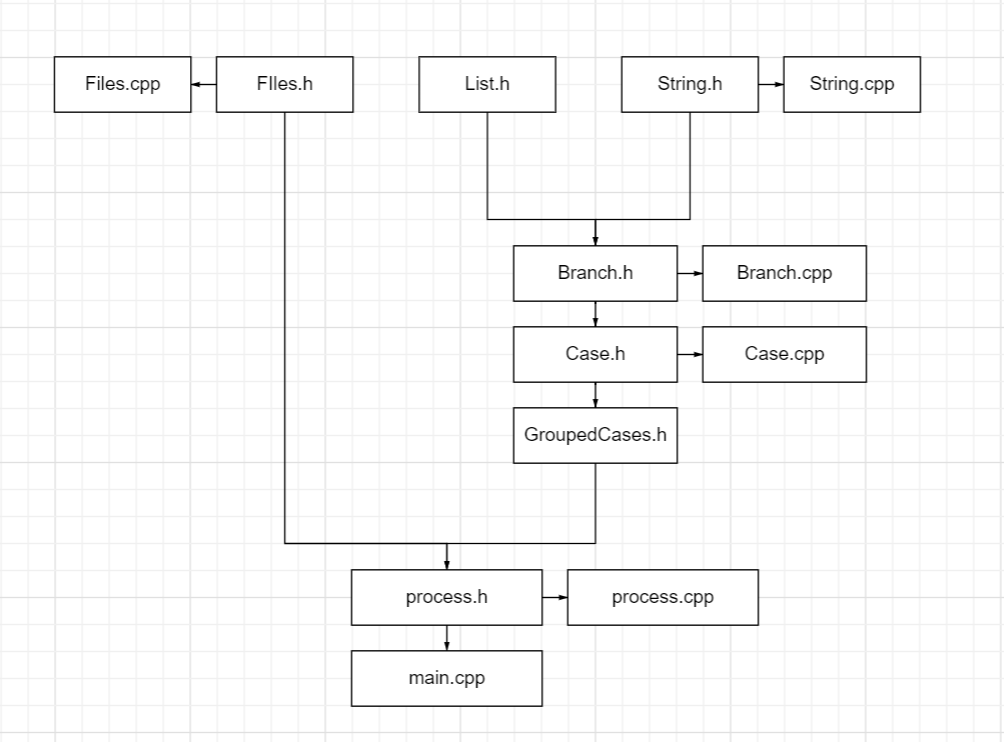
Тип страхования: ДМС

**Особенности реализации задания на компьютере.**

Реализация договоров через индексы накладывает дополнительные проверки — пользователь может указать неверные индексы, следовательно, такие договоры нужно отбрасывать и даже прерывать программу, если в ходе отбрасывания не осталось ни одного корректного договора.

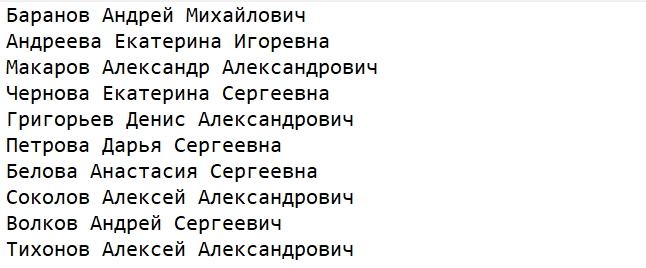
Это также касается и некорректных страховых сумм — пользователь все же мог ввести изначально корректный индекс для списка страховых сумм, но при этом укорачивание списка сумм привело к превращению корректных индексов в некорректные.

Иерархия включений модулей следующая:



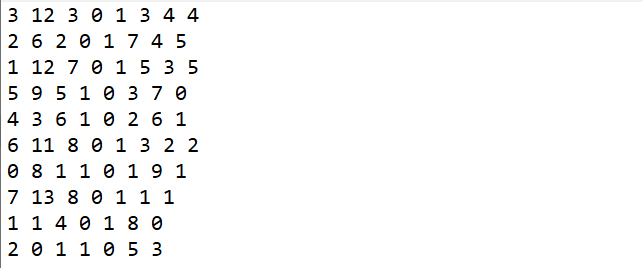
**Разработка интерфейса пользователя.**

Для всех файлов с исходными данными интерфейс един:



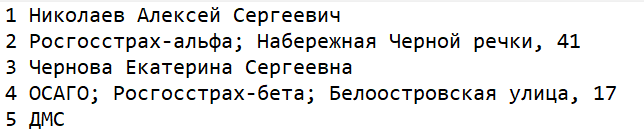
Поскольку строковые данные не зависят от корректного ввода, достаточно просто до перехода на новую строку ввести значение и запись в список готова. Со страховыми суммами аналогично, только вместо символов одно число на одной строке.

А вот дальше интерфейсы различаются: ввод договоров выглядит так:



Числа соответствуют индексам следующей последовательности: дата-срок-сумма-название-адрес-клиент-агент-тип.

И, наконец, интерфейс ввода запросов (рис. 3):



Сначала идет код (число от 1 до 5), затем обязательно пробел.

Подробнее по каждому коду: коды 1, 3 и 5 — результаты работы агента, база договоров клиента и определенного типа, соответственно, поэтому после этих кодов достаточно ввести строчку до перехода на новую. Код 2 — результат работы филиала, но, ввиду организации структуры «филиал», нужно отдельно название, отдельно адрес, поэтому разделение по точке с запятой. Аналогично с кодом 4 — база договоров определенного типа в определенном филиале.

Записи файла протокола:

1. Ошибки при начальной проверке входных файлов
2. Подробная фиксация этапов обработки (запись в списки, формирование договоров и картотек, обработка запросов, подтверждение удаления данных)
3. Сообщения о досрочном завершении программы с описанием причин остановки

Записи выходного файла исключают пункт 2, для него характерен лишь вывод ответов на запросы пользователя, либо ошибок, не позволяющих программе отработать должным образом.

**Описание используемых данных.**

String:

const unsigned int N = 5; — фиксированный размер массива символов (блочное представление строки)

struct String {

char\* text; — массив символов

char mark; — маркер, оканчивающий содержательную часть массива (в блочном представлении хотя бы один из блоков будет заполнен не до конца)

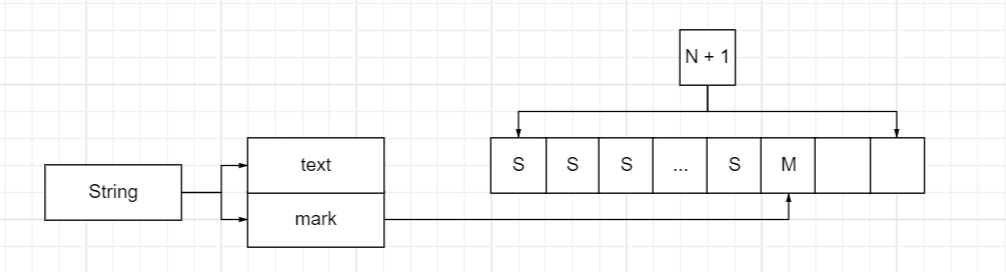
String(const unsigned int size = N); — конструктор

~String(); — деструктор

unsigned int getSize(); — метод, возвращающий размер содержательной части массива

bool isEqual(String\* other); — метод, сравнивающий, равны ли две строки (символ в символ)

};



Item:

template<typename T>

struct Item {

T\* val; — указатель на данные типа T

Item<T>\* next; -- указатель на следующий элемент списка

Item() { — конструктор

this->val = new T;

this->next = nullptr;

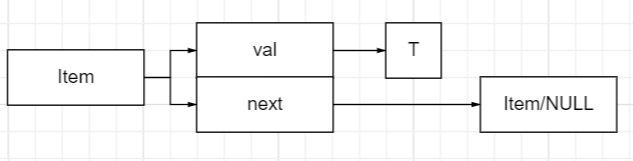
}

~Item() { — деструктор

delete this->val;

}

};



List:

template<typename T>

struct List {

Item<T>\* first; — указатель на первый элемент в списке

Item<T>\* last; — указательн на последний элемент в списке

List() { — конструктор

this->first = nullptr;

this->last = nullptr;

}

~List() { — деструктор

this->deleteList();

}

bool isEmpty() { — метод, проверяющий, пуст ли список

return (this->first == nullptr);

}

unsigned int getSize() { — метод, возвращающий размер списка

unsigned int size = 0;

Item<T>\* p = this->first;

while (p != nullptr) {

++size;

p = p->next;

}

return size;

}

Item<T>\* getItem(const int index) { — метод, возвращающий элемент списка по заданному индексу, если он корректен

if (index < 0 || index >= this->getSize()) return nullptr;

unsigned int pos = 0;

Item<T>\* p = this->first;

while (pos != index) {

++pos;

p = p->next;

}

return p;

}

void appendItem(Item<T>\* val) { — метод, добавляющий новый элемент в конец списка

if (this->isEmpty()) {

this->first = val;

this->last = val;

}

else {

this->last->next = val;

this->last = val;

}

}

void deleteItem(const int index) { — метод, удаляющий элемент списка по заданному индексу, если он корректен

if (this->isEmpty()) return;

int size = this->getSize();

if (index < 0 || index >= size) return;

if (size == 1) {

this->last = nullptr;

delete this->first;

this->first = nullptr;

}

else {

Item<T>\* toDel = this->getItem(index);

if (index == 0) {

this->first = toDel->next;

delete toDel;

}

else if (index == size - 1) {

this->last = this->getItem(index - 1);

this->last->next = nullptr;

delete toDel;

}

else {

Item<T>\* prev = this->getItem(index - 1);

prev->next = this->getItem(index + 1);

delete toDel;

prev = nullptr;

}

toDel = nullptr;

if (this->getSize() == 1) {

this->last = this->first;

}

}

}

void deleteList() { — метод, удаляющий весь список

unsigned int curIndex = this->getSize() - 1;

while (this->first != nullptr) {

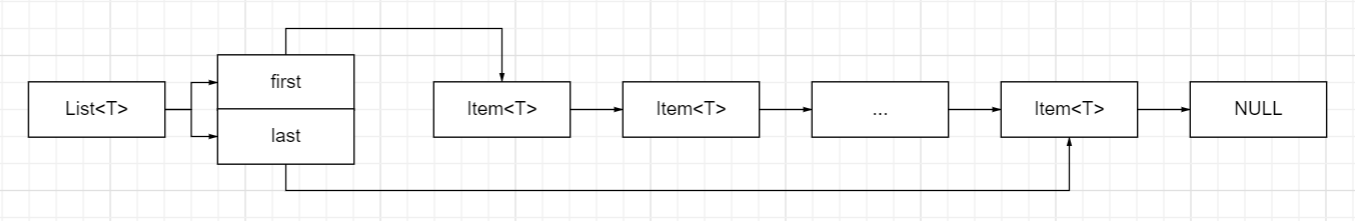
this->deleteItem(curIndex);

--curIndex;

}

}

};



Branch:

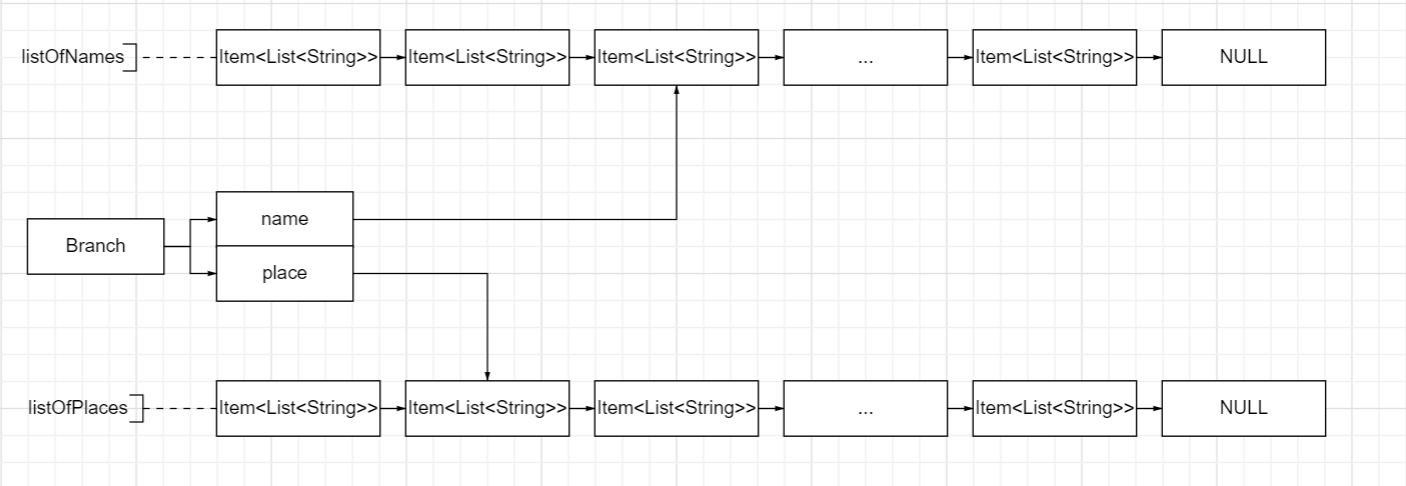
struct Branch {

List<String>\* name, \* place; — указатели на объекты из исходных списков названий и адресов

Branch(List<String>\* name = nullptr, List<String>\* place = nullptr); — конструктор

~Branch(); — деструктор

};



Case:

struct Case {

List<String>\* client, \* date, \* term, \* agent, \* type; — указатели на данные исходных списков всех категорий

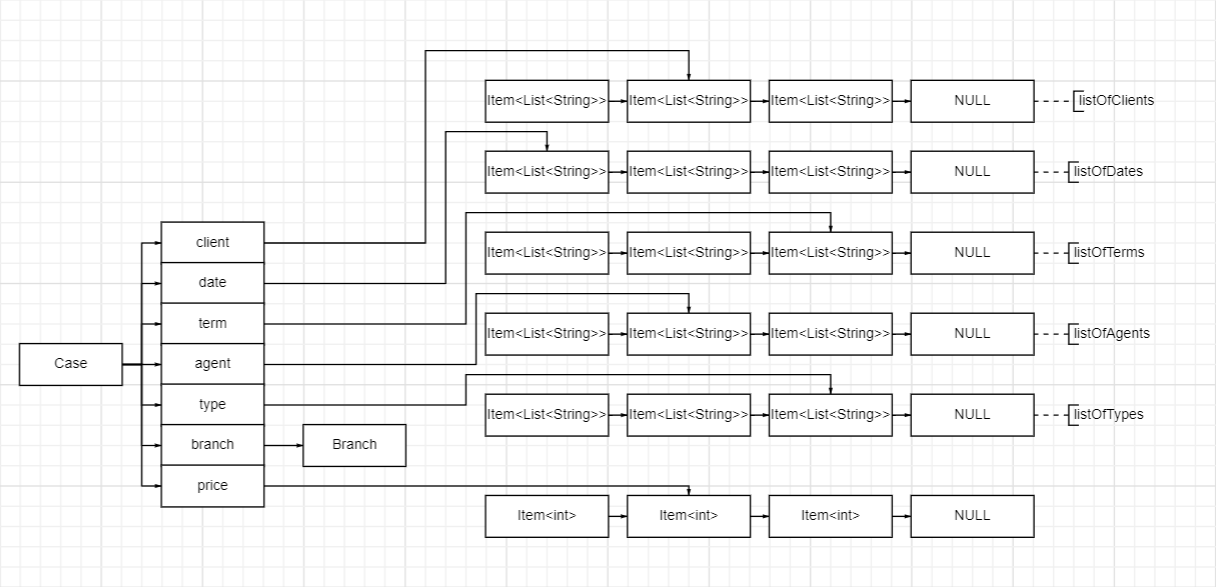
Branch\* branch; — указатель на «филиал» (выделяется память под новый объект)

int\* price; — указатель на числовые данные (страховые суммы)

Case(List<String>\* client = nullptr, List<String>\* date = nullptr, List<String>\* term = nullptr, List<String>\* agent = nullptr, List<String>\* type = nullptr, List<String>\* name = nullptr, List<String>\* place = nullptr, int\* price = nullptr); — конструктор

~Case(); — деструктор

};



GroupedCases:

template<typename T>

struct GroupedCases {

T\* val; — указатель на значение, которому соответствует список договоров

List<Case>\* cases; — список договоров, имеющих общее значение

GroupedCases(T\* val = nullptr) { — конструктор

this->val = val;

this->cases = new List<Case>;

}

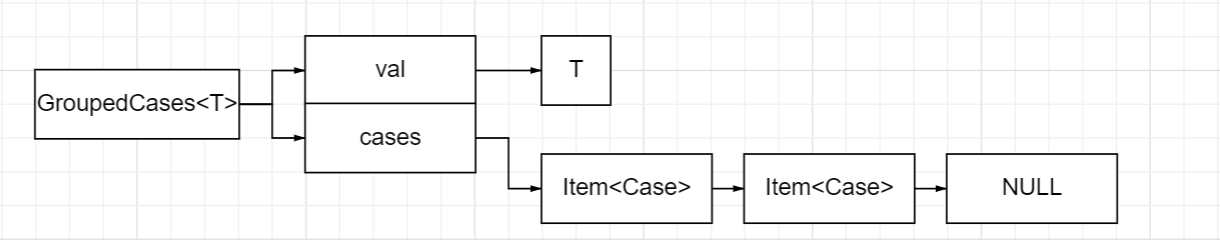
~GroupedCases() { — деструктор

delete this->cases;

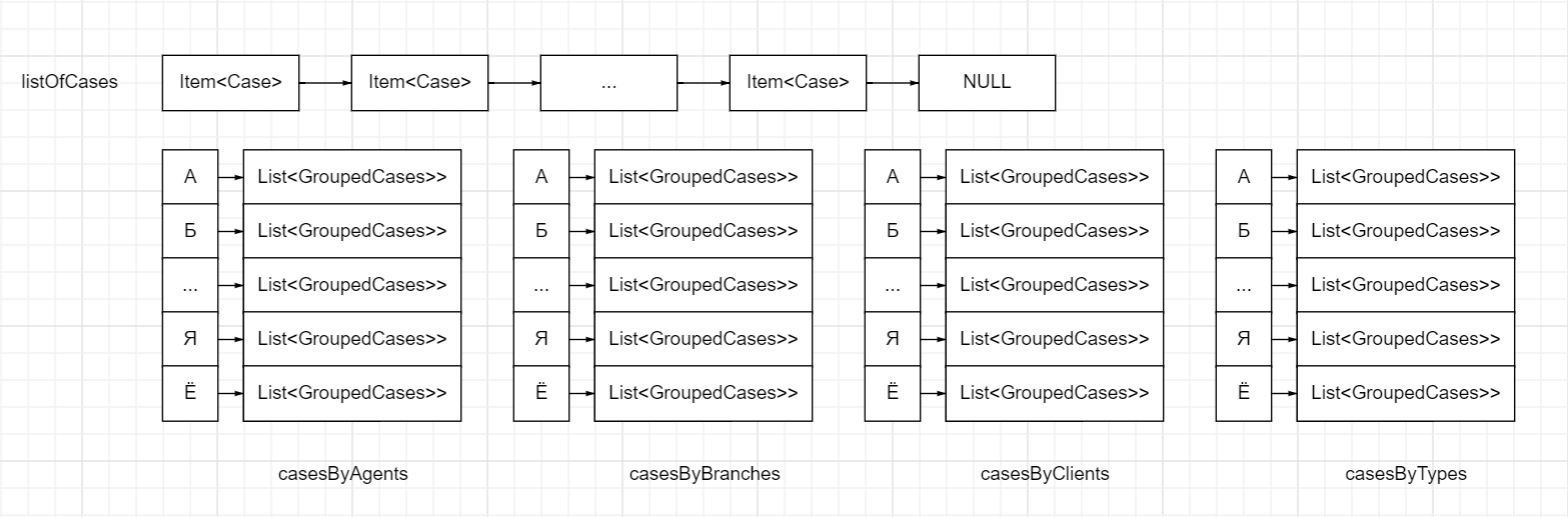
this->val = nullptr;

}

};



Итоговая структура эффективной организации данных:



**Описание используемых функций.**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Имя | Файл | Назначение | Параметры | | | |
| входные | выходные | модифицируемые | транзитные |
| checkFiles | process.cpp | первичная проверка входных файлов | files |  |  |  |
| writeToList | запись данных в список | file | list |  |  |
| writeToNumList | то же, что и выше, но с числовыми данными | file, log | list |  |  |
| printList | форматированный вывод списка | list |  | file |  |
| printString | форматированный вывод строки | list |  | file |  |
| areEqual | сравнение строк | a, b |  |  |  |
| areBranchesEqual | сравнение «филиалов» | a, b |  |  |  |
| isInRegistry | проверка вхождения объекта в картотеку | list, val |  |  |  |
| isBranchInRegistry | проверка вхождения «филиала» в картотеку | list, val |  |  |  |
| indexInRegistry | возврат индекса вхождения объекта в картотеку | list, val | pos |  |  |
| indexBranchInRegistry | возврат индекса вхождения «филиала» в картотеку | list, val | pos |  |  |
| newCase | создание нового Case объекта, наполнение данными | date, term, price, name, place, client, agent, type | tmpItem |  |  |
| printCase | форматированный вывод договоров | p |  | file |  |
| printCaseCheckBranch | то же, что и выше, но с проверкой на пересечение картотек | p, val |  | file |  |
| process | основная функция-обработчик |  |  |  |  |

**Организация ввода-вывода.**

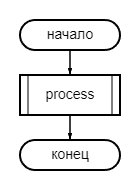
|  |  |  |  |
| --- | --- | --- | --- |
| Библиотека | Имя | Команда | Назначение |
| fstream |  | >> | считывание с файла |
|  | << | запись в файл |
|  | unsetf | снятие флага чтения/записи |
|  | eof | проверка на достижение EOF |
|  | open | открытие файла |
|  | close | закрытие файла |
|  | clear | снятие всех флагов |

**Представление алгоритма решения задачи.**

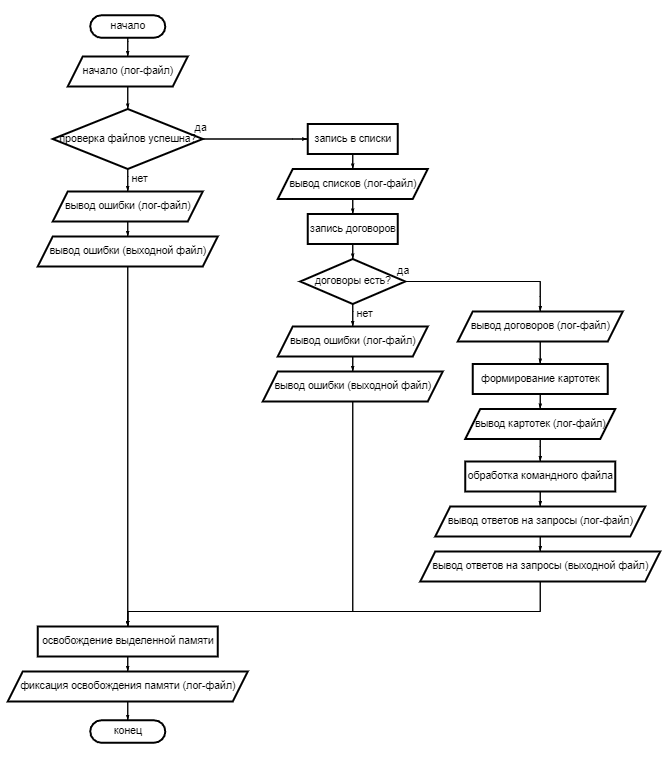
1. Проверить, что все входные файлы на месте и не пустые
2. Заполнить списки данными
3. Обработать файл с договорами, сформировать список из договоров, где указаны корректные индексы на исходные данные
4. Если список оказался не пуст, сформировать из имеющихся договоров картотеку по категориям
5. Обработать командный файл с запросами, вывести ответы на корректные запросы
6. Освободить память от исходных данных и завершить обработку

**Блок-схема.**

main:



process:



**Текст программы.**

main.cpp:

/\*

Курсовая работа за 2 семестр. Задание 7.3:

Организовать информационную систему страховой компании по договорам страхования частных лиц.

Должны храниться сведения о:

- филиалах кампании (название, местонахождение),

- страховых агентах в каждом филиале (ФИО),

- застрахованных лицах (ФИО),

- договорах страхования (дата заключения, срок, страховая сумма, филиал, агент) по виду страхования

(например, страхование автотранспорта от угона, страхование домашнего имущества, добровольное медицинское страхование и т.п.).

Факт заключения нового договора страхования фиксируется.

Необходимо обеспечить эффективную выдачу сведений:

- о результатах работы определенного страхового агента,

- о результатах работы определенного филиала,

- о договорах страхования определенного частного лица,

- о договорах страхования определенного вида (по филиалам и компании в целом).

Автор: Савин Павел Алексеевич, группа 2309.

Даты: Начало: 18.04.2023 Окончание 15.05.2023

\*/

#include "process.h"

// Иерархия включений модулей позволяет в main только вызвать подключенный process, а все остальное отработает за пределами main (условно, сказано для наглядности).

int main() {

process();

return 0;

}

Files.h (Files):

#pragma once

#include <fstream>

/\*

Класс Files - структура под все имеющиеся файлы в программе.

Использование в связке с указателями позволяет через обычный синтаксис указателей открывать и закрывать все файлы сразу.

\*/

struct Files {

std::ifstream names;

std::ifstream places;

std::ifstream agents;

std::ifstream people;

std::ifstream dates;

std::ifstream terms;

std::ifstream prices;

std::ifstream types;

std::ifstream cases;

std::ifstream commands;

std::ofstream logFile;

std::ofstream outFile;

Files(const char\* names = "names.txt", const char\* places = "places.txt", const char\* agents = "agents.txt", const char\* people = "people.txt", const char\* dates = "dates.txt", const char\* terms = "terms.txt", const char\* prices = "prices.txt", const char\* types = "types.txt", const char\* cases = "cases.txt", const char\* commands = "commands.txt", const char\* log = "log.txt", const char\* out = "out.txt");

~Files();

};

Files.cpp (Files):

#include "Files.h"

Files::Files(const char\* names, const char\* places, const char\* agents, const char\* people, const char\* dates, const char\* terms, const char\* prices, const char\* types, const char\* cases, const char\* commands, const char\* log, const char\* out) {

this->names.open(names);

this->places.open(places);

this->agents.open(agents);

this->people.open(people);

this->dates.open(dates);

this->terms.open(terms);

this->prices.open(prices);

this->types.open(types);

this->cases.open(cases);

this->commands.open(commands);

this->logFile.open(log);

this->outFile.open(out);

}

Files::~Files() {

this->names.close();

this->places.close();

this->agents.close();

this->people.close();

this->dates.close();

this->terms.close();

this->prices.close();

this->types.close();

this->cases.close();

this->commands.close();

this->logFile.close();

this->outFile.close();

}

String.h (String):

#pragma once

/\*

Класс String, являющийся реализацией строки.

Строка представлена как массив символов, оканчивающийся заданным маркером.

Ввиду требования на поблочное считывание введен константный счетчик, ограничивающий размерность массива.

Счетчик настраивается изменением значения до запуска программы.

\*/

const unsigned int N = 5;

struct String {

char\* text;

char mark;

String(const unsigned int size = N);

~String();

unsigned int getSize();

bool isEqual(String\* other);

};

String.cpp (String):

#include "String.h"

String::String(const unsigned int size) {

this->text = new char[size + 1];

this->text[0] = '\0';

this->mark = '\0';

}

String::~String() {

delete[] this->text;

}

unsigned int String::getSize() {

unsigned int i = 0;

while (this->text[i++] != this->mark);

return i;

}

bool String::isEqual(String\* other) {

if (this->getSize() != other->getSize()) return false;

for (int i = 0; i < this->getSize(); ++i) {

if (this->text[i] != other->text[i]) return false;

}

return true;

}

List.h (List, Item):

#pragma once

/\*

Классы Item и List - элемент списка и сам список (технически, формуляр - класс, управляющий списком).

Создание нескольких классов для реализации списков под каждый тип - плохая идея, поэтому классы под элемент списка и сам список шаблонизированы.

Все описание содержится внутри одного файла - по правилу описание шаблонов нельзя разделять на объявление и определение в разные файлы.

Все методы списка изначально построены так, что не зависят от принимаемого типа.

\*/

template<typename T>

struct Item {

T\* val;

Item<T>\* next;

Item() {

this->val = new T;

this->next = nullptr;

}

~Item() {

delete this->val;

}

};

template<typename T>

struct List {

Item<T>\* first;

Item<T>\* last;

List() {

this->first = nullptr;

this->last = nullptr;

}

~List() {

this->deleteList();

}

bool isEmpty() {

return (this->first == nullptr);

}

unsigned int getSize() {

unsigned int size = 0;

Item<T>\* p = this->first;

while (p != nullptr) {

++size;

p = p->next;

}

return size;

}

Item<T>\* getItem(const int index) {

if (index < 0 || index >= this->getSize()) return nullptr;

unsigned int pos = 0;

Item<T>\* p = this->first;

while (pos != index) {

++pos;

p = p->next;

}

return p;

}

void appendItem(Item<T>\* val) {

if (this->isEmpty()) {

this->first = val;

this->last = val;

}

else {

this->last->next = val;

this->last = val;

}

}

void deleteItem(const int index) {

if (this->isEmpty()) return;

int size = this->getSize();

if (index < 0 || index >= size) return;

if (size == 1) {

this->last = nullptr;

delete this->first;

this->first = nullptr;

}

else {

Item<T>\* toDel = this->getItem(index);

if (index == 0) {

this->first = toDel->next;

delete toDel;

}

else if (index == size - 1) {

this->last = this->getItem(index - 1);

this->last->next = nullptr;

delete toDel;

}

else {

Item<T>\* prev = this->getItem(index - 1);

prev->next = this->getItem(index + 1);

delete toDel;

prev = nullptr;

}

toDel = nullptr;

if (this->getSize() == 1) {

this->last = this->first;

}

}

}

void deleteList() {

unsigned int curIndex = this->getSize() - 1;

while (this->first != nullptr) {

this->deleteItem(curIndex);

--curIndex;

}

}

};

Branch.h (Branch):

#pragma once

#include "List.h"

#include "String.h"

/\*

Класс Branch, который просто объединяет два объекта из исходных данных - название филиала и его адрес.

Является неудобным костылем, вводящим трудности в генерализацию функций, но имеет плюс в виде промежуточной ветки в иерархии подключении модулей.

\*/

struct Branch {

List<String>\* name, \* place;

Branch(List<String>\* name = nullptr, List<String>\* place = nullptr);

~Branch();

};

Branch.cpp (Branch):

#include "Branch.h"

Branch::Branch(List<String>\* name, List<String>\* place) {

this->name = name;

this->place = place;

}

Branch::~Branch() {

this->name = nullptr;

this->place = nullptr;

}

Case.h (Case):

#pragma once

#include "Branch.h"

/\*

Класс Case, являющийся структурой для страхового договора.

По требованию не содержит в себе новых данных, не считая выделение памяти под новый экземпляр, который будет эти не "новые" данные хранить.

\*/

struct Case {

List<String>\* client, \* date, \* term, \* agent, \* type;

Branch\* branch;

int\* price;

Case(List<String>\* client = nullptr, List<String>\* date = nullptr, List<String>\* term = nullptr, List<String>\* agent = nullptr, List<String>\* type = nullptr, List<String>\* name = nullptr, List<String>\* place = nullptr, int\* price = nullptr);

~Case();

};

Case.cpp (Case):

#include "Case.h"

Case::Case(List<String>\* client, List<String>\* date, List<String>\* term, List<String>\* agent, List<String>\* type, List<String>\* name, List<String>\* place, int\* price) {

this->client = client;

this->date = date;

this->term = term;

this->branch = new Branch(name, place);

this->agent = agent;

this->price = price;

this->type = type;

}

Case::~Case() {

this->client = nullptr;

this->date = nullptr;

this->term = nullptr;

this->agent = nullptr;

this->type = nullptr;

delete this->branch;

this->price = nullptr;

}

GroupedCases.h (GroupedCases):

#pragma once

#include "Case.h"

/\*

Класс GroupedCases, являющийся этаким словарем для значения.

Привязанному к объекту из исходных данных привязывается список всех страховых договоров, где этот объект фигурирует.

\*/

template<typename T>

struct GroupedCases {

T\* val;

List<Case>\* cases;

GroupedCases(T\* val = nullptr) {

this->val = val;

this->cases = new List<Case>;

}

~GroupedCases() {

delete this->cases;

this->val = nullptr;

}

};

process.h:

#pragma once

#include "Files.h"

#include "GroupedCases.h"

bool checkFiles(Files\* files);

void writeToList(std::ifstream& file, List<List<String>>\* list);

void writeToNumList(std::ifstream& file, std::ofstream& log, List<int>\* list);

void printList(List<List<String>>\* list, std::ofstream& file);

void printString(List<String>\* list, std::ofstream& file);

bool areEqual(List<String>\* a, List<String>\* b);

bool areBranchesEqual(Branch\* a, Branch\* b);

bool isInRegistry(List<GroupedCases<List<String>>>\* list, List<String>\* val);

bool isBranchInRegistry(List<GroupedCases<Branch>>\* list, Branch\* val);

int indexInRegistry(List<GroupedCases<List<String>>>\* list, List<String>\* val);

int indexBranchInRegistry(List<GroupedCases<Branch>>\* list, Branch\* val);

Item<Case>\* newCase(Item<List<String>>\* date, Item<List<String>>\* term, Item<int>\* price, Item<List<String>>\* name, Item<List<String>>\* place, Item<List<String>>\* client, Item<List<String>>\* agent, Item<List<String>>\* type);

void printCase(Item<Case>\* p, std::ofstream& file);

int printCaseCheckBranch(Item<Case>\* p, std::ofstream& file, Branch\* val);

void process();

process.cpp:

#include "process.h"

/\*

Функция checkFiles, занимающаяся первичной проверкой всех входных файлов.

Отсекает возможность продолжения работы, если хоть один из файлов не найден/пуст, и записывает все ошибки в протокол (и уведомляет пользователя о них)

\*/

bool checkFiles(Files\* files) {

bool flag = true;

char tmp;

if (files->names.is\_open()) {

files->names.unsetf(std::ios::skipws);

files->names >> tmp;

if (files->names.eof()) {

flag = false;

files->logFile << "Файл с названиями филиалов пустой.\n";

files->outFile << "Ошибка: файл с названиями филиалов пустой.\n";

}

else {

files->names.clear();

files->names.seekg(0);

files->names.setf(std::ios::skipws);

}

}

else {

flag = false;

files->logFile << "Файл с названиями филиалов не найден.\n";

files->outFile << "Ошибка: файл с названиями филиалов не найден.\n";

}

if (files->places.is\_open()) {

files->places.unsetf(std::ios::skipws);

files->places >> tmp;

if (files->places.eof()) {

flag = false;

files->logFile << "Файл с адресами филиалов пустой.\n";

files->outFile << "Ошибка: файл с адресами филиалов пустой.\n";

}

else {

files->places.clear();

files->places.seekg(0);

files->places.setf(std::ios::skipws);

}

}

else {

flag = false;

files->logFile << "Файл с адресами филиалов не найден.\n";

files->outFile << "Ошибка: файл с адресами филиалов не найден.\n";

}

if (files->agents.is\_open()) {

files->agents.unsetf(std::ios::skipws);

files->agents >> tmp;

if (files->agents.eof()) {

flag = false;

files->logFile << "Файл с ФИО агентов пустой.\n";

files->outFile << "Ошибка: файл с ФИО агентов пустой.\n";

}

else {

files->agents.clear();

files->agents.seekg(0);

files->agents.setf(std::ios::skipws);

}

}

else {

flag = false;

files->logFile << "Файл с ФИО агентов не найден.\n";

files->outFile << "Ошибка: файл с ФИО агентов не найден.\n";

}

if (files->people.is\_open()) {

files->people.unsetf(std::ios::skipws);

files->people >> tmp;

if (files->people.eof()) {

flag = false;

files->logFile << "Файл с ФИО клиентов пустой.\n";

files->outFile << "Ошибка: файл с ФИО клиентов пустой.\n";

}

else {

files->people.clear();

files->people.seekg(0);

files->people.setf(std::ios::skipws);

}

}

else {

flag = false;

files->logFile << "Файл с ФИО клиентов не найден.\n";

files->outFile << "Ошибка: файл с ФИО клиентов не найден.\n";

}

if (files->dates.is\_open()) {

files->dates.unsetf(std::ios::skipws);

files->dates >> tmp;

if (files->dates.eof()) {

flag = false;

files->logFile << "Файл с датами пустой.\n";

files->outFile << "Ошибка: файл с датами пустой.\n";

}

else {

files->dates.clear();

files->dates.seekg(0);

files->dates.setf(std::ios::skipws);

}

}

else {

flag = false;

files->logFile << "Файл с датами не найден.\n";

files->outFile << "Ошибка: файл с датами не найден.\n";

}

if (files->terms.is\_open()) {

files->terms.unsetf(std::ios::skipws);

files->terms >> tmp;

if (files->terms.eof()) {

flag = false;

files->logFile << "Файл со сроками пустой.\n";

files->outFile << "Ошибка: файл со сроками пустой.\n";

}

else {

files->terms.clear();

files->terms.seekg(0);

files->terms.setf(std::ios::skipws);

}

}

else {

flag = false;

files->logFile << "Файл со сроками не найден.\n";

files->outFile << "Ошибка: файл со сроками не найден.\n";

}

if (files->prices.is\_open()) {

files->prices.unsetf(std::ios::skipws);

files->prices >> tmp;

if (files->prices.eof()) {

flag = false;

files->logFile << "Файл с суммами пустой.\n";

files->outFile << "Ошибка: файл с суммами пустой.\n";

}

else {

files->prices.clear();

files->prices.seekg(0);

files->prices.setf(std::ios::skipws);

}

}

else {

flag = false;

files->logFile << "Файл с суммами не найден.\n";

files->outFile << "Ошибка: файл с суммами не найден.\n";

}

if (files->types.is\_open()) {

files->types.unsetf(std::ios::skipws);

files->types >> tmp;

if (files->types.eof()) {

flag = false;

files->logFile << "Файл с типами страхования пустой.\n";

files->outFile << "Ошибка: файл с типами страхования пустой.\n";

}

else {

files->types.clear();

files->types.seekg(0);

files->types.setf(std::ios::skipws);

}

}

else {

flag = false;

files->logFile << "Файл с типами страхования не найден.\n";

files->outFile << "Ошибка: файл с типами страхования не найден.\n";

}

if (files->cases.is\_open()) {

files->cases.unsetf(std::ios::skipws);

files->cases >> tmp;

if (files->cases.eof()) {

flag = false;

files->logFile << "Файл со случаями страхования пустой.\n";

files->outFile << "Ошибка: файл со случаями страхования пустой.\n";

}

else {

files->cases.clear();

files->cases.seekg(0);

files->cases.setf(std::ios::skipws);

}

}

else {

flag = false;

files->logFile << "Файл со случаями страхования не найден.\n";

files->outFile << "Ошибка: файл со случаями страхования не найден.\n";

}

if (files->commands.is\_open()) {

files->commands.unsetf(std::ios::skipws);

files->commands >> tmp;

if (files->commands.eof()) {

flag = false;

files->logFile << "Командный файл пустой.\n";

files->outFile << "Ошибка: командный файл пустой.\n";

}

else {

files->commands.clear();

files->commands.seekg(0);

files->commands.setf(std::ios::skipws);

}

}

else {

flag = false;

files->logFile << "Командный файл не найден.\n";

files->outFile << "Ошибка: командный файл не найден.\n";

}

return flag;

}

// Далее идет блок функций, которые заменяют повторяющиеся алгоритмы в исходной функции обработки.

/\*

Функция writeToList, заменяющая алгоритм записи исходных данных из входного файла в соответствующий список.

\*/

void writeToList(std::ifstream& file, List<List<String>>\* list) {

file.unsetf(std::ios::skipws);

char tmp;

Item<String>\* tmpItem = new Item<String>;

Item<List<String>>\* tmpList = new Item<List<String>>;

int charsCount = 0, blocksCount = 0, strCount = 0;

do {

file >> tmp;

if (file.eof()) {

if (charsCount || blocksCount || strCount) {

tmpList->val->appendItem(tmpItem);

list->appendItem(tmpList);

tmpList = nullptr;

tmpItem = nullptr;

charsCount = 0;

++blocksCount;

++strCount;

}

else {

delete tmpItem;

delete tmpList;

}

break;

}

if (tmp == '\n') {

tmpList->val->appendItem(tmpItem);

++blocksCount;

list->appendItem(tmpList);

++strCount;

tmpItem = new Item<String>;

tmpList = new Item<List<String>>;

charsCount = 0;

}

else {

if (charsCount == N) {

tmpList->val->appendItem(tmpItem);

++blocksCount;

tmpItem = new Item<String>;

charsCount = 0;

}

tmpItem->val->text[charsCount] = tmp;

tmpItem->val->text[charsCount + 1] = '\0';

++charsCount;

}

}

while (true);

}

/\*

Функция writeToNumList, делающая тоже самое, что и writeToList, но в список числовых данных и с нюансом: алгоритм отбрасывает посторонние данные - отрицательные числа.

\*/

void writeToNumList(std::ifstream& file, std::ofstream& log, List<int>\* list) {

long int tmp;

Item<int>\* item;

do {

file >> tmp;

if (file.eof()) break;

else if (tmp < 0) log << "Обнаружено отрицательное число - " << tmp << ". Оно будет отброшено.\n";

else {

item = new Item<int>;

\*(item->val) = tmp;

list->appendItem(item);

item = nullptr;

log << tmp << "->\n";

}

}

while (true);

log << "NULL\n\n";

}

/\*

Функция printList, заменяющая алгоритм форматированного вывода списка.

\*/

void printList(List<List<String>>\* list, std::ofstream& file) {

Item<List<String>>\* tmpList = list->first;

while (tmpList != nullptr) {

Item<String>\* tmpItem = tmpList->val->first;

while (tmpItem != nullptr) {

int i = 0;

while (tmpItem->val->text[i] != '\0') {

file << tmpItem->val->text[i];

++i;

}

file << "->";

tmpItem = tmpItem->next;

}

file << "NULL\n|\nV\n";

tmpList = tmpList->next;

}

file << "NULL\n\n";

}

/\*

Функция printString, заменяющая алгоритм вывода строки как строки, а не как списка массивов символов.

\*/

void printString(List<String>\* list, std::ofstream& file) {

Item<String>\* tmpItem = list->first;

while (tmpItem != nullptr) {

int i = 0;

while (tmpItem->val->text[i] != '\0') {

file << tmpItem->val->text[i];

++i;

}

tmpItem = tmpItem->next;

}

}

/\*

Функция areEqual, сравнивающая два списка из строк.

Реализована здесь из-за шаблонности реализации класса List (нет смысла писать один метод, который еще и является специализацией шаблона строки).

Сравнение происходит строка в строку.

\*/

bool areEqual(List<String>\* a, List<String>\* b) {

if (a->getSize() != b->getSize()) return false;

Item<String>\* pa = a->first, \* pb = b->first;

while (pa != nullptr) {

if (!pa->val->isEqual(pb->val)) return false;

pa = pa->next; pb = pb->next;

}

return true;

}

/\*

Функция areBranchesEqual, сравнивающая два объекта филиалов.

Реализована из-за существования класса Branch, который свзяывает между собой два члена в один.

На деле просто является адаптацией алгоритма сравнения для двух пар строк сразу.

\*/

bool areBranchesEqual(Branch\* a, Branch\* b) {

if (areEqual(a->name, b->name) && areEqual(a->place, b->place)) return true;

return false;

}

/\*

Функция isInRegistry, проверяющая, есть ли запись об объекте в картотеке.

\*/

bool isInRegistry(List<GroupedCases<List<String>>>\* list, List<String>\* val) {

Item<GroupedCases<List<String>>>\* p = list->first;

while (p != nullptr) {

if (areEqual(p->val->val, val)) return true;

p = p->next;

}

return false;

}

/\*

Функция isBranchInRegistry, делающая тоже самое, что и isInRegistry, но с объектом класса Branch.

\*/

bool isBranchInRegistry(List<GroupedCases<Branch>>\* list, Branch\* val) {

Item<GroupedCases<Branch>>\* p = list->first;

while (p != nullptr) {

if (areBranchesEqual(p->val->val, val)) return true;

p = p->next;

}

return false;

}

/\*

Функция indexInRegistry, возвращающая индекс заданного объекта в картотеке.

Возвращает -1, если объект не записан в картотеку.

\*/

int indexInRegistry(List<GroupedCases<List<String>>>\* list, List<String>\* val) {

int pos = 0;

Item<GroupedCases<List<String>>>\* p = list->first;

while (p != nullptr) {

if (areEqual(p->val->val, val)) return pos;

++pos;

p = p->next;

}

return -1;

}

/\*

Функция indexBranchInRegistry, делающая тоже самое, что и indexInRegistry, но с объектом класса Branch.

\*/

int indexBranchInRegistry(List<GroupedCases<Branch>>\* list, Branch\* val) {

int pos = 0;

Item<GroupedCases<Branch>>\* p = list->first;

while (p != nullptr) {

if (areBranchesEqual(p->val->val, val)) return pos;

++pos;

p = p->next;

}

return -1;

}

/\*

Функция newCase, заменяющая алгоритм создания нового страхового случая.

\*/

Item<Case>\* newCase(Item<List<String>>\* date, Item<List<String>>\* term, Item<int>\* price, Item<List<String>>\* name, Item<List<String>>\* place, Item<List<String>>\* client, Item<List<String>>\* agent, Item<List<String>>\* type) {

Item<Case>\* tmpItem = new Item<Case>;

tmpItem->val->date = date->val;

tmpItem->val->term = term->val;

tmpItem->val->price = price->val;

tmpItem->val->branch->name = name->val;

tmpItem->val->branch->place = place->val;

tmpItem->val->client = client->val;

tmpItem->val->agent = agent->val;

tmpItem->val->type = type->val;

return tmpItem;

}

/\*

Функция printCase, заменяющая алгоритм форматированного вывода страховых договоров.

\*/

void printCase(Item<Case>\* p, std::ofstream& file) {

while (p != nullptr) {

file << "Дата заключения: "; printString(p->val->date, file);

file << "\nСрок: "; printString(p->val->term, file);

file << "\nСтраховая сумма: " << \*(p->val->price);

file << "\nФилиал: \""; printString(p->val->branch->name, file); file << "\", "; printString(p->val->branch->place, file);

file << "\nФИО клиента: "; printString(p->val->client, file);

file << "\nФИО агента: "; printString(p->val->agent, file);

file << "\nТип страхования: "; printString(p->val->type, file); file << "\n\n";

p = p->next;

}

}

/\*

Функция printCaseCheckBranch, являющаяся адаптацией printCase под вывод запроса по типу и по филиалу.

Если договор находится, происходит вывод, если ни одного не нашлось, об этом будет соответствующее сообщение - все по счетчику.

\*/

int printCaseCheckBranch(Item<Case>\* p, std::ofstream& file, Branch\* val) {

int count = 0;

while (p != nullptr) {

if (areBranchesEqual(p->val->branch, val)) {

file << "Дата заключения: "; printString(p->val->date, file);

file << "\nСрок: "; printString(p->val->term, file);

file << "\nСтраховая сумма: " << \*(p->val->price);

file << "\nФилиал: \""; printString(p->val->branch->name, file); file << "\", "; printString(p->val->branch->place, file);

file << "\nФИО клиента: "; printString(p->val->client, file);

file << "\nФИО агента: "; printString(p->val->agent, file);

file << "\nТип страхования: "; printString(p->val->type, file); file << "\n\n";

++count;

}

p = p->next;

}

return count;

}

// Конец блока замещающих функций.

/\*

Функция process, занимающаяся обработкой данных.

Алгоритм обработки:

1. Проверить, что все входные файлы на месте и не пустые

2. Заполнить списки данными

3. Обработать файл с договорами, сформировать список из договоров, где указаны корректные индексы на исходные данные

4. Если список оказался не пуст, сформировать из имеющихся договоров картотеку по категориям

5. Обработать командный файл с запросами, вывести ответы на корректные запросы

6. Освободить память от исходных данных и завершить обработку

\*/

void process() {

Files\* files = new Files;

files->logFile << "Программа начала свою работу.\n\nПроизводится проверка входных файлов на существование и наличие содержимого.\n\n";

if (checkFiles(files)) {

List<List<String>>\* listOfNames = new List<List<String>>, \* listOfPlaces = new List<List<String>>, \* listOfAgents = new List<List<String>>, \* listOfClients = new List<List<String>>, \* listOfDates = new List<List<String>>, \* listOfTerms = new List<List<String>>, \* listOfTypes = new List<List<String>>;

List<int>\* listOfPrices = new List<int>;

files->logFile << "Все входные файлы найдены и не пустые.\nПроизводится запись исходнных данных в списки.\n\nЗаполняется список названий филиалов.\n\n";

writeToList(files->names, listOfNames);

printList(listOfNames, files->logFile);

files->logFile << "Заполнен список названий филиалов.\n\nЗаполняется список адресов филиалов.\n\n";

writeToList(files->places, listOfPlaces);

printList(listOfPlaces, files->logFile);

files->logFile << "Заполнен список адресов филиалов.\n\nЗаполняется список ФИО агентов.\n\n";

writeToList(files->agents, listOfAgents);

printList(listOfAgents, files->logFile);

files->logFile << "Заполнен список ФИО агентов.\n\nЗаполняется список ФИО клиентов.\n\n";

writeToList(files->people, listOfClients);

printList(listOfClients, files->logFile);

files->logFile << "Заполнен список ФИО клиентов.\n\nЗаполняется список дат страхования.\n\n";

writeToList(files->dates, listOfDates);

printList(listOfDates, files->logFile);

files->logFile << "Заполнен список дат страхования.\n\nЗаполняется список сроков страхования.\n\n";

writeToList(files->terms, listOfTerms);

printList(listOfTerms, files->logFile);

files->logFile << "Заполнен список сроков страхования.\n\nЗаполняется список типов страхования.\n\n";

writeToList(files->types, listOfTypes);

printList(listOfTypes, files->logFile);

files->logFile << "Заполнен список типов страхования.\n\nЗаполняется список страховых сумм.\n\n";

writeToNumList(files->prices, files->logFile, listOfPrices);

if (listOfPrices->isEmpty()) {

files->logFile << "В ходе отбрасывания список страховых сумм остался пустым.\nДальнейшая работа программы не имеет смысла.\n";

files->outFile << "В ходе отбрасывания список страховых сумм остался пустым.\nДальнейшая работа программы не имеет смысла.\n";

}

else {

files->logFile << "Заполнен список страховых сумм.\n\nЗаполняется список заключенных договоров.\n\n";

List<Case>\* listOfCases = new List<Case>;

Item<GroupedCases<List<String>>>\* tmpReg;

Item<GroupedCases<Branch>>\* tmpBranchReg;

List<GroupedCases<List<String>>>\*\* casesByAgents = new List<GroupedCases<List<String>>>\* [33];

for (int i = 0; i < 33; ++i) casesByAgents[i] = new List<GroupedCases<List<String>>>;

List<GroupedCases<Branch>>\*\* casesByBranches = new List<GroupedCases<Branch>>\* [33];

for (int i = 0; i < 33; ++i) casesByBranches[i] = new List<GroupedCases<Branch>>;

List<GroupedCases<List<String>>>\*\* casesByClients = new List<GroupedCases<List<String>>>\* [33];

for (int i = 0; i < 33; ++i) casesByClients[i] = new List<GroupedCases<List<String>>>;

List<GroupedCases<List<String>>>\*\* casesByTypes = new List<GroupedCases<List<String>>>\* [33];

for (int i = 0; i < 33; ++i) casesByTypes[i] = new List<GroupedCases<List<String>>>;

Item<Case>\* tmpItem;

int tmpDate, tmpTerm, tmpPrice, tmpName, tmpPlace, tmpClient, tmpAgent, tmpType;

bool flag;

do {

flag = false;

files->cases >> tmpDate >> tmpTerm >> tmpPrice >> tmpName >> tmpPlace >> tmpClient >> tmpAgent >> tmpType;

files->logFile << tmpDate << ' ' << tmpTerm << ' ' << tmpPrice << ' ' << tmpName << ' ' << tmpPlace << ' ' << tmpClient << ' ' << tmpAgent << ' ' << tmpType << "\n";

Item<List<String>>\* date = listOfDates->getItem(tmpDate);

if (date == nullptr) {

flag = true;

files->logFile << "Некорректный индекс для списка дат - " << tmpDate << ".\n";

}

Item<List<String>>\* term = listOfTerms->getItem(tmpTerm);

if (term == nullptr) {

flag = true;

files->logFile << "Некорректный индекс для списка сроков - " << tmpTerm << ".\n";

}

Item<int>\* price = listOfPrices->getItem(tmpPrice);

if (price == nullptr) {

flag = true;

files->logFile << "Некорректный индекс для списка страховых сумм - " << tmpPrice << ".\n";

}

Item<List<String>>\* name = listOfNames->getItem(tmpName);

if (name == nullptr) {

flag = true;

files->logFile << "Некорректный индекс для списка названий - " << tmpName << ".\n";

}

Item<List<String>>\* place = listOfPlaces->getItem(tmpPlace);

if (place == nullptr) {

flag = true;

files->logFile << "Некорректный индекс для списка адресов - " << tmpPlace << ".\n";

}

Item<List<String>>\* client = listOfClients->getItem(tmpClient);

if (client == nullptr) {

flag = true;

files->logFile << "Некорректный индекс для списка ФИО клиентов - " << tmpClient << ".\n";

}

Item<List<String>>\* agent = listOfAgents->getItem(tmpAgent);

if (agent == nullptr) {

flag = true;

files->logFile << "Некорректный индекс для списка ФИО агентов - " << tmpAgent << ".\n";

}

Item<List<String>>\* type = listOfTypes->getItem(tmpType);

if (type == nullptr) {

flag = true;

files->logFile << "Некорректный индекс для списка типов страхования - " << tmpType << ".\n";

}

if (flag) files->logFile << "Договор не подлежит записи.\n\n";

else {

char a;

tmpItem = newCase(date, term, price, name, place, client, agent, type);

listOfCases->appendItem(tmpItem);

tmpItem = newCase(date, term, price, name, place, client, agent, type);

a = tmpItem->val->agent->first->val->text[0];

if (!isInRegistry(casesByAgents[a == 'Ё' ? 32 : a + 64], tmpItem->val->agent)) {

tmpReg = new Item<GroupedCases<List<String>>>;

tmpReg->val->val = tmpItem->val->agent;

tmpReg->val->cases->appendItem(tmpItem);

casesByAgents[a == 'Ё' ? 32 : a + 64]->appendItem(tmpReg);

}

else casesByAgents[a == 'Ё' ? 32 : a + 64]->getItem(indexInRegistry(casesByAgents[a == 'Ё' ? 32 : a + 64], tmpItem->val->agent))->val->cases->appendItem(tmpItem);

tmpItem = newCase(date, term, price, name, place, client, agent, type);

a = tmpItem->val->branch->name->first->val->text[0];

if (!isBranchInRegistry(casesByBranches[a == 'Ё' ? 32 : a + 64], tmpItem->val->branch)) {

tmpBranchReg = new Item<GroupedCases<Branch>>;

tmpBranchReg->val->val = tmpItem->val->branch;

tmpBranchReg->val->cases->appendItem(tmpItem);

casesByBranches[a == 'Ё' ? 32 : a + 64]->appendItem(tmpBranchReg);

}

else casesByBranches[a == 'Ё' ? 32 : a + 64]->getItem(indexBranchInRegistry(casesByBranches[a == 'Ё' ? 32 : a + 64], tmpItem->val->branch))->val->cases->appendItem(tmpItem);

tmpItem = newCase(date, term, price, name, place, client, agent, type);

a = tmpItem->val->client->first->val->text[0];

if (!isInRegistry(casesByClients[a == 'Ё' ? 32 : a + 64], tmpItem->val->client)) {

tmpReg = new Item<GroupedCases<List<String>>>;

tmpReg->val->val = tmpItem->val->client;

tmpReg->val->cases->appendItem(tmpItem);

casesByClients[a == 'Ё' ? 32 : a + 64]->appendItem(tmpReg);

}

else casesByClients[a == 'Ё' ? 32 : a + 64]->getItem(indexInRegistry(casesByClients[a == 'Ё' ? 32 : a + 64], tmpItem->val->client))->val->cases->appendItem(tmpItem);

tmpItem = newCase(date, term, price, name, place, client, agent, type);

a = tmpItem->val->type->first->val->text[0];

if (!isInRegistry(casesByTypes[a == 'Ё' ? 32 : a + 64], tmpItem->val->type)) {

tmpReg = new Item<GroupedCases<List<String>>>;

tmpReg->val->val = tmpItem->val->type;

tmpReg->val->cases->appendItem(tmpItem);

casesByTypes[a == 'Ё' ? 32 : a + 64]->appendItem(tmpReg);

}

else casesByTypes[a == 'Ё' ? 32 : a + 64]->getItem(indexInRegistry(casesByTypes[a == 'Ё' ? 32 : a + 64], tmpItem->val->type))->val->cases->appendItem(tmpItem);

files->logFile << "Договор успешно записан.\n\n";

}

}

while (!files->cases.eof());

files->logFile << "Заполнен список заключенных договоров.\n\nДоговоры:\n\n";

printCase(listOfCases->first, files->logFile);

files->logFile << "По агентам:\n\n";

for (int i = 0; i < 33; ++i) {

Item<GroupedCases<List<String>>>\* p = casesByAgents[i]->first;

while (p != nullptr) {

printString(p->val->val, files->logFile);

files->logFile << "\n\n";

printCase(p->val->cases->first, files->logFile);

p = p->next;

}

}

files->logFile << "По филиалам:\n\n";

for (int i = 0; i < 33; ++i) {

Item<GroupedCases<Branch>>\* p = casesByBranches[i]->first;

while (p != nullptr) {

files->logFile << '"'; printString(p->val->val->name, files->logFile); files->logFile << "\", "; printString(p->val->val->place, files->logFile);

files->logFile << "\n\n";

printCase(p->val->cases->first, files->logFile);

p = p->next;

}

}

files->logFile << "По клиентам:\n\n";

for (int i = 0; i < 33; ++i) {

Item<GroupedCases<List<String>>>\* p = casesByClients[i]->first;

while (p != nullptr) {

printString(p->val->val, files->logFile);

files->logFile << "\n\n";

printCase(p->val->cases->first, files->logFile);

p = p->next;

}

}

files->logFile << "По типам:\n\n";

for (int i = 0; i < 33; ++i) {

Item<GroupedCases<List<String>>>\* p = casesByTypes[i]->first;

while (p != nullptr) {

printString(p->val->val, files->logFile);

files->logFile << "\n\n";

printCase(p->val->cases->first, files->logFile);

p = p->next;

}

}

if (listOfCases->isEmpty()) {

files->logFile << "В ходе отбрасывания список заключенных договоров остался пустым.\nДальнейшая работа программы не имеет смысла.\n";

files->outFile << "В ходе отбрасывания список заключенных договоров остался пустым.\nДальнейшая работа программы не имеет смысла.\n";

}

else {

files->commands.unsetf(std::ios::skipws);

files->logFile << "Производится обработка запросов.\n\n";

int mode;

char tmp;

do {

files->commands >> mode;

if (mode < 1 || mode > 5) {

files->logFile << "Считан неверный код запроса - " << mode << ". Запрос проигнорирован.\n\n";

files->outFile << "Ошибка: считан неверный код запроса - " << mode << " (допустимые коды - от 1 до 5).\nЗапрос проигнорирован.\n\n";

files->commands >> tmp;

while (tmp != '\n' && !files->commands.eof()) files->commands >> tmp;

}

else if (mode == 1 || mode == 3 || mode == 5) {

List<String>\* val = new List<String>;

Item<String>\* tmpItem = new Item<String>;

int charsCount = 0;

files->commands >> tmp; files->commands >> tmp;

do {

if (tmp == '\n' || files->commands.eof()) {

if (charsCount) {

val->appendItem(tmpItem);

tmpItem = nullptr;

}

else delete tmpItem;

break;

}

else {

if (charsCount == N) {

val->appendItem(tmpItem);

tmpItem = new Item<String>;

charsCount = 0;

}

tmpItem->val->text[charsCount] = tmp;

tmpItem->val->text[charsCount + 1] = '\0';

++charsCount;

}

files->commands >> tmp;

}

while (true);

char a = val->first->val->text[0];

if (mode == 1) {

files->logFile << "Запрос с кодом 1 - результаты работы агента:\n";

files->outFile << "Запрос с кодом 1 - результаты работы агента:\n";

if (isInRegistry(casesByAgents[a == 'Ё' ? 32 : a + 64], val)) {

int summary = 0;

files->logFile << "\n"; printString(val, files->logFile); files->logFile << "\n\nДоговоры:\n\n";

files->outFile << "\n"; printString(val, files->outFile); files->outFile << "\n\nДоговоры:\n\n";

printCase(casesByAgents[a == 'Ё' ? 32 : a + 64]->getItem(indexInRegistry(casesByAgents[a == 'Ё' ? 32 : a + 64], val))->val->cases->first, files->logFile);

printCase(casesByAgents[a == 'Ё' ? 32 : a + 64]->getItem(indexInRegistry(casesByAgents[a == 'Ё' ? 32 : a + 64], val))->val->cases->first, files->outFile);

Item<Case>\* t = casesByAgents[a == 'Ё' ? 32 : a + 64]->getItem(indexInRegistry(casesByAgents[a == 'Ё' ? 32 : a + 64], val))->val->cases->first;

while (t != nullptr) {

summary += \*(t->val->price);

t = t->next;

}

files->logFile << "Выручка агента: " << summary << "\n\n";

files->outFile << "Выручка агента: " << summary << "\n\n";

}

else {

files->logFile << "Агент "; printString(val, files->logFile); files->logFile << " не найден в картотеке. Запрос проигнорирован.\n\n";

files->outFile << "Агент "; printString(val, files->outFile); files->outFile << " не найден в картотеке. Запрос проигнорирован.\n\n";

}

}

else if (mode == 3) {

files->logFile << "Запрос с кодом 3 - страховые договоры клиента:\n";

files->outFile << "Запрос с кодом 3 - страховые договоры клиента:\n";

if (isInRegistry(casesByClients[a == 'Ё' ? 32 : a + 64], val)) {

files->logFile << "\n"; printString(val, files->logFile); files->logFile << "\n\n";

files->outFile << "\n"; printString(val, files->outFile); files->outFile << "\n\n";

printCase(casesByClients[a == 'Ё' ? 32 : a + 64]->getItem(indexInRegistry(casesByClients[a == 'Ё' ? 32 : a + 64], val))->val->cases->first, files->logFile);

printCase(casesByClients[a == 'Ё' ? 32 : a + 64]->getItem(indexInRegistry(casesByClients[a == 'Ё' ? 32 : a + 64], val))->val->cases->first, files->outFile);

}

else {

files->logFile << "Клиент "; printString(val, files->logFile); files->logFile << " не найден в картотеке. Запрос проигнорирован.\n\n";

files->outFile << "Клиент "; printString(val, files->outFile); files->outFile << " не найден в картотеке. Запрос проигнорирован.\n\n";

}

}

else {

files->logFile << "Запрос с кодом 5 - база договоров компании по типу:\n";

files->outFile << "Запрос с кодом 5 - база договоров компании по типу:\n";

if (isInRegistry(casesByTypes[a == 'Ё' ? 32 : a + 64], val)) {

files->logFile << "\n"; printString(val, files->logFile); files->logFile << "\n\n";

files->outFile << "\n"; printString(val, files->outFile); files->outFile << "\n\n";

printCase(casesByTypes[a == 'Ё' ? 32 : a + 64]->getItem(indexInRegistry(casesByTypes[a == 'Ё' ? 32 : a + 64], val))->val->cases->first, files->logFile);

printCase(casesByTypes[a == 'Ё' ? 32 : a + 64]->getItem(indexInRegistry(casesByTypes[a == 'Ё' ? 32 : a + 64], val))->val->cases->first, files->outFile);

}

else {

files->logFile << "Тип \""; printString(val, files->logFile); files->logFile << "\" не найден в картотеке. Запрос проигнорирован.\n\n";

files->outFile << "Тип \""; printString(val, files->outFile); files->outFile << "\" не найден в картотеке. Запрос проигнорирован.\n\n";

}

}

delete val;

}

else if (mode == 2) {

files->logFile << "Запрос с кодом 2 - результаты работы филиала:\n";

files->outFile << "Запрос с кодом 2 - результаты работы филиала:\n";

List<String>\* val1 = new List<String>;

Item<String>\* tmpItem = new Item<String>;

int charsCount = 0;

files->commands >> tmp; files->commands >> tmp;

do {

if (tmp == ';' || files->commands.eof()) {

if (charsCount) {

val1->appendItem(tmpItem);

tmpItem = nullptr;

}

else delete tmpItem;

break;

}

else {

if (charsCount == N) {

val1->appendItem(tmpItem);

tmpItem = new Item<String>;

charsCount = 0;

}

tmpItem->val->text[charsCount] = tmp;

tmpItem->val->text[charsCount + 1] = '\0';

++charsCount;

}

files->commands >> tmp;

}

while (true);

List<String>\* val2 = new List<String>;

tmpItem = new Item<String>;

charsCount = 0;

files->commands >> tmp; files->commands >> tmp;

do {

if (tmp == '\n' || files->commands.eof()) {

if (charsCount) {

val2->appendItem(tmpItem);

tmpItem = nullptr;

}

else delete tmpItem;

break;

}

else {

if (charsCount == N) {

val2->appendItem(tmpItem);

tmpItem = new Item<String>;

charsCount = 0;

}

tmpItem->val->text[charsCount] = tmp;

tmpItem->val->text[charsCount + 1] = '\0';

++charsCount;

}

files->commands >> tmp;

}

while (true);

char a = val1->first->val->text[0];

Branch\* val = new Branch(val1, val2);

if (isBranchInRegistry(casesByBranches[a == 'Ё' ? 32 : a + 64], val)) {

files->logFile << "\n\""; printString(val->name, files->logFile); files->logFile << "\", "; printString(val->place, files->logFile); files->logFile << "\n\n";

files->outFile << "\n\""; printString(val->name, files->outFile); files->outFile << "\", "; printString(val->place, files->outFile); files->outFile << "\n\n";

int summary = 0;

printCase(casesByBranches[a == 'Ё' ? 32 : a + 64]->getItem(indexBranchInRegistry(casesByBranches[a == 'Ё' ? 32 : a + 64], val))->val->cases->first, files->logFile);

printCase(casesByBranches[a == 'Ё' ? 32 : a + 64]->getItem(indexBranchInRegistry(casesByBranches[a == 'Ё' ? 32 : a + 64], val))->val->cases->first, files->outFile);

Item<Case>\* t = casesByBranches[a == 'Ё' ? 32 : a + 64]->getItem(indexBranchInRegistry(casesByBranches[a == 'Ё' ? 32 : a + 64], val))->val->cases->first;

while (t != nullptr) {

summary += \*(t->val->price);

t = t->next;

}

files->logFile << "Выручка филиала: " << summary << "\n\n";

files->outFile << "Выручка филиала: " << summary << "\n\n";

}

else {

files->logFile << "Филиал \""; printString(val->name, files->logFile); files->logFile << "\", "; printString(val->place, files->logFile); files->logFile << " не найден в картотеке. Запрос проигнорирован.\n\n";

files->outFile << "Филиал \""; printString(val->name, files->outFile); files->outFile << "\", "; printString(val->place, files->outFile); files->outFile << " не найден в картотеке. Запрос проигнорирован.\n\n";

}

delete val;

delete val1;

delete val2;

}

else if (mode == 4) {

files->logFile << "Запрос с кодом 4 - база договоров филиала по типу:\n";

files->outFile << "Запрос с кодом 4 - база договоров филиала по типу:\n";

List<String>\* val1 = new List<String>;

Item<String>\* tmpItem = new Item<String>;

int charsCount = 0;

files->commands >> tmp; files->commands >> tmp;

do {

if (tmp == ';' || files->commands.eof()) {

if (charsCount) {

val1->appendItem(tmpItem);

tmpItem = nullptr;

}

else delete tmpItem;

break;

}

else {

if (charsCount == N) {

val1->appendItem(tmpItem);

tmpItem = new Item<String>;

charsCount = 0;

}

tmpItem->val->text[charsCount] = tmp;

tmpItem->val->text[charsCount + 1] = '\0';

++charsCount;

}

files->commands >> tmp;

}

while (true);

List<String>\* val2 = new List<String>;

tmpItem = new Item<String>;

charsCount = 0;

files->commands >> tmp; files->commands >> tmp;

do {

if (tmp == ';' || files->commands.eof()) {

if (charsCount) {

val2->appendItem(tmpItem);

tmpItem = nullptr;

}

else delete tmpItem;

break;

}

else {

if (charsCount == N) {

val2->appendItem(tmpItem);

tmpItem = new Item<String>;

charsCount = 0;

}

tmpItem->val->text[charsCount] = tmp;

tmpItem->val->text[charsCount + 1] = '\0';

++charsCount;

}

files->commands >> tmp;

}

while (true);

List<String>\* val3 = new List<String>;

tmpItem = new Item<String>;

charsCount = 0;

files->commands >> tmp; files->commands >> tmp;

do {

if (tmp == '\n' || files->commands.eof()) {

if (charsCount) {

val3->appendItem(tmpItem);

tmpItem = nullptr;

}

else delete tmpItem;

break;

}

else {

if (charsCount == N) {

val3->appendItem(tmpItem);

tmpItem = new Item<String>;

charsCount = 0;

}

tmpItem->val->text[charsCount] = tmp;

tmpItem->val->text[charsCount + 1] = '\0';

++charsCount;

}

files->commands >> tmp;

}

while (true);

Branch\* branch = new Branch(val2, val3);

char a = val1->first->val->text[0], b = val2->first->val->text[0];

if (isInRegistry(casesByTypes[a == 'Ё' ? 32 : a + 64], val1) && isBranchInRegistry(casesByBranches[b == 'Ё' ? 32 : b + 64], branch)) {

files->logFile << "\nТип: "; printString(val1, files->logFile);

files->outFile << "\nТип: "; printString(val1, files->outFile);

files->logFile << "\nФилиал: \""; printString(branch->name, files->logFile); files->logFile << "\", "; printString(branch->place, files->logFile); files->logFile << "\n\n";

files->outFile << "\nФилиал: \""; printString(branch->name, files->outFile); files->outFile << "\", "; printString(branch->place, files->outFile); files->outFile << "\n\n";

if (!printCaseCheckBranch(casesByTypes[a == 'Ё' ? 32 : a + 64]->getItem(indexInRegistry(casesByTypes[a == 'Ё' ? 32 : a + 64], val1))->val->cases->first, files->logFile, branch)) {

files->logFile << "Записей, совпадающих по двум параметрам сразу, в картотеке не было обнаружено.\n\n";

}

if (!printCaseCheckBranch(casesByTypes[a == 'Ё' ? 32 : a + 64]->getItem(indexInRegistry(casesByTypes[a == 'Ё' ? 32 : a + 64], val1))->val->cases->first, files->outFile, branch)) {

files->outFile << "Записей, совпадающих по двум параметрам сразу, в картотеке не было обнаружено.\n\n";

}

}

else {

if (!isInRegistry(casesByTypes[a == 'Ё' ? 32 : a + 64], val1)) {

files->logFile << "Тип \""; printString(val1, files->logFile); files->logFile << "\" не найден в картотеке.\n";

files->outFile << "Тип \""; printString(val1, files->outFile); files->outFile << "\" не найден в картотеке.\n";

}

if (!isBranchInRegistry(casesByBranches[b == 'Ё' ? 32 : b + 64], branch)) {

files->logFile << "Филиал \""; printString(branch->name, files->logFile); files->logFile << "\", "; printString(branch->place, files->logFile); files->logFile << " не найден в картотеке.\n";

files->outFile << "Филиал \""; printString(branch->name, files->outFile); files->outFile << "\", "; printString(branch->place, files->outFile); files->outFile << " не найден в картотеке.\n";

}

files->logFile << "Запрос проигнорирован.\n\n";

files->outFile << "Запрос проигнорирован.\n\n";

}

delete branch;

delete val1;

delete val2;

delete val3;

}

}

while (!files->commands.eof());

files->logFile << "Командный файл достиг конца.\n\n";

files->outFile << "Командный файл достиг конца.";

}

files->logFile << "Блок проверок на корректные удаления картотек:\n\nПо агентам:\n\n";

for (int i = 0; i < 33; ++i) {

files->logFile << "До: " << casesByAgents[i]->first << " " << casesByAgents[i]->last;

delete casesByAgents[i];

files->logFile << "\nПосле: " << casesByAgents[i]->first << " " << casesByAgents[i]->last << "\n\n";

}

delete[] casesByAgents;

files->logFile << "По филиалам:\n\n";

for (int i = 0; i < 33; ++i) {

files->logFile << "До: " << casesByBranches[i]->first << " " << casesByBranches[i]->last;

delete casesByBranches[i];

files->logFile << "\nПосле: " << casesByBranches[i]->first << " " << casesByBranches[i]->last << "\n\n";

}

delete[] casesByBranches;

files->logFile << "По клиентам:\n\n";

for (int i = 0; i < 33; ++i) {

files->logFile << "До: " << casesByClients[i]->first << " " << casesByClients[i]->last;

delete casesByClients[i];

files->logFile << "\nПосле: " << casesByClients[i]->first << " " << casesByClients[i]->last << "\n\n";

}

delete[] casesByClients;

files->logFile << "По типам:\n\n";

for (int i = 0; i < 33; ++i) {

files->logFile << "До: " << casesByTypes[i]->first << " " << casesByTypes[i]->last;

delete casesByTypes[i];

files->logFile << "\nПосле: " << casesByTypes[i]->first << " " << casesByTypes[i]->last << "\n\n";

}

delete[] casesByTypes;

files->logFile << "Список договоров:\n\nДо: " << listOfCases->first << " " << listOfCases->last;

delete listOfCases;

files->logFile << "\nПосле: " << listOfCases->first << " " << listOfCases->last << "\n\nБлок завершен успешно.\n\n";

}

files->logFile << "Блок проверок на корректные удаления исходных списков:\n\nНазвания филиалов:\n\nДо: " << listOfNames->first << " " << listOfNames->last;

delete listOfNames;

files->logFile << "\nПосле: " << listOfNames->first << " " << listOfNames->last << "\n\nАдреса филиалов:\n\nДо: " << listOfPlaces->first << " " << listOfPlaces->last;

delete listOfPlaces;

files->logFile << "\nПосле: " << listOfPlaces->first << " " << listOfPlaces->last << "\n\nФИО агентов:\n\nДо: " << listOfAgents->first << " " << listOfAgents->last;

delete listOfAgents;

files->logFile << "\nПосле: " << listOfAgents->first << " " << listOfAgents->last << "\n\nФИО клиентов:\n\nДо: " << listOfClients->first << " " << listOfClients->last;

delete listOfClients;

files->logFile << "\nПосле: " << listOfClients->first << " " << listOfClients->last << "\n\nДаты страхования:\n\nДо: " << listOfDates->first << " " << listOfDates->last;

delete listOfDates;

files->logFile << "\nПосле: " << listOfDates->first << " " << listOfDates->last << "\n\nСроки страхования:\n\nДо: " << listOfTerms->first << " " << listOfTerms->last;

delete listOfTerms;

files->logFile << "\nПосле: " << listOfTerms->first << " " << listOfTerms->last << "\n\nТипы страхования:\n\nДо: " << listOfTypes->first << " " << listOfTypes->last;

delete listOfTypes;

files->logFile << "\nПосле: " << listOfTypes->first << " " << listOfTypes->last << "\n\nСтраховые суммы:\n\nДо: " << listOfPrices->first << " " << listOfPrices->last;

delete listOfPrices;

files->logFile << "\nПосле: " << listOfPrices->first << " " << listOfPrices->last << "\n\n";

files->logFile << "Блок завершен успешно.\n\n";

}

files->logFile << "Программа завершила свою работу.";

delete files;

}

**Результаты работы программы.**

Для контрольного примера результаты следующие:

|  |  |
| --- | --- |
| Файл протокола | Выходной файл |
| Программа начала свою работу.  Производится проверка входных файлов на существование и наличие содержимого.  Все входные файлы найдены и не пустые.  Производится запись исходнных данных в списки.  Заполняется список названий филиалов.  Росго->сстра->х-аль->фа->NULL  |  V  Росго->сстра->х-бет->а->NULL  |  V  NULL  Заполнен список названий филиалов.  Заполняется список адресов филиалов.  Белоо->стров->ская ->улица->, 17->NULL  |  V  Набер->ежная-> Черн->ой ре->чки, ->41->NULL  |  V  NULL  Заполнен список адресов филиалов.  Заполняется список ФИО агентов.  Григо->рьева-> Елен->а Сер->геевн->а->NULL  |  V  Корне->ев Ан->дрей ->Алекс->андро->вич->NULL  |  V  Сокол->ова А->наста->сия А->лекса->ндров->на->NULL  |  V  Макар->ов Да->ниил ->Алекс->андро->вич->NULL  |  V  Никол->аев А->лексе->й Сер->гееви->ч->NULL  |  V  Корол->ева А->наста->сия В->иктор->овна->NULL  |  V  Шеста->ков Д->митри->й Але->ксанд->рович->NULL  |  V  Васил->ьев И->ван И->ванов->ич->NULL  |  V  Петро->в Сер->гей Д->митри->евич->NULL  |  V  Гусев->а Ана->стаси->я Але->ксанд->ровна->NULL  |  V  NULL  Заполнен список ФИО агентов.  Заполняется список ФИО клиентов.  Баран->ов Ан->дрей ->Михай->лович->NULL  |  V  Андре->ева Е->катер->ина И->горев->на->NULL  |  V  Макар->ов Ал->ексан->др Ал->ексан->дрови->ч->NULL  |  V  Черно->ва Ек->атери->на Се->ргеев->на->NULL  |  V  Григо->рьев ->Денис-> Алек->сандр->ович->NULL  |  V  Петро->ва Да->рья С->ергее->вна->NULL  |  V  Белов->а Ана->стаси->я Сер->геевн->а->NULL  |  V  Сокол->ов Ал->ексей-> Алек->сандр->ович->NULL  |  V  Волко->в Анд->рей С->ергее->вич->NULL  |  V  Тихон->ов Ал->ексей-> Алек->сандр->ович->NULL  |  V  NULL  Заполнен список ФИО клиентов.  Заполняется список дат страхования.  30.08->.2022->NULL  |  V  17.04->.2021->NULL  |  V  27.02->.2022->NULL  |  V  11.07->.2022->NULL  |  V  06.06->.2021->NULL  |  V  03.01->.2023->NULL  |  V  20.05->.2022->NULL  |  V  24.12->.2022->NULL  |  V  15.08->.2021->NULL  |  V  07.11->.2021->NULL  |  V  NULL  Заполнен список дат страхования.  Заполняется список сроков страхования.  150 д->ней->NULL  |  V  40 дн->ей->NULL  |  V  17 ле->т->NULL  |  V  25 дн->ей->NULL  |  V  9 лет->NULL  |  V  15 дн->ей->NULL  |  V  270 д->ней->NULL  |  V  30 ле->т->NULL  |  V  10 ле->т->NULL  |  V  600 д->ней->NULL  |  V  35 ле->т->NULL  |  V  14 ле->т->NULL  |  V  6 лет->NULL  |  V  4 год->а->NULL  |  V  90 дн->ей->NULL  |  V  NULL  Заполнен список сроков страхования.  Заполняется список типов страхования.  КАСКО->NULL  |  V  ОСАГО->NULL  |  V  ДМС->NULL  |  V  НС->NULL  |  V  СМР->NULL  |  V  Страх->овани->е иму->ществ->а->NULL  |  V  Страх->овани->е от ->после->дстви->й COV->ID-19->NULL  |  V  NULL  Заполнен список типов страхования.  Заполняется список страховых сумм.  356182->  111347->  340832->  970513->  834828->  832404->  926120->  232581->  600413->  NULL  Заполнен список страховых сумм.  Заполняется список заключенных договоров.  3 12 3 0 1 3 4 4  Договор успешно записан.  2 6 2 0 1 7 4 5  Договор успешно записан.  1 12 7 0 1 5 3 5  Договор успешно записан.  5 9 5 1 0 3 7 0  Договор успешно записан.  4 3 6 1 0 2 6 1  Договор успешно записан.  6 11 8 0 1 3 2 2  Договор успешно записан.  0 8 1 1 0 1 9 1  Договор успешно записан.  7 13 8 0 1 1 1 1  Договор успешно записан.  1 4 0 1 8 0 2 0  Некорректный индекс для списка адресов - 8.  Договор не подлежит записи.  1 1 0 5 3 0 2 0  Некорректный индекс для списка названий - 5.  Некорректный индекс для списка адресов - 3.  Договор не подлежит записи.  Заполнен список заключенных договоров.  Договоры:  Дата заключения: 11.07.2022  Срок: 6 лет  Страховая сумма: 970513  Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41  ФИО клиента: Чернова Екатерина Сергеевна  ФИО агента: Николаев Алексей Сергеевич  Тип страхования: СМР  Дата заключения: 27.02.2022  Срок: 270 дней  Страховая сумма: 340832  Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41  ФИО клиента: Соколов Алексей Александрович  ФИО агента: Николаев Алексей Сергеевич  Тип страхования: Страхование имущества  Дата заключения: 17.04.2021  Срок: 6 лет  Страховая сумма: 232581  Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41  ФИО клиента: Петрова Дарья Сергеевна  ФИО агента: Макаров Даниил Александрович  Тип страхования: Страхование имущества  Дата заключения: 03.01.2023  Срок: 600 дней  Страховая сумма: 832404  Филиал: "Росгосстрах-бета", Белоостровская улица, 17  ФИО клиента: Чернова Екатерина Сергеевна  ФИО агента: Васильев Иван Иванович  Тип страхования: КАСКО  Дата заключения: 06.06.2021  Срок: 25 дней  Страховая сумма: 926120  Филиал: "Росгосстрах-бета", Белоостровская улица, 17  ФИО клиента: Макаров Александр Александрович  ФИО агента: Шестаков Дмитрий Александрович  Тип страхования: ОСАГО  Дата заключения: 20.05.2022  Срок: 14 лет  Страховая сумма: 600413  Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41  ФИО клиента: Чернова Екатерина Сергеевна  ФИО агента: Соколова Анастасия Александровна  Тип страхования: ДМС  Дата заключения: 30.08.2022  Срок: 10 лет  Страховая сумма: 111347  Филиал: "Росгосстрах-бета", Белоостровская улица, 17  ФИО клиента: Андреева Екатерина Игоревна  ФИО агента: Гусева Анастасия Александровна  Тип страхования: ОСАГО  Дата заключения: 24.12.2022  Срок: 4 года  Страховая сумма: 600413  Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41  ФИО клиента: Андреева Екатерина Игоревна  ФИО агента: Корнеев Андрей Александрович  Тип страхования: ОСАГО  По агентам:  Васильев Иван Иванович  Дата заключения: 03.01.2023  Срок: 600 дней  Страховая сумма: 832404  Филиал: "Росгосстрах-бета", Белоостровская улица, 17  ФИО клиента: Чернова Екатерина Сергеевна  ФИО агента: Васильев Иван Иванович  Тип страхования: КАСКО  Гусева Анастасия Александровна  Дата заключения: 30.08.2022  Срок: 10 лет  Страховая сумма: 111347  Филиал: "Росгосстрах-бета", Белоостровская улица, 17  ФИО клиента: Андреева Екатерина Игоревна  ФИО агента: Гусева Анастасия Александровна  Тип страхования: ОСАГО  Корнеев Андрей Александрович  Дата заключения: 24.12.2022  Срок: 4 года  Страховая сумма: 600413  Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41  ФИО клиента: Андреева Екатерина Игоревна  ФИО агента: Корнеев Андрей Александрович  Тип страхования: ОСАГО  Макаров Даниил Александрович  Дата заключения: 17.04.2021  Срок: 6 лет  Страховая сумма: 232581  Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41  ФИО клиента: Петрова Дарья Сергеевна  ФИО агента: Макаров Даниил Александрович  Тип страхования: Страхование имущества  Николаев Алексей Сергеевич  Дата заключения: 11.07.2022  Срок: 6 лет  Страховая сумма: 970513  Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41  ФИО клиента: Чернова Екатерина Сергеевна  ФИО агента: Николаев Алексей Сергеевич  Тип страхования: СМР  Дата заключения: 27.02.2022  Срок: 270 дней  Страховая сумма: 340832  Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41  ФИО клиента: Соколов Алексей Александрович  ФИО агента: Николаев Алексей Сергеевич  Тип страхования: Страхование имущества  Соколова Анастасия Александровна  Дата заключения: 20.05.2022  Срок: 14 лет  Страховая сумма: 600413  Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41  ФИО клиента: Чернова Екатерина Сергеевна  ФИО агента: Соколова Анастасия Александровна  Тип страхования: ДМС  Шестаков Дмитрий Александрович  Дата заключения: 06.06.2021  Срок: 25 дней  Страховая сумма: 926120  Филиал: "Росгосстрах-бета", Белоостровская улица, 17  ФИО клиента: Макаров Александр Александрович  ФИО агента: Шестаков Дмитрий Александрович  Тип страхования: ОСАГО  По филиалам:  "Росгосстрах-альфа", Набережная Черной речки, 41  Дата заключения: 11.07.2022  Срок: 6 лет  Страховая сумма: 970513  Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41  ФИО клиента: Чернова Екатерина Сергеевна  ФИО агента: Николаев Алексей Сергеевич  Тип страхования: СМР  Дата заключения: 27.02.2022  Срок: 270 дней  Страховая сумма: 340832  Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41  ФИО клиента: Соколов Алексей Александрович  ФИО агента: Николаев Алексей Сергеевич  Тип страхования: Страхование имущества  Дата заключения: 17.04.2021  Срок: 6 лет  Страховая сумма: 232581  Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41  ФИО клиента: Петрова Дарья Сергеевна  ФИО агента: Макаров Даниил Александрович  Тип страхования: Страхование имущества  Дата заключения: 20.05.2022  Срок: 14 лет  Страховая сумма: 600413  Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41  ФИО клиента: Чернова Екатерина Сергеевна  ФИО агента: Соколова Анастасия Александровна  Тип страхования: ДМС  Дата заключения: 24.12.2022  Срок: 4 года  Страховая сумма: 600413  Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41  ФИО клиента: Андреева Екатерина Игоревна  ФИО агента: Корнеев Андрей Александрович  Тип страхования: ОСАГО  "Росгосстрах-бета", Белоостровская улица, 17  Дата заключения: 03.01.2023  Срок: 600 дней  Страховая сумма: 832404  Филиал: "Росгосстрах-бета", Белоостровская улица, 17  ФИО клиента: Чернова Екатерина Сергеевна  ФИО агента: Васильев Иван Иванович  Тип страхования: КАСКО  Дата заключения: 06.06.2021  Срок: 25 дней  Страховая сумма: 926120  Филиал: "Росгосстрах-бета", Белоостровская улица, 17  ФИО клиента: Макаров Александр Александрович  ФИО агента: Шестаков Дмитрий Александрович  Тип страхования: ОСАГО  Дата заключения: 30.08.2022  Срок: 10 лет  Страховая сумма: 111347  Филиал: "Росгосстрах-бета", Белоостровская улица, 17  ФИО клиента: Андреева Екатерина Игоревна  ФИО агента: Гусева Анастасия Александровна  Тип страхования: ОСАГО  По клиентам:  Андреева Екатерина Игоревна  Дата заключения: 30.08.2022  Срок: 10 лет  Страховая сумма: 111347  Филиал: "Росгосстрах-бета", Белоостровская улица, 17  ФИО клиента: Андреева Екатерина Игоревна  ФИО агента: Гусева Анастасия Александровна  Тип страхования: ОСАГО  Дата заключения: 24.12.2022  Срок: 4 года  Страховая сумма: 600413  Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41  ФИО клиента: Андреева Екатерина Игоревна  ФИО агента: Корнеев Андрей Александрович  Тип страхования: ОСАГО  Макаров Александр Александрович  Дата заключения: 06.06.2021  Срок: 25 дней  Страховая сумма: 926120  Филиал: "Росгосстрах-бета", Белоостровская улица, 17  ФИО клиента: Макаров Александр Александрович  ФИО агента: Шестаков Дмитрий Александрович  Тип страхования: ОСАГО  Петрова Дарья Сергеевна  Дата заключения: 17.04.2021  Срок: 6 лет  Страховая сумма: 232581  Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41  ФИО клиента: Петрова Дарья Сергеевна  ФИО агента: Макаров Даниил Александрович  Тип страхования: Страхование имущества  Соколов Алексей Александрович  Дата заключения: 27.02.2022  Срок: 270 дней  Страховая сумма: 340832  Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41  ФИО клиента: Соколов Алексей Александрович  ФИО агента: Николаев Алексей Сергеевич  Тип страхования: Страхование имущества  Чернова Екатерина Сергеевна  Дата заключения: 11.07.2022  Срок: 6 лет  Страховая сумма: 970513  Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41  ФИО клиента: Чернова Екатерина Сергеевна  ФИО агента: Николаев Алексей Сергеевич  Тип страхования: СМР  Дата заключения: 03.01.2023  Срок: 600 дней  Страховая сумма: 832404  Филиал: "Росгосстрах-бета", Белоостровская улица, 17  ФИО клиента: Чернова Екатерина Сергеевна  ФИО агента: Васильев Иван Иванович  Тип страхования: КАСКО  Дата заключения: 20.05.2022  Срок: 14 лет  Страховая сумма: 600413  Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41  ФИО клиента: Чернова Екатерина Сергеевна  ФИО агента: Соколова Анастасия Александровна  Тип страхования: ДМС  По типам:  ДМС  Дата заключения: 20.05.2022  Срок: 14 лет  Страховая сумма: 600413  Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41  ФИО клиента: Чернова Екатерина Сергеевна  ФИО агента: Соколова Анастасия Александровна  Тип страхования: ДМС  КАСКО  Дата заключения: 03.01.2023  Срок: 600 дней  Страховая сумма: 832404  Филиал: "Росгосстрах-бета", Белоостровская улица, 17  ФИО клиента: Чернова Екатерина Сергеевна  ФИО агента: Васильев Иван Иванович  Тип страхования: КАСКО  ОСАГО  Дата заключения: 06.06.2021  Срок: 25 дней  Страховая сумма: 926120  Филиал: "Росгосстрах-бета", Белоостровская улица, 17  ФИО клиента: Макаров Александр Александрович  ФИО агента: Шестаков Дмитрий Александрович  Тип страхования: ОСАГО  Дата заключения: 30.08.2022  Срок: 10 лет  Страховая сумма: 111347  Филиал: "Росгосстрах-бета", Белоостровская улица, 17  ФИО клиента: Андреева Екатерина Игоревна  ФИО агента: Гусева Анастасия Александровна  Тип страхования: ОСАГО  Дата заключения: 24.12.2022  Срок: 4 года  Страховая сумма: 600413  Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41  ФИО клиента: Андреева Екатерина Игоревна  ФИО агента: Корнеев Андрей Александрович  Тип страхования: ОСАГО  СМР  Дата заключения: 11.07.2022  Срок: 6 лет  Страховая сумма: 970513  Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41  ФИО клиента: Чернова Екатерина Сергеевна  ФИО агента: Николаев Алексей Сергеевич  Тип страхования: СМР  Страхование имущества  Дата заключения: 27.02.2022  Срок: 270 дней  Страховая сумма: 340832  Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41  ФИО клиента: Соколов Алексей Александрович  ФИО агента: Николаев Алексей Сергеевич  Тип страхования: Страхование имущества  Дата заключения: 17.04.2021  Срок: 6 лет  Страховая сумма: 232581  Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41  ФИО клиента: Петрова Дарья Сергеевна  ФИО агента: Макаров Даниил Александрович  Тип страхования: Страхование имущества  Производится обработка запросов.  Запрос с кодом 1 - результаты работы агента:  Николаев Алексей Сергеевич  Договоры:  Дата заключения: 11.07.2022  Срок: 6 лет  Страховая сумма: 970513  Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41  ФИО клиента: Чернова Екатерина Сергеевна  ФИО агента: Николаев Алексей Сергеевич  Тип страхования: СМР  Дата заключения: 27.02.2022  Срок: 270 дней  Страховая сумма: 340832  Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41  ФИО клиента: Соколов Алексей Александрович  ФИО агента: Николаев Алексей Сергеевич  Тип страхования: Страхование имущества  Выручка агента: 1311345  Запрос с кодом 2 - результаты работы филиала:  "Росгосстрах-альфа", Набережная Черной речки, 41  Дата заключения: 11.07.2022  Срок: 6 лет  Страховая сумма: 970513  Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41  ФИО клиента: Чернова Екатерина Сергеевна  ФИО агента: Николаев Алексей Сергеевич  Тип страхования: СМР  Дата заключения: 27.02.2022  Срок: 270 дней  Страховая сумма: 340832  Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41  ФИО клиента: Соколов Алексей Александрович  ФИО агента: Николаев Алексей Сергеевич  Тип страхования: Страхование имущества  Дата заключения: 17.04.2021  Срок: 6 лет  Страховая сумма: 232581  Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41  ФИО клиента: Петрова Дарья Сергеевна  ФИО агента: Макаров Даниил Александрович  Тип страхования: Страхование имущества  Дата заключения: 20.05.2022  Срок: 14 лет  Страховая сумма: 600413  Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41  ФИО клиента: Чернова Екатерина Сергеевна  ФИО агента: Соколова Анастасия Александровна  Тип страхования: ДМС  Дата заключения: 24.12.2022  Срок: 4 года  Страховая сумма: 600413  Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41  ФИО клиента: Андреева Екатерина Игоревна  ФИО агента: Корнеев Андрей Александрович  Тип страхования: ОСАГО  Выручка филиала: 2744752  Запрос с кодом 3 - страховые договоры клиента:  Чернова Екатерина Сергеевна  Дата заключения: 11.07.2022  Срок: 6 лет  Страховая сумма: 970513  Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41  ФИО клиента: Чернова Екатерина Сергеевна  ФИО агента: Николаев Алексей Сергеевич  Тип страхования: СМР  Дата заключения: 03.01.2023  Срок: 600 дней  Страховая сумма: 832404  Филиал: "Росгосстрах-бета", Белоостровская улица, 17  ФИО клиента: Чернова Екатерина Сергеевна  ФИО агента: Васильев Иван Иванович  Тип страхования: КАСКО  Дата заключения: 20.05.2022  Срок: 14 лет  Страховая сумма: 600413  Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41  ФИО клиента: Чернова Екатерина Сергеевна  ФИО агента: Соколова Анастасия Александровна  Тип страхования: ДМС  Запрос с кодом 4 - база договоров филиала по типу:  Тип: ОСАГО  Филиал: "Росгосстрах-бета", Белоостровская улица, 17  Дата заключения: 06.06.2021  Срок: 25 дней  Страховая сумма: 926120  Филиал: "Росгосстрах-бета", Белоостровская улица, 17  ФИО клиента: Макаров Александр Александрович  ФИО агента: Шестаков Дмитрий Александрович  Тип страхования: ОСАГО  Дата заключения: 30.08.2022  Срок: 10 лет  Страховая сумма: 111347  Филиал: "Росгосстрах-бета", Белоостровская улица, 17  ФИО клиента: Андреева Екатерина Игоревна  ФИО агента: Гусева Анастасия Александровна  Тип страхования: ОСАГО  Запрос с кодом 5 - база договоров компании по типу:  ДМС  Дата заключения: 20.05.2022  Срок: 14 лет  Страховая сумма: 600413  Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41  ФИО клиента: Чернова Екатерина Сергеевна  ФИО агента: Соколова Анастасия Александровна  Тип страхования: ДМС  Командный файл достиг конца. | Запрос с кодом 1 - результаты работы агента:  Николаев Алексей Сергеевич  Договоры:  Дата заключения: 11.07.2022  Срок: 6 лет  Страховая сумма: 970513  Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41  ФИО клиента: Чернова Екатерина Сергеевна  ФИО агента: Николаев Алексей Сергеевич  Тип страхования: СМР  Дата заключения: 27.02.2022  Срок: 270 дней  Страховая сумма: 340832  Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41  ФИО клиента: Соколов Алексей Александрович  ФИО агента: Николаев Алексей Сергеевич  Тип страхования: Страхование имущества  Выручка агента: 1311345  Запрос с кодом 2 - результаты работы филиала:  "Росгосстрах-альфа", Набережная Черной речки, 41  Дата заключения: 11.07.2022  Срок: 6 лет  Страховая сумма: 970513  Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41  ФИО клиента: Чернова Екатерина Сергеевна  ФИО агента: Николаев Алексей Сергеевич  Тип страхования: СМР  Дата заключения: 27.02.2022  Срок: 270 дней  Страховая сумма: 340832  Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41  ФИО клиента: Соколов Алексей Александрович  ФИО агента: Николаев Алексей Сергеевич  Тип страхования: Страхование имущества  Дата заключения: 17.04.2021  Срок: 6 лет  Страховая сумма: 232581  Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41  ФИО клиента: Петрова Дарья Сергеевна  ФИО агента: Макаров Даниил Александрович  Тип страхования: Страхование имущества  Дата заключения: 20.05.2022  Срок: 14 лет  Страховая сумма: 600413  Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41  ФИО клиента: Чернова Екатерина Сергеевна  ФИО агента: Соколова Анастасия Александровна  Тип страхования: ДМС  Дата заключения: 24.12.2022  Срок: 4 года  Страховая сумма: 600413  Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41  ФИО клиента: Андреева Екатерина Игоревна  ФИО агента: Корнеев Андрей Александрович  Тип страхования: ОСАГО  Выручка филиала: 2744752  Запрос с кодом 3 - страховые договоры клиента:  Чернова Екатерина Сергеевна  Дата заключения: 11.07.2022  Срок: 6 лет  Страховая сумма: 970513  Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41  ФИО клиента: Чернова Екатерина Сергеевна  ФИО агента: Николаев Алексей Сергеевич  Тип страхования: СМР  Дата заключения: 03.01.2023  Срок: 600 дней  Страховая сумма: 832404  Филиал: "Росгосстрах-бета", Белоостровская улица, 17  ФИО клиента: Чернова Екатерина Сергеевна  ФИО агента: Васильев Иван Иванович  Тип страхования: КАСКО  Дата заключения: 20.05.2022  Срок: 14 лет  Страховая сумма: 600413  Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41  ФИО клиента: Чернова Екатерина Сергеевна  ФИО агента: Соколова Анастасия Александровна  Тип страхования: ДМС  Запрос с кодом 4 - база договоров филиала по типу:  Тип: ОСАГО  Филиал: "Росгосстрах-бета", Белоостровская улица, 17  Дата заключения: 06.06.2021  Срок: 25 дней  Страховая сумма: 926120  Филиал: "Росгосстрах-бета", Белоостровская улица, 17  ФИО клиента: Макаров Александр Александрович  ФИО агента: Шестаков Дмитрий Александрович  Тип страхования: ОСАГО  Дата заключения: 30.08.2022  Срок: 10 лет  Страховая сумма: 111347  Филиал: "Росгосстрах-бета", Белоостровская улица, 17  ФИО клиента: Андреева Екатерина Игоревна  ФИО агента: Гусева Анастасия Александровна  Тип страхования: ОСАГО  Запрос с кодом 5 - база договоров компании по типу:  ДМС  Дата заключения: 20.05.2022  Срок: 14 лет  Страховая сумма: 600413  Филиал: "Росгосстрах-альфа", Набережная Черной речки, 41  ФИО клиента: Чернова Екатерина Сергеевна  ФИО агента: Соколова Анастасия Александровна  Тип страхования: ДМС  Командный файл достиг конца.  **ИЗ ЛОГ-ФАЙЛА:**  Блок проверок на корректные удаления картотек:  По агентам:  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000026A748D1840 0000026A748D1840  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000026A748D3A80 0000026A748D3A80  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000026A748D3DA0 0000026A748D3DA0  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000026A748D1980 0000026A748D1980  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000026A748CF4A0 0000026A748CF4A0  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000026A748D1D90 0000026A748D1D90  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000026A748D0DA0 0000026A748D0DA0  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  По филиалам:  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000026A748D0490 0000026A748D1B60  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  По клиентам:  До: 0000026A748D4C50 0000026A748D4C50  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000026A748D1200 0000026A748D1200  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000026A748D13E0 0000026A748D13E0  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000026A748D0A30 0000026A748D0A30  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000026A748D0080 0000026A748D0080  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  По типам:  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000026A748D44D0 0000026A748D44D0  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000026A748D0AD0 0000026A748D0AD0  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000026A748D1D40 0000026A748D1D40  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000026A748CF680 0000026A748D0FD0  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  До: 0000000000000000 0000000000000000  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  Список договоров:  До: 0000026A748CFDB0 0000026A748D3EE0  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  Блок завершен успешно.  Блок проверок на корректные удаления исходных списков:  Названия филиалов:  До: 0000026A748B0A80 0000026A748BE910  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  Адреса филиалов:  До: 0000026A748BE7D0 0000026A748BEFA0  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  ФИО агентов:  До: 0000026A748BF470 0000026A748C2270  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  ФИО клиентов:  До: 0000026A748C3C30 0000026A748C7060  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  Даты страхования:  До: 0000026A748C7590 0000026A748C8990  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  Сроки страхования:  До: 0000026A748C8DF0 0000026A748CACC0  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  Типы страхования:  До: 0000026A748CA950 0000026A748CCE60  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  Страховые суммы:  До: 0000026A748CC190 0000026A748CC1E0  После: DDDDDDDDDDDDDDDD DDDDDDDDDDDDDDDD  Блок завершен успешно.  Программа завершила свою работу. |

**Вывод о проделанной работе.**

Изучен на практике мультисписок как структура, получен опыт в проектировании сложной информационной системы, реализующей эффективную выдачу сведений.