



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**

**ЛЕМЕШ ВЛАДИСЛАВ ЕВГЕНЬЕВИЧ**

**РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ ДЛЯ АВТОМАТИЗАЦИИ РАБОТЫ СО СПРАВОЧНИКАМИ «КОНСТРУКТОР ПК»**

**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

В.Е. Лемеш

(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
1 Анализ предметной области (ПО) .....	4
1.1 Модель ПО .....	4
1.2 Постановки задач обработки .....	6
2 Теоретическая часть .....	8
2.1 Хеш-таблица .....	8
2.1.1 Хеш-функция .....	8
2.1.2 Разрешение коллизий методом цепочек .....	9
2.1.3 Односвязный динамический список .....	10
2.2 АВЛ дерево .....	10
3 Требования к информационной системе .....	14
3.1 Функциональные требования .....	14
3.2 Требования к данным .....	15
3.2.1 Требования к входным данным .....	15
3.2.1 Требования к выходным данным .....	16
3.3 Требования к интерфейсу .....	16
4 Реализация .....	17
4.1 Диаграмма классов .....	17
4.2 Описание классов .....	17
4.3 Описание интерфейса .....	24
4.4 Тестирование .....	28
Заключение .....	30
Список литературы .....	31

## **Введение**

В области розничной торговли часто возникает потребность в хранении большого количества информации, её изменении и удобного доступа, что и послужило выбором данной предметной области, поэтому в рамках курсового проекта будет рассмотрена информационная система для магазинов и их товаров.

Целью курсового проекта является: разработка информационной системы для автоматизации работы со справочниками организации, предоставляющей услуги доступа к информации о товарах, находящихся в наличии.

Для достижения поставленной цели необходимо выполнить следующие задачи:

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

## 1 Анализ предметной области (ПО)

Требуется разработать информационную систему для автоматизации работы со справочниками организации, предоставляющей услуги доступа к информации о товарах и их наличии.

Система должна решать следующие задачи:

- 1) хранить информацию о товарах, находящихся в наличии;
- 2) позволять просматривать всю информацию о товарах, находящихся в наличии;
- 3) позволять добавлять информацию о товарах, находящихся в наличии;
- 4) позволять удалять информацию о товарах, находящихся в наличии;
- 5) позволять искать информацию о товарах, находящихся в наличии;
- 6) предусмотреть проверку целостности информации, представленной в справочниках товаров и товаров, находящихся в наличии.

### 1.1 Модель ПО

Предметная область – торговля цифровыми товарами.

Профессионал предметной области – сотрудник компании, предоставляющей услуги продажи цифровой техники.

#### **Объекты предметной области:**

Объект **Список товаров** – информация о нем представляется в справочнике, который содержит информацию о товаре, находящемся в наличии магазина.

Объект **Товар** – информация о нем содержит: наименование, цена.

**Наименование товара** – строка, содержащая как цифры, так и буквы латинского алфавита и кириллицы, а также допустим символ '-'. Наименование является ключом, то есть оно должно быть уникальным. Длина строки не должна превышать 30 символов

**Цена товара** – число, в диапазоне от 0 до 999999.

Ниже представлен пример справочника, который содержит информацию для каждого товара:

MSI Geforce RTX 3060 VENTUS 105800  
Palit Geforce 1050 Ti StormX 20990  
Gigabyte Geforce GTX 1660 OC 6G 58000  
XFX Radeon RX 550 RX-550P4PFG5 14990  
AMD FX-4320 OEM 3999  
AMD Athlon 3000G OEM 6999  
Intel Celeron G5905 OEM 6999  
AMD PRO A10-8770 OEM 7699  
Intel Core i3-10100F OEM 9299  
AMD FX-4230 OEM 10000

Объект **Список товаров в магазинах** – информация о нем представляется в справочнике, который содержит информацию о наличии товара в магазине.

Объект **Товар в магазине** – информация о нем содержит: магазин, товар.

Объект **Магазин** – информация о нем содержит: название магазина, адрес магазина.

**Название** – строка, содержащая как цифры, так и буквы латинского алфавита и кириллицы. Название является первым ключом. Длина строки не должна превышать 15 символов

**Адрес** – строка, содержащая как цифры, так и буквы латинского алфавита и кириллицы. Адрес является вторым ключом, то есть пара <название, адрес> должна быть уникальной. Длина строки не должна превышать 20 символов.

Ниже представлен пример справочника, который содержит информацию для каждого товара:

Мвидео Авиационная 88 Intel Core i3-10100F OEM 9299  
Мвидео Юбилейная 5 Intel Core i3-10100F OEM 9299  
Мвидео Проспект 60-лет победы Intel Core i3-10100F OEM 9299  
DNS Фрунзе 46 MSI Geforce RTX 3060 VENTUS 105800  
Мвидео Флегонтова 42 AMD PRO A10-8770 OEM 7699

Технопоинт Пушкина 7 MSI Geforce RTX 3060 VENTUS 105800  
Electronic Запарина 5 MSI Geforce RTX 3060 VENTUS 105800  
Интерстеп Павла-Морозова 24 MSI Geforce RTX 3060 VENTUS 105800  
220Volt Ким-ю-Чена 89 MSI Geforce RTX 3060 VENTUS 105800  
CyberMall Ленина 112 MSI Geforce RTX 3060 VENTUS 105800

## **1.2 Постановки задач обработки**

1. Поиск в справочнике «список товаров».

Входные данные: список товаров, наименование товара.

Выходные данные: список товаров.

Связь:

наименование товара = Список товаров.Товар.Наименование

2. Добавление в справочник «список товаров».

Входные данные: список товаров, наименование товара, цена товара.

Выходные данные: список товаров.

Связь:

Если в списке товаров не содержится Товар.Наименование && Товар.Цена то добавляем. Иначе не добавляем.

3. Удаление из справочника «список товаров».

Входные данные: список товаров, наименование товара, список товаров в магазинах.

Выходные данные: список товаров, список товаров в магазинах.

Связь:

Если в списке товаров содержится Товар.Наименование, то удаляем из списка товаров. Если удалили из списка товаров, то проверяем, если в списке товаров в магазинах содержится Товар.Наименование, то удаляем из списка товаров в магазинах.

4. Проверка целостности при добавлении в справочник «Список товаров в магазинах».

Входные данные: список товаров в магазинах, название магазина, адрес магазина, наименование товара, список товаров.

Выходные данные: список товаров в магазинах.

Связь:

Если в списке товаров в магазинах не содержится Магазин.Название && Магазин.Адрес && Товар.Наименование, то добавляем. Иначе не добавляем.

5. Удаление из справочника «Список товаров в магазинах».

Входные данные: список товаров в магазинах, название магазина, адрес магазина, наименование товара.

Выходные данные: список товаров в магазинах.

Связь:

Если в списке товаров в магазинах содержится Магазин.Название && Магазин.Адрес && Товар.Наименование, то удаляем. Иначе не удаляем.

6. Поиск по Товару в справочнике «Список товаров в магазинах».

Входные данные: список товаров в магазинах, наименование товара.

Выходные данные: список товаров в магазинах.

Связь:

Наименование товара = Список товаров в магазинах.Товар.Наименование

.

## 2 Теоретическая часть

В рамках курсового проекта необходимо хранить информацию о магазинах и товарах, а также осуществлять поиск по товару. Для этих целей используется АВЛ-дерево [2], которое является сбалансированным в следующем смысле: для любого узла дерева высота его правого поддерева отличается от высоты левого поддерева не более чем на единицу, что позволяет значительно увеличить скорость поиска.

Помимо хранения информации о магазинах и товарах, необходимо отдельно хранить информацию о товарах и осуществлять поиск по названию товара. Хеш-таблицы [1] позволяют осуществить наиболее быстрый доступ к информации. Это преимущество более очевидно, когда количество записей велико.

### 2.1 Хеш-таблица

Хеш-таблица [1] представляет собой эффективную структуру данных для реализации словарей. Хеширование представляет из себя применение хеш-функции [1] на ключах, поступающих к ней на вход, и преобразовании множества этих ключей в ячейки хеш-таблицы.

#### 2.1.1 Хеш-функция

Пусть элемент с ключом  $k$  хранится в ячейке  $k$ . При хешировании этот элемент хранится в ячейке  $h(k)$ , где  $h$  – хеш-функция. Функция  $h$  отображает совокупность ключей  $U$  на ячейки хеш-таблицы  $T[0 \dots m-1]: h: U \rightarrow \{0, 1, \dots, m-1\}$ , где размер  $m$  хеш-таблицы обычно меньше значения  $|U|$ . Мы говорим, что элемент с ключом  $k$  хешируется в ячейку  $h(k)$ ; величина  $h(k)$  называется хеш-значением ключа  $k$  [1].

Рассмотрим хеш-функцию, основанную на методе деления. Пусть  $a_1, a_2, \dots, a_n$  – все цифры данного ключа  $k$ . Тогда ключ  $k$  отображается в ячейку  $h(k)$  как остаток от деления суммы цифр ключа на размер хеш-таблицы, то есть



$$h(k) = \sum_{i=1}^n a_i \bmod m$$

В рамках курсового проекта ключом является строка, содержащая наименование товара (см. 1.1). Число  $k$  является суммой кодов таблицы `ascii` символов ключа

Пусть размер массива хеш-таблицы равен 100 и на вход поступают записи товаров из модели ПО (п. 1.1).

Наименование товара является ключом, поэтому на вход хеш-функции будут поступать только наименования товаров. Наименование товара будем представлять в виде суммы `char` значений в кодировке таблицы `ascii`.

AMD FX-4320 OEM = 65 + 77 + 68 + 32 + 70 + 88 + 45 + 52 + 51 + 50 + 48 + 32 + 79 + 69 + 77 = 903

Тогда  $h(\text{AMD FX-4320 OEM}) = 903 \bmod 100 = 3$

Таким образом:

$h(\text{MSI Geforce RTX 3060 VENTUS}) = 0$

$h(\text{Palit Geforce 1050 Ti StormX}) = 41$

$h(\text{Gigabyte Geforce GTX 1660 OC 6G}) = 31$

$h(\text{AMD Athlon 3000G OEM}) = 11$

$h(\text{XFX Radeon RX 550 RX-550P4PFG5}) = 30$

$h(\text{Intel Celeron G5905 OEM}) = 23$

$h(\text{AMD PRO A10-8770 OEM}) = 20$

$h(\text{Intel Core i3-10100F OEM}) = 24$

$h(\text{AMD FX-4230 OEM}) = 3$

Как видно из примера, значения хеш-функций могут совпадать для разных ключей. Для разрешения коллизий в данной работе будет использоваться метод цепочек.

### 2.1.2 Разрешение коллизий методом цепочек

При разрешении коллизий методом цепочек [1], все элементы, имеющие одинаковое значение хеш-функции, добавляются в связный список. В данной

работе будет использоваться односвязный динамический список с добавлением в конец списка.

### 2.1.3 Односвязный динамический список с добавлением в конец

Односвязный динамический список [1] — это динамическая структура данных, состоящая из узлов. Каждый узел содержит как минимум два поля, данные и указатель на следующий узел. В данной работе реализовано добавление элементов в конец динамического списка.

Пример хеш-таблицы приведен на рисунке 1:

0	MSI <u>Geforce</u> RTX 3060 VENTUS, 105800 → <u>null</u>
3	AMD FX-4320 OEM, 3999 → AMD FX-4230 OEM, 10000 → null
11	AMD Athlon 3000G OEM, 6999 → <u>null</u>
23	Intel Celeron G5905 OEM, 6999 → <u>null</u>
35	Intel Core i3-10100F OEM, 9299 → <u>null</u>
41	<u>Palit Geforce 1050 Ti StormX</u> , 20990 → <u>null</u>
74	XFX Radeon RX 550 RX-550P4PFG5, 14990 → <u>null</u>
90	Gigabyte <u>Geforce</u> GTX 1660 OC 6G, 58000 → <u>null</u>
93	AMD PRO A10-8770 OEM, 7699 → <u>null</u>

Рисунок 1 – хеш-таблица с разрешением коллизий методом цепочек с добавлением в конец динамического односвязного списка

## 2.2 AVL дерево

Сбалансированным называется такое двоичное дерево поиска, в котором высота каждого из поддеревьев, имеющих общий корень, отличается не более чем на некоторую константу  $k$ , и при этом выполняются условия характерные для двоичного дерева поиска.

AVL-дерево – сбалансированное двоичное дерево поиска с  $k = 1$ . Для его узлов определен коэффициент сбалансированности (balance factor). Balance factor – это разность высот правого и левого поддеревьев, принимающая одно значение из множества  $\{-1, 0, 1\}$  [1].

Если после выполнения операции добавления или удаления, коэффициент сбалансированности какого-либо узла AVL-дерева становится равен 2, то необходимо выполнить операцию балансировки. Она осуществляется путем вращения (поворота) узлов – изменения связей в поддереве. Вращения не меняют свойств бинарного дерева поиска, и выполняются за константное время. Всего различают 4 их типа:

1. малое правое вращение;
2. большое правое вращение;
3. малое левое вращение;
4. большое левое вращение.

Возможны два случая нарушения сбалансированности. Первый из них исправляется 1 и 3 типом, а второй – 2 и 4. Рассмотрим первый случай. Пусть имеется следующее сбалансированное поддерево (Рисунок 2):

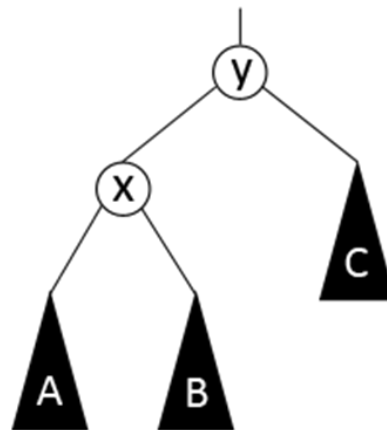


Рисунок 2 – сбалансированное дерево

Здесь  $x$  и  $y$  – узлы, а  $A$ ,  $B$ ,  $C$  – поддеревья. После добавления к поддереву  $A$  узла  $v$ , баланс нарушится, и потребуется балансировка. Она осуществляется правым поворотом (тип 1) узла  $y$  (Рисунок 3):

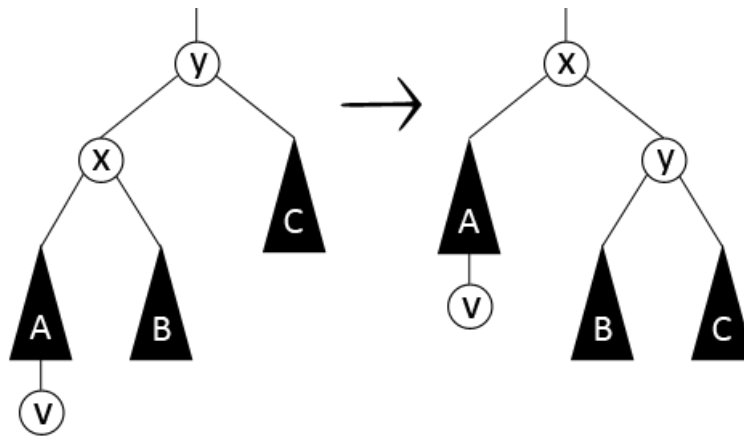


Рисунок 3 – правый поворот АВЛ дерева

Второй случай дисбаланса исправляется большим правым или большим левым вращением. Пусть имеется следующее сбалансированное поддереве (Рисунок 4):

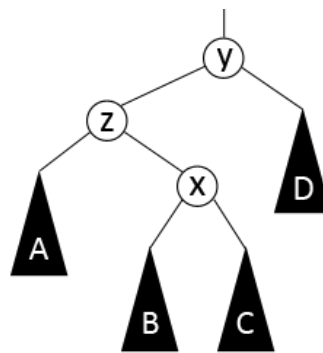


Рисунок 4 – сбалансированное дерева

Вставка узлов в поддереве A или D, не нарушит сбалансированности, но добавление их в B или C приведет к необходимости произвести балансировку вращением 2-го типа (Рисунок 5):

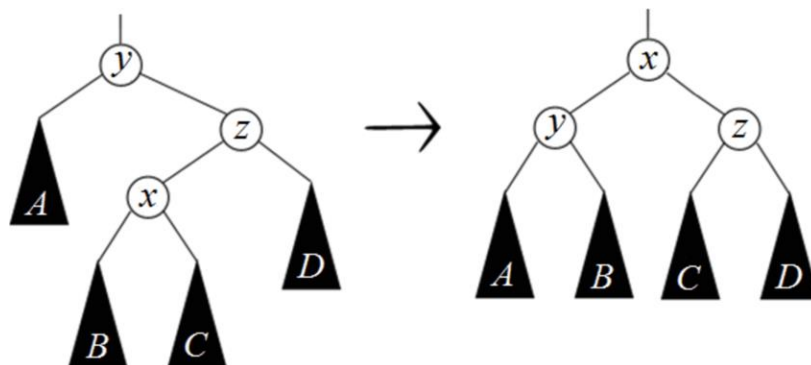


Рисунок 5 – большой левый поворот

### Добавление узлов

Операция вставки нового узла в АВЛ-дерево выполняется рекурсивно. По ключу данного узла производится поиск места вставки: спускаясь вниз по дереву, алгоритм сравнивает ключ добавляемого узла со встречающимися ключами, далее происходит вставка нового элемента; по возвращению из рекурсии, выполняется проверка всех показателей сбалансированности узлов и, в случае необходимости, выполняется балансировка. Для осуществления балансировки следует знать, с каким из рассмотренных выше случаев дисбаланса имеем дело. Допустим, мы добавили узел  $x$  в левое поддерево, для которого выполнялось  $h(T_i, R) < h(T_i, L)$ , т. е. высота левого поддерева изначально превышала высоту правого. Если левое поддерево этого узла выше правого, то потребуется большое вращение, иначе – малое.

### **Удаление узлов**

В данной работе, при удалении узла, имеющего дочерние узлы, на его место встаёт минимальный справа узел.

Также как и вставку узла, его удаление удобно задать рекурсивно. Пусть  $x$  – удаляемый узел, тогда если  $x$  – лист (терминальный узел), то алгоритм удаления сводится к простому исключению узла  $x$ , и подъему к корню с переопределением *balance factor*’ов узлов. Если же  $x$  не является листом, то он либо имеет правое поддерево, либо не имеет его. Во втором случае, из свойства АВЛ-дерева, следует, что левое поддерево имеет высоту 1, и здесь алгоритм удаления сводится к тем же действиям, что и при терминальном узле. Остается ситуация, когда у  $x$  есть правое поддерево. В таком случае нужно в правом поддерево отыскать следующий по значению за  $x$  узел  $y$ , заменить  $x$  на  $y$ , и рекурсивно вернуться к корню, переопределяя коэффициенты сбалансированности узлов. Из свойства двоичного дерева поиска следует, что узел  $y$  имеет наименьшее значение среди всех узлов правого поддерева узла  $x$ .

Ниже представлен пример АВЛ дерева на записях из анализа предметной области (Рисунок 6):

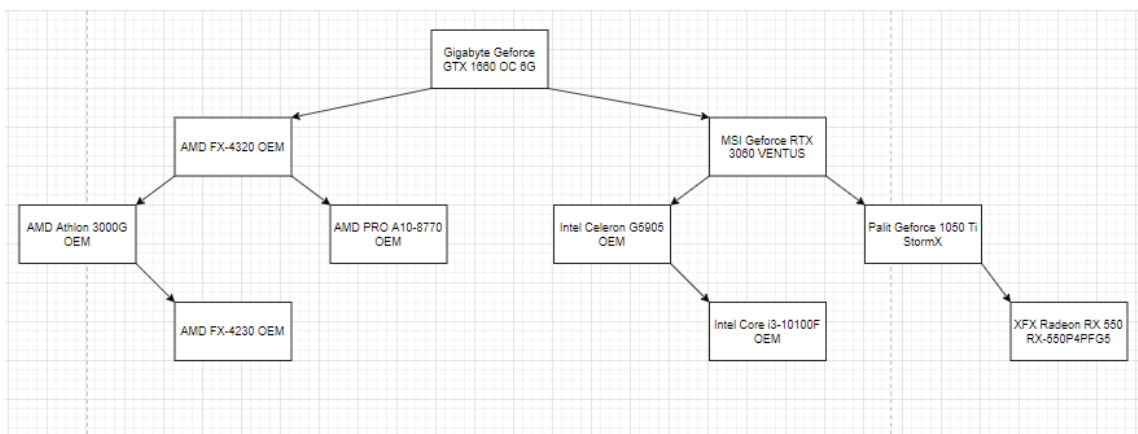


Рисунок 6 – AVL дерево, построенное на записях из анализа предметной области

### 3 Требования к информационной системе

#### 3.1 Функциональные требования

Информационная система для автоматизации работы со справочниками «Конструктор ПК», должна позволять:

1. хранить информацию о товарах и их ценах;
2. позволять просматривать всю информацию о магазинах их адресах, товарах и их ценах;
3. позволять добавлять информацию о товарах и их ценах (при добавлении дублирующей информации выводить соответствующее сообщение). При добавлении в справочник список товаров в магазинах проверять, присутствует ли наименование товара в списке товаров, если нет, то не добавлять данную запись и выводить соответствующее сообщение;
4. позволять удалять информацию о товарах и их ценах, при удалении проверять содержится ли данная запись в справочнике список товаров, если нет, то выводить соответствующее сообщение. Иначе проверять содержится ли данная запись в справочнике список товаров в магазинах, если содержится, то удалить соответствующую запись в справочнике список товаров в магазинах;
5. позволять искать информацию о товарах и их ценах, при поиске проверять содержится ли данная запись в справочнике список товаров в магазинах, если нет, то выводить соответствующее сообщение.

## **3.2 Требования к данным**

### **3.2.1 Требования к входным данным**

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

– текстовый файл General.txt, каждая строка файла содержит информацию об объекте список товаров в магазинах, а именно название магазина, адрес магазина, наименование товара разделенные символом «;»

Пример текстового файла:

Мвидео;Авиационная 88;Intel Core i3-10100F OEM;9299

Мвидео;Юбилейная 5;Intel Core i3-10100F OEM;9299

Мвидео;Проспект 60-лет победы;Intel Core i3-10100F OEM;9299

DNS;Фрунзе 46;MSI Geforce RTX 3060 VENTUS;105800

Мвидео;Флегонтова 42;AMD PRO A10-8770 OEM;7699

Технопоинт;Пушкина 7;MSI Geforce RTX 3060 VENTUS;105800

Electronic;Запарина 5;MSI Geforce RTX 3060 VENTUS;105800

Интерстеп;Павла-Морозова 24;MSI Geforce RTX 3060 VENTUS;105800

220Volt;Ким-ю-Чена 89;MSI Geforce RTX 3060 VENTUS;105800

CyberMall;Ленина 112;MSI Geforce RTX 3060 VENTUS;105800

– текстовый файл Price.txt, каждая строка файла содержит информацию об объекте список товара, а именно наименование товара, цена товара разделенный символом «;»

Пример текстового файла:

MSI Geforce RTX 3060 VENTUS;105800

Palit Geforce 1050 Ti StormX;20990

Gigabyte Geforce GTX 1660 OC 6G;58000

XFX Radeon RX 550 RX-550P4PFG5;14990

AMD FX-4320 OEM;3999

AMD Athlon 3000G OEM;6999

Intel Celeron G5905 OEM;6999

AMD PRO A10-8770 OEM;7699

Intel Core i3-10100F OEM;9299

AMD FX-4230 OEM;10000

- Наименование, Цена (см. п.1.1);
- Магазин, Товар (см. п.1.1).

### **3.2.1 Требования к выходным данным**

Выходными данными для работы со справочниками являются:

- текстовый файл General.txt, каждая строка файла содержит информацию об объекте наличие товара в магазине, а именно название магазина, адрес магазина, наименование товара разделенные символом «;»
- текстовый файл Price.txt, каждая строка файла содержит информацию об объекте товар, а именно наименование товара, цена товара разделенный символом «;»
- сообщения об ошибках должны выводиться в отдельном окне и содержать информацию об ошибке. Всевозможные сообщения об ошибках приведены ниже:

1. “Некорректный ввод!”
2. “Введите уникальный ключ!”
3. “Запись не найдена!”
4. “Запись уже существует”
5. “Совпадения не найдены”

### **3.3 Требования к интерфейсу**

Оконный интерфейс. Русская локализация.



## 4 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 7.

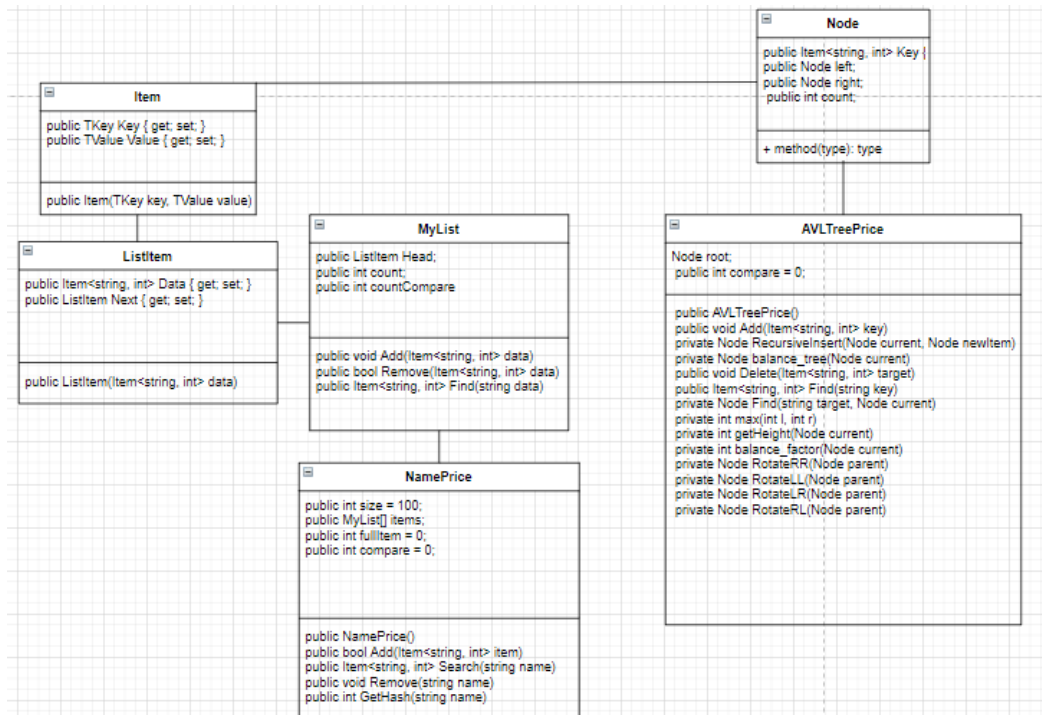


Рисунок 7 – UML диаграмма

### 4.2 Описание классов

Класс Item - класс, описывающий объект товар.

Поля:

public TKey Key – хранит наименование товара

public TValue Value – хранит цену товара

Методы:

public Item(TKey key, TValue value) – конструктор класса, присваивает значения полям Key и Value

Входные данные: наименование товара, цена товара

Выходные данные: объект товара

Список формальных параметров: TKey key – наименование товара, TValue value – цена товара

Класс ListItem - класс, описывающий узел односвязного списка.

Поля:

public Item<string, int> Data – хранит объект товар

public ListItem Next – хранит адрес следующего узла

Методы:

public ListItem(Item<string, int> data)– конструктор класса, присваивает значения полю Data

Входные данные: объект товар

Выходные данные: элемент односвязного списка

Список формальных параметров: Item<string, int> data – объект товар

Класс NamePrice - класс, описывающий хеш-таблицу.

Поля:

public int size; - хранит размер таблицы

public MyList[] items; - массив списков, являющихся ячейками таблицы

public int fullItem; - хранит количество ячеек хранящих данные

public int compare; - хранит количество сравнений при поиске.

Методы:

public NamePrice() – конструктор класса, заполняет таблицу с файла

Входные данные: файл ввода, пустая хеш-таблица

Выходные данные: хеш-таблица

Список формальных параметров: string path – хранит название файла ввода

public bool Add(Item<string, int> item) – метод добавляет в таблицу значение item

Входные данные: объект товар, хеш-таблица

Выходные данные: хеш-таблица

Список формальных параметров: Item<string, int> item – объект товар

public Item<string, int> Search(string name) – метод поиска объекта товар, по наименованию товара.

Входные данные: хеш-таблица, наименование товара

Выходные данные: хеш-таблица

Список формальных параметров: string name – наименование товара

public void Remove(string name) – метод удаляет объект товар, по наименованию товара.

Входные данные: наименование товара, хеш-таблица

Выходные данные: хеш-таблица

Список формальных параметров: string name – наименование товара

public int GetHashCode(string name) – метод вычисляет значение хеш-функции по наименованию товара.

Входные данные: наименование товара

Выходные данные: значение хеш-функции

Список формальных параметров: string name – наименование товара

Предположим, что размер хеш-таблицы равен 100, а на вход поступает запись товара из модели предметной области. Приведем пример:

$$\text{GetHash}(\text{"AMD FX-4320 OEM"}) = (65 + 77 + 68 + 32 + 70 + 88 + 45 + 52 + 51 + 50 + 48 + 32 + 79 + 69 + 77) \bmod 100 = 903 \bmod 100 = 3$$

Таким образом:

$\text{GetHash}(\text{"MSI Geforce RTX 3060 VENTUS"}) = 0$

$\text{GetHash}(\text{"Palit Geforce 1050 Ti StormX"}) = 41$

$\text{GetHash}(\text{"AMD Athlon 3000G OEM"}) = 11$

$\text{GetHash}(\text{"Gigabyte Geforce GTX 1660 OC 6G"}) = 31$

$\text{GetHash}(\text{"XFX Radeon RX 550 RX-550P4PFG5"}) = 30$

$\text{GetHash}(\text{"Intel Celeron G5905 OEM"}) = 23$

$\text{GetHash}(\text{"AMD PRO A10-8770 OEM"}) = 20$

$\text{GetHash}(\text{"Intel Core i3-10100F OEM"}) = 24$

$\text{GetHash}(\text{"AMD FX-4230 OEM"}) = 3$

Класс MyList - класс, описывающий односвязный список.

Поля:

public ListItem Head; - хранит голову списка

`public int count;` - хранит количество элементов в списке

`public int countCompare;` - хранит количество сравнений при поиске

Методы:

`public void Add(Item<string, int> data)` – метод добавляет объект товар в односвязный список

Входные данные: односвязный список, объект товар

Выходные данные: односвязный список

Список формальных параметров: `Item<string, int> data` – объект товар

`public bool Remove(Item<string, int> data)` – метод удаляет объект товар из односвязного списка

Входные данные: объект товар, односвязный список

Выходные данные: односвязный список

Список формальных параметров: `Item<string, int> data` – объект товар

`public Item<string, int> Find(string data)` – метод осуществляет поиск объекта товар в односвязном списке по наименованию товара

Входные данные: наименование товара, односвязный список

Выходные данные: объект товар, динамический список

Список формальных параметров: `string data` – хранит наименование товара

Класс `Node` - класс, описывающий узел бинарного дерева.

Поля:

`public Item<string, int> Key` – хранит объект товара

`public Node left` – хранит адрес левого поддерева

`public Node right` – хранит адрес правого поддерева

`public int count` – хранит количество вхождений одинаковых объектов товара

Методы:

`public Node(Item<string, int> key)` – конструктор класса присваивает значение полю `Key`

Входные данные: объект товар

Выходные данные: узел АВЛ дерева

Список формальных параметров: `Item<string, int> key` – объект товар

Класс `AVLTreePrice` - класс, описывающий АВЛ дерево.

Поля:

`Node root` – хранит корень дерева

`public int compare` – хранит количество сравнений при поиске элемента

Методы:

`public AVLTreePrice()` – конструктор класса присваивает `null` значение полю `root`

Входные данные: пустой корень АВЛ дерева

Выходные данные: АВЛ дерево

Список формальных параметров: -

`public void Add(Item<string, int> key)` – метод присваивает корню дерева, дерево с добавленным объектом товар

Входные данные: АВЛ дерево, объект товар

Выходные данные: АВЛ дерево

Список формальных параметров: `Item<string, int> key` – хранит объект товар

`private Node Insert(Node current, Node newItem)` – метод возвращает поддерево с добавленным объектом товара

Входные данные: АВЛ дерево, узел дерева, который необходимо добавить

Выходные данные: АВЛ дерево

Список формальных параметров: `Node current` – хранит корень дерева, `Node newItem` – хранит узел дерева, который необходимо добавить

`private Node balance_tree(Node current)` – метод балансирует дерево

Входные данные: АВЛ дерево

Выходные данные: АВЛ дерево

Список формальных параметров: `Node current` – хранит корень дерева

`public void Delete(Item<string, int> target)` – метод присваивает корню дерева, дерево с удалённым объектом товара

Входные данные: АВЛ дерево, объект товар

Выходные данные: AVL дерево

Список формальных параметров: `Item<string, int> target` – хранит объект товар

`private Node Delete(Node current, Item<string, int> target)`– метод возвращает дерево с удаленным объектом товара

Входные данные: AVL дерево, объект товар

Выходные данные: AVL дерево

Список формальных параметров: `Node current` – хранит корень дерева, `Item<string, int> target` - хранит объект товара

`private Node Delete(Node current, Item<string, int> target)`– метод возвращает дерево с удаленным объектом товара

Входные данные: AVL дерево, объект товар

Выходные данные: AVL дерево

Список формальных параметров: `Node current` – хранит корень дерева, `Item<string, int> target` – хранит объект товара

`public Item<string, int> Find(string key)` – метод возвращает найденный по наименованию объект товар

Входные данные: динамический список, наименование товара

Выходные данные: элемент динамического списка

Список формальных параметров: `string key` - наименование товара

`private Node Find(string target, Node current)` – метод возвращает найденный по наименованию узел дерева содержащий объект товар

Входные данные: наименование товара, AVL дерево

Выходные данные: узел дерева

Список формальных параметров: `string target` – хранит наименование товара, `Node current` – хранит корень дерева

`private int max(int l, int r)` – метод возвращает наибольшую высоту среди левого и правого поддеревьев

Входные данные: число равное высоте левого поддерева, число равное высоте правого поддерева

Выходные данные: целое число

Список формальных параметров: int l – хранит высоту левого поддерева,  
int r – хранит высоту правого поддерева

private int balance\_factor(Node current) – метод возвращает разницу высот  
левого и правого поддеревьев

Входные данные: AVL дерево

Выходные данные: AVL дерево

Список формальных параметров: Node current – хранит корень дерева

private Node RotateRR(Node parent) – метод осуществляет правый поворот

Входные данные: AVL дерево

Выходные данные: AVL дерево

Список формальных параметров: Node parent – хранит корень дерева

private Node RotateLL(Node parent) – метод осуществляет левый поворот

Входные данные: AVL дерево

Выходные данные: AVL дерево

Список формальных параметров: Node parent – хранит корень дерева

private Node RotateLR(Node parent) – метод осуществляет большой правый  
поворот

Входные данные: AVL дерево

Выходные данные: AVL дерево

Список формальных параметров: Node parent – хранит корень дерева

private Node RotateRL(Node parent) – метод осуществляет большой левый  
поворот

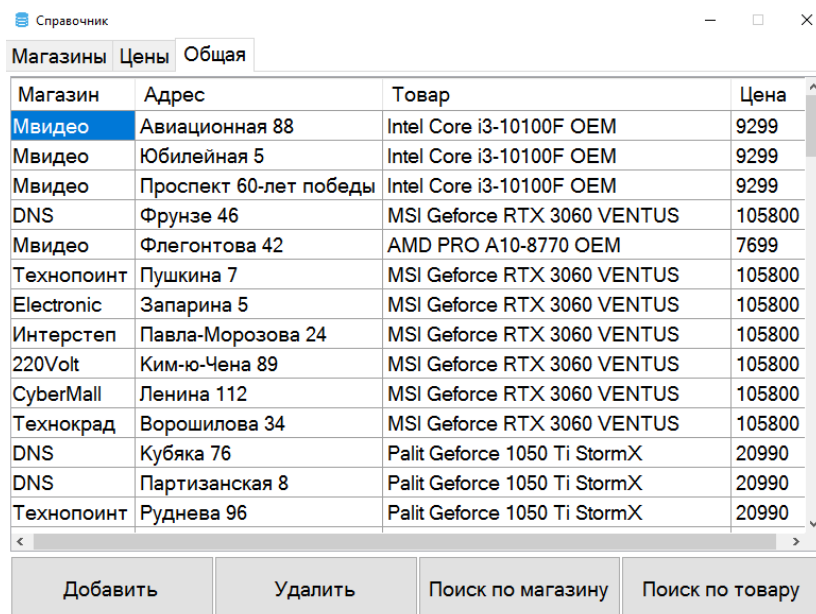
Входные данные: AVL дерево

Выходные данные: AVL дерево

Список формальных параметров: Node parent – хранит корень дерева

### 4.3 Описание интерфейса

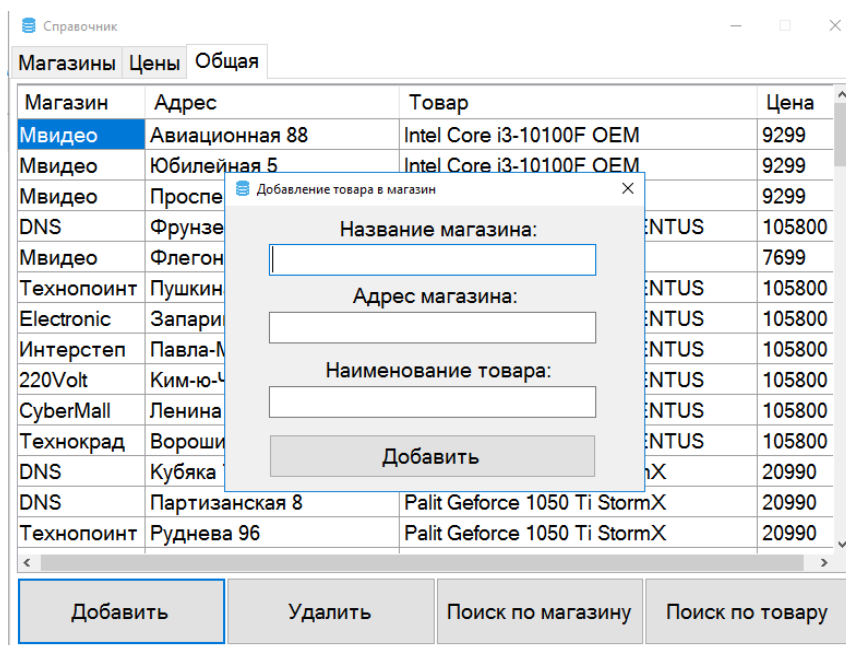
На рисунке 8 представлено главное окно программы:



Магазин	Адрес	Товар	Цена
Мвидео	Авиационная 88	Intel Core i3-10100F OEM	9299
Мвидео	Юбилейная 5	Intel Core i3-10100F OEM	9299
Мвидео	Проспект 60-лет победы	Intel Core i3-10100F OEM	9299
DNS	Фрунзе 46	MSI Geforce RTX 3060 VENTUS	105800
Мвидео	Флегонтова 42	AMD PRO A10-8770 OEM	7699
Технопоинт	Пушкина 7	MSI Geforce RTX 3060 VENTUS	105800
Electronic	Запарина 5	MSI Geforce RTX 3060 VENTUS	105800
Интерстеп	Павла-Морозова 24	MSI Geforce RTX 3060 VENTUS	105800
220Volt	Ким-ю-Чена 89	MSI Geforce RTX 3060 VENTUS	105800
CyberMall	Ленина 112	MSI Geforce RTX 3060 VENTUS	105800
Технокрад	Ворошилова 34	MSI Geforce RTX 3060 VENTUS	105800
DNS	Кубяка 76	Palit Geforce 1050 Ti StormX	20990
DNS	Партизанская 8	Palit Geforce 1050 Ti StormX	20990
Технопоинт	Руднева 96	Palit Geforce 1050 Ti StormX	20990

Рисунок 8 – главное окно программы

Для добавления новой записи в справочник список товаров в магазинах необходимо нажать кнопку «Добавить» после чего появится форма, изображенная на рисунке 9:



Добавление товара в магазин

Название магазина:

Адрес магазина:

Наименование товара:

Добавить

Рисунок 9 – форма добавления в список товаров в магазинах



Для удаления из справочника список товаров в магазинах, необходимо выбрать запись в справочнике и нажать кнопку «Удалить» после чего появится форма, изображенная на рисунке 10:

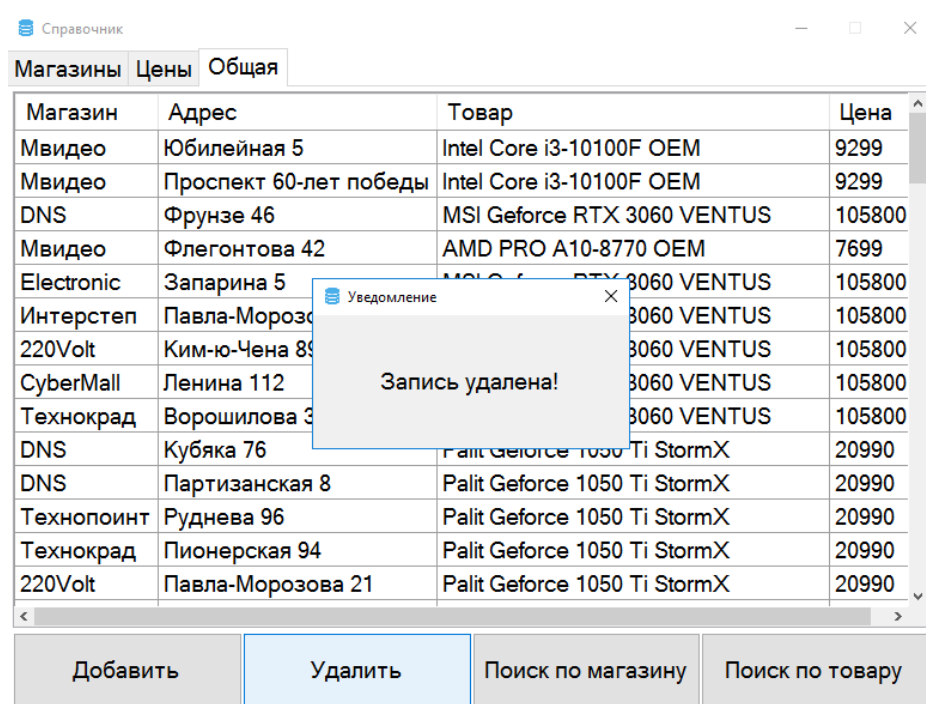


Рисунок 10 – форма успешного удаления из справочника список товаров в магазинах

Чтобы произвести поиск по наименованию товара, необходимо нажать кнопку «поиск по товару» после чего появится форма, изображенная на рисунке 11:

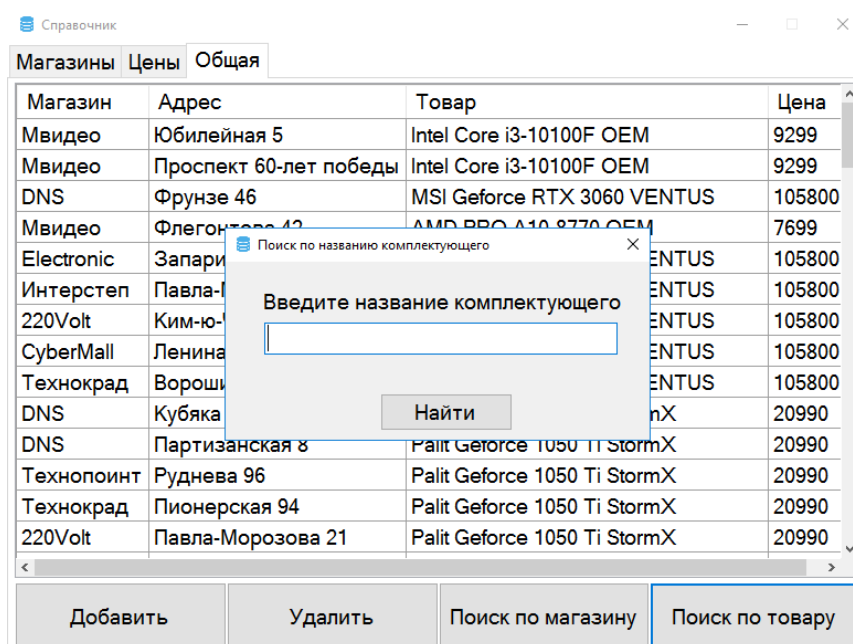


Рисунок 11 – форма поиска по наименованию товара

Для добавления в справочник список товаров, нужно перейти на вкладку Цены и нажать кнопку «Добавить», соответствующая форма приведена на рисунке 12:

The screenshot shows the 'Справочник' application window with the 'Цены' tab selected. A table lists items with columns 'Хэш', 'Наименование', and 'Цена'. A dialog box titled 'Добавление комплектующего' is open, containing input fields for 'Название товара:' and 'Цена товара:', and a 'Добавить' button. Below the table are buttons for 'Добавить', 'Удалить', and 'Найти', along with a search input field labeled 'Введите название'.

Хэш	Наименование	Цена
0	MSI Geforce RTX 3060 VENTUS	105800
3	AMD FX-4320 OEM	3999
3	AMD FX-4230 OEM	10000
9	AMD FX-4	
18	AMD Ryzen	
23	Intel Celeron	
35	Intel Core i	
41	Palit Geforce	
74	XFX Radeon	
90	Gigabyte G	
93	AMD PRO	

Рисунок 12 – форма добавления в справочник список товаров

Для удаления из справочника список товаров, нужно перейти на вкладку Цены, выбрать запись и нажать кнопку «Удалить», соответствующая форма приведена на рисунке 13:

The screenshot shows the 'Справочник' application window with the 'Цены' tab selected. A table lists items with columns 'Хэш', 'Наименование', and 'Цена'. A notification dialog box titled 'Уведомление' is open, displaying the message 'Связка удалена!'. Below the table are buttons for 'Добавить', 'Удалить', and 'Найти', along with a search input field labeled 'Введите название'.

Хэш	Наименование	Цена
3	AMD FX-4320 OEM	3999
3	AMD FX-4230 OEM	10000
9	AMD FX-4300 BOX	4499
18	AMD Ryzen 3 PRO 1200 OEM	9399
23	Intel Celeron G5905 OEM	6000
35	Intel Core i3-10100F	
41	Palit Geforce 1050 Ti	
74	XFX Radeon RX 550	
90	Gigabyte Geforce GT	
93	AMD PRO A10-8770 OEM	7099

Рисунок 13 – форма успешного удаления из справочника список товаров

Для поиска по наименованию товара в справочнике список товаров, необходимо ввести название товара в правом нижнем углу формы и нажать кнопку «Найти», соответствующая форма приведена на рисунке 14:

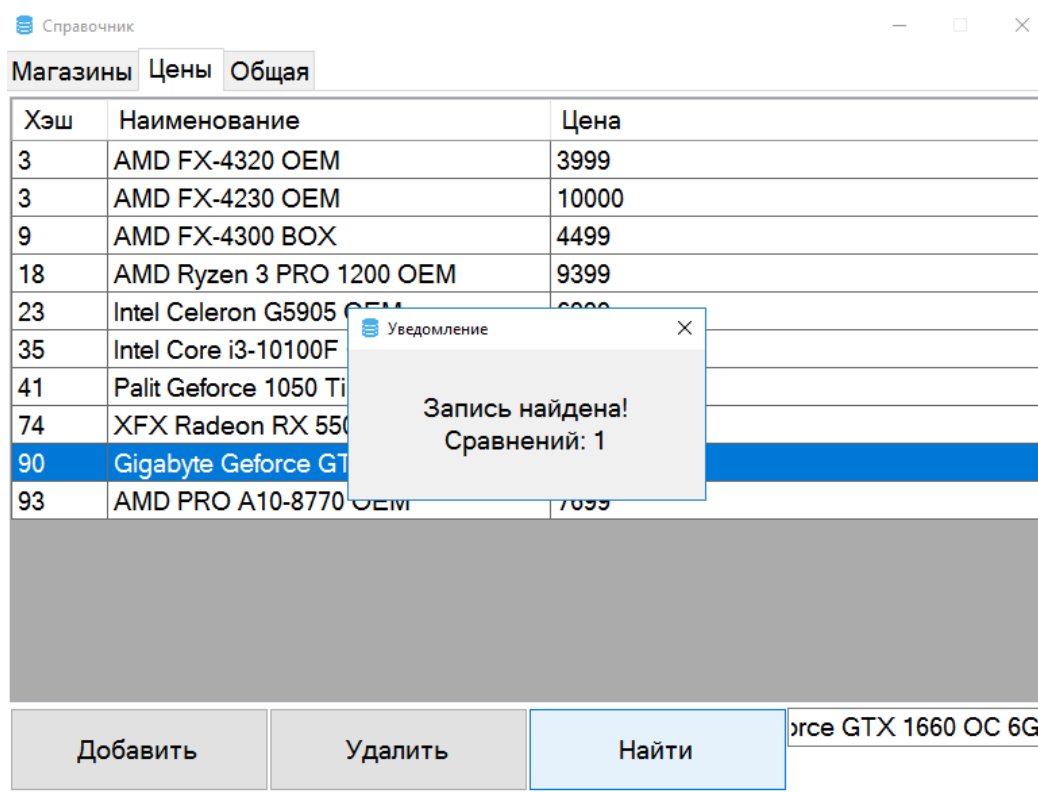


Рисунок 14 – форма успешного поиска в справочнике список товаров

При выходе из программы, пользователю будет предложено сохранить результаты работы, рисунок 15:

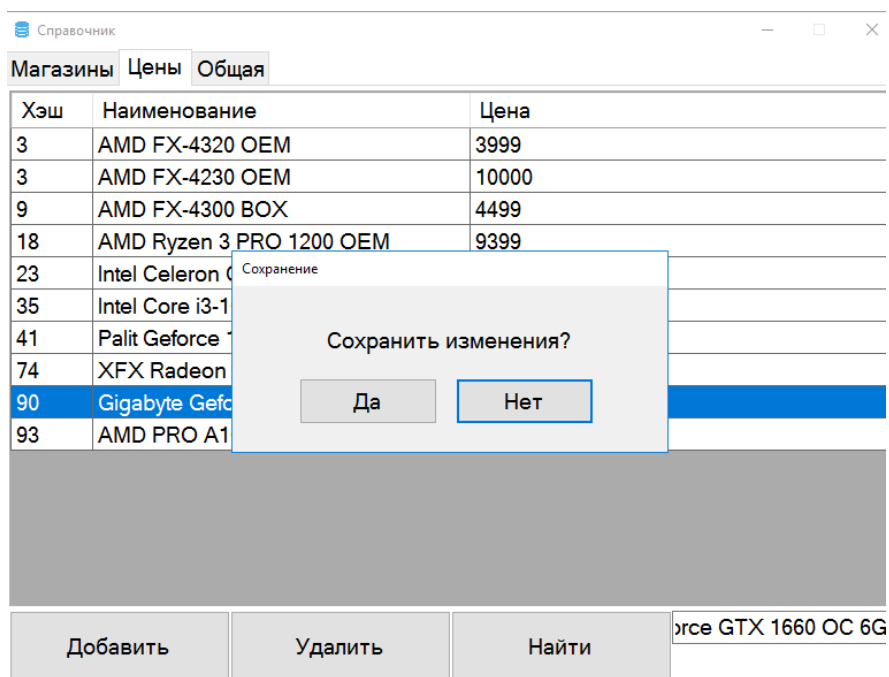


Рисунок 15 – форма сохранения результата работы программы

## 4.4 Тестирование

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

Таблица 1 – тестирование справочника список товаров

Описание тестовой ситуации		Входные данные		Выходные данные	
		Хеш-таблица (метод цепочек)	Наименование товара, цена товара	Хеш-таблица (метод цепочек)	Сообщение
Добавление					
1	Добавление некорректных данных	-	AMD FX-4230 OEM, АБВГ	-	Некорректный ввод
2	Добавление корректных данных	-	MSI Geforce RTX 3090, 40000	MSI Geforce RTX 3090, 40000	Запись добавлена
3	Проверка уникальности ключа	MSI Geforce RTX 3090, 40000	MSI Geforce RTX 3090, 40000	MSI Geforce RTX 3090, 40000	Введите уникальный ключ
4	Добавление при коллизии	AMD FX-4320 OEM, 10000	AMD FX-4230 OEM, 10000	AMD FX-4320 OEM, 10000 AMD FX-4230 OEM, 10000	Запись добавлена
Поиск					
5	Запись не существует	-	Geforce RTX	-	Запись не найдена, сравнений: 1
6	Запись существует	MSI Geforce RTX 3090, 40000	MSI Geforce RTX 3090	MSI Geforce RTX 3090, 40000	Запись найдена, сравнений: 1
7	Поиск при коллизии	AMD FX-4320 OEM, 10000 AMD FX-4230 OEM, 10000	AMD FX-4230	AMD FX-4320 OEM, 10000 AMD FX-4230 OEM, 10000	Запись найдена, сравнений: 2
Удаление					
8	Запись не существует	-	MSI Geforce RTX 3090	-	-
9	Запись существует	MSI Geforce RTX 3090	MSI Geforce RTX 3090	-	Запись удалена
10	Удаление при коллизии	AMD FX-4320 OEM, 10000 AMD FX-4230 OEM, 10000	AMD FX-4230	AMD FX-4320 OEM, 10000	Запись удалена

Таблица 2 – Тестирование справочника список товаров в магазинах

Описание тестовой ситуации		Входные данные		Выходные данные	
		АВЛ дерево	наименование товара, цена товара	АВЛ дерево	Сообщение
Добавление					
1	Добавление некорректных данных	-	-	-	Записи не существует
2	Добавление корректных данных	-	MSI Geforce RTX 3090, 40000	MSI Geforce RTX 3090, 40000	Запись добавлена
Поиск					

4	Запись не существует	-	Geforce RTX	-	Совпадения не найдены, сравнений: 4
5	Запись существует	MSI Geforce RTX 3090, 40000	MSI Geforce RTX 3090	MSI Geforce RTX 3090, 40000	Найдены совпадения, сравнений: 1
Удаление					
7	Запись не существует	-	MSI Geforce RTX 3090	-	-
8	Запись существует	MSI Geforce RTX 3090	MSI Geforce RTX 3090	-	Запись удалена

## **Заключение**

Целью курсового проекта было: разработать информационную систему для автоматизации работы со справочниками «Конструктор ПК».

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

1) Проведён анализ предметной области «Конструктор ПК» и построена её модель;

2) Изучены теоретические основы методов построения справочников;

3) Определены требования к информационной системе;

4) Информационная система была реализована и протестирована;

a) Изучен язык разработки C# версии 9.0;

b) Изучен .NET Framework 4.7.2;

c) Во время разработки, в качестве среды выполнения, была использована Visual Studio Community 2019.

## Список литературы

1. Т. Х. Кормен, Ч. И. Лейзерсон, Р. Л. Ривест, К. Штайн. Алгоритмы: построение и анализ, 3-е изд.: Пер. с англ. – М.: ООО «И.Д.Вильямс», 2013. – 1328 с.: ил. парал. тит. англ.
2. Н. Вирт. Алгоритмы и структуры данных. Новая версия для Оберона. Издательство ДМК Пресс, 2010 год.