

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

**«Дальневосточный федеральный университет»**

(ДВФУ)

**ШКОЛА ЕСТЕСТВЕННЫХ НАУК**

**Кафедра прикладной математики, механики, управления и программного обеспечения**

САЗОНТОВА МАРИЯ ДМИТРИЕВНА

РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ ДЛЯ

АВТОМАТИЗАЦИИ РАБОТЫ СО СПРАВОЧНИКАМИ ОРГАНИЗАЦИЙ ДЛЯ ПРОЕКТА «MYMAP»

# КУРСОВОЙ ПРОЕКТ

по дисциплине «Фундаментальные структуры данных и алгоритмы» по образовательной программе подготовки бакалавров по направлению

09.03.04 - Программная инженерия

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Студент гр. Б8119-09.03.04прогин | | | | | | |
|  |  |  |  |  |  | Сазонтова М.Д. |
|  |  |  |  | (подпись) |  |  |
| Защищен с оценкой | |  |  | Руководитель |  |  |
| Ученая степень  ст. преподаватель | | | | | | |
|  |  |  |  | должность |  | О.А. Крестникова |
|  | (подпись) | (И.О. Фамилия) |  | (подпись) |  | (И.О. Фамилия) |
| « | » | 2021 г. |  |  |  |  |

г. Владивосток

2021

Оглавление

[Введение 3](#_Toc75865877)

[1 Анализ предметной области (ПО) 4](#_Toc75865878)

[1.1 Модель ПО 4](#_Toc75865879)

[Объекты предметной области: 4](#_Toc75865880)

[1.2 Постановки задач обработки 6](#_Toc75865881)

[2.Теоретическая часть 7](#_Toc75865882)

[2.1 Хеш-таблица 7](#_Toc75865883)

[2.1.1 Хеш-функция 7](#_Toc75865884)

[2.1.2 Разрешение коллизий методом цепочек 9](#_Toc75865885)

[2.1.3 Упорядоченный по убыванию связный список 9](#_Toc75865886)

[2.2 Бинарное дерево поиска 10](#_Toc75865887)

[3 Требования к информационной системе 13](#_Toc75865888)

[3.1 Функциональные требования 13](#_Toc75865889)

[3.2 Требования к данным 14](#_Toc75865890)

[3.2.1 Требования к входным данным 14](#_Toc75865891)

[3.2.2 Требования к выходным данным 15](#_Toc75865892)

[3.3 Требования к интерфейсу 15](#_Toc75865893)

[4 Реализация 16](#_Toc75865894)

[4.1 Диаграмма классов 16](#_Toc75865895)

[4.2 Описание классов 17](#_Toc75865896)

[4.3 Описание интерфейса 22](#_Toc75865897)

[4.4 Тестирование 28](#_Toc75865898)

[Заключение 30](#_Toc75865899)

[Список литературы 31](#_Toc75865900)

# Введение

Зачастую сложно самому уследить за всеми акциями, которые предоставляют различные организации. Приложение «MyMap» поможет решить эту проблему путем систематизации данных об организациях и проводимых там акциях в удобном для пользователя формате таблицы с возможностью ее редактирования.

Целью курсового проекта является: разработка информационной системы для автоматизации работы со справочниками, содержащими основную информацию об организациях и проводимых ими акциях.

Для достижения поставленной цели необходимо выполнить следующие задачи:

1. Провести анализ предметной области («MyMap») и построить ее модель.
2. Изучить теоретические основы методов построения справочников.
3. Определить требования к информационной системе.
4. Реализовать и провести тестирование.

# Анализ предметной области (ПО)

Требуется разработать информационную систему для автоматизации работы со справочниками организаций, содержащими основную информаций об организациях и проводимых ими акциях.

Система должна решать следующие задачи:

1) хранить информацию об организациях и акциях, проводимых ими;

2) позволять просматривать всю информацию об организациях и акциях, проводимых ими;

3) позволять добавлять информацию об организациях и акциях, проводимых ими;

4) позволять удалять информацию об организациях и акциях, проводимых ими;

5) позволять искать информацию об организациях и акциях;

6) предусмотреть проверку целостности информации, представленной в справочниках об организациях и акциях, проводимых ими, при добавлении, удалении и поиске элемента.

# Модель ПО

Предметная область – справочник организаций для проекта «MYMAP».

Профессионал предметной области – пользователи и представители организаций.

# Объекты предметной области:

Объект **Справочник проводимых организациями акций** – информация о нем представляется в справочнике, который содержит информацию по каждой организации и проводимых акциях.

**Название организации**– строка, содержащая строчные и прописные символы русского и латинского алфавитов, цифры и знаки препинания.

**Адрес** – строка, содержащая строчные и прописные символы русского и латинского алфавитов, цифры и знаки препинания.

**Название акции** – строка, содержащая строчные и прописные символы русского и латинского алфавитов, цифры и знаки препинания.

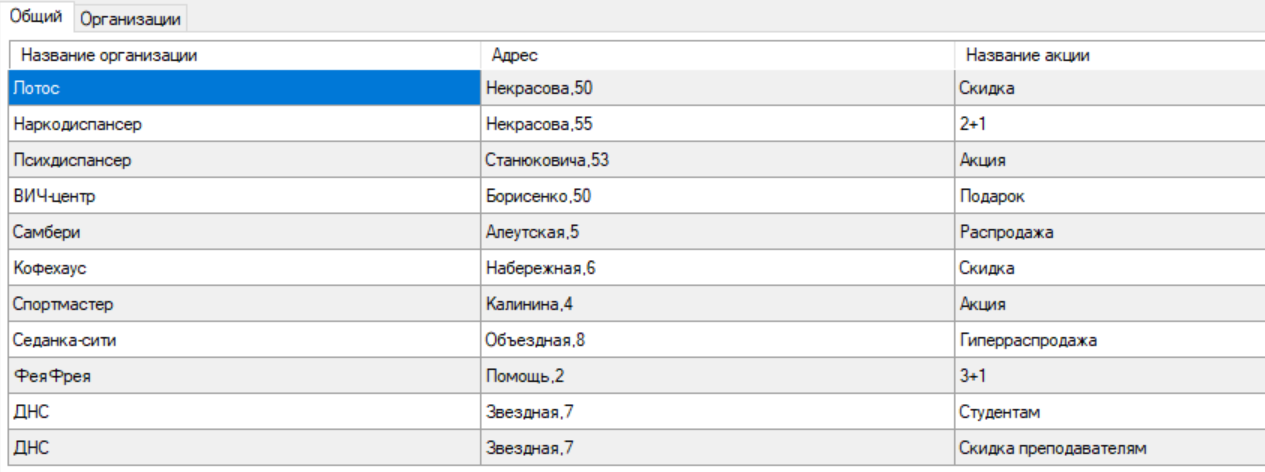
На Рисунке 1.1 представлен пример справочника, который содержит информацию по каждой организации.

Рисунок 1.1. Справочник «Общий»

Объект **Справочник Организаций** – информация о нем представляется в справочнике, который содержит информацию по каждой организации.

**Название организации**– строка, содержащая строчные и прописные символы русского и латинского алфавитов, цифры и знаки препинания.

**Адрес** – строка, содержащая строчные и прописные символы русского и латинского алфавитов, цифры и знаки препинания.

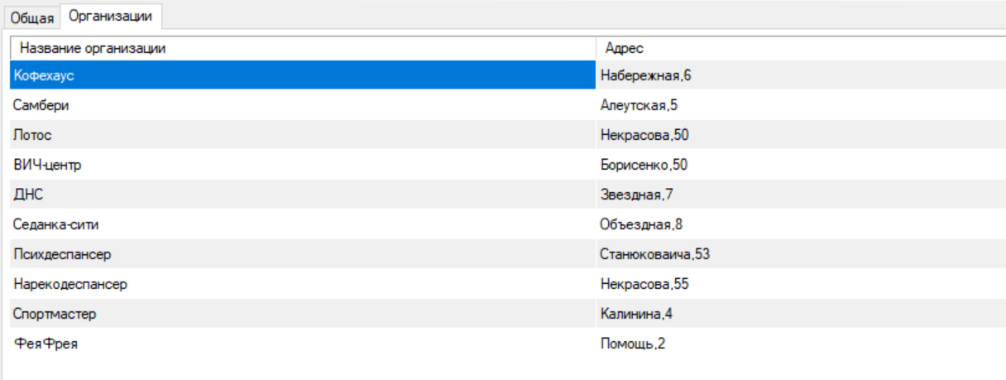
 На Рисунке 2 представлен пример справочника, который содержит информацию об организациях.

Рисунок 2. Справочник «Организации»

# Постановки задач обработки

Поиск в справочнике «Организации».

Входные данные: список организаций «Организации», название организации.

Выходные данные: организация с данным названием.

Связь: если «название организации» = «Организации».«название организации», то вывести «название организации» + данные о ней из справочника.

Поиск в справочнике «Общий» .

Входные данные: список организаций «Общий», название организации.

Выходные данные: организация с данным названием.

Связь: если «название организации» = «Общий».«название организации», то вывести «название организации»+ данные о ней из справочника.

Проверка целостности при добавлении в «Общий».

Входные данные: название, адрес, акция, список «Организации», список «Общий».

Выходные данные: список акций и организаций.

Связь :При добавлении «название»+ «адрес»+ «акция» в «Общий», если «название»+ «адрес» нет в «Организации», добавить «название»+«адрес» также в «Организации».

Проверка целостности при удалении из справочника «Организации».

Входные данные: список «Организации», список «Общий», название.

Выходные данные: список «Организаций», список «Общий»

Связь: Если «название»=«Общий».«название», то удалить «Организации».«название» и «Общий».«название».

# 2.Теоретическая часть

В рамках курсового проекта должны быть реализованы: хранение информации об организациях, их адресах, а также акциях, проводимых ими - а именно необходимо динамическое множество, поддерживающее операции добавления, поиска и удаления элемента. В таком случае удобно использовать хеш-таблицу и бинарное дерево поиска.

# Хеш-таблица

Хеш-таблица [1] – это структура данных, позволяющая хранить пары, состоящие из ключа и значения, и выполнять следующие операции: операцию добавления новой пары, операцию поиска и операцию удаления пары по ключу. Хеширование [1] – это определение ячейки для каждой пары ключ-значение с помощью заданной хеш-функции, принимающей ключ пары на вход и преобразовывающей его в номер ячейки таблицы.

В худшем случае поиск в хеш-таблице может занимать столько же времени, сколько поиск в списке (ϴ(n)), но на практике хеширование весьма эффективно. При выполнении некоторых естественных условий математическое ожидание времени поиска элемента в хеш-таблице есть О(1).

# Хеш-функция

При наличии хеш-таблицы, которая может содержать A элементов, требуется функция, которая преобразует множество ключей L в целые числа в диапазоне [0, A-1]. Пусть элемент k должен храниться в ячейке с номером b, b ϵ [0, A-1]. Тогда хеш-функция h будет отображать множество ключей L на множество ячеек в таблице, то есть

h(k): L → [0, 1, …, A-1]

В данном случае элемент k хешируется в ячейку h(k), где h – хеш-функция, а величина h(k) – хеш-значение ключа k. На практике обычно размер хеш-таблицы значительно меньше количества ключей [1].

Рассмотрим хеш-функцию, основанную на методе деления. Хеш-функция в данном случае будет отображать ключ k на одну из m ячеек хеш-таблицы путём получения остатка от деления k на m, то есть

h(k) = k mod m

В рамках курсового проекта ключом является название организации, поэтому на вход функции h(k) будут поступать только названия организаций. В свою очередь, названия организаций будем представлять в виде суммы кодов символов в таблицы ASCII.

Предположим, что размер хеш-таблицы m = 10 и нам на вход поступают названия организаций из модели предметной области (см п. 1.1). Приведем несколько примеров работы хеш-функции:

h(Лотос) = (1051+1086+1090+1086+1089) mod 10 = 2

h(Наркодиспансер)=(1053+1072+1088+1082+1086+1076+1080+1089+1087+ 1072+ 1085+1089+ 1077+1088)mod10 = 4

h(Психдиспансер)=(1055+1089+1080+1093+1076+1080+1089+1087+1072+1085+1089+1077+ 1088)mod10 = 0

h(ВИЧ-центр)=( 1042 + 1048 + 1063 + 45 + 1094 + 1077 + 1085 + 1090 + 1088)mod10= 2 h(Кофехаус)=(1050+1086+1092+1077+1093+1072+1091+1089)mod10=(8650)mod10=0

h(ФеяФрея)= 8

h(Спортмастер)=8

h(Седанка-сити)=5

h(ДНС)=4

h(Самбери)=1

При использовании такой хеш-функции можно заметить, что два и более ключей были хешированы в одну и ту же ячейку. Такая ситуация называется коллизией. Существует несколько способов разрешения коллизий, например метод цепочек или метод открытой адресации. Остановимся на методе цепочек.

# Разрешение коллизий методом цепочек

При разрешении коллизии с помощью цепочек помещаются все элементы, хешированные в одну и ту же ячейку, в связный список. Связный список – это динамическая структура данных, состоящая из узлов, каждый из которых содержит как собственно данные, так и ссылки на следующий и/или предыдущий узел списка [1]. Ячейка содержит указатель на головной элемент списка всех элементов. Если таких элементов нет, ячейка содержит значение NULL. В рамках курсового проекта рассмотрим цепочку, представляющую собой двусвязный упорядоченный по убыванию список

# Упорядоченный по убыванию связный список

Каждый узел двусвязного списка, помимо собственных данных, содержит два поля указателей — на следующий и на предыдущий узлы. Указатель на предыдущий узел головного элемента списка содержит нулевое значение. Указатель на следующий узел хвостового элемента также содержит нулевое значение [1]. Упорядоченный по убыванию список строится путем сравнения ключей по таблице кодов символов ASCII. На рисунке представлено использование двусвязного списка как способ разрешения коллизий в хеш-таблице. На Рисунке 3 представлен пример хеш-таблицы с цепочкой в ячейке 3.

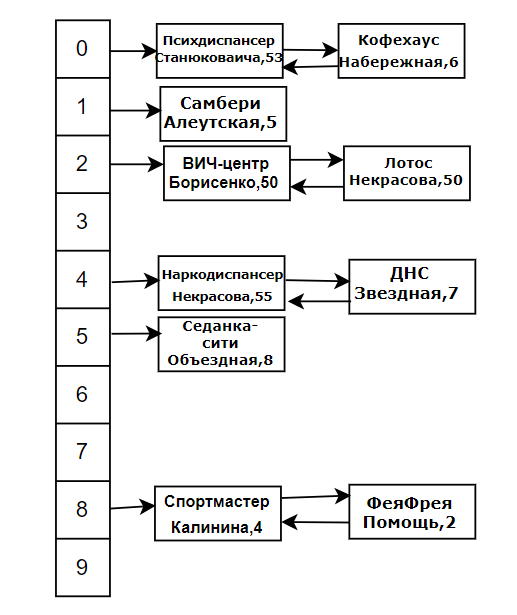


Рисунок 3. Метод цепочек

# Бинарное дерево поиска

Бинарное дерево поиска – структура данных, в которой каждый узел содержит поле данных, а также указатель на левый и правый дочерние узлы (потомки). Ключ левого потомка меньше или равен ключу родителя, ключ правого потомка больше или равен ключу родителя[1].

Бинарное дерево поиска поддерживает следующие операции: поиск узла, в котором хранится пара ключ-значение, добавление в дерево пары ключ- значение и удаление узла, в котором хранится пара ключ-значение.

Поиск узла

Пусть имеется дерево Treeи задан ключ парой (КeyAdress, KeyName). Необходимо проверить, если ли узел с этим ключом в дереве Tree, и если да, то вернуть ссылку на этот узел. Поиск выполняется рекурсивно. Если дерево пуcто, то поиск останавливается, ключ не найден. Иначе, ключ сравнивает со значением ключа корневого узла X:

1. если (КeyAdress, KeyName)= *X*, вернуть ссылку на корневой узел. Поиск останавливается;
2. если пусто, то рекурсивно искать ключ (КeyAdress, KeyName) в левом поддереве Tree;
3. если пусто, то рекурсивно искать ключ (КeyAdress, KeyName) в правом поддереве Tree.

Добавление элемента

Пусть имеется дерево Treeи пара (K, V),

1. если Tree пустое, то создать новое дерево с родительским узлом (K,V);
2. если узел (K,V) меньше родительского, то происходит вставка в левое поддерево;
3. если узел (K,V) больше или равен родительскому, то происходит вставка в правое поддерево;

Реализация подразумевает, что при добавлении повторных ключей в дерево, новый ключ идет вправо.

Удаление узла

Пусть имеется дерево Treeс корневым узлом Hи задана пара (*K*, *V*). Необходимо удалить из дерева Treeпару (*K, V*). Алгоритм удаления элемента:

1. Ищем элемент в дереве, если его нет – возвращаем пусто
2. Если узел равен Head – удаляем его
3. Если у узла нет потомков, удаляем его, а на его место встает единственный потомок
4. Если у элемента есть оба потомка, то находим максимальный слева элемент и ставим его на место удаленного элемента

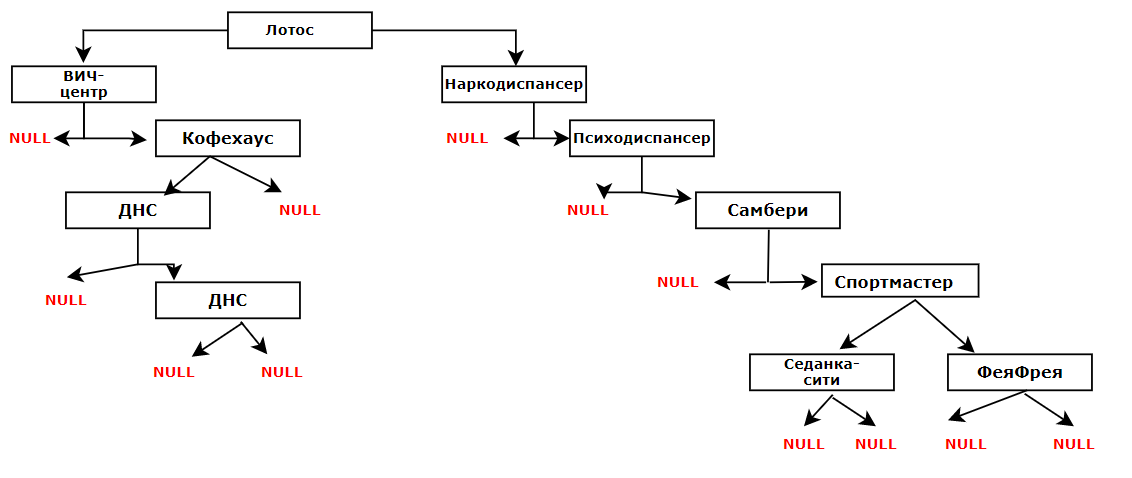
 Ниже представлен пример бинарного дерева поиска:

Рисунок 4. Пример бинарного дерева поиска

# Требования к информационной системе

# Функциональные требования

Информационная система для автоматизации работы со справочниками организаций и акций, должна позволять:

1. Хранить информацию об организациях;
2. Просматривать всю информацию об организации;

проверки: если справочник пустой, то ничего не должно выводится.

1. Добавлять информацию об организации;

проверки: при добавлении повторяющегося ключа, справочник не должен изменяться; при добавлении в справочник организаций и акций организации, которой нет в справочнике организаций, она добавляется и в справочник организаций; также справочник организаций и акций могут быть добавлены дубли организаций, не может быть добавлена организация с разными адресами.

1. Удалять информацию об организации и акции при выборе;

проверки: справочник организаций и акций не должен изменяться при попытке удалить организацию, которого нет в справочнике; при удалении из справочника организаций и акций организации, которая есть в справочнике организаций, она не должна удалятся из справочника организаций; при удалении организации из справочника организаций, она должна удалиться и из справочника организаций и акций;

1. Искать информацию о мероприятии;

проверки: если организация была найдено, то должна выводиться таблица, содержащая только эти организации; если не был найден, то должна выводится пустая таблица.

# Требования к данным

# Требования к входным данным

Основываясь на анализе ПО, входными данными для работы со справочниками является:

1. текстовый файл формата .tbl с произвольным названием, каждая строка файла содержит информацию об объектах одного из справочников, либо является строкой разделителем справочников “r\”. Первым записывается справочник организаций и акций: Название Адрес Акция (через пробел), далее идет разделительная строка (программа записывает ее двумя невидимыми знаками, когда при ручном вводе данных ее записываем одним невидимым знаком через Enter), далее справочник организаций: Название Адрес (через пробел).

Пример текстового файла:

Лотос Некрасова,50 Скидка

Нарекодеспансер Некрасова,55 2+1

Психдеспансер Станюковаича,53 Акция

ВИЧ-центр Борисенко,50 Подарок

Самбери Алеутская,5 Распродажа

Кофехаус Набережная,6 Скидка

Спортмастер Калинина,4 Акция

Седанка-сити Объездная,8 Гиперраспродажа

ФеяФрея Помощь,2 3+1

ДНС Звездная,7 Студентам

ДНС Звездная,7 Преподавателям

Кофехаус Набережная,6

Самбери Алеутская,5

Лотос Некрасова,50

ВИЧ-центр Борисенко,50

ДНС Звездная,7

Седанка-сити Объездная,8

Психдеспансер Станюковаича,53

Нарекодеспансер Некрасова,55

Спортмастер Калинина,4

ФеяФрея Помощь,2

1. Название организации, адрес организации и название акции (см. п. 1.1)

# 3.2.2 Требования к выходным данным

1. Текстовый файл формата .tbl с произвольным названием вида, описанного в пункте 3.2.1
2. Сообщения для пользователя:

«Попытка добавить уже имеющуюся или пустую организацию!»

«[BTree] Добавлен новый элемент в дерево»

«[BTree] Был удален элемент из дерева»

«[Хэш:Организация-Адрес] Добавлен новый элемент:»

«Элемент найден, адрес»

«Кол-во сравнений: »

«Ошибочка, Уважаемый:)»

«Элемент уже находится в справочнике»

«Нет таких элементов»

«Не все строки заполнены»

«Что то тут не чисто…»

# Требования к интерфейсу

Интерфейс должен быть оконным, все элементы и выводимые сообщения должны быть на русском языке.

# Реализация

# 4.1 Диаграмма классов

Основываясь на анализе ПО и на функциональных требованиях к информационной системе, определены типы классов и связи между ними, которые представлены в виде UML-диаграммы классов на Рисунке 5.

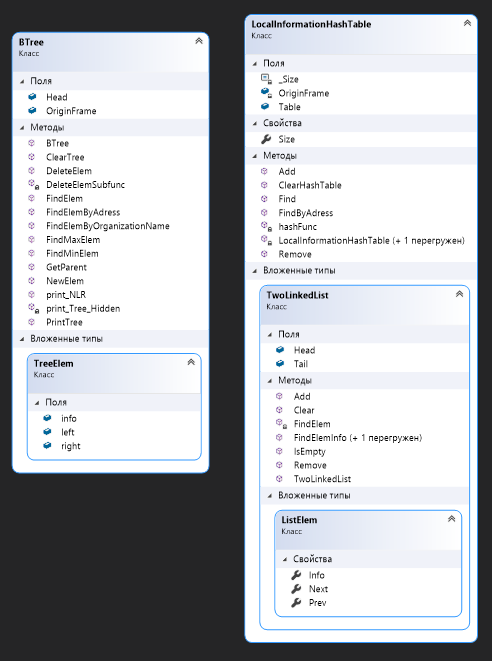
/ 

Рисунок 5 UML-диаграмма классов

# 

# Описание классов

**Класс LocalInformationHashTable**–класс, описывающий хэш-таблицу с модульной хеш-функцией и методом цепочек по убыванию.

Поля:

* \_Size - размер хеш-таблицы
* OriginFrame – форма, создавшая экземпляр
* Table - таблица

Свойства:

Size - размер

Методы:

* LocalInformationHashTable – конструктор.
* hashFunc(info) - Хэш-функция. Формальные параметры: info – ключ (название организации). Входные данные: ключ. Выходные данные: ключ.

Предположим, что размер массива 10 и на вход поступает запись, содержащая название организации из анализа предметной области (см. п. 1).

hashFunc (Лотос) = 2

hashFunc (Наркодеспансер) = 4

hashFunc (Психдеспансер) = 0

hashFunc (ВИЧ-центр = 2

hashFunc (ФеяФрея)= 8

hashFunc (Спортмастер) =8

hashFunc (Седанка-сити) =5

hashFunc (ДНС) =4

hashFunc (Самбери) =1

hashFunc (Кофехаус) =0

* Add(info) – функция добавления информации в хэш-таблицу. Формальные параметры: информация для добавления. Входные данные: информация для добавления, хэш-таблица. Выходные данные: true/false в случае удачного/неудачного добавления соответственно.
* FindByAdress(Adress) –функция поиск в кеш-таблице по адресу. Формальные параметры: ключ – адрес организации. Входные данные: хеш-таблица, ключ. Выходные данные: информацию об узле.
* Find(info) - функция поиска элемента в хэш-таблице. Формальные параметры: информация равная информации искомого элемента. Входные данные: информация равная информации искомого элемента, хэш-таблица. Выходные данные: искомый элемент либо null;
* Remove(info) - функция удаления информации из хэш-таблицы. Формальные параметры: информация равная информации удаляемого элемента. Входные данные: информация равная информации удаляемого элемента, хэш-таблица. Выходные данные: true/false в случае удачного/неудачного удаления соответственно.
* ClearHashTable() - очистка хеш-таблицы. Формальные параметры: нет.Входные данные: хеш-таблица. Выходные данные: пустая хеш-таблица.

Вложенные типы:

**Класс TwoLinkedList –** класс, описывающий узел хеш-таблицы методом цепочек.

Поля:

* Head – первый узел.
* Tail – последний узел

Методы:

* Clear() – очистка списка. Формальные параметры: нет. Входные данные: нет. Выходные данные: нет.
* IsEmpty() – проверка на пустоту. Формальные параметры: iнет. Входные данные: нет. Выходные данные: нет.
* Add(info) - добавление элемента по убыванию. Формальные параметры: info –элемент. Входные данные: элемент, список. Выходные данные: нет.
* Remove(info) – удалить элемент из списка. Формальные параметры: info –элемент. Входные данные: элемент, список. Выходные данные: элемент.
* FindElemInfo(OrganizationName) - поиск по названию организации. Формальные параметры: OrganizationName –название организации (одно из полей элемента). Входные данные: элемент список. Выходные данные: нет.
* FindElemInfo(info) – поиск по всей информации. Формальные параметры: info –элемент. Входные данные: элемент, список. Выходные данные: элемент.
* FindElem(info) – поиск элемента в списке. Формальные параметры: info – элемент. Входные данные: ключ, список. Выходные данные: элемент.

Вложенные типы:

* **Класс ListElem –** класс элемента в списке.

Свойства:

* Prev – указатель на предыдущий узел.
* Info – информация об узле.
* Next – указатель на следующий узел.

**Класс BTree** – класс, описывающий бинарное дерево поиска.

Поля:

* Head – корень дерева.
* OriginFrame – окно, из которого создаётся дерево.

Методы**:**

* NewElem(info)– добавление нового элемента в дерево. Формальные параметры: info –данные нового элемента. Входные данные: корень дерева, данные нового элемента. Выходные данные: true/false в случае удачи/неудачи соответственно.
* ClearTree() – очистка дерева. Формальные параметры: нет. Входные данные: дерево. Выходные данные: нет.
* FindElem(elem, info) – поиск элемента в дереве. Формальные параметры: elem – корень дерева, info – искомые данные. Входные данные: корень дерева, исходные данные, дерево. Выходные данные: найденный элемент либо null.
* FindElemByAdress (elem, Adress) – поиск элемента в дереве по адресу. Формальные параметры: elem – корень дерева, Adress – адрес искомого элемента. Входные данные: корень дерева, логин искомого элемента, дерево. Выходные данные: найденный элемент либо null.
* FindElemByOrganizationName(elem,OrganizationName) – поиск элемента в дереве по названию организации\. Формальные параметры: elem – корень дерева, OrganizationName – название искомого элемента. Входные данные: корень дерева, название искомого элемента, дерево. Выходные данные: найденный элемент либо null.
* FindMaxElem(elem) – поиск максимального элемента в дереве. Формальные параметры: elem – корень дерева/поддерева. Входные данные: корень дерева/поддерева, дерево. Выходные данные: найденный элемент либо null
* FindMinElem (elem) – поиск минимального элемента в дереве. Формальные параметры: elem – корень дерева/поддерева. Входные данные: корень дерева/поддерева, дерево. Выходные данные: найденный элемент либо null
* DeleteElem (info) – удаление элемента из дерева. Если список значений оказывается пустым, удаляется сам узел. Формальные параметры: информация равная информации удаляемого элемента. Входные данные: информация равная информации удаляемого элемента, дерева. Выходные данные: true/false в случае удачи/неудачи соответственно.
* DeleteElemSubfunc(elem)– метод (вызывается в методе DeleteElem) удаляет элемент, у которого нет детей или ребенок только один, и корректно меняет связи. Формальные параметры: ударяемый элемент. Входные данные: корень дерева, удаляемый элемент. Выходные данные: нет.
* GetParent(child) - поиск родителя элемента-наследника(если таковой есть) Формальные параметры - элемент child. Входные данные: корень дерева, элемент child. Выходные данные - значения элементов дерева.
* PrintTree() - получение информации о дереве. Формальные параметры: нет. Входные данные: дерево, ключ ,значение узла. Выходные данные: строка с информацией.
* print\_Tree\_Hidden(TreeElem p, int level, ref string result) - внутренняя подфункция для получения информации о дереве. Формальные параметры: p – текущий элемент, level – глубина, result – ссылка строки для вывода. Входные данные: дерево, p – текущий элемент, level – глубина, result – ссылка строки для вывода. Выходные данные: нет.
* print\_NLR(TreeElem Head) - Прямой обход(кто первый тот и выводится). Формальные параметры: корень дерева. Входные данные: корень дерева. Выходные данные: значение элементов дерева.

Вложенные типы:

**Класс TreeElem** - класс, описывающий структуру элемента.

Поля:

* Info – информация элемента.
* Left – ссылка на левый узел.
* Right - ссылка на правый узел.

# Описание интерфейса

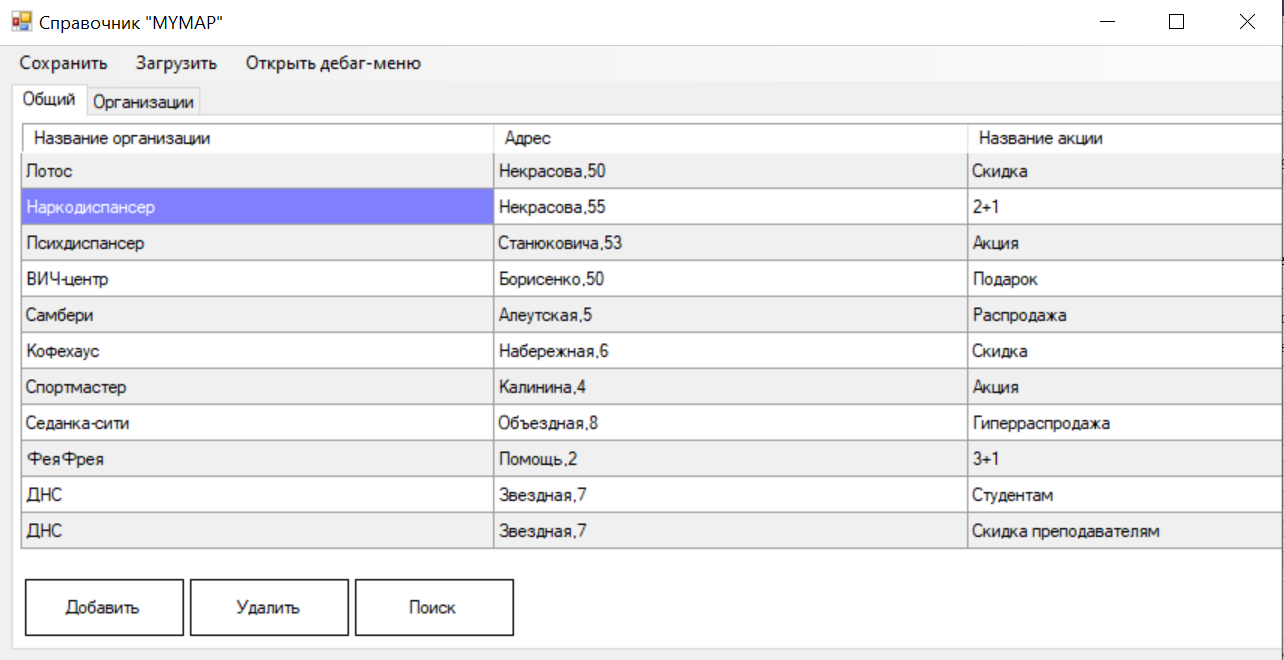
На Рисунке 6 представлено главное окно программы. В верхней части окна располагаются кнопки «Загрузить» - для заполнений справочников из файла, и «Сохранить» - для сохранения текущего справочника в файл и «Открыть дебаг-меню» - для открытия окна отладки.

Рисунок 6. Главное окно программы

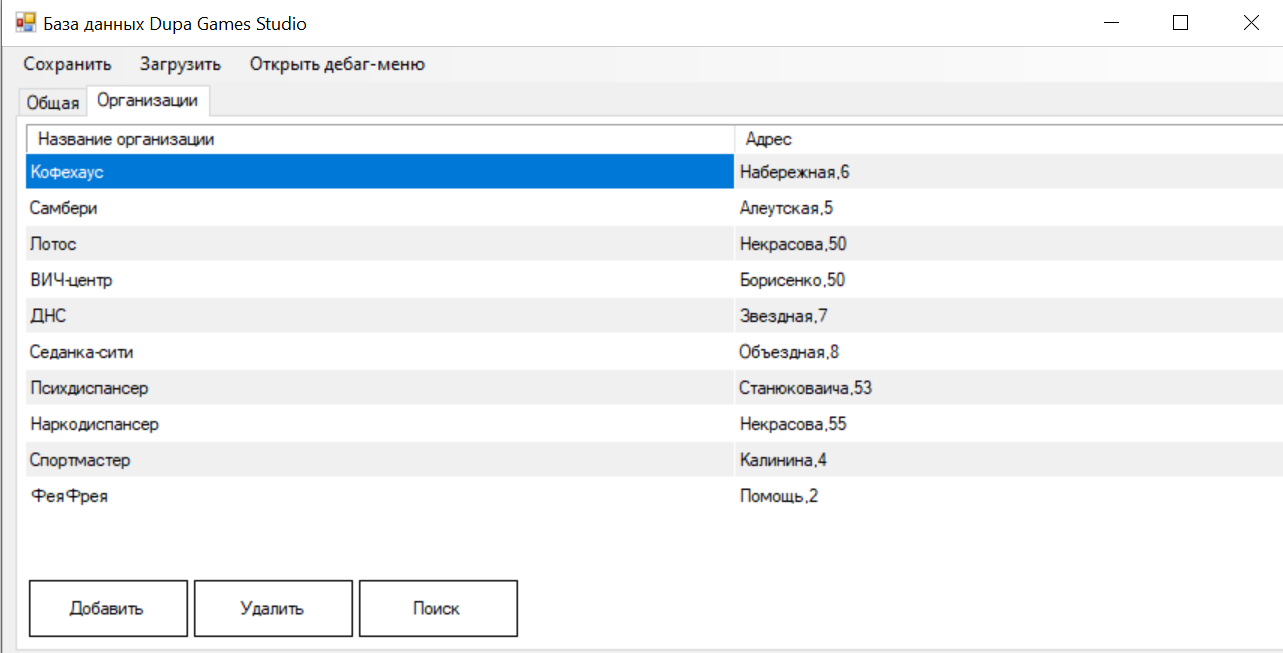
В приложении можно переключаться между справочниками, используя вкладки «Общий» и «Организации» в верхней части окна (Рисунок 7). Кнопки в нижней части окна «Поиск», «Добавить» и «Удалить» работают с выбранным с помощью вкладок справочником.

Рисунок 7. Переключение между справочниками

Рассмотрим работу кнопок «Поиск», «Добавить», «Удалить» в «Организации».

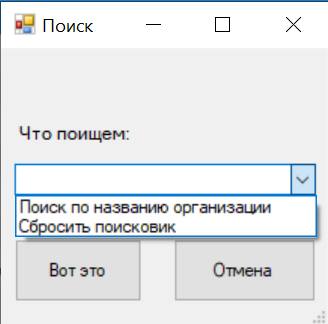
При нажатии кнопки «Найти» откроется диалоговое окно (Рисунок 8), в котором пользователю выбрать параметры поиска справочнике.

Рисунок 8. Окно для выбора параметра поиска

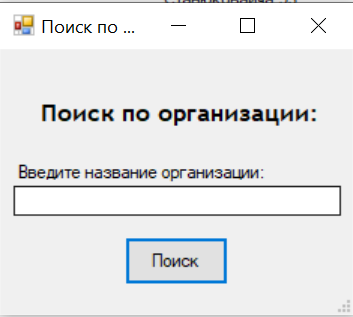
При выборе варианта «Поиск по названию организации» и нажатии кнопки «Вот это» в диалоговом окне программа выведет окно для ввода названия организации (Рисунок 9).

Рисунок 9. Поиск организации по названию в справочнике «Организации»

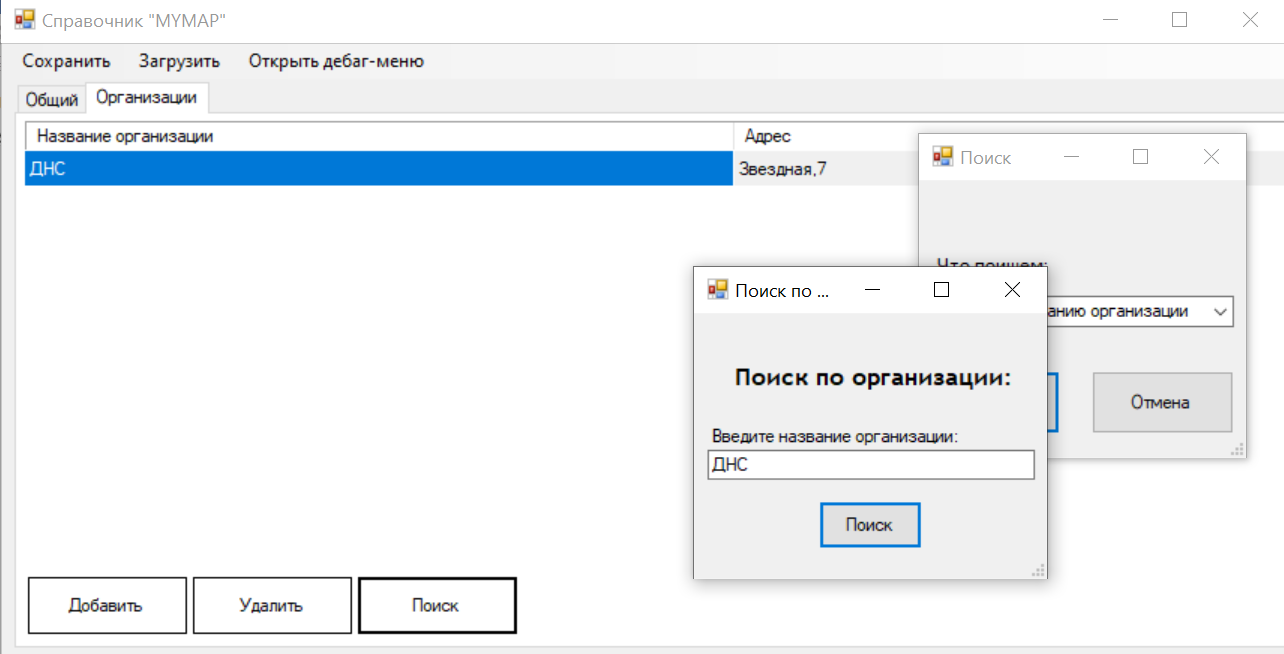
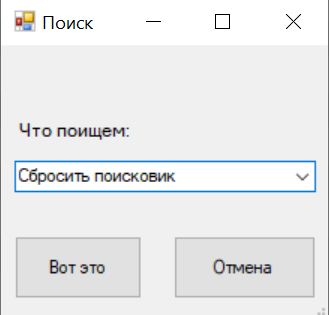
После ввода названия и нажатия кнопки «Поиск», останутся только те записи в справочнике, которые удовлетворяют условиям поиска (Рисунок 10).

Рисунок 10. Результаты поиска по названию

После закрытия диалогового окна справочник не вернётся в своё первоначальное состояние. Для этого необходимо выбрать в параметрах поиска «Сбросить поисковик» и нажать кнопку «Вот это» (Рисунок 11)

Рисунок 11. Сброс фильтра поиска по названию организации

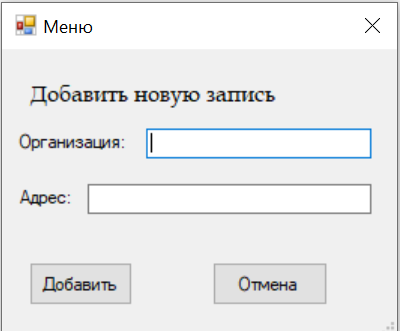
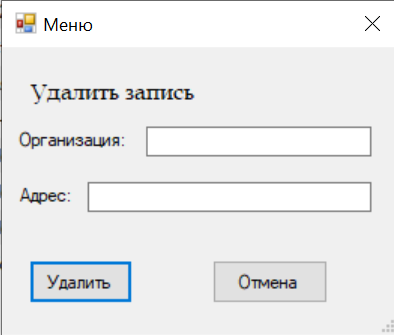
Кнопка «Добавить» откроется диалоговое окно (Рисунок 12), в котором пользователю необходимо ввести название и адрес организации, которую необходимо добавить в справочник.

Рисунок 12. Добавление организаций в справочник «Организации»

Если данные введены корректно (нет дублей), то после нажатия кнопки «Добавить» данные добавятся в справочник.

При нажатии кнопки «Удалить» откроется диалоговое окно, в котором пользователю необходимо ввести название и адрес организации, которую надо удалить (Рисунок 13).

Рисунок 13. Удаление в справочнике «Организации»

При удалении организации из справочника «Организации», она удаляется и из справочника «Общий».

Рассмотрим работу кнопок «Поиск», «Добавить» и

«Удалить» в справочнике «Общий».

Кнопка «Поиск» дублируется из справочника «Организации».

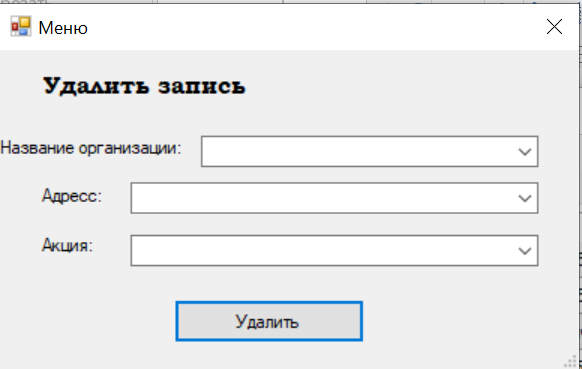
При нажатии кнопки «Удалить» откроется диалоговое окно, в котором пользователю необходимо ввести акцию, название и адрес организации, которую надо удалить (Рисунок 14).

Рисунок 14. Удаление в справочнике «Общий»

При удалении организации из справочника «Общий», она не удаляется из справочника «Организации».

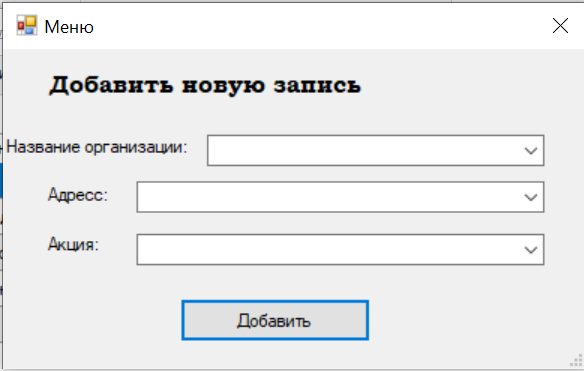
При нажатии кнопки «Добавить» откроется диалоговое окно (Рисунок 15), в котором пользователю необходимо ввести данные об организации (название, адрес и проводимую ею акцию)

Рисунок 15. Добавление организации в справочнике «Общий».

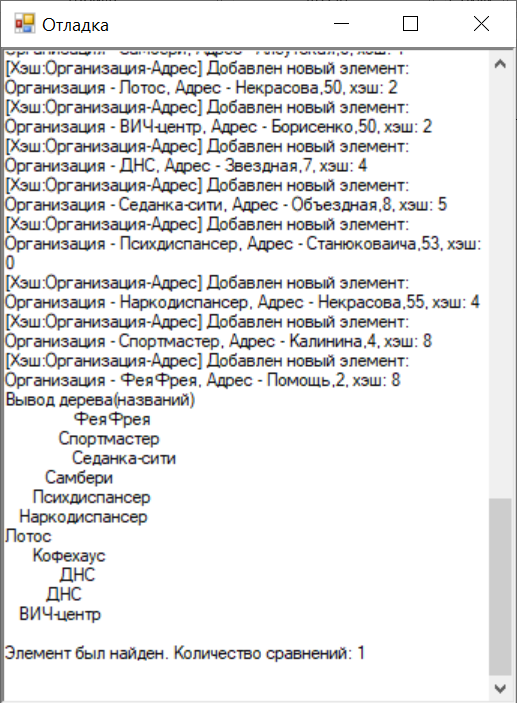
При добавлении организации в справочнике «Общий», она автоматически добавляется в справочник «Организации». При нажатии кнопки «Открыть дебаг-меню» в верхней части окна, появляется окно откладки (Рисунок 16), в котором представлена техническая информация, которая обновляется автоматически.

Рисунок 16. Окно отладки

При некорректно вводе данных для пользователя выводится окно с сообщением об ошибке (Рисунок 17), текст сообщения меняется в зависимости от формы и особенностей ввода.

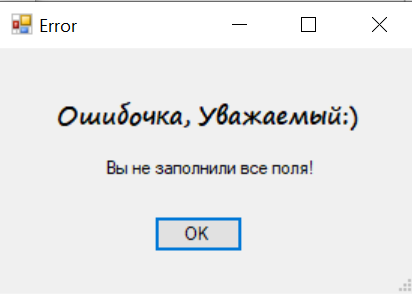


Рисунок 17.Окно ошибок

# Тестирование

Тестирование проводилось методом черного ящика, результаты которого представлены в Таблицах 1, 2.

Таблица 1. Тестирование работы со справочником организаций

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Описание тестовой ситуации | | Входные данные | | Выходные данные | |
| Хеш таблица с методом цепочек | Название организации, Адрес | Хеш-таблица с методом цепочек | Результат |
| Добавление | | | | | |
| 1 | Добавление корректных данных |  | ДВФУ Аякс,10 | ДВФУ Аякс,10 | Данные добавлены |
| 2 | Добавление некорректных данных |  | ДВФУ |  | Данные не добавлены |
| ДВФУ Аякс,10 |  | ДВФУ Аякс,10 | Данные не добавлены |
| 3 | Проверка уникальности ключа | ДВФУ Аякс,10 | ДВФУ Аякс,10 | ДВФУ Аякс,10 | Запись не добавлена |
| 4 | Добавление при коллизии | ДВФУ Аякс,10 | Банк Тихая,5 | ДВФУ Аякс,10 && Банк Тихая,5 | Данные добавлены |
| 5 | Добавление не уникального ключа | ДВФУ Аякс,10 | ДВФУ Аякс,16 | ДВФУ Аякс,10 | Данные не добавлены |
| Поиск | | | | | |
| 6 | Запись не существует |  | ДВФУ |  | Пустой справочник «Организации» |
| 7 | Запись существует | ДВФУ Аякс,10 | ДВФУ | ДВФУ Аякс,10 | ДВФУ Аякс,10 в справочнике «Организации» |
| 8 | Поиск при коллизии | ДВФУ Аякс,10 && Банк Тихая,5 | ДВФУ | ДВФУ Аякс,10 && Банк Тихая,5 | ДВФУ Аякс,10 в справочнике «Организации» |
| Удаление | | | | | |
| 9 | Запись не существует | ДВФУ Аякс,10 | Банк Тихая,5 | ДВФУ Аякс,10 | Данные не удалены |
| 10 | Запись существует | *Хеш-таблица:*  ДВФУ Аякс,10  *«Общий»:*  ДВФУ Аякс,10 Скидка | ДВФУ Аякс,10 |  | Данные удалены из справочников «Организации» и «Общий» |
| 11 | Удаление при коллизии | ДВФУ Аякс,10 && Банк Тихая,5 | ДВФУ Аякс,10 | Банк Тихая,5 | Данные удалены |

Таблица 2. Тестирование работы с общим справочником

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Описание тестовой ситуации | | | Входные данные | | Выходные данные | | |
| Бинарное дерево поиска | Название организации, адрес и акция | Бинарное дерево поиска | | Результат |
| Добавление | | | | | | | |
| 1 | Добавление корректных данных при существовании связанной записи | |  | ДВФУ Аякс,10 2+1 | ДВФУ | | Данные добавлены + занесены в справочник «Организации» |
| 2 | Добавление корректных данных при существовании связанной записи | | ДВФУ Аякс,10 2+1 |  | ДВФУ Аякс,10 2+1 | | Данные не добавлены |
| 3 | Добавление при наличии ключа в дереве | | ДВФУ | ДВФУ Аякс,10 2+1 | ДВФУ->ДВФУ  (Если акция не повторялась) | | Дубль добавлен вправо |
| ДВФУ(если акция повторялась) | | Данные не добавлены |
| 4 | Добавление при наличии ключа в дереве | | ДВФУ | Банк Тихая,5 Акция | ДВФУ->Банк, если Банк больше | | Данные добавлены |
| Банк<-ДВФУ, если Банк меньше | |
| Поиск | | | | | | | |
| 5 | Запись не существует | |  | ДВФУ |  | | Пустой справочник «Общий» |
| 6 | Запись существует | | ДВФУ | ДВФУ | ДВФУ | | Название организации, адрес и акция «Общий» |
| 7 | Существует несколько записей | | ДВФУ  ДВФУ | ДВФУ | ДВФУ  ДВФУ | | Две организации с адресом, названием и акцией в справочнике «Общий» |
| Удаление | | | | | | | |
| 8 | Запись не существует |  | ДВФУ Аякс,10 2+1 | |  | Справочник «Общий» не изменяется | |
| 9 | Запись существует | ДВФУ Аякс,10 2+1 | ДВФУ Аякс,10 2+1 | |  | Данные удалены | |
| 10 | Некорректные входные данные | ДВФУ Аякс,10 2+1 |  | | ДВФУ Аякс,10 2+1 | Данные не удалены | |
| 11 | Замена в дереве | ООН<-ДВФУ->ШКОЛА | ДВФУ | | ООН->ШКОЛА | При удалении узел меняется на потомка или максимальный слева | |

# Заключение

Целью курсового проекта было: разработать информационную систему для автоматизации работы со справочниками организаций.

Цель достигнута. Для достижения поставленной цели были выполнены следующие задачи:

1. проведён анализ предметной области проекта «MYMAP» и построена её модель,
2. изучены теоретические основы методов построения справочников,
3. определены требования к информационной системе,
4. информационная система была реализована и протестирована,
   1. изучен язык разработки C# версии 9.0;
   2. изучен .NET Framework 4.8.04084;
   3. во время разработки Microsoft Visual Studio Community 2019 была использована в качестве среды выполнения

# Список литературы

1. Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн. Алгоритмы: построение и анализ, 3-е изд.: Пер. с англ. – М.: ООО «И.Д.Вильямс», 2013. – 1328 с.: ил. парал. тит. англ.
2. Роберт Седжвик. Фундаментальные алгоритмы на С++. Анализ/Структуры данных/Сортировка/Поиск: Пер. с англ – К.: Издательство «ДиаСофт», 2001. – 688 с.