Пермский филиал федерального государственного автономного  
образовательного учреждения высшего образования  
«Национальный исследовательский университет  
«Высшая школа экономики»

*Факультет социально-экономических и компьютерных наук*

Мальшаков Александр Аркадьевич

**Самостоятельная работа**

студента образовательной программы «Разработка информационных систем для бизнеса» по направлению подготовки 38.03.05-Бизнес - информатика

Руководитель

Старший преподаватель кафедры ИТБ \_\_\_\_\_\_\_\_\_\_\_\_

Дацун Н. Н.

Пермь, 2025 год

# Оглавление

[Оглавление 2](#_Toc191319430)

[Запись 4](#_Toc191319431)

[Исходная табличка 5](#_Toc191319432)

[Массив 6](#_Toc191319433)

[Функция ввода данных 6](#_Toc191319434)

[С консоли 7](#_Toc191319435)

[С файла 7](#_Toc191319436)

[Случайно 8](#_Toc191319437)

[Вывод элементов массива 8](#_Toc191319438)

[Сортировка массива 9](#_Toc191319439)

[Функция поиска элементов по значениям ключевых атрибутов 10](#_Toc191319440)

[Бинарный поиск 10](#_Toc191319441)

[Функция редактирования записи в массиве 12](#_Toc191319442)

[Функция удаления элемента из массива 13](#_Toc191319443)

[Бинарное дерево 15](#_Toc191319444)

[Функция добавления записи в дерево 15](#_Toc191319445)

[Функция вывода данных 17](#_Toc191319446)

[Функция поиска элементов по значениям ключевых атрибутов 18](#_Toc191319447)

[Функция удаления записи из дерева 19](#_Toc191319448)

[Линейный список 23](#_Toc191319449)

[Функция добавление с упорядочиванием 23](#_Toc191319450)

[Просмотр записей в порядке ввода 24](#_Toc191319451)

[Просмотр записей в отсортированном порядке 25](#_Toc191319452)

[Поиск записи по значениям ключевых атрибутов 26](#_Toc191319453)

[Удаление записи 27](#_Toc191319454)

[Дополнительно 31](#_Toc191319455)

[Таблица 31](#_Toc191319456)

[Хеш Таблица 31](#_Toc191319457)

[Интерфейс 33](#_Toc191319458)

[Самооценка 35](#_Toc191319459)

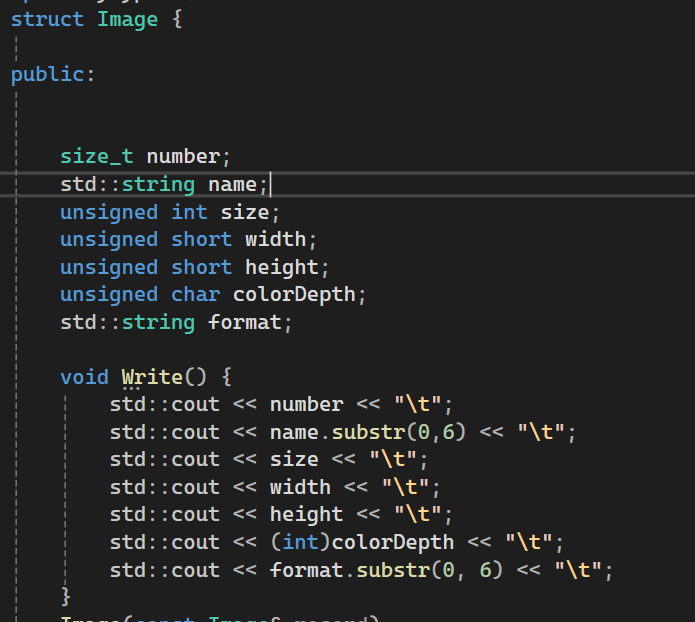
[Библиографический список 36](#_Toc191319460)

[Приложение А 37](#_Toc191319461)

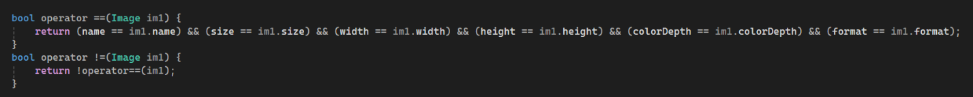
# Запись

Запись представляет собой структуру, хранящую номер, имя, размер в килобайтах, ширину, высоту, глубину цвета и формат картинки в базе данных. Для размера выбран тип unsigned int, так с помощью него можно задать картинке размер в 4 терабайта, изображений большего размера в мире не очень много. Для высоты и ширины выбран тип unsigned short так как картинки размером примерно 65тыс или больше пикселей очень-очень редки. Для глубины цвета использован один байт так как при глубине цвета больше 32 человеческий глаз перестает замечать разницу.

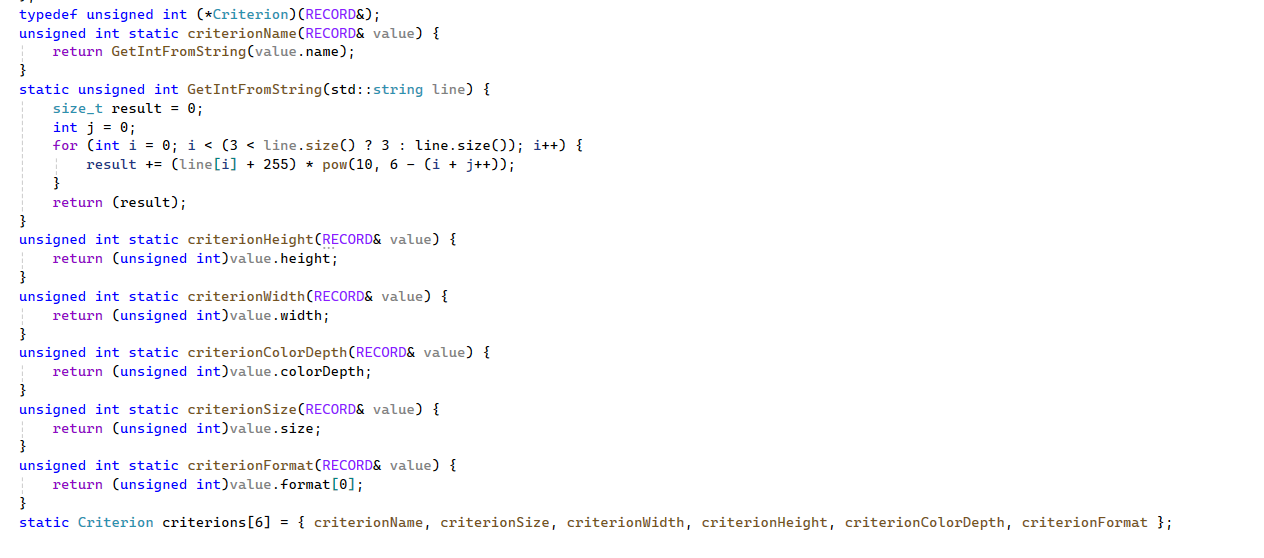
В записи есть также функция вывода, длинные названия сокращаются до 6 символов для красивого вывода.



Функция сравнения записи сравнивает все ее элементы.



Для того чтобы из записи получать ключи были написаны 6 функций. Все они позволяют сравнивать между собой ключи записи возвращая целое без знакового число, так как это самое большое поле в записи, и к нему все остальные будут приводиться без потери данных.



Ключ имени преобразуется в unsigned int следующим образом:

Суммируются первые 3 кода букв умноженные на 10 в степени их позиции, так сортировка будет лексикографической, коллизии могу происходить, но это достаточно редко.

## Исходная табличка

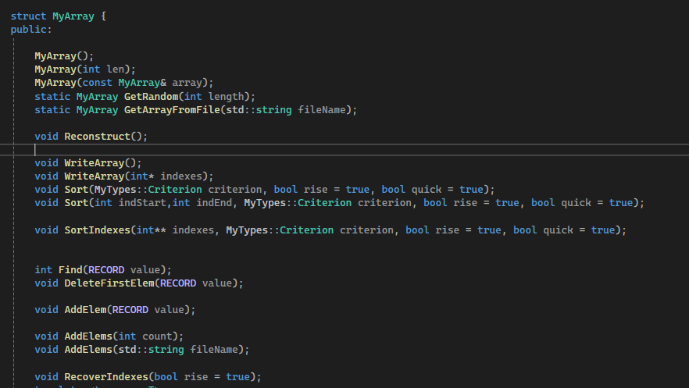
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Номер** | **Имя** | **Размер (кб)** | **Ширина** | **Высота** | **Глубина цвета** | **Формат** |
| 1 | Кот | 451 | 1262 | 713 | 8 | png |
| 2 | Собака на траве | 317 | 1400 | 1050 | 24 | jpg |
| 3 | Луна | 26 | 736 | 615 | 24 | jpg |
| 4 | Эйфелева башня | 428 | 3000 | 1688 | 32 | webp |
| 5 | Кремль | 165 | 1280 | 853 | 24 | jpg |
| 6 | Бабочка | 437 | 823 | 630 | 32 | jpg |
| 7 | Клубок | 430 | 820 | 635 | 32 | jpg |
| 8 | арахис | 45 | 100 | 100 | 32 | webp |
| 9 | Человек | 470 | 200 | 1630 | 32 | jpg |
| 10 | Шелёпа | 900 | 1280 | 1000 | 16 | png |
| 11 | Соболь | 300 | 1800 | 1350 | 8 | jpg |
| 12 | Карандаш | 200 | 3200 | 3223 | 8 | webp |
| 13 | Тополь | 500 | 456 | 355 | 16 | webp |

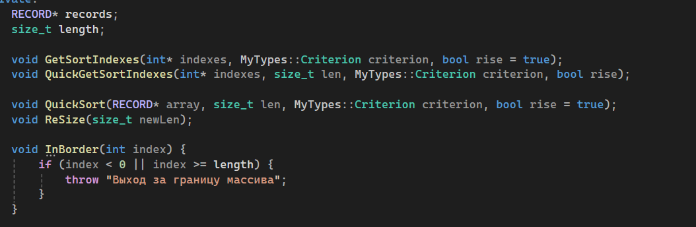
Ключи – имя и высота картинки.

# Массив

Массив хранит саму таблицу, список и дерево хранят указатели на массив так как это удобнее для вывода и проектирования функций, а также это не занимает больше памяти чем хранить индексы массива. Минус в том, что нужно заново создавать деревья и списки при изменении размера массива, но это проблема бы решилась если бы использовались умные указатели.

Массив представляет собой структуру, которая хранит указатель на массив записей, а также их количество и методы для работы с ними.

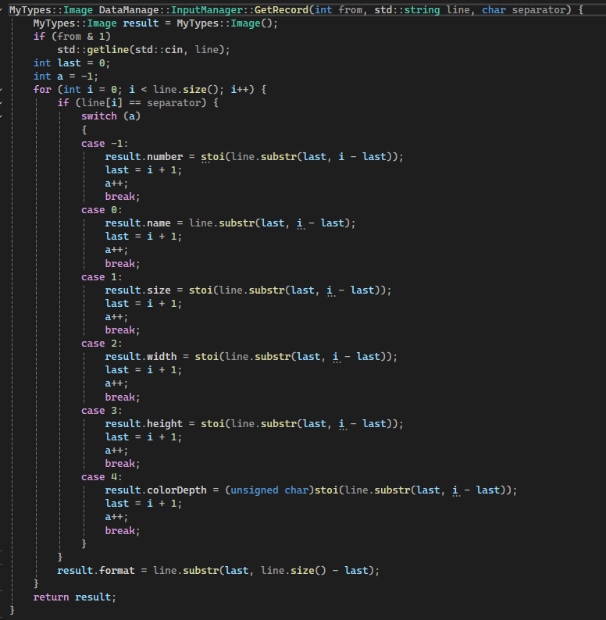




## Функция ввода данных

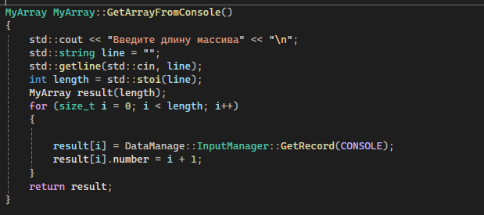
Для ввода массива были спроектированы 3 функции ввода, с консоли, с файла, с помощью генератора псевдослучайных чисел.

Для преобразования строки в запись была спроектирована функция.



### С консоли

Функция ввода массива с консоли.

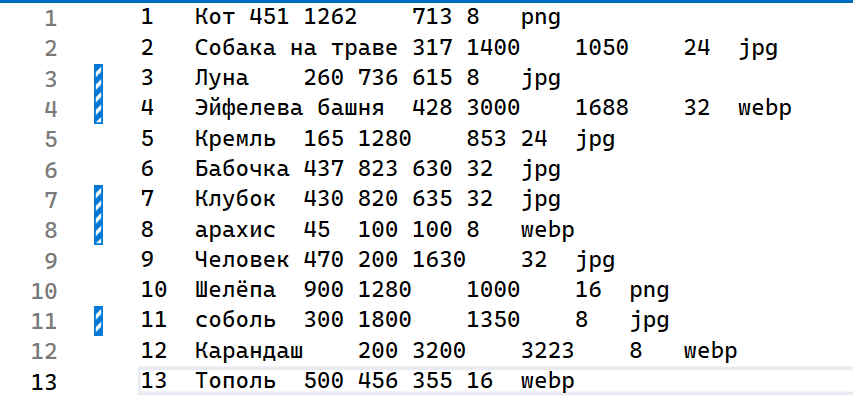


### С файла

Функция ввода массива с файла.

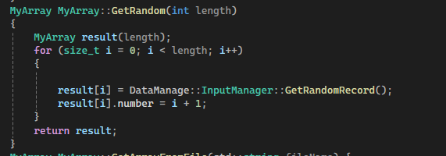


Сам файл, поля записи разделены символом “\t“.



### Случайно

Функция получения массива псевдо случайно.



Функция получения случайной записи.

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

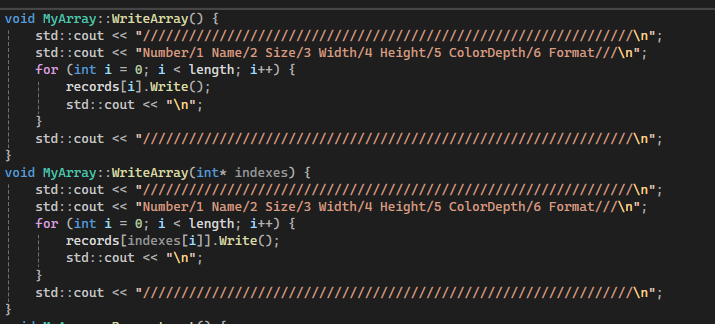
Функция получения случайной строки

Изображение выглядит как текст, снимок экрана, Шрифт

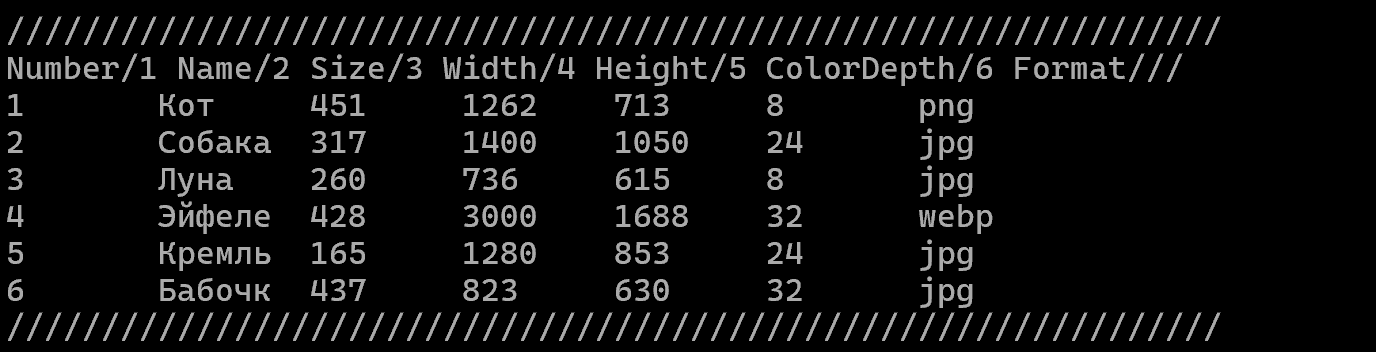
Контент, сгенерированный ИИ, может содержать ошибки.

## Вывод элементов массива

Для вывода массива были спроектированы 2 функции, одна выводит массив просто как таблицу, другая выводит массив по массив индексов.



Пример работы функции, выводящей просто массив.



Пример работы функции, выводящей по индексам (отсортировано по именам в обратном порядке).

Изображение выглядит как текст, снимок экрана, Шрифт, черный

Контент, сгенерированный ИИ, может содержать ошибки.

## Сортировка массива

Были созданы 2 метода, для быстрой сортировки и сортировки пузырьком. Они оба принимают пустой указатель на индексы и сортируют его.

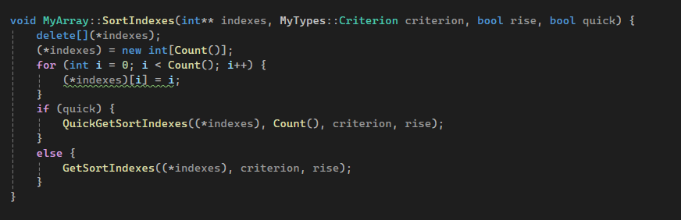


Сортировка пузырьком.

Изображение выглядит как текст, программное обеспечение, Шрифт, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Для создания указателя на индексы и передачи его в эти функции есть дополнительная функция.



Вывод массива по отсортированным индексам (сортировка оп высоте).

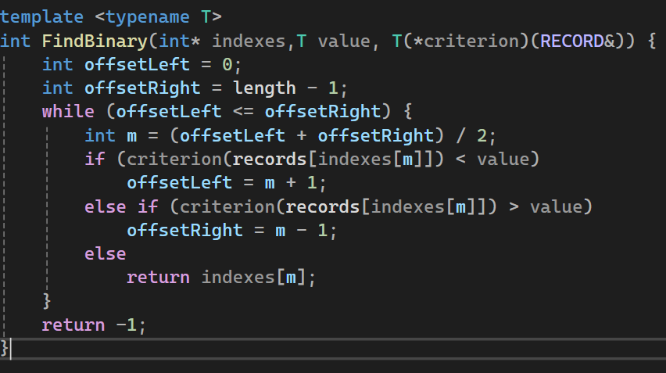
Изображение выглядит как текст, снимок экрана, Шрифт, черный

Контент, сгенерированный ИИ, может содержать ошибки.

## Функция поиска элементов по значениям ключевых атрибутов

### Бинарный поиск

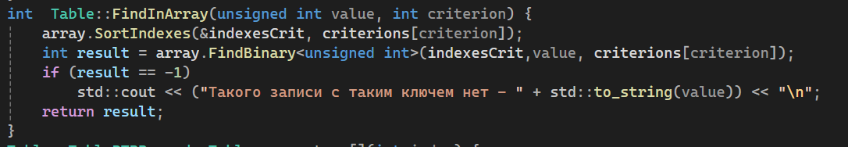
Для бинарного поиска спроектирована функция ниже итерационная и рекурсивная. Она принимает указатель на функцию для того, чтобы из записи получать ключ, по которому она будет искать, а также отсортированные индексы.



**Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.**

Функция, которая находит элемент в массиве, ниже. Так как в записи все ключи приводятся к типу unsigned int, то она передает в функцию значения типа unsigned int.



Ниже пример работы.

Изображение выглядит как текст, Шрифт, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

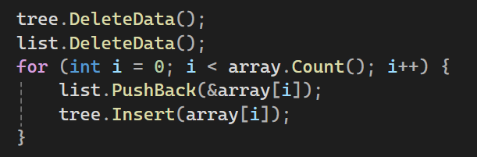
Изображение выглядит как текст, снимок экрана, Шрифт, черный

Контент, сгенерированный ИИ, может содержать ошибки.

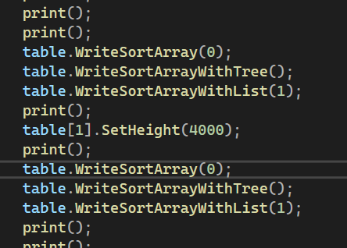


## Функция редактирования записи в массиве

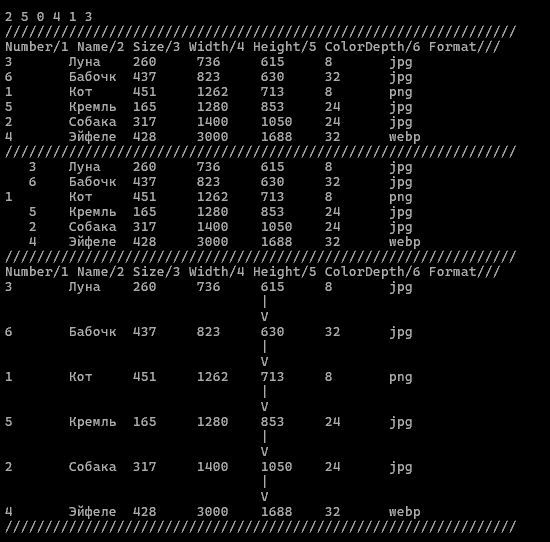
При редактировании нужно изменять индексы не только массива, а также дерева и списка. При редактировании записи будут заново создаваться списки и деревья.



Код.



Массив, дерево и список до изменения значения высоты у 2 записи.



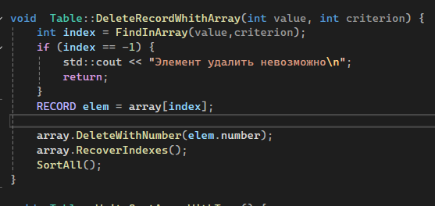
После изменения 2 записи.

Изображение выглядит как снимок экрана, текст

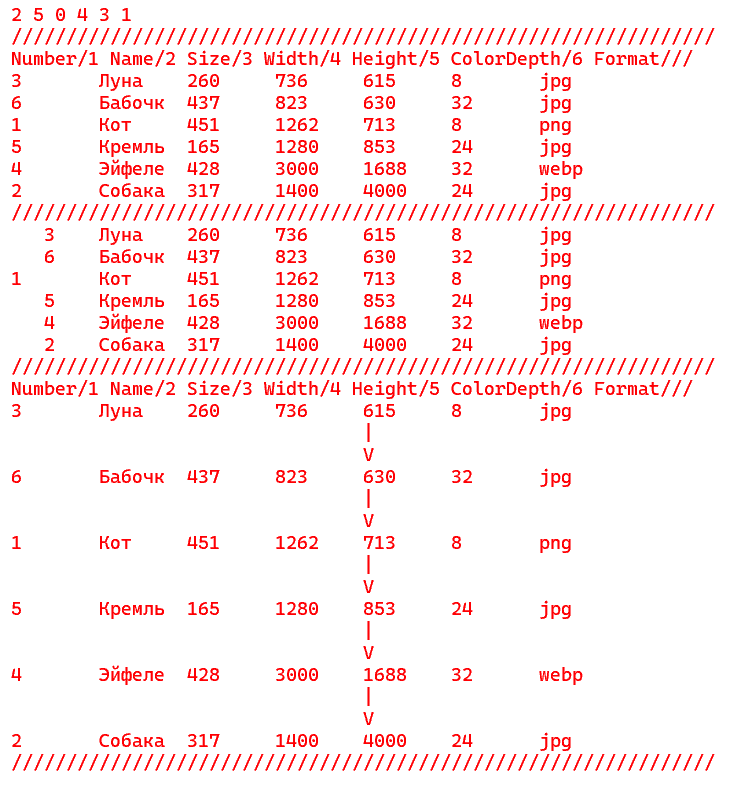
Контент, сгенерированный ИИ, может содержать ошибки.

## Функция удаления элемента из массива

Для удаления также надо заново создавать дерево и список, так как массив копирует все элементы в новую память и старые указатели начинают указывать на мусор. Функция удаления находит элемент в массиве с помощью предыдущей функции, потом удаляет его по индексу, восстанавливает номера записей и заново сортирует дерево и список.



До удаления элемента из массива с 713 высотой.



После

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

# Бинарное дерево

Дерево представляет собой структуру, хранящую корень – изначальный узел, а также функции для работы с ним. Для дерева выбран тип B-Tree для реализации балансировки. У B дерева есть порядок для него использовался шаблон, который позволяет задать переменную как константу при указании типа. За основу были взяты функции из статьи на хабре[1], но в них было несколько ошибок, которые пришлось исправлять.

Узел хранит массив детей, а также массив указателей на записи.



## Функция добавления записи в дерево

Принимает ссылку на запись, ищет лист если в листе есть место ставит запись туда, если же нет, то в функции Restruct, отправляет элемент в узел выше и делит текущий узел на 2, и присваивает эти 2 узла верхнему.

Изображение выглядит как текст, линия, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

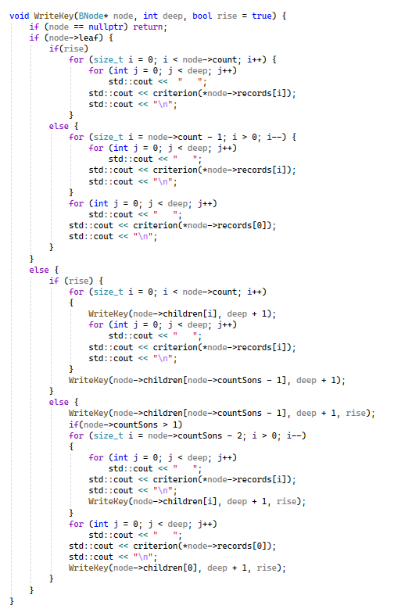
И так делает рекурсивно пока не останется переполненных узлов.

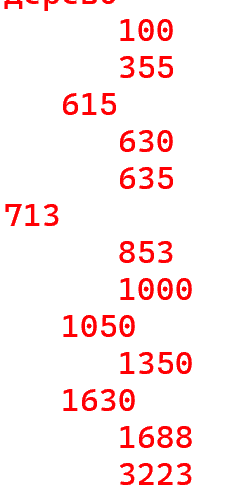
Изображение выглядит как текст, снимок экрана, документ, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

## Функция вывода данных

Функция для вывода только ключей из дерева.





Функция для вывода записей из дерева.

Изображение выглядит как текст, снимок экрана, Шрифт, документ

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, Шрифт, число, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

## Функция поиска элементов по значениям ключевых атрибутов

Итерационная версия поисковой функции.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рекурсивная

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

**Пример**

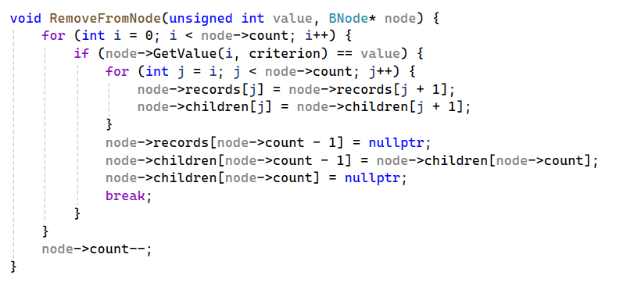
****

## Функция удаления записи из дерева

Для начала функция удаления проверяет нахождение элемента в дереве. После либо удаляет элемент из массива узла, либо из листа, либо из произвольного узла.



Метод удаления из массива узла.

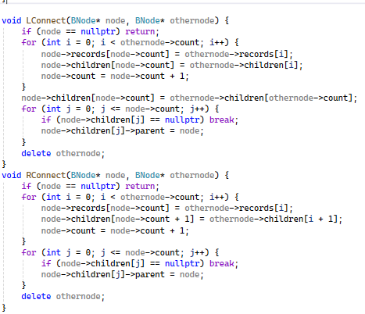


Метод удаления из листа.

Изображение выглядит как текст, снимок экрана, документ, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Методы левого и правого присоединения



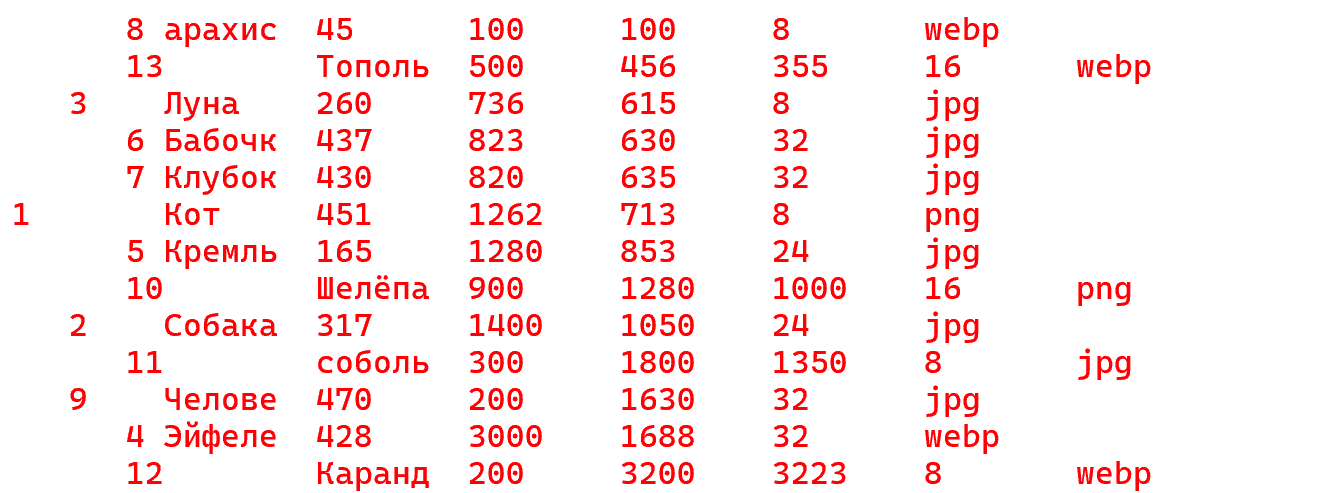
И метод починки узла.

Изображение выглядит как текст, снимок экрана, документ, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

**Пример работы**

До удаления элемента с высотой 355



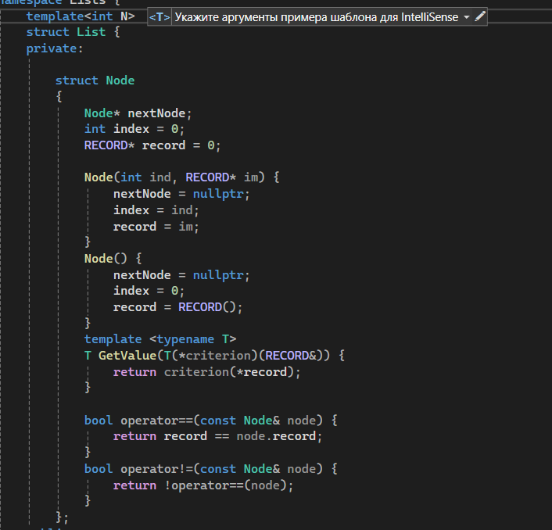
После удаления элемента с высотой 355

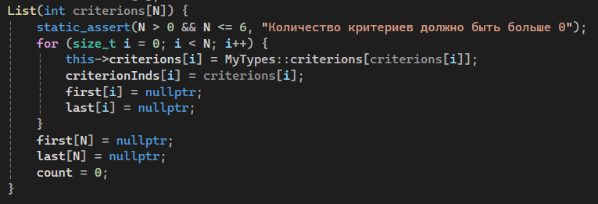
Изображение выглядит как текст, Шрифт, снимок экрана, число

Контент, сгенерированный ИИ, может содержать ошибки.

# Линейный список

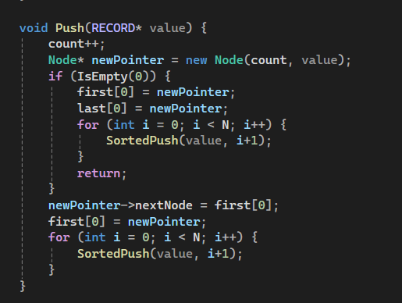
Список — это структура, хранящая массив из N указателей на первые узлы и на последние. Каждый указатель отвечает за сортируемый ключ, это удобнее чем если бы каждый узел указывал бы на 3 потомков, а памяти занимает столько же. Ключи передаются при создании списка, количество ключей определяется шаблоном.





## Функция добавление с упорядочиванием

Для начала функция принимает ссылку на запись, и либо вставляет в начало списка, либо в конец (функции Push и PushBack).

 Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

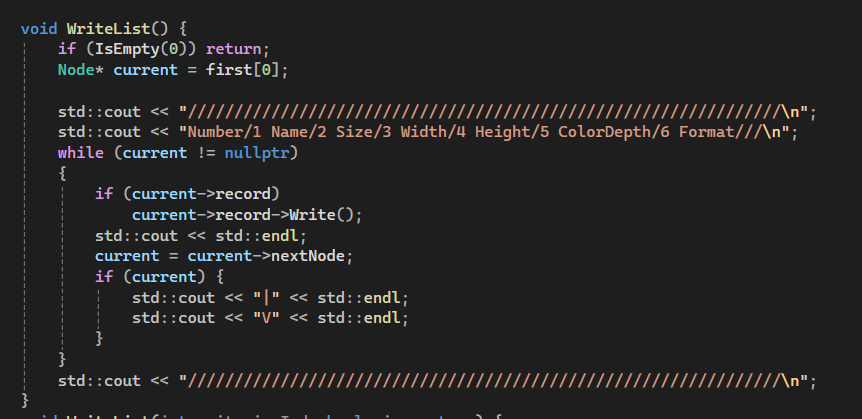
Дальше эти функции передают элемент в сортировочные функции для сортировки по ключам.



Так получаются N + 1 указатель, каждый отсортированный по своему ключу.

## Просмотр записей в порядке ввода

Нулевой указатель всегда указывает на элементы в порядке вставки. Ниже функция для вывода.

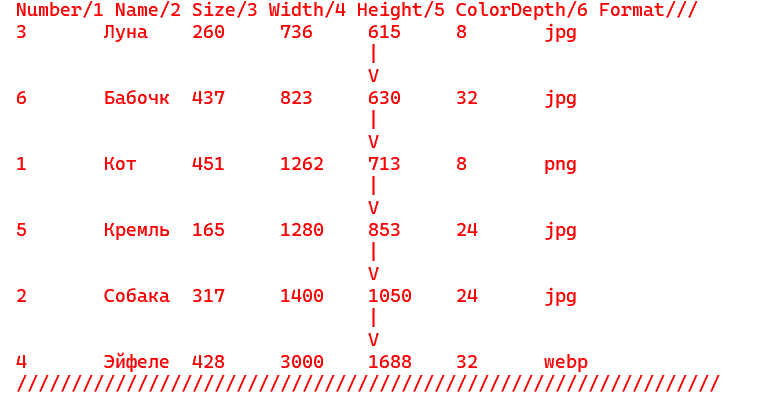


## Просмотр записей в отсортированном порядке

Для сортированного вывода нужно также вывести все элементы, но теперь уже не от 0-го указателя, для вывода по убыванию нужна функция переворачивания списка.



Список выводит записи со стрелочками, которые показывают на отсортированные ключи.



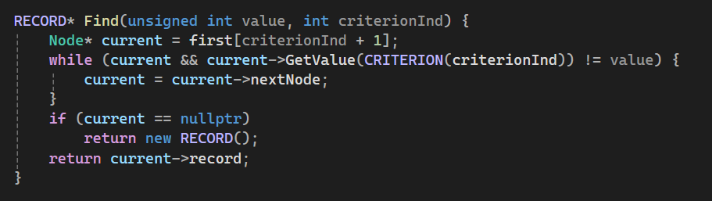
Функция переворачивания списка, итерационная и рекурсивная.

Изображение выглядит как текст, снимок экрана, Шрифт

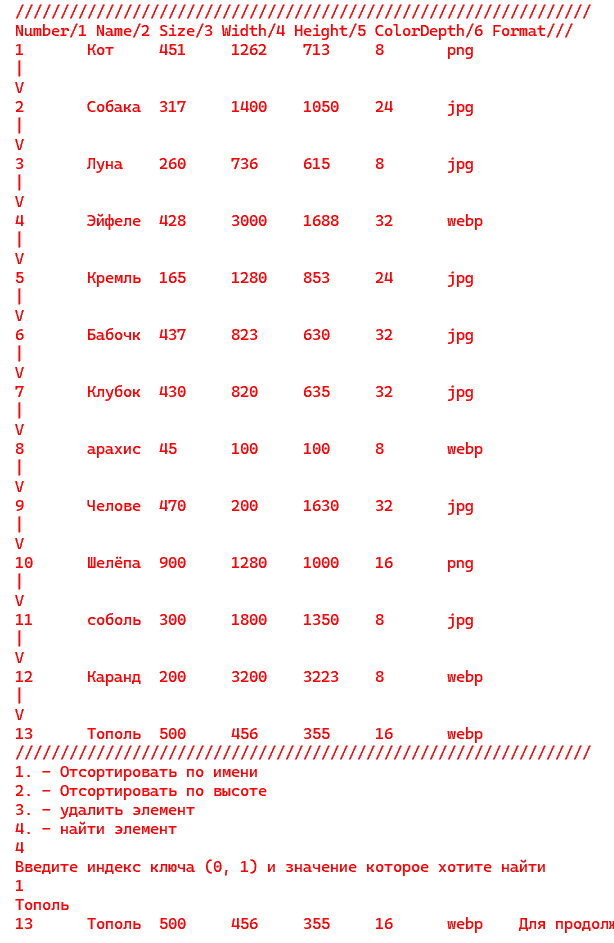
Контент, сгенерированный ИИ, может содержать ошибки. Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

## Поиск записи по значениям ключевых атрибутов



**Пример**



## Удаление записи

Чтобы удалить запись для начала ее нудно найти, потом удалить последовательно из всех списков по ключам.

Изображение выглядит как текст, снимок экрана, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Функции удаления первого/последнего элемента.

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Функция удаления узла из всех под списков.

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

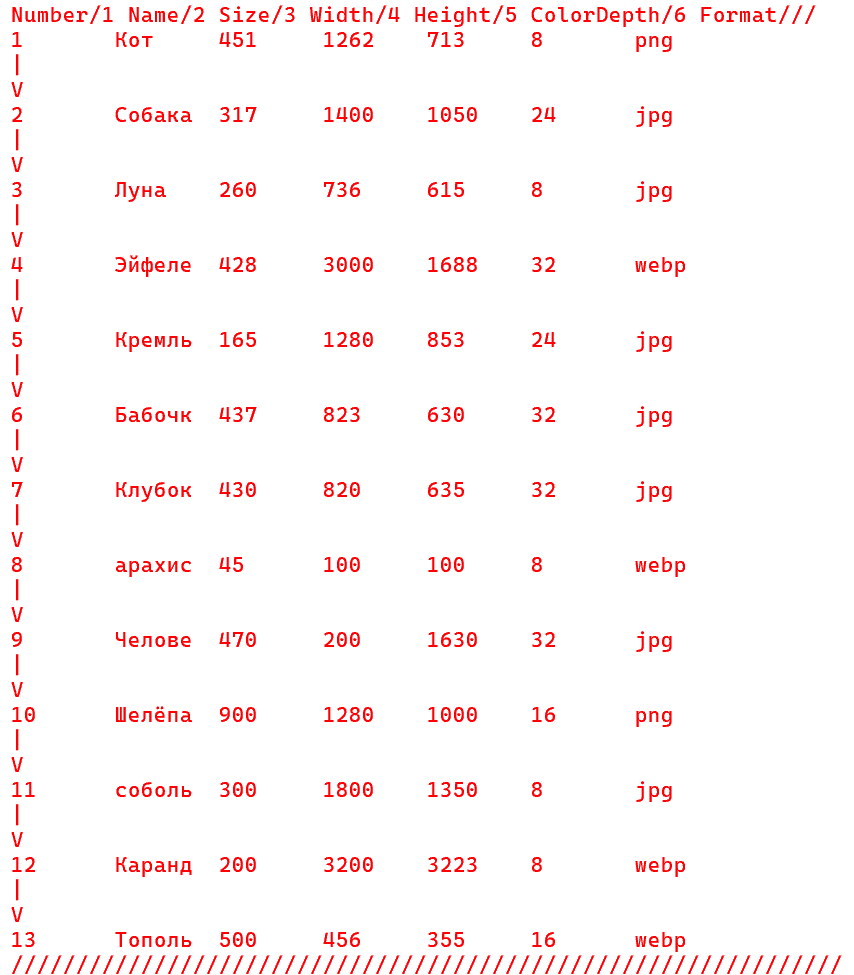
Функция удаления первого/последнего узла из подсписка.

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

**Пример**

До удаления элемента с высотой 3223



После удаления элемента с высотой 3223

Изображение выглядит как текст, снимок экрана, Шрифт, меню

Контент, сгенерированный ИИ, может содержать ошибки.

# Дополнительно

## Таблица

Для объединения массива, дерева и списка сделана структура таблицы.



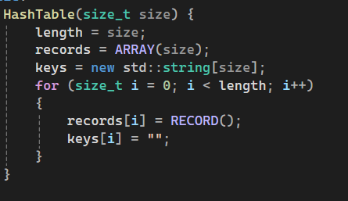
Через нее идет работа с 3 структурами.

В ней также есть структура табличного указателя на запись для того, чтобы, когда изменяли значение в записи, происходила сортировка всех структур.

Функции таблицы — это комбинации функций структур.

## Хеш Таблица

Хеш таблица сделана и работает, но ее негде применить.



Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Для того чтобы положить запись она высчитывает хэш строки, если есть колизия идет на следующую ячейку, если она так прошлась по всей таблице и не нашла куда вставить, то функция ничего не делает.

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

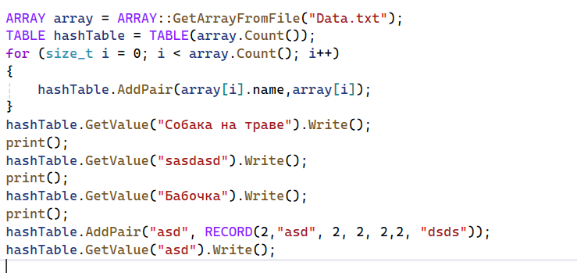
Контент, сгенерированный ИИ, может содержать ошибки.

Аналогичная функция получения только после она возвращает значение если оно было найдено, если же нет то возвращает пустую запись.

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Пример работы, ниже код для теста.



Как видно она находит элемент по названию с помощью хэша.

Изображение выглядит как текст, Шрифт, красный

Контент, сгенерированный ИИ, может содержать ошибки.

## Интерфейс

Интерфейс позволяет просматривать таблицу через массив, список, дерево по 2 критериям – высоте и имени. Также из каждой структуры можно удалить или найти элементы.

Блок для массива



Блок для дерева

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

Блок для списка

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

# Самооценка

Сделаны все функции, которые были в задании, +ключи есть возможность поменять 2 действиями в коде программе + балансировка + хеш-таблица + некоторые функции рекурсивно и итерационно + интерфейс – мало проверок на правильный ввод в функциях и в интерфейсе

9

# Библиографический список

[1] Хабр “Сбалансированное дерево поиска B-Tree” [Электронный ресурс] -URL: https://habr.com/ru/articles/337594/(дата обращения: 20.02.2025)

# Приложение А

<https://github.com/Poruch/LabWorks/tree/main/IndividualWorkTOI> гит хаб