

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение   
высшего образования

**«Дальневосточный федеральный университет»**

(ДВФУ)

**ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ**

**Департамент программной инженерии и искусственного интеллекта**

Перепечин Владимир Владимирович

Разработка информационной системы

**КУРСОВОЙ ПРОЕКТ**

по дисциплине «Фундаментальные структуры данных и алгоритмы»

по образовательной программе подготовки бакалавров по направлению

09.03.04 - Программная инженерия

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | |  |  | Студент гр. Б9120-09.03.04прогин | | |
|  |  |  | | В.В. Перепечин |
|  | | | | | | |  |  | (подпись) | |  |
|  | | | | | | | | | | | |
| Защищен с оценкой | | | | | | |  |  | Руководитель | | |
|  | | | | | | |  |  | ст. преподаватель | | |
|  | | | |  |  | |  |  | ученая степень, должность |  | О.А. Крестникова |
| (подпись) | | | |  | (И.О. Фамилия) | |  |  | (подпись) |  | (И.О. Фамилия) |
| « |  | » |  | | | 2022 г. |  |  |  | | |
|  | | | | | | | | | | | |

г. Владивосток

2022

# Оглавление

[Оглавление 2](#_Toc108150541)

[Введение 3](#_Toc108150542)

[1 Анализ предметной области 4](#_Toc108150543)

[1.1 Объекты предметной области 4](#_Toc108150544)

[1.2 Законы ПО 6](#_Toc108150545)

[1.3 Постановка задачи «Формирование отчета «Кандидаты»» 6](#_Toc108150546)

[2 Теоретическая часть 8](#_Toc108150547)

[2.1 Хеш-таблица 8](#_Toc108150548)

[2.1.1 Хеш-функция 8](#_Toc108150549)

[2.1.2 Разрешение коллизий методом открытой адресации 10](#_Toc108150550)

[2.2 Красно-чёрное дерево 11](#_Toc108150551)

[3 Требования к информационной системе 18](#_Toc108150552)

[3.1 Требования к данным 18](#_Toc108150553)

[3.1.1 Требования к входным данным 18](#_Toc108150554)

[3.1.2 Требования к выходным данным 19](#_Toc108150555)

[3.2 Функциональные требования 20](#_Toc108150556)

[4 Реализация 22](#_Toc108150557)

[4.1 Спецификация структур данных 22](#_Toc108150558)

[4.2 Описание среды разработки 28](#_Toc108150559)

[4.3 Руководство пользователя 28](#_Toc108150560)

[4.4 Тестирование 30](#_Toc108150561)

[Заключение 33](#_Toc108150562)

[Список литературы 34](#_Toc108150563)

# Введение

В современном мире многие люди испытывают недостаток времени, однако занятие спортом является важной составляющей жизни человека. Отсутствие социальных связей создает проблемы, ведь вокруг есть много мест сбора (фитнес-клубы, площадки), но зачастую сложно найти в своем окружении достаточное количество знакомых, которые готовы объединится в одном месте в одно время.

Проект «Team of players» предоставляет возможность любому заинтересованному пользователю найти себе товарища для спортивного времяпровождения .

Целью курсового проекта является: разработка информационной системы «Teams Of Players». Подсистема работы с справочниками «Игроки» и «Игроки команды», а также отчетом «Кандидаты».

Для достижения поставленной цели необходимо выполнить следующие задачи:

1. Провести анализ предметной области «Приложения для спортивных знакомств».

2. Изучить теоретические основы методов построения справочников.

3. Определить требования к информационной системе.

4. Реализовать и провести тестирование.

# 1 Анализ предметной области

Требуется разработать информационную систему для предметной области (ПО) «Приложения для спортивных знакомств».

Система должна решать следующие задачи:

1. хранить, позволять просматривать, добавлять и удалять информацию об игроках и игроках команды;
2. позволять искать информацию по ФИО и названию команды.
3. формировать список игроков по возрасту и роли в команде.
4. Предусмотреть проверку целостности информации, представленной в справочниках.

1.1 Объекты предметной области

Исходя из задач, которые должна решать информационная система, выделены объекты ПО, представленные в Таблице 1.

Таблица 1 – Объекты ПО

|  |  |
| --- | --- |
| Тип и название объекта | Описание объекта |
| Справочник «Игроки» | Хранит информацию по каждому игроку |
| Справочник «Игроки команд» | Хранит информацию об игроках команды |
| Отчет «Кандидаты» | Хранит информацию о игроках отобранных по возрасту и роли |

Каждый игрок характеризуется следующими параметрами: ФИО, дата рождения, виды спорта.

**ФИО** – аббревиатура, расшифровывается как: фамилия, имя, отчество. Фамилия, имя и отчество являются словами, состоящими из букв английского алфавита, первая буква каждого слова – заглавная, остальные – строчные. Слова разделены одним пробелом.

**Дата рождения** – три целых числа разделенных одной точкой. Первое число в диапазоне от 1 до 31 – день, второе число в диапазоне от 1 до 12 – месяц, третье число в диапазоне от 1965 до 2020 – год.

**Виды спорта** – множество строк, состоящих из слов, состоящих из букв английского алфавита, первая буква строки – заглавная, остальные – строчные. Строки разделены одной запятой, слова разделены одним пробелом.

Пример справочника «Игроки» представлен в таблице 2.

Таблица 2 – Справочник «Игроки»

|  |  |  |
| --- | --- | --- |
| **ФИО** | **Дата рождения** | **Виды спорта** |
| SHesodi Hyrolavacu Gocaewoqufo | 15.03.1988 | Daelibazhi, Lymezhe |
| SHesaemo Beqavitu Sywaelyloka | 08.12.2003 | Mushijezhynae, Mecusykaeshi |
| Xawacocaesu Xaesobe Wucaeja | 05.01.1998 | Nycyvaefo |
| Piraezhi Gaqilago Nacaxose | 02.12.2007 | Bucujuwaqi, Mahejae |
| ZHefaexi Buwygadi ZHaevory | 02.02.1998 | Cusobae |
| Lymaelu Daehyconu Bukoqabe | 01.10.1980 | Caejikashama |
| Menady Delilae Gymaegiwuwy | 01.08.1996 | Xidyshi, Gebaeve |
| Paedenaeny SHudexa ZHaezharaebu | 01.07.1977 | Dyshynaxo, Rejuqomovy, Hysaezhuzho, Kulefedo |
| Kaejushapa Jagaepu Dexeropo | 01.06.1969 | Qiwylyqo, Rereta |
| Risowe ZHysyku SHecaevushae | 01.07.1996 | Juqalima, Pyjeshelikae, Maexuwiga, Jaepylemu |

Каждый игрок команды характеризуется следующими параметрами: ФИО, команда, роль, вид спорта.

**ФИО** – аббревиатура, расшифровывается как: фамилия, имя, отчество. Фамилия, имя и отчество являются словами, состоящими из букв английского алфавита, первая буква каждого слова – заглавная, остальные – строчные. Слова разделены одним пробелом.

**Команда** – слово из букв английского алфавита, первая буква которого – заглавная, остальные – строчные.

**Роль** – слово из букв английского алфавита, первая буква которого – заглавная, остальные – строчные.

**Вид спорта** – слово из букв английского алфавита, первая буква которого – заглавная, остальные – строчные.

Пример справочника «Игроки команд» представлен в таблице 3.

Таблица 3 – Справочник «Игроки команд»

|  |  |  |  |
| --- | --- | --- | --- |
| **ФИО** | **Команда** | **Роль** | **Виды спорта** |
| Shesaemo Beqavitu Sywaelyloka | Wyrymae | Mixutylo | Daelibazhi |
| Kaejushapa Jagaepu Dexeropo | Lotoqe | Fysaebinae | Daelibazhi |
| Laetuzharece Pyfaeri Waebymugy | Xaeshaetyraeki | Daesaeshyki | Gobaeta |
| Qapoce Jytavova ZHilolopae | ZHebucoma | Betonamexa | Baecurexu |
| Bypoxema Qagaeshabi SHubaefu | Cizhuvu | Bemyxyboly | Xaevopizhy |
| Lymaelu Daehyconu Bukoqabe | Nonaqo | Bemyxyboly | Xidyshi |
| Fybokecaeqi Lylyby Mejujasho | Jadaemifezhae | Balepi | Daelibazhi |
| Lymaelu Daehyconu Bukoqabe | Solotasy | Balepi | Lafofashae |
| Soxytide Muriwo Dishahelae | ZHepida | Balepi | Pyjeshelikae |
| Zhylaefafa Wejudepa Gihyloshovae | Fipadyqimy | Balepi | Hilisolu |

1.2 Законы ПО

1. у одного игрока может быть несколько команд;
2. каждый игрок может находится в команде, только в единственном экземпляре;
3. игрок не обязан иметь команды;
4. не может быть команды у несуществующего игрока;
5. у каждой команды только один вид спорта;
6. каждое ФИО игрока уникально;

1.3 Постановка задачи «Формирование отчета «Кандидаты»»

Входные данные: Справочник «Игроки», Справочник «Игроки команд», возраст – целое число, роль - слово.

Выходные данные: Отчет «Кандидаты».

Связь: Справочник«Игроки».Игрок.ФИО = Справочник «Игроки команд».Игрок.ФИО

# 2 Теоретическая часть

В рамках курсового проекта необходимо искать информацию о игроках и игроках команды. Для ускорения поиска используются такие структуры данных, например, как сбалансированные бинарные деревья поиска (АВЛ - деревья, красно-чёрные деревья, B-деревья и т.д.) и хеш-таблицы. Красно чёрное дерево — это один из видов самобалансирующихся двоичных деревьев поиска, гарантирующий линейно-логарифмический показатель сложности выполнения основных операций: добавление, удаление, поиск. Сбалансированность дерева достигается за счёт введения дополнительного атрибута для узла дерева - «Цвет». «Цвет» принимает одно из двух значений - «чёрный» или «красный» [1].

Хеш-таблицы часто применяются в базах данных, а также в языковых процессорах типа компиляторов и ассемблеров, где они изящно обслуживают таблицы идентификаторов. В таких приложениях хеш-таблица - наилучшая структура данных.

2.1 Хеш-таблица

Хеш-таблица представляет собой эффективную структуру данных для реализации ассоциативных массивов, в частности словарей. Ассоциативным массивом называется такой абстрактный тип данных, который позволяет хранить пары вида «ключ, значение» и поддерживающий операции добавления пары, а также поиска и удаления пары по ключу [2]. Словарь - одна из разновидностей таких массивов. Хешированием называют отображение множества ключей в множество индексов массива по некоторому правилу, которое называют хеш-функцией. Несмотря на худший случай при поиске - О(п), в среднем время поиска составляет 0(1), что и подтверждается на практике [1].

2.1.1 Хеш-функция

Пусть элемент с ключом хранится в ячейке . При хешировании этот элемент хранится в ячейке , где - хеш-функция. Функция h отображает совокупность ключей на ячейки хеш-таблицы , где размер хеш-таблицы обычно гораздо меньше значения . Мы говорим, что элемент с ключом к хешируется в ячейку ; величина называется хеш-значеиием ключа [1].

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

, где символ ключа.

В рамках курсового проекта ключом является совокупность двух полей (ФИО и Роль). Для полей «ФИО» и «Роль» значение , будет кодом соответствующего символа в .

Предположим, что размер хеш-таблицы равен 20, а на вход поступают записи игроков команды из модели предметной области. Приведем 10 примеров:

Аналогично:

Как видно из примера, два ключа могут быть хешированы в одну и ту же ячейку, например, для ключей и . Такая ситуация называется коллизией. Существует несколько эффективных методов разрешения коллизий. В рамках курсового проекта будет рассмотрен метод открытой адресации [1].

2.1.2 Разрешение коллизий методом открытой адресации

Пусть хеш-таблица представлена массивом . Тогда если соответствующая ячейка таблицы пуста, то заносим в неё значение; иначе по некому алгоритму заносим значение в другую свободную ячейку. В рамках курсового проекта рассмотрим метод линейного пробирования. [1].

Пусть — адрес ячейки, — ключ, — номер попытки при разрешении коллизии, - величина шага. Тогда при линейном пробировании выбор ячейки производится по формуле:

Таким образом, линейное пробирование сводится к последовательному перебору ячеек с некоторым фиксированным шагом.

Однако по мере заполнения хеш-таблицы можно достигнуть её полного переполнения что приведёт к невозможности вставки нового элемента. Для решения этой проблемы необходимо проводить рехеширование – увеличение размеров таблицы с последующим её повторным заполнением. В рамках курсового проекта рехеширование происходит при 75% и 25% заполненности хеш-таблицы [3].

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

2.2 Красно-чёрное дерево

Бинарное дерево поиска — это структура данных для работы с упорядоченными множествами [1].

Красно-чёрное дерево представляет из себя бинарное дерево поиска с дополнительным атрибутом цвета для каждого узла. Цвет узла может быть либо красным (RED), либо чёрным (BLACK). В соответствии с накладываемыми на узлы дерева ограничениями ни один простой путь от корня в красно-чёрном дереве не отличается от другого по длине более чем в два раза, так что красно-чёрные деревья являются приближённо сбалансированными [1].

Каждый дочерний узел содержит атрибуты color, key, left, right и р. Если не существует дочернего или родительского узла по отношению к данному, соответствующий указатель принимает значение NULL. Мы будем рассматривать эти значения NULL как указатели на внешние узлы (листья) бинарного дерева поиска. При этом все «нормальные» узлы, содержащие поле ключа, становятся внутренними узлами дерева [1].

Бинарное дерево поиска является красно-чёрным деревом, если оно удовлетворяет следующим красно-чёрным свойствам:

1) Каждый узел является либо красным, либо чёрным.

2) Корень дерева является чёрным узлом.

3) Каждый лист дерева (NULL) является чёрным узлом.

4) Если узел красный, то оба его дочерних узла чёрные.

5) Для каждого узла все простые пути от него до листьев, являющихся потомками данного узла, содержат одно и то же количество чёрных узлов [1].

Красно-чёрное дерево поддерживает следующие операции:

1. Вставка.
2. Удаление.
3. Повороты (малый левый и правый повороты, большой левый и большой правый повороты).
4. Обходы (прямой, обратный, симметричный).

Узел, с которым мы работаем, на картинках имеет имя х.

Вставка элемента [1].

Каждый элемент вставляется вместо листа, поэтому для выбора места вставки идём от корня до тех пор, пока указатель на следующего сына не станет NULL (то есть этот сын — лист). Вставляем вместо него новый элемент с нулевыми потомками и красным цветом. Теперь проверяем балансировку. Если отец нового элемента черный, то никакое из свойств дерева не нарушено. Если же он красный, то нарушается свойство 3, для исправления достаточно рассмотреть два случая:

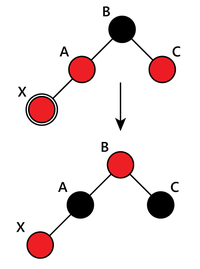
1. "Дядя" этого узла тоже красный. Тогда, чтобы сохранить свойства 3 и 4, просто перекрашиваем "отца" и "дядю" в чёрный цвет, а "деда" в красный. В таком случае черная высота в этом поддереве одинакова для всех листьев и у всех красных вершин "отцы" черные. Проверяем, не нарушена ли балансировка. Если в результате этих перекрашиваний мы дойдём до корня, то в нём в любом случае ставим чёрный цвет, чтобы дерево удовлетворяло свойству 2.

Рисунок 2 – Красно-чёрное дерево, вставка элемента 1 случай

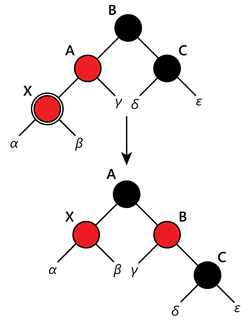
1. "Дядя" чёрный. Если выполнить только перекрашивание, то может нарушиться постоянство чёрной высоты дерева по всем ветвям. Поэтому выполняем поворот. Если добавляемый узел был правым потомком, то необходимо сначала выполнить левое вращение, которое сделает его левым потомком. Таким образом, свойство 3 и постоянство черной высоты сохраняются.

Рисунок 3 – Красно-чёрное дерево, вставка элемента, 2 случай

Удаление вершины [1].

При удалении вершины могут возникнуть три случая в зависимости от количества её детей[1].:

1. Если у вершины нет детей, то изменяем указатель на неё у родителя на NULL [1].
2. Если у неё только один ребёнок, то делаем у родителя ссылку на него вместо этой вершины [1].

3) Если же имеются оба ребёнка, то находим вершину со следующим значением ключа. У такой вершины нет левого ребёнка. Удаляем уже эту вершину описанным во втором пункте способом, скопировав её ключ в изначальную вершину [1], [4].

Проверим балансировку дерева. Так как при удалении красной вершины свойства дерева не нарушаются, то восстановление балансировки потребуется только при удалении чёрной. Рассмотрим ребёнка удалённой вершины [1].

Если брат этого ребёнка красный, то делаем вращение вокруг ребра между отцом и братом, тогда брат становится родителем отца. Красим его в чёрный, а отца — в красный цвет, сохраняя таким образом черную высоту дерева (Рисунок 4). Хотя все пути по-прежнему содержат одинаковое количество чёрных узлов, сейчас x имеет чёрного брата и красного отца. Таким образом, мы можем перейти к следующему шагу [1],[4].

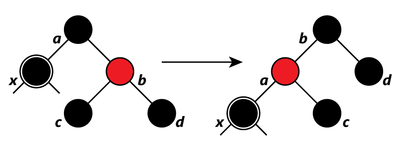


Рисунок 4 – Красно-чёрное дерево, удаление вершины, 3 случай.

Если брат текущей вершины был чёрным, то получаем три случая:

1) Оба ребёнка у брата чёрные. Красим брата в красный цвет и рассматриваем далее отца вершины. Делаем его черным, это не повлияет на количество чёрных узлов на путях, проходящих через b, но добавит один к числу чёрных узлов на путях, проходящих через x, восстанавливая тем самым влияние удаленного чёрного узла. Таким образом, после удаления вершины черная глубина от отца этой вершины до всех листьев в этом поддереве будет одинаковой (Рисунок 5) [1], [4].

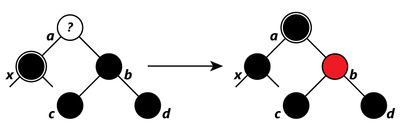


Рисунок 5 – Красно-чёрное дерево, удаление вершины, 1 подпункт.

2) Если у брата правый ребёнок чёрный, а левый красный, то перекрашиваем брата и его левого сына и делаем вращение. Все пути по-прежнему содержат одинаковое количество чёрных узлов, но теперь у x есть чёрный брат с красным правым потомком, и мы переходим к следующему случаю. Ни x, ни его отец не влияют на эту трансформацию (Рисунок 6) [1]. [4].

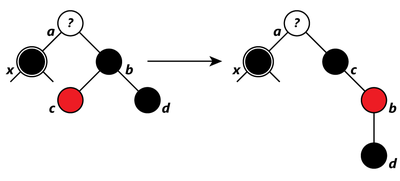


Рисунок 6 – Красно-чёрное дерево, удаление вершины, 2 подпункт.

3) Если у брата правый ребёнок красный, то перекрашиваем брата в цвет отца, его ребёнка и отца — в чёрный, делаем вращение. Поддерево по-прежнему имеет тот же цвет корня, поэтому свойство 3 и 4 не нарушаются. Но у x теперь появился дополнительный чёрный предок: либо a стал чёрным, или он и был чёрным и b был добавлен в качестве чёрного дедушки. Таким образом, проходящие через x пути проходят через один дополнительный чёрный узел. Выходим из алгоритма (Рисунок 7) [1], [4].

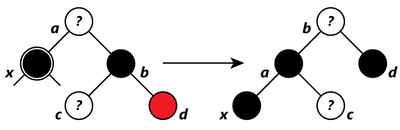


Рисунок 7 – Красно-чёрное дерево, удаление вершины, 3 подпункт.

В рамках курсового проекта при удалении узла с двумя дочерними узлами, на его месте появится узел, который является максимальным слева (самый правый потомок левого поддерева), относительно удаляемого узла.

Ниже представлен пример красно-чёрного дерева (Рисунок 8) [1].

Kaejushapa Jagaepu Dexeopo;  
Lotoqe; Blondas; Daelibazhi

Fybokaeqi Lylyby Mejujasho; Eventac; Daelibazhi; Balepi

NILL

Bypoema Qagashabi Subaefu; Artyse; Bexyboly; Xaevopizhy

;

Lymaelu Daehconu Bukoqabe; Celadoko; Bexyboly; Xidyshi

Laetuece Pyfaeri Waebymugy; Blondas; Dayki; Gobaeta

;

Qapoce Jytavova ZHilolopae;  
Dremado; Baeurexu; Solotasy

Рисунок 8 – Пример красно-чёрного дерева на записях из анализа предметной области.

# 3 Требования к информационной системе

3.1 Требования к данным

3.1.1 Требования к входным данным

Основываясь на анализе ПО, входными данными является:

* текстовый файл Players.txt, каждая строка файла содержит информацию об объекте Игрок, а именно поля ФИО, Дата рождения, Виды спорта разделенные символом «;». Разделителем внутри поля Виды спорта является «,».

Пример текстового файла:

Sesodi Hyrolavacu Gocaewoqufo;15.03.1988;Daelibazhi, Lymezhe

Sesaemo Beqavitu Sywaelyloka;08.12.2003;MuSijezhynae, MecusykaeSi

Xawacocaesu Xaesobe Wucaeja;05.01.1998;Nycyvaefo

Piraezhi Gaqilago Nacaxose;02.12.2007;Bucujuwaqi, Mahejae

ZHefaexi Buwygadi ZHaevory;02.02.1998;Cusobae

Lymaelu Daehyconu Bukoqabe;01.10.1980;CaejikaSama

Menady Delilae Gymaegiwuwy;01.08.1996;XidySi, Gebaeve

Paedenaeny Sudexa ZHaezharaebu;01.07.1977;DySynaxo

KaejuSapa Jagaepu Dexeropo;01.06.1969;Qiwylyqo, Rereta

Risowe ZHysyku SecaevuSae;01.07.1996;Juqalima

* текстовый файл Teams.txt, каждая строка файла содержит информацию об объекте Игрок команды, а именно поля ФИО, Название команды, Роль, Вид спорта, разделенные символом «;»;

Пример текстового файла:

Sesaemo Beqavitu Sywaelyloka;Wyrymae;Mixutylo;Daelibazhi

KaejuSapa Jagaepu Dexeropo;Lotoqe;Fysaebinae;Daelibazhi

Laetuzharece Pyfaeri Waebymugy;XaeSaetyraeki;DaesaeSyki;Gobaeta

Qapoce Jytavova ZHilolopae;ZHebucoma;Betonamexa;Baecurexu

Bypoxema QagaeSabi Subaefu;Cizhuvu;Bemyxyboly;Xaevopizhy

Lymaelu Daehyconu Bukoqabe;Nonaqo;Bemyxyboly;XidySi

Fybokecaeqi Lylyby MejujaSo;Jadaemifezhae;Balepi;Daelibazhi

Lymaelu Daehyconu Bukoqabe;Solotasy;Balepi;LafofaSae

Soxytide Muriwo DiSahelae;Zepida;Balepi;PyjeSelikae

Zhylaefafa Wejudepa GihyloSovae;Fipadyqimy;Balepi;Hilisolu

* Пользователь вводит с клавиатуры: Возраст – натуральное число, Роль – слово.

3.1.2 Требования к выходным данным

Выходными данными являются:

* текстовый файл Players.txt, каждая строка файла содержит информацию об объекте Игрок, а именно поля ФИО, Дата рождения, Виды спорта разделенные символом «;». Разделителем внутри поля Виды спорта является «,».
* текстовый файл Teams.txt, каждая строка файла содержит информацию об объекте Игрок команды, а именно поля ФИО, Название команды, Роль, Вид спорта, разделенные символом «;»;
* сообщения об ошибках должны выводится в отдельной форме и содержать информацию об ошибке. Всевозможные сообщения об ошибках приведены ниже:

1. Не удается получить число
2. Ожидалось натуральное число
3. Игрок с таким именем уже есть
4. Такого игрока нету в таблице Игроки
5. Поиск пуст
6. Такой игрок в этой команде уже есть
7. Такого игрока нет в списке
8. Запись не найдена
9. Файл не существует
10. Не действительная дата
11. Не соответствие количества аргументов
12. Пустое поле ФИО
13. ФИО состоит не из трех слов
14. Неправильное ФИО
15. Пустое поле названия команды
16. Неправильное название команды
17. Пустое поле роли
18. Неправильное название роли
19. Пустое поле вида спорта
20. Неправильное название вида спорта
21. Пустое поле даты
22. Неправильный формат даты

3.2 Функциональные требования

Информационная система, должна позволять:

1. Хранить информацию о игроках и игроках команд;
2. Позволять просматривать всю информацию о игроках и игроках команд;
3. Позволять добавлять информацию о игроке и игроке команды (каждая запись уникальна, при добавлении дублирующей информации выводить соответствующее сообщение. При добавлении игрока команды, в списке игроков должна существовать запись, у которой поля ФИО, Дата рождения, Виды спорта должны совпадать с соответствующими значениями, вводимые пользователем. В противном случае программа должна выводить соответствующее сообщение);
4. Позволять удалять информацию о игроке, игроке команды (в случае, если такой записи не существует, выводить соответствующее сообщение. Если существует игрок, что его поля ФИО, Дата рождения, Виды спорта совпадают с соответствующими значениями, введёнными пользователям, то удалить игрока и игрока команд со связанными полями);
5. Позволять искать информацию о игроке, игроке команды (в случае, если такой записи не существует, выводить соответствующее сообщение);

# 4 Реализация

4.1 Спецификация структур данных

Класс Player - класс, описывающий Игрока.

Поля:

* String Name – переменная, хранит ФИО
* DateTime Birthday – переменная, хранит дату рождения
* List<string> SportTypes – переменная, хранит виды спорта

Методы:

* Player(string name, DateTime birthday, List<string> sportTypes) – конструктор класса Player, создает экземпляр объекта класса;

Входные данные: строка, дата, список строк

Выходные данные: объект класса Player

Класс TeamPlayer - класс, описывающий Игрока команды.

Поля:

* String PlayerName – переменная, хранит ФИО
* String TeamName – переменная, хранит название команды
* String Role – переменная, хранит роль в команде
* String SportType – переменная, хранит вид спорта

Методы:

* TeamPlayer(string playerName, string teamName, string role, string sportType) – конструктор класса TeamPlayer, создает экземпляр объекта класса;

Входные данные: 4 строки

Выходные данные: объект класса TeamPlayer

Класс FileManager - класс, описывающий работу с файлами.

Поля:

* отсутствуют

Методы:

* ReadPlayers(string fileName) – считывает весь заданный файл и добавляет все записи о игроках в общих список;

Входные данные: путь до файла (строка)

Выходные данные: -

* ReadTeams(string fileName) – считывает весь заданный файл и добавляет все записи о игроках команды в общих список;

Входные данные: путь до файла (строка)

Выходные данные: -

* SavePlayers(string fileName) – записывает в заданный файл весь общий список содержащий записи о игроках;

Входные данные: путь до файла (строка)

Выходные данные: -

* SaveTeams(string fileName) – записывает в заданный файл весь общий список содержащий записи о игроках команды;

Входные данные: путь до файла (строка)

Выходные данные: -

* SaveReport(string fileName) – записывает в заданный файл весь общий список содержащий записи о кандидатах;

Входные данные: путь до файла (строка)

Выходные данные: -

* Player ParsePlayer(string parse) – обрабатывает строку и получает из неё экземпляр класса Player;

Входные данные: запись о игроке (строка)

Выходные данные: экземпляр класса Player или ошибку

* Player ParseTeam(string parse) – обрабатывает строку и получает из неё экземпляр класса TeamPlayer;

Входные данные: запись о игроке команды (строка)

Выходные данные: экземпляр класса TeamPlayer или ошибку

* String ChooseFile() – открывает пользователю форму выбора файла;

Входные данные: -

Выходные данные: путь до файла (строка)

Класс Node – класс, описывающий узел красно-чёрного дерева

Поля:

− TData Data

− TKey Key

− Color Color

− Node Left

− Node Right

− Node Parent

Методы:

* Node() – конструктор класса Node, создает экземпляр объекта класса;
* Входные данные: -
* Выходные данные: -

Класс RBTree - класс, описывающий красно-чёрное дерево.

Поля:

* Node Root – переменная, хранит указатель на корень дерева.

Методы:

* void Insert(Key itemKey) - Добавляет в дерево новый элемент

Входные данные: экземпляр класса Key

Выходные данные: -

− bool Delete(Key key) - Удаление элемента из дерева

Входные данные: экземпляр класса Key

Выходные данные: false, если элемент не найден. Иначе true.

− bool Delete(Node node) - Удаление элемента из дерева. Выполняет ту же функцию, что и Delete Key. Нужна для удобства.

Входные данные: экземпляр класса Node

Выходные данные: отсутствуют

* void DisplayTree() - Процедура балансировки дерева после удаления.

Входные данные: экземпляр класса Node

Выходные данные: -

* InOrderDisplay(Node current) - Выводит значения элементов по порядку

Входные данные: экземпляр класса Node

Выходные данные: -

* Node Find(Key key)- Поиск элемента в дереве

Входные данные: экземпляр класса Key

Выходные данные: экземпляр класса Node

* List<Node> Find(Key key)- Поиск всех элементов подходящих под условие поиска

Входные данные: экземпляр класса Key

Выходные данные: список экземпляров класса Node

Класс HashTable - класс, описывающий хеш-таблицу.

Поля:

* Int size – переменная, хранит текущую заполненность таблицы;
* int capacity – переменная, хранит максимальную заполненность;
* Data[] arr – массив, состоящий из объектов класса Data
* Bool[] deleted – массив, состоящий из меток типа bool
* Const int startSize – константа, определяющая начальный размер массива;

Методы:

* HashTable() – Конструктор класса HashTable, создаёт экземпляр объекта класса;

Входные данные: -

Выходные данные: объекта класса HashTable

* Void Clear() – метод отчистки хеш-таблицы;

Входные данные: экземпляр класса HashTable

Выходные данные: объект класса HashTable с нулевыми полями

* Int Hash(Key key, int i) – возвращает целое число, являющееся значение хеш-функции, примененной к экземпляру key класса request;

Входные данные: экземпляр key класса Data, размерность хеш-таблицы, номер попытки

Выходные данные: целое число, являющееся значение хеш-функции, и в то же время индексом массива arr;

Пример функции Hash(Key key, int i);

Предположим, что размер хеш-таблицы равен 20, а на вход поступает запись игрока команды из модели предметной области, а именно:

* Void Add(Data K) – метод добавления экземпляра K класса Data в хеш-таблицу;

Входные данные: экземпляр класса Data, экземпляр класса HashTable;

Выходные данные: экземпляр класса HashTable с новым экземпляром K класса Data;

* Void Resize() – метод изменения размера хеш-таблицы и рехеширования;

Входные данные: -

Выходные данные: экземпляр класса HashTable с измененным размером и проведенным рехешированием;

* Void Delete(Data data) – метод удаления экземпляра data класса Data из хеш-таблицы;

Входные данные: экземпляр класса Data, экземпляр класса HashTable;

Выходные данные: экземпляр класса HashTable без экземпляра K класса Data;

* Int Find(Data data) – метод поиска экземпляра data класса Data в хеш-таблице. Возвращает индекс в таблице или -1, если таковой отсутствует.

Входные данные: экземпляр класса Data, экземпляр класса HashTable;

Выходные данные: индекс в хеш-таблице или -1

* Void Print() – метод вывода данных в хеш-таблице на консоль;

Входные данные: экземпляр класса HashTable;

Выходные данные: -

4.2 Описание среды разработки

Интегрированная среда разработки Visual Studio — это стартовая площадка для написания, отладки и сборки кода, а также последующей публикации приложений. Помимо стандартного редактора и отладчика, которые есть в большинстве сред IDE, Visual Studio включает в себя компиляторы, средства автозавершения кода, графические конструкторы и многие другие функции для улучшения процесса разработки.

Выбрана из-за уже присутствующих навыков работы в этой среде, а также из-за наиболее корректной работы с дизайнером Windows Forms.

4.3 Руководство пользователя

На рисунке 10 представлено главное окно программы. При запуске программа автоматически подгружает данные из файла.

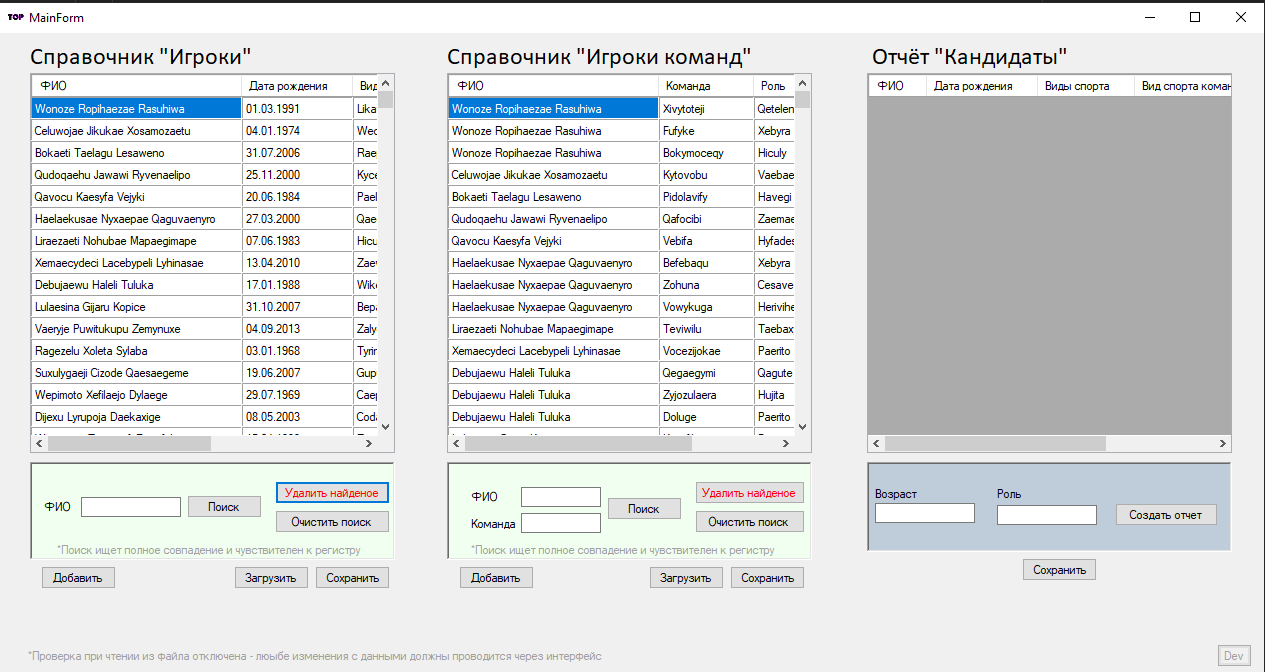


Рисунок 10 – главное окно программы.

Данные сохраняются при нажатии кнопки «Сохранить». Под каждой из таблиц присутствуют соответствующие кнопки.

При нажатии кнопки «Добавить» под таблицей игроков откроется окно добавления игрока (Рисунок 11).

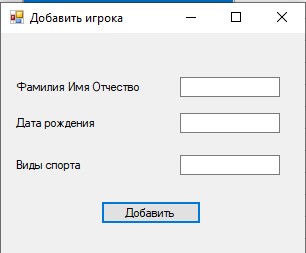


Рисунок 11 – форма добавления игрока.

В текстовые поля ФИО, Дата рождения, Виды спорта пользователю необходимо ввести соответствующие данные, а затем нажать кнопку «Добавить». При некорректности входных данных выплывающее окно уведомит пользователя об этом.

Для выполнения поиска, пользователю необходимо заполнить поле ФИО и нажать кнопку «Поиск». При некорректности входных данных выплывающее окно уведомит пользователя об этом.

Результат поиска появится в таблице выше кнопки поиска.

Для удаления должен быть успешно выполнен поиск, после чего требуется нажать кнопку «Удалить найденное».

При нажатии кнопки «Добавить» под таблицей игроков команды откроется окно добавления игрока (Рисунок 12).

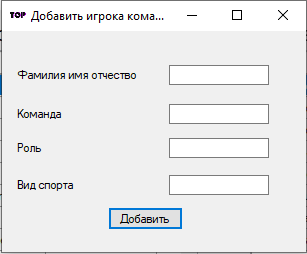


Рисунок 12 – Форма добавления игрока команды

В текстовые поля ФИО, Команда, Роль, Вид спорта пользователю необходимо ввести соответствующие данные, а затем нажать кнопку «Добавить». При некорректности входных данных выплывающее окно уведомит пользователя об этом.

Для выполнения поиска, пользователю необходимо заполнить поля ФИО и Команда, и нажать кнопку «Поиск». При некорректности входных данных выплывающее окно уведомит пользователя об этом.

Результат поиска появится в таблице выше кнопки поиска.

Для удаления должен быть успешно выполнен поиск, после чего требуется нажать кнопку «Удалить найденное».

Для создания отчета «Кандидаты» требуется заполнить поля Возраст и Роль, после нажать на кнопку «Создать отчет». При некорректности входных данных выплывающее окно уведомит пользователя об этом. Результат появится в таблице выше самой кнопки.

4.4 Тестирование

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Описание тестовой ситуации | | Входные данные | | Выходные данные | |
| Хеш-таблица (метод открытой адресации) | (ФИО, Дата рождения, Виды спорта) | Хеш-таблица (метод открытой адресации) | Сообщения |
| Добавление | | | | | |
| 1 | Добавление некорректных данных | - | Vlad Dak; 01.02.2003; Swim | - | Неправильное ФИО |
| - | Vlad Dak Toster; 01.02.0001; Swim | - | Недействительная дата |
| 2 | Добавление корректных данных | - | Vlad Dak Toster; 01.02.2001; Swim | - | Игрок успешно добавлен |
| 3 | Проверка уникальности ключа | Vlad Dak Toster; 01.02.2001; Swim | Vlad Dak Toster; 01.02.2001; Swim | Vlad Dak Toster; 01.02.2001; Swim | Такой игрок уже есть |
| 4 | Добавление при коллизии | Vlad Dak Toster; 01.02.2001; Swim | Dlav Dak Toster; 01.02.2001; Swim | Vlad Dak Toster; 01.02.2001; Swim  && Dlav Dak Toster; 01.02.2001; Swim | Игрок успешно добавлен |
| Поиск | | | | | |
| 5 | Запись не существует | - | Vlad Dak Toster; 01.02.2001; Swim | - | Не найдено ни одной записи |
| 6 | Запись существует | Vlad Dak Toster; 01.02.2001; Swim | Vlad Dak Toster; 01.02.2001; Swim | Vlad Dak Toster; 01.02.2001; Swim | Vlad Dak Toster; 01.02.2001; Swim |
| 7 | Поиск при коллизии | Vlad Dak Toster; 01.02.2001; Swim  && Dlav Dak Toster; 01.02.2001; Swim | Dlav Dak Toster; 01.02.2001; Swim | Vlad Dak Toster; 01.02.2001; Swim  && Dlav Dak Toster; 01.02.2001; Swim | Dlav Dak Toster; 01.02.2001; Swim |
| Удаление | | | | | |
| 8 | Запись не существует | - | Vlad Dak Toster; 01.02.2001; Swim | - | Не найдено ни одной записи |
| 9 | Запись существует | Vlad Dak Toster; 01.02.2001; Swim | Vlad Dak Toster; 01.02.2001; Swim | - | Запись успешно удалена |
| 10 | Удаление при коллизии | Vlad Dak Toster; 01.02.2001; Swim  && Dlav Dak Toster; 01.02.2001; Swim | Dlav Dak Toster; 01.02.2001; Swim | Vlad Dak Toster; 01.02.2001; Swim | Запись успешно удалена |

Таблица 1 – Тестирование Хеш-таблицы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Описание тестовой ситуации | | Входные данные | | Выходные данные | |
| Красно-чёрное дерево | (ФИО, Название команды, Роль, Вид спорта) | Красно-чёрное дерево | Сообщения |
| Добавление | | | | | |
| 1 | Добавление некорректных данных | - | vlad Dak Toster; Squad; Monster; Swim | - | Неправильное ФИО |
| - | Vlad Dak Toster; Squad; Monster | - | Не соответствие количества аргументов |
| - | Vlad Dak; Squad; Monster; Swim | - | Неправильное ФИО |
| - | Vlad Dak Toster; Squad; Monster; Swim, Golf | - | Неправильный вид спорта |
| 2 | Добавление корректных данных | - | Vlad Dak Toster; Squad; Monster; Golf | Vlad Dak Toster; Squad; Monster; Golf | Игрок команды успешно добавлен |
| 3 | Проверка уникальности ключа | Vlad Dak Toster; Squad; Monster; Golf | Vlad Dak Toster; Squad; Monster; Golf | Vlad Dak Toster; Squad; Monster; Golf | Такой игрок команды уже есть |
| 4 | Добавление при коллизии | Vlad Dak Toster; Squad; Monster; Golf | Dlav Dak Toster; Squad; Monster; Golf | Vlad Dak Toster; Squad; Monster; Golf  &&  Dlav Dak Toster; Squad; Monster; Golf | Игрок команды успешно добавлен |
| Поиск | | | | | |
| 5 | Запись не существует | - | Vlad Dak Toster; Squad; Monster; Golf | - | Не найдено ни одной записи |
| 6 | Запись существует | Vlad Dak Toster; Squad; Monster; Golf | Vlad Dak Toster; Squad; Monster; Golf | Vlad Dak Toster; Squad; Monster; Golf | Vlad Dak Toster; Squad; Monster; Golf |
| 7 | Поиск при коллизии | Vlad Dak Toster; Squad; Monster; Golf  &&  Dlav Dak Toster; Squad; Monster; Golf | Dlav Dak Toster; Squad; Monster; Golf | Vlad Dak Toster; Squad; Monster; Golf  &&  Dlav Dak Toster; Squad; Monster; Golf | Dlav Dak Toster; Squad; Monster; Golf |
| Удаление | | | | | |
| 8 | Запись не существует | - | Vlad Dak Toster; Squad; Monster; Golf | - | Не найдено ни одной записи |
| 9 | Запись существует | Vlad Dak Toster; Squad; Monster; Golf | Vlad Dak Toster; Squad; Monster; Golf | - | Запись успешно удалена |
| 10 | Удаление при коллизии | Vlad Dak Toster; Squad; Monster; Golf  &&  Dlav Dak Toster; Squad; Monster; Golf | Dlav Dak Toster; Squad; Monster; Golf | Vlad Dak Toster; Squad; Monster; Golf | Запись успешно удалена |

Таблица 2 – Тестирование Красно-Чёрного дерева

# Заключение

Целью курсового проекта было: разработка информационной системы «Teams Of Players». Подсистема работы с справочниками «Игроки» и «Игроки команды», а также отчетом «Кандидаты».

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

1. Проведён анализ предметной области «Приложения для спортивных знакомств»;
2. Изучены теоретические основы методов построения справочников;
3. Определены требования к информационной системе;
4. Информационная система была реализована и протестирована;
5. Изучен язык разработки C# версии 9.0;
6. Приобретены навыки для работы с Windows Forms
7. Изучена среда разработки Rider JetBrains

# Список литературы

1. Томас Х. Кормен, Чарль И. Лейзерсон, Роналд Л. Ривест, Клиффорд Штайн, Алгоритмы: построение и анализ, 3-е изд.: Пер. с англ – М.: ООО «И.Д.Вильямс», 2013. – 1328с.:ил. парал. тит. Англ
2. Paul E. Black, “associative array”, Dictionary of Algorithms and Data Structures, изм. 2004
3. Седжвик Р. Фундаментальные алгоритмы на C++. Анализ/Структура данных/Сортировка/Поиск – К.:Диасофт, 2001. – 688с
4. <https://neerc.ifmo.ru/wiki>