

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**

АГАПОВА ДАРЬЯ РУСЛАНОВНА

РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ ДЛЯ АВТОМАТИЗАЦИИ РАБОТЫ СО СПРАВОЧНИКАМИ ПРЕДМЕТНОЙ ОБЛАСТИ ПРОЕКТА «YOUR SEWING PATTERN»

**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 | | | |
|  |  |  | | Д.Р. Агапова | |
|  | | | | | | |  |  | (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

[Оглавление 2](#_Toc75939526)

[Введение 3](#_Toc75939527)

[1 Анализ предметной области (ПО) 4](#_Toc75939528)

[1.1 Модель ПО 4](#_Toc75939529)

[1.2 Постановки задач обработки 7](#_Toc75939530)

[2 Теоретическая часть 10](#_Toc75939531)

[2.1 Хеш-таблица 10](#_Toc75939532)

[2.1.1 Хеш-функция 10](#_Toc75939533)

[2.1.2 Разрешение коллизий методом цепочек 12](#_Toc75939534)

[2.1.3 Неупорядоченный двусвязный кольцевой список 13](#_Toc75939535)

[2.2 Бинарное дерево поиска 13](#_Toc75939536)

[3 Требования к информационной системе 18](#_Toc75939537)

[3.1 Функциональные требования 18](#_Toc75939538)

[3.2 Требования к данным 19](#_Toc75939539)

[3.2.1 Требования к входным данным 19](#_Toc75939540)

[3.2.1 Требования к выходным данным 21](#_Toc75939541)

[3.3 Требования к интерфейсу 22](#_Toc75939542)

[4 Реализация 23](#_Toc75939543)

[4.1 Диаграмма классов 23](#_Toc75939544)

[4.2 Описание классов 23](#_Toc75939545)

[4.3 Описание интерфейса 30](#_Toc75939546)

[4.4 Тестирование 32](#_Toc75939547)

[Заключение 36](#_Toc75939548)

[Список литературы 37](#_Toc75939549)

# Введение

В области шитья очень часто возникает вопрос о хранении информации о категории одежды и нужных мерках и удобного доступа к ней, что послужило выбором данной предметной области, поэтому в рамках курсового проекта будет рассмотрена информационная система для ведения справочников одежды и их мерок.

Целью курсового проекта является: разработка информационной системы для автоматизации работы со справочниками одежды, хранящими информацию о названии и нужных мерках каждой категории.

1. Провести анализ предметной области, а именно области шитья 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 приложения, предоставляющим пользователю доступ к информации о категориях изделий, их мерках и его персонализированным данным.

Система должна решать следующие задачи:

1. хранить информацию о категориях изделий, их мерках и персонализированных данных пользователей;
2. позволять просматривать всю информацию о категориях изделий, их мерках и своих персонализированных данных;
3. позволять добавлять информацию о мерках изделия и своих персональных данных;
4. позволять удалять информацию о мерках изделия и своих персональных данных;
5. позволять искать информацию о категориях изделий и их мерках;
6. предусмотреть проверку целостности информации, представленной в справочниках категорий изделий, их мерках и персонализированных данных пользователей.

1.1 Модель ПО

Предметная область – пошив одежды.

Профессионал предметной области – швея.

**Объекты предметной области:**

Объект **Список одежды** – информация о нем представляется в справочнике, который содержит информацию о категории каждого отдельного вида одежды.

Объект **Изделие** – информация о нем содержит: категорию одежды, её название.

**Категория изделия** – строка, состоящая из прописных символов кириллицы. Длина строки должна состоять из одного слова и не превышать 30 символов.

**Название** **изделия** – строка, состоящая из прописных символов кириллицы. Слова могут разделяться между собой пробелом или тире. Длина строки не должна превышать 30 символов.

Строка, состоящая из комбинации строк «Категория одежды» и «Название», является ключом, то есть оно уникально.

Пример справочника, который содержит информацию по каждой категории одежды:

ЮБКА СОЛНЦЕ ЮБКА

ЮБКА ПАЧКА ЮБКА

ЮБКА КАРАНДАШ ЮБКА

ЮБКА ГОДЕ ЮБКА

ЮБКА ТЮЛЬПАН ЮБКА

БРЮКИ КЛЕШ БРЮКИ

БРЮКИ КАРГО БРЮКИ

БРЮКИ ГАЛИФЕ БРЮКИ

БРЮКИ БАНАНЫ БРЮКИ

РУБАШКА ШЕМИЗЬЕ РУБАШКА

ПЛАТЬЕ ФУТЛЯР ПЛАТЬЕ

Объект **Избранные изделия пользователей** – информация о нем представляется в справочнике, который содержит информацию о логине пользователя, категории и названии изделия, мерках, требующихся для каждого вида одежды, мерках пользователя.

Объект **Избранная пользователем** – информация о нем содержит: логин пользователя, название изделия, категорию изделия, мерки изделия, индивидуальные мерки пользователя.

**Логин** – строка, начинающаяся с прописного символа латиницы и все остальные символы – прописные буквы алфавита латиницы. Так же допустимы арабские цифры. Логин так же является ключом, то есть оно должно быть уникальным! Длина строки не должна превышать 8 символов.

**Категория изделия** – строка, состоящая из прописных заглавных символов кириллицы. Длина строки должна состоять из одного слова и не превышать 30 символов.

**Название изделия** – строка, состоящая из прописных заглавных символов кириллицы. Слова могут разделяться между собой пробелом или тире. Длина строки не должна превышать 30 символов.

**Мерки изделия** – строка, начинающаяся с прописного символа латиницы и все остальные символы – прописные буквы алфавита латиницы. Так же допустимы пробелы. Длина строки не должна превышать 100 символов.

**Мерки пользователя** – строка из 11 натуральных чисел, разделенных пробелом и обозначающих размер разных частей тела в следующем порядке: полуобхват шеи, бедренный полуохват, ширина грудной клетки, ширина спины, высота проймы с задней части, длина спинной части до линии талии, высота плеча, ширина плечевого ската, длина рукава, окружность плеча, окружность запястья.

Пример справочника, который содержит информацию об избранных видах одежды пользователя:

Aitaaan ЮБКА ЮБКА КАРАНДАШ От Об Дсб Дсп Дсз Дю Дтб 60 100 90 120 31 110 40 52 89 33 22

Nikidsad ЮБКА ЮБКА ТРАПЕЦИЯ От Об Дсб Дсп Дсз Дю Дтб 70 110 97 110 31 113 40 52 83 34 23

NiShiGara ПЛАТЬЕ ПЛАТЬЕ ФУТЛЯР Ош Ош От Об Уб Г1 Шг2 Цг Шп Шс Дтс Дтс1 Дтп Впк Впрз 87 120 80 130 31 114 40 52 81 44 24

TyU ЮБКА ЮБКА КАРАНДАШ От Об Дсб Дсп Дсз Дю Дтб 67 110 85 122 31 117 40 52 84 44 27

Aitaaan ПЛАТЬЕ ПЛАТЬЕ ФУТЛЯР Ош Ош От Об Уб Г 1 Шг2 Цг Шп Шс Дтс Дтс1 Дтп Впк Впрз 60 100 90 120 31 110 40 52 89 33 22

Eloi ЮБКА ЮБКА КАРАНДАШ От Об Дсб Дсп Дсз Дю Дтб 60 100 90 123 51 140 40 92 39 33 22

RTu23 ЮБКА ЮБКА ТРАПЕЦИЯ От Об Дсб Дсп Дсз Дю Дтб 70 110 117 190 31 123 40 32 83 34 23

ShiZaoka ЮБКА ЮБКА ГОДЕ Ст Сб Дсп Дзб Диз 87 120 80 130 31 114 40 52 81 44 24

Pingiuin БРЮКИ БРЮКИ БАНАНЫ Дсш От Об Дб Дбш 67 110 85 122 31 117 40 52 84 44 27

Aitaaan РУБАШКА РУБАШКА ШЕМИЗЬЕ Ош Ош От Об Уб Г 1 Шг2 Цг Шп Шс Дтс Дтс1 Дтп Впк Впрз 62 103 111 120 31 110 40 52 89 33 22

1.2 Постановки задач обработки

Поиск в справочнике «Список одежды».

Входные данные:

1. **Список одежды**
2. **Название категории изделия**
3. **Название изделия**

Выходные данные: **Список одежды**

Связь:

Список одежды.Изделие.Категория изделия = Категория

Список одежды.Изделие.Название изделия = Название

Поиск в справочнике «Избранные изделия пользователей».

Входные данные:

1. **Список избранных изделий пользователей**
2. **Название категории изделия**
3. **Название изделия**

Выходные данные: **Список избранных изделий пользователей**

Связь:

Список избранных изделий пользователей.Избранное пользователя.Категория && Список избранных изделий пользователей.Избранное пользователя. Название = Категория && Название

Проверка целостности при добавлении в справочник «Список одежды».

Входные данные:

1. **Список одежды**
2. **Название категории изделия**
3. **Название изделия**

Выходные данные: **Список одежды**

Связь:

Не существует (Список одежды.Изделие.Категория = Категория && Список одежды.Изделие.Название = Название)

Проверка целостности при удалении из справочника «Список одежды».

Входные данные:

1. **Список одежды**
2. **Название категории изделия**
3. **Название изделия**

Выходные данные: **Список одежды и список избранных изделий пользователей**

Связь:

Не существует (Избранные изделия пользователей.Избранное.Категория && Избранные изделия пользователей.Избранное.Название = Список одежды.Изделие)

Проверка целостности при удалении из справочника «Избранные изделия пользователей».

Входные данные:

**1) Список избранных изделий пользователей**

**2) Логин**

**3) Категория изделия**

**4) Название изделия**

**5) Мерки изделия**

**6) Мерки пользователя**

Выходные данные: Список избранных изделий пользователей

Связь:

Существует Список избранных изделий пользователей.Избранное пользователя = Избранное пользователя.

# 2 Теоретическая часть

В рамках курсового проекта необходимо искать информацию о категориях изделий, их мерках и его персонализированных данных.

Для ускорения поиска используются такие структуры данных, например, как сбалансированные бинарные деревья поиска[1] (АВЛ-деревья, красно-черные деревья, B-деревья и т.д.) и хеш-таблицы.

Деревья поиска представляют собой структуры данных, которые поддерживают многие операции с динамическими множествами, включая Search, Minimum, Maximum, Predecessor, Successor, Insert и Delete. Основные операции в бинарном дереве поиска выполняются за время, пропорциональное его высоте. Для полного бинарного дерева с узлами эти операции выполняются за время в наихудшем случае. Однако, если дерево представляет собой линейную цепочку из узлов, те же операции выполняются в наихудшем случае за время .

Источник: Т. Х. Кормен, Ч. И. Лейзерсон, Р. 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.

2.1 Хеш-таблица

Хеш-таблица[1] представляет собой эффективную структуру данных для реализации словарей. Хеширование представляет из себя применение хеш-функции на ключах, поступающих к ней на вход, и преобразовании множества этих ключей в ячейки хеш-таблицы. Хотя на поиск элемента в хеш-таблицу может в наихудшем случае потребоваться – , на практике хеширование исключительно эффективно. При вполне обоснованных допущениях среднее время поиска элемента в хеш-таблице составляет .

Источник: Т. Х. Кормен, Ч. И. Лейзерсон, Р. 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.

2.1.1 Хеш-функция

Пусть элемент с ключом хранится в ячейке . При хешировании этот элемент хранится в ячейке , т.е. мы используем хеш-функцию[1] для вычисления ячейки данного ключа . Функция отображает совокупность ключей на ячейки хеш-таблицы :

где размер хеш-таблицы обычно гораздо меньше значения . Мы говорим, что элемент с ключом хешируется в ячейку ; величина называется хеш-значением ключа .

Рассмотрим пример построения хеш-функции на примере метода свертки. Построение хеш-функции этим методом состоит в отображении ключа k на одну из m ячеек путем получения остатка от деления k на m, где k – сумма всех цифр в обоих ключах. Тогда хеш-функция имеет вид:

В рамках курсового проекта ключом является совокупность двух полей: название категории изделия и название изделия. Для полей «Категория» и «Название» значение будет кодом соответствующего символа в UNICODE.

Предположим, что размер хеш-таблицы равен 10, а на вход поступают записи заявок из модели предметной области. Приведем пример:

В силу особенности построения нашей хеш-функции:

Проделываем тоже самое с остальными записями:

Как видно из примера, два ключа могут быть хешированы в одну и ту же ячейку, например, «ЮБКА ТРАПЕЦИЯЮБКА» и «». Такая ситуация называется коллизией. Имеются эффективные технологии разрешения конфликтов, вызываемых коллизиями. Например, линейное разрешение коллизий[2], двойное хеширование, на основе связанных списков, разрешение коллизий с помощью цепочек и т.д. Последний метод (п. 2.1.2) будет использоваться в данной работе.

Источник:

1. Т. Х. Кормен, Ч. И. Лейзерсон, Р. 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.
2. А. А. Навроцкий Основы алгоритмизации и программирования в среде Visual C++: учеб.-метод. пособие / А. А. Навроцкий. – Минск: БГУИР, 2014. – 160 с.: ил.

### **2.1.2 Разрешение коллизий методом цепочек**

При разрешении коллизий с помощью цепочек помещаются все элементы, хешированные в одну и ту же ячейку, в связанный список. Связанный список – это динамическая структура данных, состоящая из узлов, каждый из которых содержит как собственно данные, так и одну или две ссылки на следующий и/или предыдущий узел списка. Ячейка содержит указатель на заголовок списка всех элементов. Если таких элементов нет, ячейка содержит значение .

На рисунке 1 представлен пример хеш-таблицы с цепочкой на записях предметной области.



Рисунок 1 Пример хеш-таблицы с цепочкой на записях предметной области

### **2.1.3 Неупорядоченный двусвязный кольцевой список**

Двусвязный кольцевой список[1] – это структура данных, в которой объекты расположены в линейном порядке. Однако, в отличие от массива, в котором этот порядок определяется индексами, порядок в списке определяется указателями на каждый объект. Каждый элемент двусвязного кольцевого списка обладает одним атрибутом и двумя атрибутами-указателями (следующий) и (предыдущий), которые указывают на соответствующие элементы в списке. В кольцевом списке атрибут последнего в списке элемента указывает на первый элемент списка.

Пример двусвязного кольцевого списка приведен на рисунке 1

Источник: Т. Х. Кормен, Ч. И. Лейзерсон, Р. 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.

## **2.2 Бинарное дерево поиска**

Бинарное дерево поиска[1] может быть представлено с помощью связанной структуры данных, в которой каждый узел является объектом. В дополнение к атрибуту ключа и сопутствующим данным каждый узел содержит атрибуты , , , который указывают на левый и правый дочерние узлы и на родительский узел соответственно. Если дочерний или родительский узел отсутствуют, соответствующее поле равно . Единственный узел, указатель которого равен , – это корневой узел дерева.

Ключи в бинарном дереве поиска хранятся таким образом, чтобы в любой момент удовлетворять следующему свойству: пусть представляет собой узел бинарного дерева поиска, если является узлом в левом поддереве , то , если является узлом в правом поддереве , то .

Свойство бинарного дерева поиска позволяет вывести все ключи, находящиеся в дереве, в отсортированном порядке с помощью простого рекурсивного алгоритма, называемого центрированным (симметричным) обходом дерева (incorder tree walk). В таком алгоритме ключ в корне поддерева выводится между значениями ключей левого поддерева и правого поддерева. Имеются и другие способы обхода, а именно – обход в прямом порядке (preorder tree walk), при котором сначала выводится корень, а затем – значения левого и правого поддеревьев (именно этот обход будет использован в работе), и обход в обратном порядке (postorder tree walk), когда первыми выводятся значения левого и правого поддеревьев, а уже затем – корня.

Наиболее частой операцией, выполняемой с бинарным деревом поиска, является поиск в нем определенного ключа. Для поиска узла с заданным ключом в бинарном дереве используется алгоритм, который получает в качестве параметров указатель на корень бинарного дерева и ключ , а возвращает указатель на узел с этим ключом (если таковой существует; в противном случае возвращается значение ).

Процедура поиска начинается с корня дерева и проходит вниз по дереву. Для каждого встреченного на пути вниз узла его ключ сравнивается с переданным в качестве параметра ключом . Если ключи одинаковы, поиск завершается. Если меньше , поиск продолжается в левом поддереве . Узлы, которые мы посещаем при рекурсивном поиске, образуют простой нисходящий путь от корня дерева, так что время работы процедуры равно , где - высота дерева.

Элемент с минимальным значением ключа всегда можно найти, следуя по дочерним указателям left от корневого узла до тех пор, пока не встретится значение . Элемент с максимальным значением находится симметрично. Процедура находит минимальный (максимальный) элемент дерева за время , где - высота дерева.

Иногда для заданного узла в бинарном дереве поиска требуется определить, какой узел следует за ним в отсортированной последовательности, определяемой порядком центрированного обхода бинарного дерева, и какой узел предшествует данному. Если все ключи различны, последующим по отношению к узлу является узел с наименьшим ключом, большим . Структура бинарного дерева поиска позволяет найти этот узел, не выполняя сравнение ключей.

Если правое поддерево узла непустое, то следующий за элемент является крайним слева узлом в правом поддереве , который выявляется вызовом процедуры поиска минимального элемента.

С другой стороны, если правое поддерево узла пустое и у имеется следующий за ним элемент , то является наименьшим предком , левый наследник которого также является предком . Время работы процедуры в дереве высотой составляет . Процедура поиска предшествующего симметрична процедуре поиска последующего.

Для добавления нового значения в бинарное дерево поиска, мы воспользуемся следующим алгоритмом.

В качестве параметра используется узел , атрибуты которого , и . Алгоритм начинает работу с корневого узла деррева и проходит по простому нисходящему пути. Указатель отмечает путь, проходимый в поисках , который должен быть заменен входным элементом . Алгоритм поддерживает также замыкающий указатель , который представляет собой указатель на родительский по отношению к узел. После инициализации в цикле эти указатели перемещаются вниз влево или вправо по дереву, в зависимости от результата сравнения ключей , до тех пор, пока не станет равным . Это значение находится именно в той позиции, в которую следует поместить элемент . Замыкающий указатель нужен для того, чтобы знать, какой узел должен быть изменен.

Стратегия удаления узла из бинарного дерева поиска имеет три основные ситуации.

* Если у нет дочерних узлов, то мы просто удаляем его, внося изменения в его родительский узел, а именно – заменяя дочерний узел на .
* Если у только один дочерний узел, то мы удаляем узел создавая новую связь между родительским и дочерним узлами узла .
* Если у узла два дочерних узла, то мы находим следующий за ним узел , который должен находиться в правом поддереве и который занимает в дереве место . Остаток исходного правого поддерева становится новым поддеревом , а левое поддерево становится новым левым поддеревом . Это самый сложный случай, поскольку, как мы увидим, здесь играет роль, является ли правым дочерним узлом .

Процедура удаления данного узла из бинарного дерева поиска получает в качестве аргумента указатели на дерево и на . Она организована несколько иначе, чем описано ранее, и рассматривает не три, а пять случаев.

* Если z не имеет левого дочернего узла, то мы заменяем z его правым дочерним узлом, который может быть (или не быть) NIL. Если правый дочерний узел z представляет собой NIL, то мы оказываемся в ситуации, когда у узла z имеется единственный дочерний узел, являющийся правым дочерним узлом
* Если z имеет только один дочерний узел, являющийся его левым дочерним узлом, то мы заменяем z его левым дочерним узлом.
* В противном случае z имеет и 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 из его текущего положения и заменить им в дереве узел z.
* Если y является правым дочерним узлом z, то мы заменяем 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 мы сначала заменяем y его собственным правым дочерним узлом, а затем заменяем z на y.

На рисунке 2 представлен пример бинарного дерева поиска на записях предметной области.



Рисунок 2 Пример бинарного дерева поиска на записях предметной области

В рамках курсового проекта при удалении узла с двумя дочерними узлами, на его месте появится узел, который является минимальным справа (самый левый потомок правого поддерева), относительно удаляемого узла.

В каждом узле дерева будет храниться двусвязный кольцевой список с добавлением в конец. Пример двусвязного кольцевого списка приведен на рисунке 1.

Источник: Т. Х. Кормен, Ч. И. Лейзерсон, Р. 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.

# 3 Требования к информационной системе

## **3.1 Функциональные требования**

Информационная система для автоматизации работы со справочниками предметной области проекта «Your Sewing Pattern», должна позволять:

Система должна решать следующие задачи:

1. хранить информацию о категориях изделий, их мерках и персонализированных данных пользователей;
2. позволять просматривать всю информацию о категориях изделий, их мерках и своих персонализированных данных;
3. позволять добавлять информацию о мерках изделия и своих персональных данных (каждая запись уникальна, при добавлении дублирующей информации выводить соответствующее сообщение. При добавлении новой записи, в списке должны существовать записи, у которых поля категория изделия и название изделия, логин и мерки изделия и мерки пользователя, должны совпадать с соответствующими значениями, вводимые пользователем. В противном случае программа должна выводить сообщение об отсутствии таких данных);
4. позволять удалять информацию о мерках изделия и своих персональных данных (в случае, если такой записи не существует, выводить соответствующее сообщение. Если в списках существует запись, у которой поля логин, категория изделия, название изделия, мерки изделия и мерки пользователя совпадают с соответствующими значениями, вводимые пользователем, то удалить запись);
5. позволять искать информацию о категориях изделий и их мерках (в случае, если такой записи не существует, вывести соответствующее сообщение. Если в списках существует запись, у которой поля логин, категория изделия, название изделия, мерки изделия и мерки пользователя совпадают с соответствующими значениями, вводимые пользователем, то вывести соответствующее сообщение и количество сравнений);
6. предусмотреть проверку целостности информации, представленной в справочниках, содержащих избранные изделия пользователей (при добавлении записи в справочник «Список избранных изделий пользователей» допускается добавление только тех данных, поля название категории изделия и название изделия которых являются полями справочника «Список одежды», в противном случае выводить соответствующее сообщение; при удалении записи из справочника «Список одежды» удаляются записи из справочника «Список избранных изделий пользователей», поля название категории изделия и название изделия которых совпадают с полем название категории изделия и название изделия удаляемой записи).

## **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 CATEGORY.txt, каждая строка файла содержит информацию об объекте Список одежды, а именно поля название изделия, название категории изделия, разделенные символом «\_». Возможным разделителем внутри поля название изделия является « ».

Пример текстового файла:

ЮБКА КАРАНДАШ\_ЮБКА

ЮБКА КЛЕШ\_ЮБКА

ЮБКА ТРАПЕЦИЯ\_ЮБКА

ЮБКА СОЛНЦЕ\_ЮБКА

ЮБКА ПАЧКА\_ЮБКА

ЮБКА ГОДЕ\_ЮБКА

ЮБКА ТЮЛЬПАН\_ЮБКА

БPЮКИ КЛЕШ\_БРЮКИ

БРЮКИ КАРГО\_БРЮКИ

БРЮКИ ГАЛИФЕ\_БРЮКИ

БРЮКИ БАНАНЫ\_БРЮКИ

РУБАШКА ШЕМИЗЬЕ\_РУБАШКА

ПЛАТЬЕ ФУТЛЯР\_ПЛАТЬЕ

* Название изделия, название категории изделия (см. п.1.1);
* текстовый файл FAVORITES.txt, каждая строка файла содержит информацию об объекте Избранные изделия пользователей, а именно поля логин пользователя, название категории изделия, название изделия, мерки данного изделия и мерки пользователя, разделенные между собой символом «\_».

Пример текстового файла:

NiShiGara\_ПЛАТЬЕ\_ПЛАТЬЕ ФУТЛЯР\_/////\_87 120 80 130 31 114 40 52 81 44 24

George\_БРЮКИ\_БРЮКИ ГАЛИФЕ\_/////\_90 102 90 129 31 111 40 52 88 34 24

Olivia\_ЮБКА\_ЮБКА КАРАНДАШ\_/////\_90 102 90 129 31 111 40 52 88 34 24

George\_ЮБКА\_ЮБКА КАРАНДАШ\_/////\_90 102 90 129 31 111 40 52 88 34 24

WioSen\_ПЛАТЬЕ\_ПЛАТЬЕ ФУТЛЯР\_/////\_65 153 29 111 61 113 40 52 89 34 20

Olivia\_ПЛАТЬЕ\_КИМОНО\_/////\_68 103 89 121 31 113 40 52 89 34 20

Kleo\_ПЛАТЬЕ\_КИМОНО\_/////\_68 103 89 121 31 113 40 52 89 34 20

LauriQW\_ЮБКА\_ЮБКА СОЛНЦЕ\_/////\_86 109 79 143 31 118 40 52 89 34 28

Iori\_РУБАШКА\_РУБАШКА ШЕМИЗЬЕ\_/////\_60 100 90 120 31 110 40 52 89 33 22

Amalia23\_РУБАШКА\_РУБАШКА ШЕМИЗЬЕ\_/////\_60 100 90 120 31 110 40 52 89 33 22

Amalia23\_ЮБКА\_ЮБКА СОЛНЦЕ\_/////\_70 110 97 110 31 113 40 52 83 34 23

Amalia23\_ПЛАТЬЕ\_КИМОНО\_/////\_44 106 88 153 31 119 40 52 86 44 29

* Логин пользователя, название категории изделия, название изделия, мерки данного изделия и мерки пользователя (см. п.1.1):

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

Выходными данными для работы со справочниками являются:

* текстовый файл FAVORITES2.txt, каждая строка файла содержит информацию об объекте Список одежды, Список пользователей и Избранные изделия пользователей, а именно поля название изделия, название категории изделия, разделенные символом «\_», поля пользователь, мерки пользователя, разделенные символом «\_» и поля логин пользователя, название категории изделия, название изделия, мерки данного изделия и мерки пользователя, разделенные символом «\_», соответственно. Возможным разделителем внутри поля название изделия является « ».
* сообщения об ошибках должны выводиться в отдельной форме и содержать информацию об ошибке. Всевозможные сообщения об ошибках приведены ниже:

1. «Ошибка при чтении файла»;
2. «Проверьте корректность строки 'Категория' или 'Название'»;
3. «Такой ключ уже существует»;
4. «Проверьте корректность введенных строк»;
5. «Данные не сходятся со всеми справочниками»;
6. «Данные не сходятся со справочниками 'Категории' и 'Мерки'»;
7. «Данные не сходятся со справочниками 'Категории' и 'Пользователи'»;
8. «Данные не сходятся со справочником 'Категории'»;
9. «Такая запись уже существует»;
10. «Такого элемента нет»;
11. «Требуется выбрать строку», «Не выбрана строка»;
12. «Ничего не введено в поля 'Категория' или 'Название'»;
13. «Проверьте корректность введенных данных в файле. Количество недобавленных строк – »;

## **3.3 Требования к интерфейсу**

Интерфейс с пользователем должен быть оконный, содержащий интуитивно понятные переходы для выполнения задач пользователя. Русская локализация, все выводимые сообщения не должны содержать орфографических ошибок.

# 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 3.

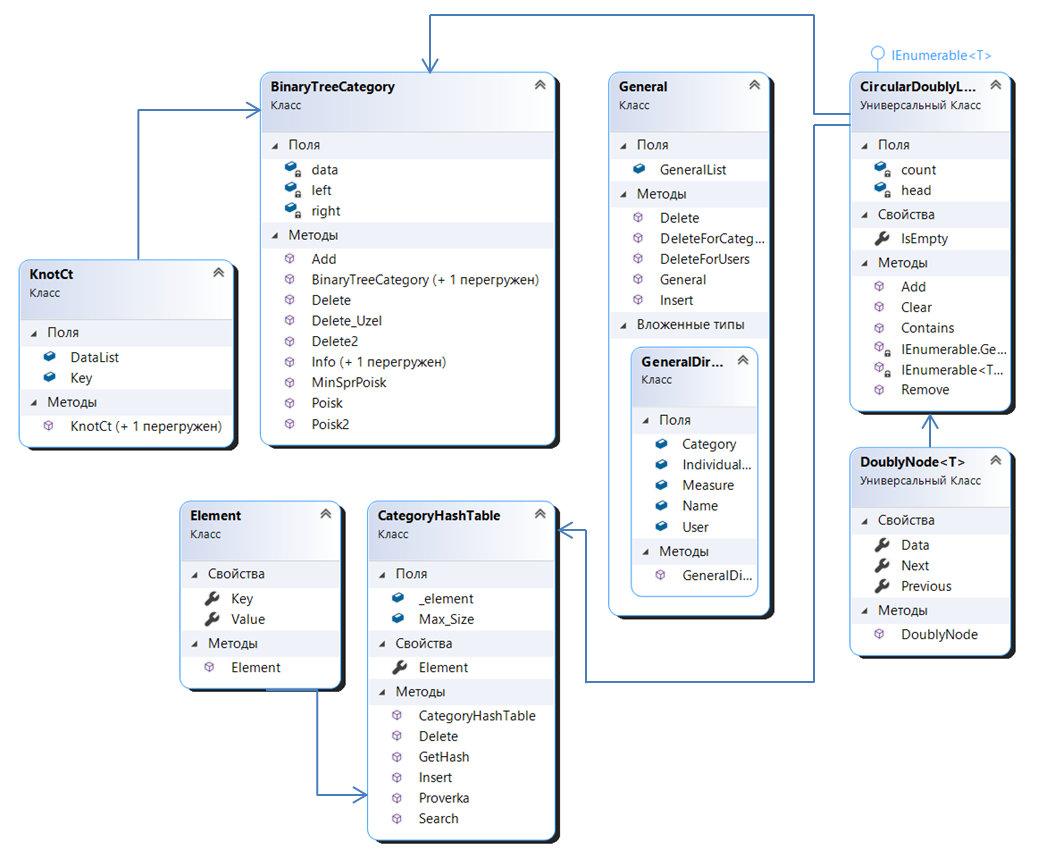


Рисунок 3 UML-диаграмма классов

## **4.2 Описание классов**

Класс **General** - класс, описывающий структура листа, используется для хранения данных общего справочника.

Поля:

* List<GeneralDirectory> GeneralList – объект классаGeneral

Методы:

* General() – конструктор класса General, создает экземпляр класса;

Входные данные: отсутствуют;

Выходные данные: объект класса General;

* void Insert(string user, string category, string name, string measure, string imeasure) – добавляет экземпляров в объект классаGeneral;

Входные данные: экземпляры user, category, name, measure, imeasure;

Выходные данные: объект класса General, с экземплярами;

* void DeleteForCategory(string category, string name) – удаление экземпляров из объекта класса CategoryHashTable;

Входные данные: экземпляры category, string name;

Выходные данные: объект класса General без экземпляров;

* void DeleteForUsers(string user, string imeasure) удаление экземпляров из объекта класса UsersHashTable;

Входные данные: экземпляры user, imeasure;

Выходные данные: объект класса General без экземпляров;

* void Delete(string user, string category, string name, string measure, string imeasure) – удаляет экземпляры из объекта классаGeneral;

Входные данные: user, category, name, measure, imeasure;

Выходные данные: объект класса General без экземпляров;

Класс **Element** - класс, описывающий элементы объекта класса **CategoryHashTable**;

Поля:

* отсутствуют;

Методы:

* Element(string value, string key) – конструктор класса Element, создает экземпляр объекта класса;

Входные данные: экземпляры value и key;

Выходные данные: объект класса Element;

Класс **CategoryHashTable** - класс, описывающий хеш-таблицу;

Поля:

* Dictionary<int, CircularDoublyLinkedList<Element>> \_element – хранимые данные;
* readonly byte MaxSize – максимальная длина ключевого поля.

Методы:

* CategoryHashTable() – конструктор класса CategoryHashTable, создает экземпляр объекта класса;

Входные данные: отсутствуют;

Выходные данные: объект класса CategoryHashTable;

* void Delete(string value, string key) – удаления экземпляров value и key из хеш-таблицы;

Входные данные: экземпляры value и key;

Выходные данные: объект класса CategoryHashTable, без экземпляров value и key класса string;

* int GetHash(string value, string key) – возвращает целое число, являющееся значением хеш-функции, примененной к экземпляру value + key;

Входные данные: экземпляры value и key;

Выходные данные: целое число, являющееся значением хеш-функции;

Предположим, что размер хеш-таблицы равен 10, а на вход поступает запись, содержащая информацию об изделии из модели предметной области (см. п.1.1), а именно:

GetHash (<ЮБКА СОЛНЦЕ>, <ЮБКА>) = (1070+1041+1050+1040+32+1057+1054+1051+1053+1062+1045+1070+1041+1050+1040) mod 10 = mod 10 = 23 mod 10 = 3

GetHash (<ЮБКА КЛЕШ>, <ЮБКА>) = 7

GetHash (<ЮБКА ТРАПЕЦИЯ>, <ЮБКА>) = 0

GetHash (<ЮБКА ПАЧКА>, <ЮБКА>) = 0

GetHash (<ЮБКА ГОДЕ>, <ЮБКА>) = 1

GetHash (<БРЮКИ КАРГО>, <БРЮКИ>) = 9

GetHash (<БРЮКИ ГАЛИФЕ>, <БРЮКИ>) = 8

GetHash (<БРЮКИ БАНАНЫ>, <БРЮКИ>) = 6

GetHash (<РУБАШКА ШЕМИЗЬЕ>, <РУБАШКА>) = 6

GetHash (<ПЛАТЬЕ ФУТЛЯР>, <ПЛАТЬЕ>) = 3

GetHash (<ПЛАТЬЕ БЕБИ ДОЛЛ>, <ПЛАТЬЕ>) = 5

GetHash (<КИМОНО>, <ПЛАТЬЕ>) = 3

* void Insert(string value, string key) – добавление экземпляров value и key в хеш-таблицу;

Входные данные: экземпляры value и key;

Выходные данные: объект класса CategoryHashTable с экземплярами value и key класса;

* int LoadCategoryHashTable() – загрузка экземпляров для класса CategoryHashTable из файла;

Входные данные: файл;

Выходные данные: объект класса CategoryHashTable и список изделий;

* int Proverka(string value, string key) – проверяет корректность введенных данных пользователем;

Входные данные: экземпляры value и key;

Выходные данные: 0, 1, 2 или 3;

* void SaveCategoryHashTable(CategoryHashTable hashTable) – сохранение экземпляров из класса CategoryHashTable в файл;

Входные данные: объект класса CategoryHashTable;

Выходные данные: файл;

* int Search(string value, string key) – поиск экземпляра в объекте класса CategoryHashTable;

Входные данные: экземпляры value и key;

Выходные данные: 0 или количество сравнений;

Класс **Node<T>** ­­– универсальный класс, описывающий узел двусвязного кольцевого списка объекта **CircularDoublyLinkedList**.

Поля:

* T Data – значение узла
* DoublyNode<T> Next – ссылка на следующий узел списка;
* DoublyNode<T> Previous – ссылка на предыдущий узел списка;

Методы:

* DoublyNode (T data) – конструктор узла;

Входные данные: отсутствуют;

Выходные данные: экземпляр объекта класса DoublyNode<T>;

Класс **CircularDoublyLinkedList** – универсальный класс, описывающий кольцевой двусвязный список;

Поля:

* int count – количество элементов в списке;
* DoublyNode<T> head – головной/первый элемент;

Методы:

* void Add(T data) – добавляет элемент в объект класса CircularDoublyLinkedList;

Входные данные: универсальный экземпляр data;

Выходные данные: объект класса CircularDoublyLinkedList с добавленным экземпляром;

– bool Contains(T data) – определяет наличие экземпляра data в объект класса CircularDoublyLinkedList

Входные данные: объект класса CircularDoublyLinkedList, экземпляр data;

Выходные данные: true или false;

– bool Remove(T data) – определяет наличие экземпляра data в объект класса CircularDoublyLinkedList

Входные данные: объект класса CircularDoublyLinkedList, экземпляр data;

Выходные данные: true или false;

Класс **KnotCt** – класс, описывающий узел в бинарном дереве поиска;

Поля:

* CircularDoublyLinkedList<DataGridViewRow> Dlist – кольцевой двусвязный список, содержащий ссылки на индексы строк;
* string Key – строка, содержащая ключ;

Методы:

* KnotCt() – конструктор узла;

Входные данные: отсутсвуют;

Выходные данные: пустой объект класса KnotCt;

* KnotCt(string key, CircularDoublyLinkedList<DataGridViewRow> Dlist) – конструктор узла;

Входные данные: экземпляры key и Dlist;

Выходные данные: объект класса KnotCt, содержащий экземпляры key и Dlist;

Класс **BinaryTreeCategory** – класс, описывающий бинарное дерево поиска, в котором объект класса KnotCt – узел дерева, который является двусвязным кольцевым списком.

Поля:

– KnotCt data – узел;

– BinaryTreeCategory left – левое поддерево;

– BinaryTreeCategory right – правое поддерево;

Методы:

– BinaryTree() – конструктор начального дерева;

Входные данные: отсутствуют;

Выходные данные: объект класса BinaryTree, с пустыми поддеревьями left, right и пустым узлом data;

– BinaryTree(string val, DataGridViewRow str) – конструктор начального дерева с данными значениями узлов;

Входные данные: строка val – ключ, список типа DataGridViewRow, содержащий ссылки на индексы строк;

Выходные данные: объект класса BinaryTreeCategory, с поддеревьями left, right и узлом data, которые имеют соответствующие входные данные;

– void Add(string val, DataGreedViewRow str) – добавления элемента в объект класса BinaryTreeCategory;

Входные данные: экземпляр val и список типа DataGridViewRow, содержащий ссылки на индексы строк;

Выходные данные: объект класса BinaryTreeCategory с новым узлом и соответствующим значением;

– void Delete(string val, DataGreedViewRow str) – удаляет узел из дерева и удаляет индекс из Dlist;

Входные данные: экземпляр val и список типа DataGridViewRow, содержащий ссылки на индексы строк;

– Выходные данные: объект класса BinaryTreeCategory, с отсутствующим узлом и индексом у Dlist;

– void Delete\_Uzel(KnotCt val) – удаляет узел из дерева вместе с индексом DList;

Входные данные: экземпляр класса KnotCt;

Выходные данные: объект класса BinaryTreeCategory, с отсутствующим узлом и индексом у Dlist;

– CircularDoublyLinkedList<DataGridViewRow> Delete2(string val) – удаляет узел (который мы решили удалить из личного справочника) из дерева;

Входные данные: экземпляр val;

Выходные данные: объект класса BinaryTreeCategory

– string Info() – выводит значения полей узлов дерева на экран с учетом связей;

Входные данные: отсутствуют;

Выходные данные: значения узлов дерева в порядке прямого обхода КЛП

– string Info(BinaryTreeCategory current) – выводит значения полей узлов поддерева на экран с учетом связей

Входные данные: объект класса BinaryTreeCategory;

Выходные данные: значения узлов объект класса BinaryTreeCategory в порядке прямого обхода КЛП;

– CircularDoublyLinkedList<DataGridViewRow> Poisk(string val) – нахождение списка индексов экземпляра val (ключа);

Входные данные: экземпляра val;

Выходные данные: список типа DataGridViewRow индексов с нужным ключом;

– KnotCt MinSprPoisk() – нахождение поддерева с минимальным правым узлом;

Входные данные: отсутствуют;

Выходные данные: Поддерево с минимальным справа узлом;

## **4.3 Описание интерфейса**

На рисунке 4 представлено изображение главного окна программы. Снизу в окне располагаются функциональные кнопки. Для добавления в Избранное необходимо нажать кнопку «Добавить». Чтобы завершить работу программы, необходимо нажать «×» в правом верхнем углу окна, и тогда программа завершит свою работу.

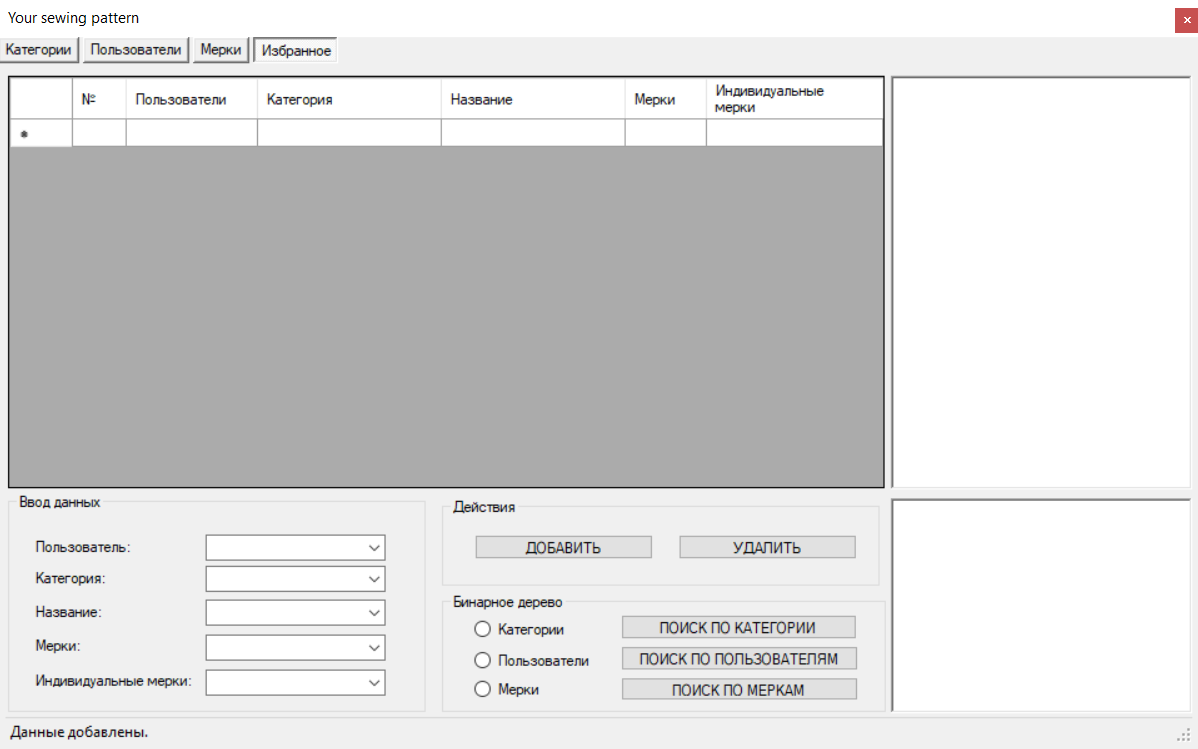


Рисунок 4 Главное окно программы

В правом верхнем углу можно переключать справочники (рисунок 5). Кнопки функционального блока «Работа с данными» работают в соответствии с выбранным справочником.

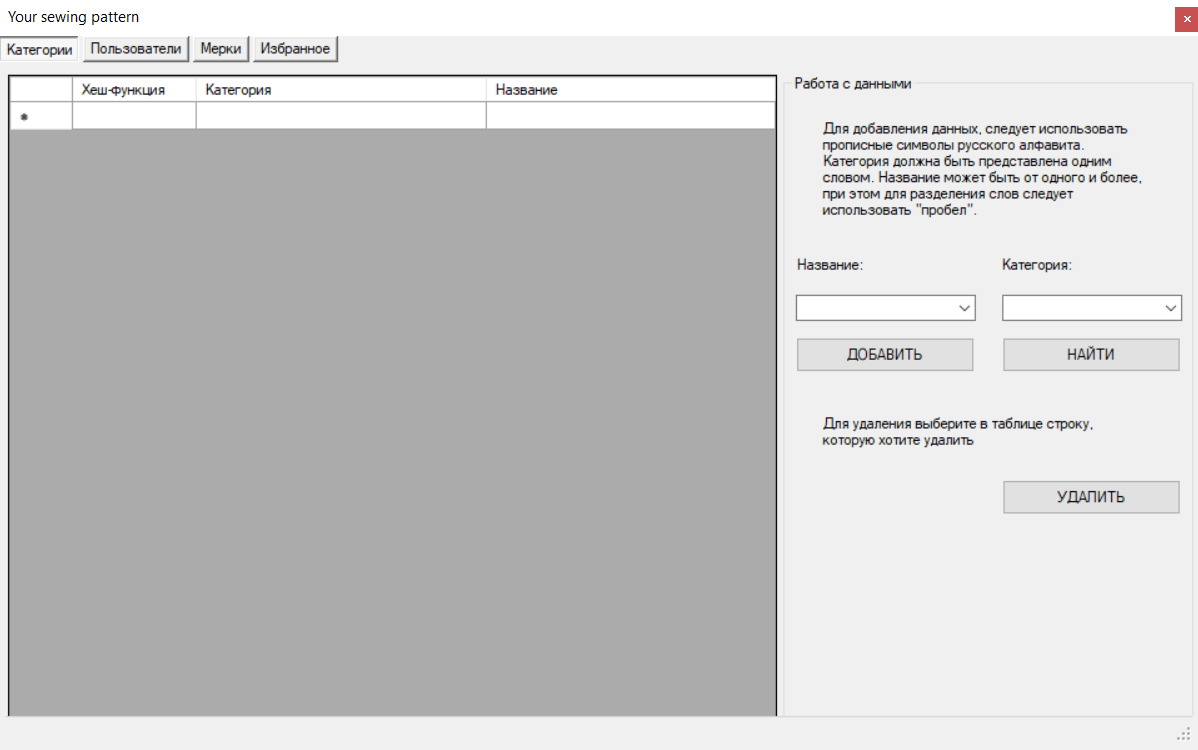


Рисунок 5 Вкладка «Категории» для работы со справочником «Список одежды»

Рассмотрим работу кнопок функционального блока «Работа с данными» на примере таблицы «Избранное».

В строках, представленных на рисунке \_ пользователь может, как сам ввести данные, так и выбрать из выпадающего списка. После этого при нажатии кнопки «Добавить» запись будет добавлена в справочник «Избранное» и в таблицу, если все поля непустые и корректны, иначе выведется сообщение об ошибке.

При нажатии кнопки «Поиск по категориям» программа выведет индексы нужного ключа в бинарном дереве, при условии, что в полях категория и название введены корректные данные.

При нажатии кнопки «Удалить» запись, выбранная в таблице, будет удалена из соответствующих справочников и таблицы, а также из бинарного дерева, причем при обнаружении связанных записей при условии, что удаление происходит не в «Избранном», записи в «Избранном» удалятся.

При нажатии кнопок «Категории», «Мерки» или «Пользователи», функционального блока «Бинарное дерево», появится информация о соответствующем бинарном дереве. Впоследствии на эти кнопки придется нажимать, чтобы информация обновилась

## **4.4 Тестирование**

Тестируете по методу черного ящика, результаты которого представлены в Таблицах 1, 2

Таблица 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Описание тестовой ситуации | | Входные данные | | Выходные данные | |
| Хеш-таблица (метод цепочек) | Изделие: категория, название | Хеш-таблица (метод цепочек) | Сообщение |
| Добавление | | | | | |
| 1 | Добавление некорректных данных | – | Категория: цифры, английские символы, строчные буквы или больше, чем одно слово раскладка  Название: цифры, английские символы, строчные буквы | – | Проверьте корректность строки «Категория» или «Название» |
| – | – | – | Проверьте корректность строки «Категория» или «Название» |
| 2 | Добавление корректных данных | – | ЮБКА КАРАНДАШ  ЮБКА | ЮБКА  ЮБКА КАРАНДАШ | Данные ЮБКА КАРАНДАШ  ЮБКА добавлены |
| 3 | Проверка уникальности ключа | ЮБКА КАРАНДАШ  ЮБКА | ЮБКА КАРАНДАШ  ЮБКА | ЮБКА КАРАНДАШ  ЮБКА | Такой ключ уже существует |
| 4 | Добавление при коллизии | БРЮКИ ГАЛИФЕ БРЮКИ | ЮБКА КАРАНДАШ  ЮБКА | БРЮКИ ГАЛИФЕ БРЮКИ  ЮБКА КАРАНДАШ  ЮБКА | Данные ЮБКА КАРАНДАШ  ЮБКА добавлены |
| Поиск | | | | | |
| 5 | Запись не существует | БРЮКИ ГАЛИФЕ БРЮКИ | ПЛАТЬЕ ФУТЛЯР ПЛАТЬЕ | – | Элемент не найден |
| 6 | Запись существует | ЮБКА СОЛНЦЕ ЮБКА | ЮБКА СОЛНЦЕ ЮБКА | – | Количество сравнений - 1 |
| 7 | Поиск при коллизии | ЮБКА СОЛНЦЕ ЮБКА  ПЛАТЬЕ ФУТЛЯР ПЛАТЬЕ | ПЛАТЬЕ ФУТЛЯР ПЛАТЬЕ | – | Количество сравнений - 2 |
| Удаление | | | | | |
| 8 | Запись не существует | ЮБКА СОЛНЦЕ ЮБКА  ПЛАТЬЕ ФУТЛЯР ПЛАТЬЕ | БРЮКИ ГАЛИФЕ БРЮКИ | – | Требуется выбрать строку |
| 9 | Запись существует и отсутствуют связанные с ней записи | ПЛАТЬЕ ФУТЛЯР ПЛАТЬЕ | ПЛАТЬЕ ФУТЛЯР ПЛАТЬЕ | – | – |
| 10 | Запись существует и присутствуют связанные с ней записи | ПЛАТЬЕ ФУТЛЯР ПЛАТЬЕ | ПЛАТЬЕ ФУТЛЯР ПЛАТЬЕ | – | – |
| 11 | Удаление при коллизии | ЮБКА СОЛНЦЕ ЮБКА  ПЛАТЬЕ ФУТЛЯР ПЛАТЬЕ | ПЛАТЬЕ ФУТЛЯР ПЛАТЬЕ | ЮБКА СОЛНЦЕ ЮБКА | – |

Таблица 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Описание тестовой ситуации | | Входные данные | | Выходные данные | |
| Бинарное дерево поиска | Избранное пользователя:  Пользователь, Категория, Название, Мерки, Индивидуальные мерки | Бинарное дерево поиска | Сообщение |
| Добавление | | | | | |
| 1 | Добавление некорректных данных | – | – | – | Проверьте корректность введенных строк |
| – | Категория: цифры, английские символы, строчные буквы или больше, чем одно слово раскладка  Название: цифры, английские символы, строчные буквы | – | Проверьте корректность введенных строк |
| 2 | Добавление корректных данных и существует связанная запись | – | Tukh4  ПЛАТЬЕ ФУТЛЯР ПЛАТЬЕ  Ош Ош От Об Уб  102 90 129 31 111 40 52 88 34 24 33 | Tukh4  ПЛАТЬЕ ФУТЛЯР ПЛАТЬЕ  Ош Ош От Об Уб  102 90 129 31 111 40 52 88 34 24 33 | Запись добавлена |
| 3 | Добавление корректных данных и не существует связанной записи | – | Tukh4  ПЛАТЬЕ ФУТЛЯР ПЛАТЬЕ | – | Данные не сходятся со справочником категории |
| 4 | Проверка уникальности ключа | Tukh4  ПЛАТЬЕ ФУТЛЯР ПЛАТЬЕ  Ош Ош От Об Уб  102 90 129 31 111 40 52 88 34 24 33 | Tukh4  ПЛАТЬЕ ФУТЛЯР ПЛАТЬЕ  Ош Ош От Об Уб  102 90 129 31 111 40 52 88 34 24 33 | Tukh4  ПЛАТЬЕ ФУТЛЯР ПЛАТЬЕ  Ош Ош От Об Уб  102 90 129 31 111 40 52 88 34 24 33 | Такая запись уже существует |
| 5 | Добавление при коллизии | Tukh4  ЮБКА ТРАПЕЦИЯ ЮБКА  Ош Ош От Об Уб  102 90 129 31 111 40 52 88 34 24 33 | Tukh4  ЮБКА ПАЧКА ЮБКА  Ош Ош От Об Уб  102 90 129 31 111 40 52 88 34 24 33 | Tukh4  ЮБКА ТРАПЕЦИЯ ЮБКА  Ош Ош От Об Уб  102 90 129 31 111 40 52 88 34 24 33  Tukh4  ЮБКА ПАЧКА ЮБКА  Ош Ош От Об Уб  102 90 129 31 111 40 52 88 34 24 33  (Внутри одного узла, представлены в виде списка) | Запись добавлена |
| Поиск | | | | | |
| 6 | Запись не существует | – | – | – | Пользователь не найден |
| 7 | Запись существует | Tukh4  ЮБКА ТРАПЕЦИЯ ЮБКА  Ош Ош От Об Уб  102 90 129 31 111 40 52 88 34 24 33 | Tukh4  ЮБКА ТРАПЕЦИЯ ЮБКА  Ош Ош От Об Уб  102 90 129 31 111 40 52 88 34 24 33 | – | Индекс 0, количество сравнений 0 |
| 8 | Поиск при коллизии | Tukh4  ЮБКА ТРАПЕЦИЯ ЮБКА  Ош Ош От Об Уб  102 90 129 31 111 40 52 88 34 24 33  Tukh4  ЮБКА ПАЧКА ЮБКА  Ош Ош От Об Уб  102 90 129 31 111 40 52 88 34 24 33 | Tukh4  ЮБКА ПАЧКА ЮБКА  Ош Ош От Об Уб  102 90 129 31 111 40 52 88 34 24 33 | – | Индекс 1, количество сравнений 1 |
| Удаление | | | | | |
| 9 | Запись не существует | – | Tukh4  ЮБКА ПАЧКА ЮБКА  Ош Ош От Об Уб  102 90 129 31 111 40 52 88 34 24 33 | – | Изделие не было найдено |
| 10 | Запись существует | Tukh4  ЮБКА ПАЧКА ЮБКА  Ош Ош От Об Уб  102 90 129 31 111 40 52 88 34 24 33 | Tukh4  ЮБКА ПАЧКА ЮБКА  Ош Ош От Об Уб  102 90 129 31 111 40 52 88 34 24 33 | – | – |
| 11 | Удаление при коллизии | {строка 0}  Tukh4  ЮБКА ПАЧКА ЮБКА  Ош Ош От Об Уб  102 90 129 31 111 40 52 88 34 24 33  {строка 1}  RuSiLGiA  ЮБКА ПАЧКА ЮБКА  Ош Ош От Об Уб  58 90 129 31 111 40 52 88 34 24 33 | {строка 1}  Еке23рк  ЮБКА ПАЧКА ЮБКА  Ош От Об Уб  129 31 111 40 52 88 34 24 | {строка 0}  Tukh4  ЮБКА ПАЧКА ЮБКА  Ош Ош От Об Уб  102 90 129 31 111 40 52 88 34 24 33 | – |

# Заключение

Целью курсового проекта было: разработать информационную систему для автоматизации работы со справочниками предметной области проекта Your Sewing Pattern.

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

1) Проведен анализ предметной области «Your Sewing Pattern» и построена ее 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 5.0;

c. во время разработки Microsoft Visual Studio Community 2019 была использована в качестве среды выполнения.

# Список литературы

1. Источник: Т. Х. Кормен, Ч. И. Лейзерсон, Р. 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.

2. Источник: А. А. Навроцкий Основы алгоритмизации и программирования в среде Visual C++: учеб.-метод. пособие / А. А. Навроцкий. – Минск: БГУИР, 2014. – 160 с.: ил.