

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**

ФЕДОТОВ АРТЕМ ЮРЬЕВИЧ

РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ ДЛЯ АВТОМАТИЗАЦИИ РАБОТЫ СО СПРАВОЧНИКАМИ ОРГАНИЗАЦИИ, ПРЕДОСТАВЛЯЮЩЕЙ УСЛУГИ

**КУРСОВОЙ ПРОЕКТ**

по дисциплине «Фундаментальные структуры данных и алгоритмы»

по образовательной программе подготовки бакалавров по направлению

09.03.04 - Программная инженерия

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | |  |  | Студент гр. Б8118-09.03.04прогин | | | |
|  |  |  | | Федотов А. Ю. | |
|  | | | | | | |  |  | (подпись) | |  | |
|  | | | | | | | | | | | | |
| Защищен с оценкой | | | | | | |  |  | Руководитель | | |  |
|  | | | | | | |  |  | Ученая степень  ст. преподаватель | | | |
|  | | | |  |  | |  |  | должность |  | О.А. Крестникова | |
| (подпись) | | | |  | (И.О. Фамилия) | |  |  | (подпись) |  | (И.О. Фамилия) | |
| « |  | » |  | | | 2020 г. |  |  |  | | | |
|  | | | | | | | | | | | | |

г. Владивосток

2020

# Оглавление

[Оглавление 2](#_Toc54680020)

[Введение 3](#_Toc54680021)

[1 Анализ предметной области (ПО) 4](#_Toc54680022)

[1.1 Модель ПО 4](#_Toc54680023)

[1.2 Постановки задач обработки 7](#_Toc54680024)

[2 Теоретическая часть 9](#_Toc54680025)

[2.1 Хеш-таблица 9](#_Toc54680026)

[2.1.1 Хеш-функция 9](#_Toc54680027)

[2.1.2 Разрешение коллизий методом открытой адресации 10](#_Toc54680028)

[2.2 Двоичное дерево поиска 11](#_Toc54680029)

[3 Требования к информационной системе 16](#_Toc54680030)

[3.1 Функциональные требования 16](#_Toc54680031)

[3.2 Требования к данным 18](#_Toc54680032)

[3.2.1 Требования к входным данным 18](#_Toc54680033)

[4 Реализация 20](#_Toc54680034)

[Заключение 21](#_Toc54680035)

[Список литературы 22](#_Toc54680036)

# Введение

При ведении складской, или связанной с ней логистической деятельности, возникает необходимость в структурировании и автоматизации подобной информации. В данной курсовой работе была разработана информационная система для хранения и обработки данных о поставщиках и товарах, поступающих и хранящихся на складе.

Целью курсового проекта является: разработка информационной системы для автоматизации работы со справочниками (списками) товаров и поставщиков.

Для достижения поставленной цели необходимо выполнить следующие задачи:

1. Провести анализ предметной области 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 склада. Система должна решать следующие задачи:

1) хранить информацию о товарах и поставщиках;

2) позволять просматривать всю информацию о товарах и поставщиках;

3) позволять добавлять информацию о товарах и поставщиках;

4) позволять удалять информацию о перевозках и транспортных средствах;

5) позволять искать информацию о товарах и поставщиках;

6) на основании информации в справочниках формировать отчеты:

* о всех товарах поставщика
* о всех поставщиках товара

7) предусмотреть сортировку данных в отчетах:

8) предусмотреть проверку целостности информации, представленной в справочниках «Поставщики» и «Товары».

# 1.1 Модель ПО

Предметная область – склад.

Профессионал предметной области – кладовщик.

**Объекты предметной области:**

Объект Поставщики товара – информация о нем представляется в справочнике, который содержит информацию по каждому поставщику, который поставляет товары на склад.

Объект Поставщик – информация о нем содержит: название поставщика, номер телефона, категория поставляемого товара.

Название поставщика – название организации или имя физического лица, осуществляющих поставки товара на склад. Объект представляет собой набор букв русского и (или) латинского алфавита. Длина набора не превышает 15 символов.

Номер телефона – последовательность цифр, присвоенная абоненту телефонной сети, зная которую можно ему позвонить. Объект представляет собой строку из цифр без пробелов и иных знаков. Длина строки не превышает 15 символов.

Категория товара - отличительная особенность одного товара от другого. Согласно одной из существующих классификаций товаров, в курсовой работе будет использована классификация товаров в зависимости от ритма совершения покупки: 1 - товары повседневного спроса. Товары, которые потребитель обычно потребляет повседневно, покупает часто, без раздумий, например, газета, мыло. 2 - товары предварительного выбора. Товары, которые потребитель в процессе выбора и покупки, как правило, сравнивает между собой по показателям пригодности и качества, например, мебель, одежда, автомобиль, электробытовые приборы. 3 - товары особого спроса. Товары с уникальными характеристиками и (или) отдельные марочные (редкие) товары. 4 - товары для экстренных случаев. Такие товары покупают при возникновении острой нужды в них, например, зонты, сапоги, лопаты.

Объем товара – единица измерения количества товара, хранящегося на складе. Измеряется в единицах (количество объектов). Объект представлен целым положительным числом в диапазоне от 1 до 10000.

На рисунке 1 представлен пример справочника, который содержит информацию по каждому поставщику.



Рисунок 1 - Справочник поставщиков

Объект Товары поставщиков – информация о нем представляется в справочнике, который содержит информацию по каждому товару.

Объект Товар - информация о нем содержит: название товара, поставщик товара, объем (количество единиц) товара.

Название товара - реальное (существующее) наименование, соответствующее конкретному товару. Объект представляет собой набор букв русского и (или) латинского алфавита, первая буква является заглавной, остальные – строчными. Длина набора не превышает 30 символов.

Название поставщика – название организации или имя физического лица, осуществляющих поставки товара на склад. Объект представляет собой набор букв русского и (или) латинского алфавита. Длина набора не превышает 15 символов.

Объем товара – единица измерения товара, выраженная в количестве конкретного товара, находящегося на складе. Измеряется в единицах. Объект представлен целым положительным числом в диапазоне от 1 до 10000.

На рисунке 2 представлен пример справочника, который содержит информацию о каждом товаре.



Рисунок 2 - Справочник товаров

# 1.2 Постановки задач обработки

На основе входных данных необходимо автоматизировать составление отчетов, которые будут нужны для анализа имеющихся данных. Ниже приведены примеры некоторых вариантов таких отчетов, которые должны автоматически выводиться программным средством, когда необходимо.

Формирование отчета - все товары поставщика.

Входные данные:

1) справочник поставщиков:

* поставщик
* номер телефона
* категория товара

2) справочник товаров:

* название товара
* поставщик
* объем

Выходные данные:

Справочник, который содержит информацию о каждом товаре поставщика:

* название товара
* объем

Связь:

Справочник\_поставщиков.Название\_Поставщика=Справочник\_Товаров.Название\_Поставщика

Формирование отчета - все поставщики товара.

Входные данные:

1) справочник товаров:

* название товара
* поставщик
* объем

2) справочник поставщиков:

* поставщик
* номер телефона
* категория товара

Выходные данные:

Справочник, который содержит информацию о каждом поставщике товара:

* название поставщика
* номер телефона
* категория товара

Связь:

Справочник\_Товаров.Название\_Поставщика=Справочник\_Поставщиков.Название\_Поставщика

# 2 Теоретическая часть

В процессе обработки больших объемов данных, необходимо, чтобы выполнение различных запросов не занимало много времени: поиск информации об объектах был быстрым, а занимаемая память меньше. Опираясь на перечисленные требования, для дальнейшей работы были выбраны к рассмотрению такие структуры данных, как хеш-таблица (для хранения данных о поставщиках) и бинарное дерево поиска (для хранения данных о товарах).

# 2.1 Хеш-таблица

Хеш-таблица – структура данных, реализующая интерфейс ассоциативного массива. В отличие от деревьев поиска, реализующих тот же интерфейс, обеспечивают меньшее время отклика в среднем. Представляет собой эффективную структуру данных для реализации словарей, а именно, она позволяет хранить пары (ключ, значение) и выполнять три операции: операцию добавления новой пары, операцию поиска и операцию удаления пары по ключу. Среднее время поиска в хеш-таблице составляет О(1).

# 2.1.1 Хеш-функция

Хеш-функция – это некоторая функция h(K), которая берет ключ K и возвращает адрес, по которому производится поиск в хеш-таблице, чтобы получить информацию, связанную с ключом K. Преобразование, производимое хеш-функцией, называется хешированием. В нашем случае, с использованием хэш-таблицы, хэш от хранимого ключа будет являться номером ячейки.

Например, K – название поставщика в справочнике поставщиков, а необходимая информация – номер телефона. Функция точно определит по какому адресу находится информация о необходимом поставщике.

Выполнение любой операции в хеш-таблице начинается с вычисления значения хеш-функции от ключа. Получающееся хеш-значение (хеш-код) h(K) играет роль индекса в массиве H. Затем, выполняемая операция (добавление, удаление или поиск) перенаправляется объекту, который хранится в соответствующей ячейке массива H[h(K)].

При добавлении элемента в хеш-таблицу, алгоритм вставки проверяет ячейки в некоторой последовательности до тех пор, пока не будет найдена первая свободная ячейка, в которую будет записан новый элемент. Алгоритм поиска просматривает ячейки хеш-таблицы в той же последовательности, что и при вставке, до тех пор, пока не найдется либо элемент с искомым ключом, либо свободная ячейка (что означает отсутствие элемента в хеш-таблице), либо пока количество сравнений не превысит размер хеш-таблицы (что также означает отсутствие искомого элемента, если хеш-таблица полностью заполнена).

Хорошая хеш-функция должна удовлетворять следующим требованиям:

1. Быстро вычисляться.

2. Минимизировать количество коллизий.

# 2.1.2 Разрешение коллизий методом открытой адресации

В большинстве случаев несколько ключей могут хешироваться одинаково, но в хеш-таблице они должны занимать разные ячейки. В связи с этим, необходимо реализовать метод разрешения коллизии для поиска различных ячеек при совпадающих хеш-значениях. Наиболее распространенными являются метод цепочек и метод открытой адресации. Рассмотрим на примере таблицы поставщиков реализацию метода отрытой адресации. На рисунке 3 возникла коллизия при добавлении поставщика со значениями {Алстеко; 824412714; 1}, хеш-функция имела значение 14 для ключа “Алстеко”. Так как ячейка с индексом 14 занята, выполняется просмотр таблицы до ближайшего свободного места. В данном случае это следующая ячейка.

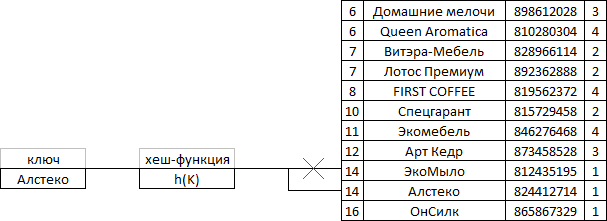


Рисунок 3 - Разрешение коллизий

Преимущество открытой адресации заключается в том, что она позволяет полностью отказаться от указателей. Вместо того чтобы следовать по указателям, мы вычисляем последовательность проверяемых ячеек, в рассматриваемом случае, указанным в п.2.1.1 способом. Дополнительная память, освобождающаяся в результате отказа от указателей, позволяет использовать хеш-таблицы большего размера при том же общем количестве памяти, потенциально приводя к меньшему количеству коллизий и более быстрой выборке.

# 2.2 Двоичное дерево поиска

Деревья поиска представляют собой структуры данных, которые поддерживают операции поиска по значению, поиск минимума и максимума, добавление и удаление, поиск предшествующего и последующего элемента (узла). Двоичное дерево поиска (ДДП, англ. binary search tree, BST) — иерархическая структура данных, для которой выполняются следующие дополнительные условия (свойства дерева поиска): оба поддерева: левое и правое — являются двоичными деревьями поиска; у всех узлов левого поддерева произвольного узла X значения ключей данных меньше, нежели значение ключа данных самого узла X; у всех узлов правого поддерева произвольного узла X значения ключей данных больше либо равно, нежели значение ключа данных самого узла X. Дерево поиска может использоваться и как словарь, и как очередь с приоритетами. Основные операции в бинарном дереве поиска выполняются за время, пропорциональное его высоте. Для полного бинарного дерева с n узлами эти операции выполняются за время O(lg n) в наихудшем случае. Однако, если дерево представляет собой линейную цепочку из n узлов, те же операции выполняются в наихудшем случае за время O(n). Теперь, рассмотрим алгоритмы перечисленных выше операций при работе с двоичным деревом поиска. Для поиска узла с заданным ключом в бинарном дереве поиска используется процедура поиска, которая получает в качестве параметров указатель на корень бинарного дерева и ключ, а возвращает указатель на узел с этим ключом (если такой существует). Рассмотрим примеры выполнения запросов в бинарном дереве на примере таблицы справочника товаров, состоящей из следующих записей.

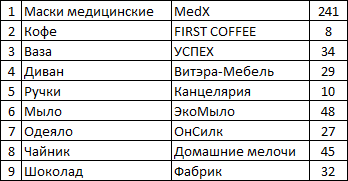


Рисунок 4 - Пример справочника товаров

Тогда бинарное дерево поиска при имеющихся данных будет выглядеть следующим образом, представленным на рисунке 5.

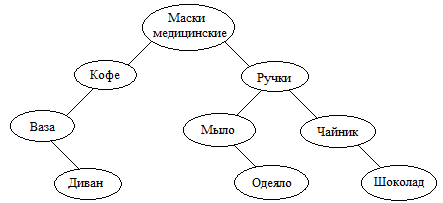


Рисунок 5 - Представление данных в виде бинарного дерева

Процедура поиска начинается с корня дерева и проходит вниз по пути к его потомкам. Например, при поиске элемента, ключ которого k = “Мыло”, для каждого встреченного на пути вниз узла X его ключ X.key сравнивается с переданным в качестве параметра ключом k = “Мыло”. Если ключи одинаковы, т. е. X.key = k, поиск завершается. Если k меньше X.key, поиск продолжается в левом поддереве X, иначе в правом. Так, для поиска заданного ключа необходимо пройти путь “Маски медицинские”, “Ручки”, “Мыло” от корня. Узлы, которые мы посещаем при рекурсивном поиске, образуют простой нисходящий путь от корня дерева, так что время работы процедуры поиска будет равно O(h), где h – высота дереве.

Для поиска элемента с минимальным значением ключа используется дочерние указатели left от корня дерева до тех пор, пока не встретится значение NIL. В качестве параметров в процедуру поиска минимального элемента подаются те же элементы, что и при поиске по заданному значению, но возвращаемое значение в данном случае будет представлено указателем на минимальный элемент поддерева с корнем в заданном узле X, который предполагается не равен NIL. Если у узла X нет левого поддерева, то, поскольку все ключи в правом поддереве X не меньше ключа X.key, минимальный ключ поддерева с корнем в узле x находится в узле X.key. Если же у узла X есть левое поддерево то, поскольку в правом поддереве не может быть узла с ключом, меньшим X.key, а все ключи в узлах левого поддерева не превышают x.key, узел с минимальным значением ключа в поддереве с корнем x находится в поддереве, корнем которого является узел X.left. Аналогичным образом работает процедура поиска максимального элемента в дереве, но при поиске используются дочерние указатели right и обход совершается по правому поддереву. Для вставки нового значения v в бинарное дерево поиска используется процедура вставки, в качестве параметров которой подаются узел z с атрибутами z.key=v, z.left=NIL, z.right=NIL. Процедура вставки изменяет некоторые поля z, что означает вставку z на конкретную позицию дерева. Процедура начинает работу с корня и проходит по простому нисходящему пути. Указатель X отмечает путь, проходимый в поисках пустого места для вставки NIL, который должен быть заменен входным элементом z. Указатели z.key и X.key сравниваются и в зависимости от результатов сравнения, указатель x перемещается вправо или влево до тех пор, пока не станет равным NIL.

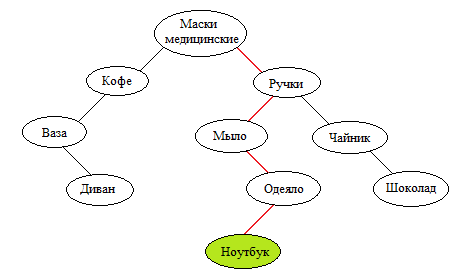


Рисунок 6 - Вставка нового элемента в бинарное дерево поиска

На рисунке приведен пример вставки нового элемента ({Ноутбук, Moscow Trading, 4}) с ключом k = ”Ноутбук” в бинарное дерево поиска. Красные линии указывают путь от корня вниз по дереву в позицию, в которую должен быть вставлен новый элемент.

Алгоритм удаления узла z из бинарного дерева поиска имеет три основные ситуации:

1. Если узел z не имеет дочерних узлов, то элемент удаляется, попутно внося изменения в родительский узел, а именно, заменяет дочерний указатель на z на NIL.

2. Если у z только один дочерний узел, то удаляется узел z и создается новая связь между родительским и дочерним узлами узла z.

3. Если у узла z два дочерних узла, то необходимо найти следующий за ним узeл y, который должен находится в правом поддереве z и который занимает в дереве место z. Остаток исходного правого поддерева z становится новым поддеревом y.

При использовании алгоритма удаления в процедуре удаления необходимо предусмотреть следующие различные случаи:

1) Если z не имеет левого дочернего узла, то необходимо заменить z его правым дочерним узлом, который может быть равен NIL. Если правый дочерний узел z не NIL, то значит, у узла z имеется единственный дочерний узел, являющийся правым дочерним узлом.

2) Если z имеет только один дочерний узел, являющийся его левым дочерним узлом, то необходимо заменить z его левым дочерним узлом.

3) В противном случае z имеет левый и правый дочерние узлы. Мы находим узел y, следующий за z. Этот узел располагается в правом поддереве z и не имеет левого дочернего узла. Мы хотим вырезать y у его текущего положения и заменить им в дереве узел z.

* Если y является правым дочерним узлом z, то необходимо заменить z на y, оставляя нетронутым правый дочерний по отношению к y узел.
* В противном случае y находится в правом поддереве узла z, но не является правым дочерним узлом z. В этом случае сначала необходимо заменить y его собственным правым дочерним узлом, а затем z на y.

# 3 Требования к информационной системе

# 3.1 Функциональные требования

Информационная система для автоматизации работы со справочниками организации, предоставляющей логистические услуги, должна позволять:

• Хранить информацию о Справочнике поставщиков и Справочнике товаров, используя структуры данных, необходимость которых была обоснована в главе 2 данной работы. Хранение информации о списках должно производиться непосредственно во время работы ПО.

• Позволять просматривать информацию, содержащуюся в Справочнике поставщиков и Справочнике товаров, используя оконный интерфейс ПО. Вывод осуществляется в оконном интерфейсе. Вывод каждого списка происходит в соответствии со следующими требованиями:

1. Таблица, содержащая информацию из Справочника поставщиков, имеет фиксированное число столбцов в количестве 5. Первый столбец представляет порядковые номера поставщиков для удобства подсчета их общего количества. Второй столбец содержит хеш-адреса (индексы) объектов хеш-таблицы, полученные при выполнении хеш-функции. Третий столбец содержит название поставщика. Четвертый столбец представлен номером телефона поставщика. Пятый столбец представляет категорию поставляемых поставщиком товаров.

2. Таблица, содержащая информацию из Справочника товаров, которые поставляют поставщики, имеет фиксированное число столбцов в количестве 4. Первый столбец представляет порядковые номера товаров для удобства подсчета их общего количества. Второй столбец содержит названия товаров, отсортированных в такой последовательности, в которой они представлены в бинарном дереве поиска. Третий столбец содержит названия поставщиков, которые поставляют конкретный товар из предыдущего столбца.

Четвертый столбец представлен объемом (количеством) товара, указанного в столбце 2.

• Позволять добавлять информацию об объектах из Справочника поставщиков и Справочника товаров обособлено. Добавление осуществляется через основной интерфейс оконного приложения при помощи соответствующих подписанных кнопок. Выполнять проверку на добавление повторяющейся информации: не могут быть добавлены объекты, содержащие информацию об Поставщиках с одинаковыми полями «Название\_поставщика»;

• Позволять удалять информацию об объектах из Справочника поставщиков и Справочника товаров обособленно. Удаление осуществляется через основной интерфейс оконного приложения при помощи соответствующих подписанных кнопок.

• Позволять выполнять поиск объектов из Справочника поставщиков и Справочника товаров с дальнейшим выводом информации об объектах, советующих условиям поиска в оконный интерфейс. Поиск должен выполняться через интерфейс оконного приложения с использованием специальных полей для ввода выбора.

• Позволять сохранять информацию, содержащуюся в списках текущей сессии в текстовые файлы отдельно для поставщиков и товаров с расширением «.txt». Сохранение происходит автоматически при изменении данных.

• Читать информацию из файлов c расширением «.txt». Если таковых нет, они создадутся автоматически. Чтение происходит автоматически, при загрузке программы, при условии, что файлы существуют.

• На основании информации из справочников в рамках текущей сессии формировать отчетность в следующем виде:

1. Формирование отчета - все товары поставщика

2. Формирование отчета - все поставщики товара

Описание формирования отчетов представлено в главе 1. Отчёты должны быть выведены в новом окне в виде таблиц. Отчеты выводятся по нажатию соответствующей подписанной кнопки в оконном приложении.

# 3.2 Требования к данным

# 3.2.1 Требования к входным данным

Основываясь на анализе ПО, входными данными для работы со справочниками являются два файла с расширениями «.txt». Первый - для поставщиков, второй - для товаров. Файлы не содержит пустых строк. Оба файла содержат информацию только о своих справочниках. Файл поставщиков имеет следующую структуру:

1. Строки файла поставщиков содержат данные Справочника Поставщиков в следующем виде:

Домашние мелочи/898612028/3

Витэра-Мебель/828966114/2

FIRST COFFEE/819562372/4

Лотос Премиум/892362888/2

2. Аналогичным способом хранятся остальные данные Справочника Товаров:

Маски медицинские/MedX/241

Кофе/FIRST COFFEE/8

Ваза/УСПЕХ/34

Диван/Витэра-Мебель/29

Данные об объектах записаны построчно в файле, без пустых строк. Необходимым требованием является корректность введённых данных. Данные должны быть записаны через слэш “/” и в определенном порядке.

1. Для Справочника Поставщиков:

* «Название\_Поставщика»/«Номер\_Телефона»/«Категория\_Товара»

1. Для Справочника Товаров:

* «Название\_Товара»/«Название\_Поставщика»/«Объем\_Товара»

При невыполнение какого-либо из перечисленных условий программа выполняется некорректно или не выполняется.

Пример файла, содержащего информацию о поставщиках – файл «Поставщики.txt» на рисунке 7:

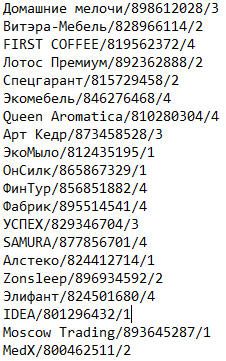


Рисунок 7 - Содержание файла «Поставщики.txt»

Пример файла, содержащего информацию о поставщиках – файл «Товары.txt» на рисунке 8:

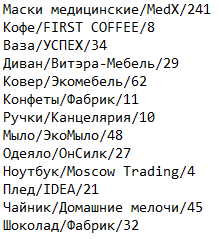


Рисунок 8 - Содержание файла «Товары.txt»

# 3.3 Требования к интерфейсу

Интерфейс должен быть реализован в виде оконного приложения, которое соответствует ряду требований:

• Добавление, удаление, поиск и демонстрация должны производиться путём нажатия специально отведенных для этого кнопок с дальнейшим вызовом соответствующих окон для ввода данных.

• При демонстрации списки должны отображаться в основной части окна в виде таблицы, требования к которой указаны в 3.2.3.

• Интерфейс должен быть реализован на русском языке.

• Вывод сообщений об ошибках производится в отдельных окнах.

• Сохранение отчётов должно быть реализовано по нажатию специальных кнопок на панели меню.

• Должна быть реализована кнопка сворачивания окна в панель задач, а также кнопка выхода из приложения.

# 4 Реализация

# 4.1 Описание классов

**Класс GlobalVariables** – класс, описывающий глобальные переменные.

**Поля:**

**treefilepath** – путь к файлу “Товары.txt”;

**hashfilepath** – путь к файлу “Поставщики.txt”;

**treesize** – переменная, хранящая количество элементов бинарного дерева поиска;

**hashsize** – переменная, хранящая количество элементов (не равных null) хеш-таблицы;

**hashMaxSize** – переменная, хранящая размер хеш-таблицы;

**gridCounter** – переменная, используемая при прохождении по строкам таблицы DataGridView поставщиков и товаров;

**Класс HashData** – класс, описывающий хеш-таблицу.

**Поля:**

**Key** – переменная, хранящая результат применения хеш-функции;

**Post** – переменная, хранящая название поставщика;

**Phone** – переменная, хранящая номер телефона;

**Kat** – переменная, хранящая категорию товара;

**Empty** - логическая переменная, указывающая, пуста ячейка хеш-таблицы или нет;

**Deleted** - логическая переменная, указывающая, удалялась информация в ячейке или нет;

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

**Поля:**

**masTovar** – переменная, хранящая название товара в массиве;

**masPost** – переменная, хранящая название поставщика в массиве;

**masObiem** – переменная, объем товара в массиве;

**Tovar** – переменная, хранящая название товара в БДП;

**Post** - переменная, хранящая название поставщика в БДП;

**Obiem** - переменная, хранящая объем товара в БДП;

**Left** - левый потомок элемента:

**Right** - правый потомок элемента;

**Parent** - родитель элемента;

**Root** - корень дерева;

**Методы:**

**Insert (string insTovar, string insPost, int insObiem)** - вставка элемента в дерево (построение дерева по товару);

**1.Входные данные:**

**• insTovar** – строка, содержащая название товара;

**• insPost** – строка, содержащая название поставщика;

**• insObiem** – число, объем товара;

**2.Выходные данные:**

**• tree** – дерево, с добавленными элементами.

**InsertPost (string insTovar, string insPost, int insObiem)** - вставка элемента в дерево (построение дерева по поставщику);

**1.Входные данные:**

**• insTovar** – строка, содержащая название товара;

**• insPost** – строка, содержащая название поставщика;

**• insObiem** – число, объем товара;

**2.Выходные данные:**

**• tree** – дерево, с добавленными элементами.

**TreeObhod(ref TreeData.TreeRoot treeRoot)** - обход дерева для получения количества его элементов;

**1.Входные данные:**

**• treeRoot** - дерево, которое нужно обойти;

**2.Выходные данные:**

**• GlobalVariables.treesize** – число, количество элементов дерева.

# 4.2 Описание интерфейса

Изображение окна программы представлено на рисунке ниже.

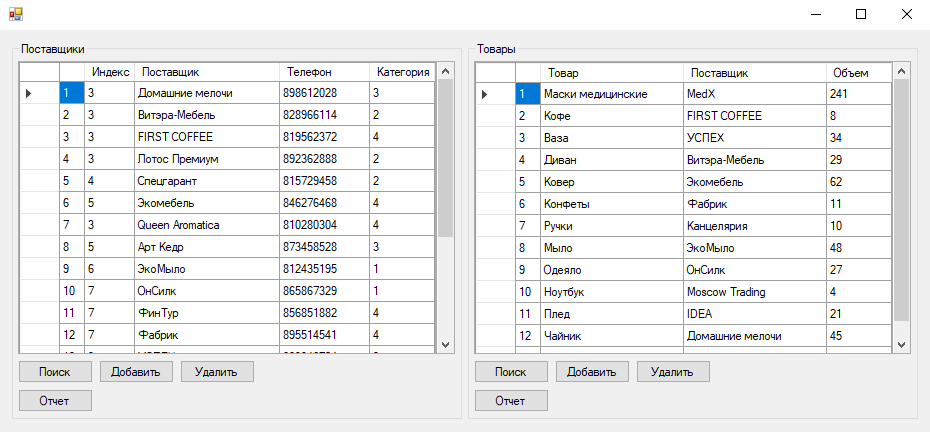


Рисунок 9 - Интерфейс программы

# 4.2.1 Загрузка данных из файла

Загрузка данных из файла происходит автоматически при запуске программы. По умолчанию файлы находятся на Локальном диске (C). В случае, если файлы не будут обнаружены в данной директории они создадутся. Если при чтении файлов будет обнаружена ошибка, программа сообщит об этом.

4.2.2 Добавление поставщика

Для добавления нового поставщика, пользователю необходимо нажать кнопку Добавить под соответствующей таблицей. После чего он увидит форму (Рисунок ниже), в которой ему будет предложено добавить новый объект Поставщик, заполнив поля этого объекта, затем нажать в форме кнопку Ок.

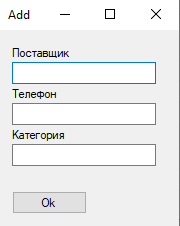


Рисунок 9 - Добавление поставщика

4.2.3 Удаление поставщика

Для удаления поставщика необходимо нажать кнопку Удалить под соответствующей таблицей. После чего появится форма (Рисунок ниже), в которой необходимо ввести название объекта Поставщик, заполнив поле, затем нажать в форме кнопку Ок.

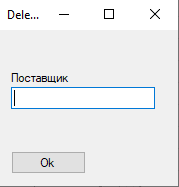


Рисунок 10 - Удаление поставщика

# 4.2.4 Поиск поставщика

Для поиска поставщика необходимо нажать кнопку Поиск под соответствующей таблицей. После чего появится форма (Рисунок ниже), в которой необходимо ввести название объекта Поставщик, заполнив поле, затем нажать в форме кнопку Ок.

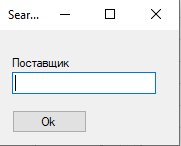


Рисунок 11 - Поиск поставщика

# 4.2.5 Добавление товара

Для добавления нового товара, пользователю необходимо нажать кнопку Добавить под соответствующей таблицей. После чего он увидит форму (Рисунок ниже), в которой ему будет предложено добавить новый объект Поставщик, заполнив поля этого объекта, затем нажать в форме кнопку Ок.

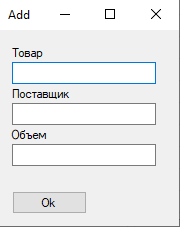


Рисунок 13 - Добавление товара

# 4.2.6 Удаление товара

Для удаления товара необходимо нажать кнопку Удалить под соответствующей таблицей. После чего появится форма (Рисунок ниже), в которой необходимо ввести название объекта Товар, заполнив поля Товар и Объем, затем нажать в форме кнопку Ок.

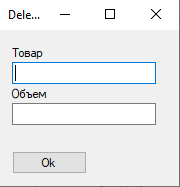


Рисунок 14 - Удаление товара

# 4.2.7 Поиск товара

Для поиска товара необходимо нажать кнопку Поиск под соответствующей таблицей. После чего появится форма (Рисунок ниже), в которой необходимо ввести название объекта Товар, заполнив поле, затем нажать в форме кнопку Ок.

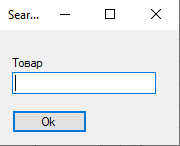


Рисунок 15 - Поиск товара

# 4.2.8 Отчёт «Товары поставщика»

Для составления отчёта «Информация о дисциплине в расписании» пользователю необходимо в верхнем выпадающем меню выбрать пункт Информация о дисциплинах в расписании (Рисунок ниже), после чего перед ним появится форма(рисунок ниже), где ему необходимо будет в текстовое поле ввести Название дисциплины, о которой он хочет составить отчёт и нажать кнопку сформировать отчёт, после чего ему будут выведен результат отчёта или сообщений об ошибке. Если пользователь хочет сохранить отчёт, то он может нажать кнопку Сохранить отчёт.

Рисунок 16 - Отчет товары поставщика

# 4.2.9 Отчёт «Поставщики товара»

Для составления отчёта «Преподавательский состав дисциплины по расписанию» пользователю необходимо в верхнем выпадающем меню выбрать пункт Преподавательский состав дисциплины по расписанию (Рисунок ниже), после чего перед ним появится форма (Рисунок ниже), где ему необходимо будет в текстовое поле ввести название дисциплины и нажать кнопку сформировать отчёт, после чего ему будут выведен результат отчёта или сообщений об ошибке. Если пользователь хочет сохранить отчёт, то он может нажать кнопку Сохранить отчёт.

Рисунок 17 - Отчет поставщики товара

# Заключение

Целью курсового проекта было: разработать информационную систему для автоматизации работы со справочниками склада, содержащего информации об поставщиках и товарах.

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

1) Проведён анализ предметной области и построена её модель;

2) Изучены теоретические основы методов построения справочников;

3) Определены требования к информационной системе;

4) Проведено тестирование;

В ходе выполнения работы были изучены язык разработки C# версии 8.0 и .NET Framework версии 4.7.2. Во время разработки, в качестве среды выполнения, была использована Visual Studio Community 2019.