



Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное  
учреждение

высшего образования

«Московский государственный технический университет

имени Н.Э. Баумана

(национальный исследовательский университет)»

(МГТУ им. Н.Э. Баумана)

---

Факультет «Информатика и системы управления»

Кафедра «Программное обеспечение ЭВМ и информационные технологии»

## ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №6 «ОБРАБОТКА ДЕРЕВЬЕВ»

Студент: Лысцев Никита Дмитриевич

Группа: ИУ7-33Б

Студент

\_\_\_\_\_  
*подпись, дата*

Лысцев Н.Д

*фамилия, и.о.*

Преподаватель

\_\_\_\_\_  
*подпись, дата*

Барышникова М.Ю.

*фамилия, и.о.*

Оценка \_\_\_\_\_

2022 г

## Оглавление

1. Цель работы.....	3
2. Описание условия задачи .....	3
3. Описание технического задания .....	3
4. Описание внутренних структур данных .....	5
5. Оценка эффективности работы алгоритмов .....	11
6. Выводы .....	11
7. Набор тестов.....	12
8. Ответы на контрольные вопросы.....	14

# 1. Цель работы

Цель работы – получить навыки применения двоичных деревьев, реализовать основные операции над деревьями: обход деревьев, включение, исключение и поиск узлов.

# 2. Описание условия задачи

Построить дерево в соответствии со своим вариантом задания. Вывести его на экран в виде дерева. Реализовать основные операции работы с деревом: обход дерева, включение, исключение и поиск узлов. Сравнить эффективность алгоритмов сортировки и поиска в зависимости от высоты деревьев и степени их ветвления.

В текстовом файле содержатся целые числа. Построить двоичное дерево из чисел файла. Вывести его на экран в виде дерева. Используя процедуру, определить количество узлов дерева на каждом уровне. Добавить число в дерево и в файл. Сравнить время добавления чисел в указанные структуры.

# 3. Описание технического задания

## 3.1. Входные данные

Целое число от 0 до 9 – номер пункта меню, позволяющий пользователю осуществить работу с бинарным деревом. Элемент дерева – целое число.

Для работы программы также можно загрузить данные из заранее подготовленного текстового файла.

## 3.2. Выходные данные

Выходными данными в зависимости от пунктов меню являются либо визуализация дерева в виде «png» картинки, либо вывод дерева на экране

консоли, либо результат сравнения времени добавления вершины в файл и в дерево.

### 3.3. Описание задачи, реализуемой программой

Задача программы – обеспечение работы с такой структурой данных, как бинарное дерево.

### 3.4. Способ обращение к программе

Ввод и вывод всех данных осуществляется через консоль.

Также для быстрого заполнения бинарного дерева есть возможность загрузить данные из заранее подготовленного текстового файла.

Программа собирается с помощью специального make-файла, и может быть запущена в командной строке с помощью слова make.

### 3.5. Описание возможных аварийных ситуаций и ошибок пользователя

Аварийные ситуации:

- Пустое поле ввода – ничего не произойдет, программа будет ждать ввода пользователя;
- Удаление вершины из пустого дерева – ничего не произойдет, будет выведено сообщение о том, что дерево пустое;
- Ввод имени файла, которого не существует – программа сгенерирует файл с данными с введенным названием.

Ошибки пользователя:

- Некорректный ввод: ввод неверного пункта меню, ввод не валидных данных при добавлении в дерево и при удалении из дерева.

## 4. Описание внутренних структур данных

Реализация вершины бинарного дерева описывается структурой «vertex\_t»

```
typedef struct vertex_t vertex_t;  
  
struct vertex_t  
{  
    int data;  
    vertex_t *left; // меньше  
    vertex_t *right; // больше  
};
```

Листинг 1. Структура «vertex\_t», содержащая описание одной вершины

Поля структуры «vertex\_t»:

1. data – целое число, данные вершины дерева;
2. left – указатель на левого (меньшего по значению) потомка;
3. right – указатель на правого (большего по значению) потомка.

Бинарное дерево описывается структурой «tree\_t».

```
typedef struct tree_t tree_t;  
  
struct tree_t  
{  
    vertex_t *root;  
};
```

Листинг 2. Реализация бинарного дерева структурой «tree\_t»

Поля структуры «tree\_t»:

1. root – корень дерева.

## 4.1. Описание алгоритма

Взаимодействие пользователя с программой осуществляется через консоль с помощью специального меню, в котором пользователю предлагается выполнить то, или иной действие.

```
-----|
Программа для обработки бинарного дерева
Вершиной дерева является целое число.

Правила:
- добавить можно только уникальное число,
  добавление дубликата будет проигнорировано;

Операции для обработки дерева:
1 - прочитать данные из файла;
2 - добавить в дерево число;
3 - добавить число в файл с данными;
4 - удалить число из дерева;
5 - вывести бинарное дерево на экран;
6 - визуализировать бинарное дерево в виде png картинки;
7 - определить количество узлов на каждом уровне;
8 - сравнить время добавления чисел в дерево и в файл;
9 - очистить дерево;

0 - выйти из программы.

-----|
Выберите пункт меню:
```

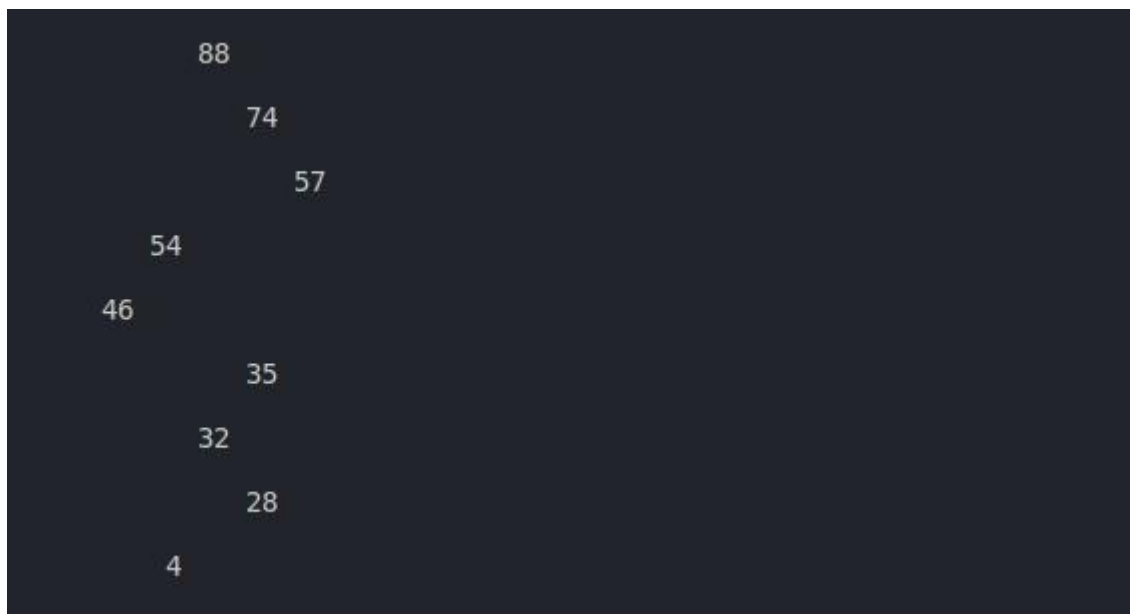
Листинг 3. Меню взаимодействия

- Примеры работы программы:



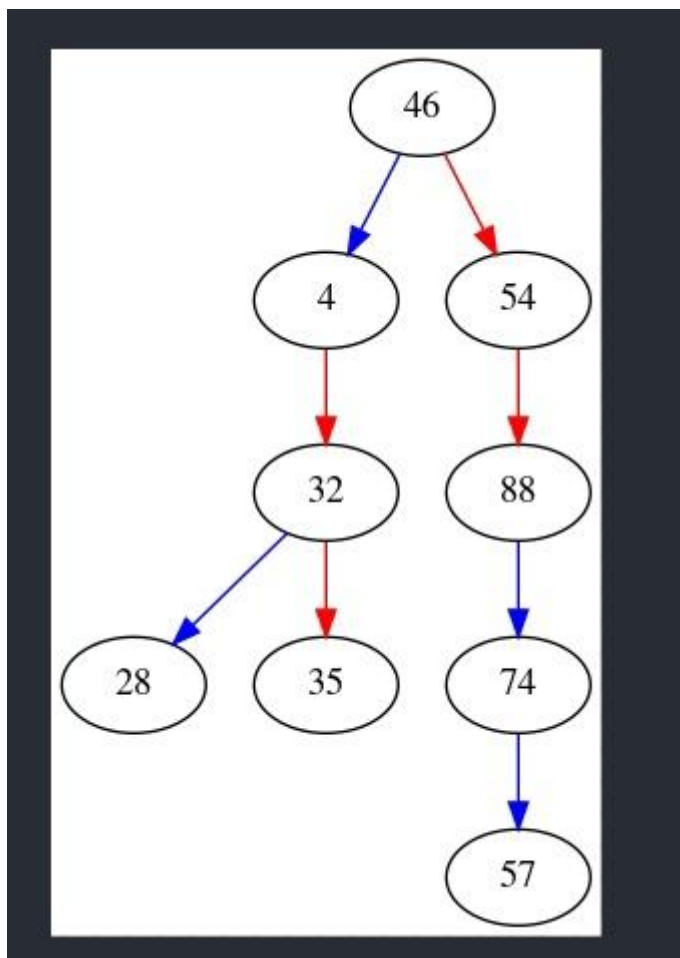
Рисунок 1. Файл с данными

Результат работы пункта меню 4 (вывод дерева на экран) на данных с картинки выше.



*Рисунок 2. Вывод бинарного дерева, считанного из файла, на экран*

Результат работы пункта меню 4 (визуализация дерева) на данных с первой картинки.

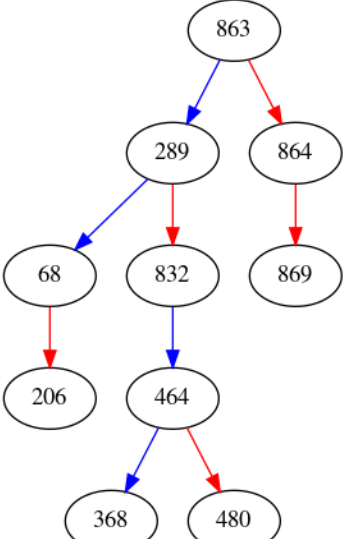
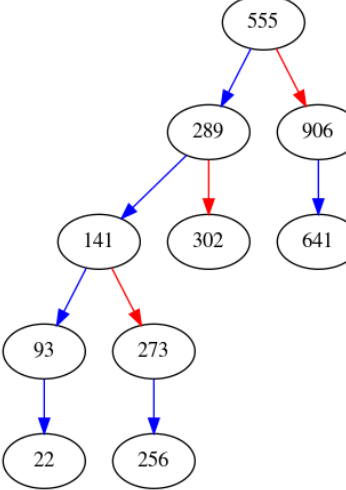
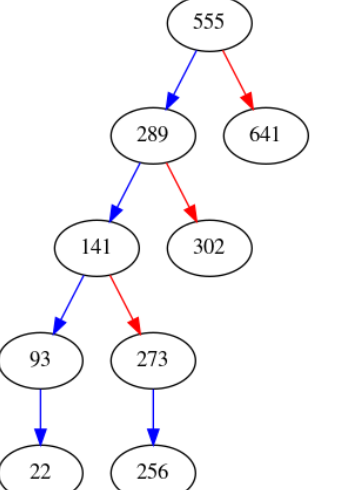
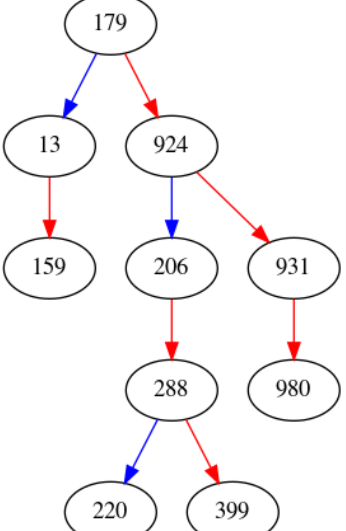
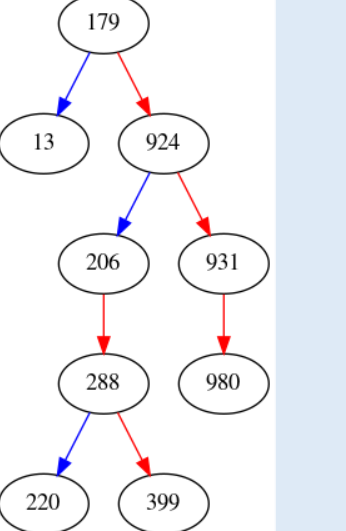


*Рисунок 3. Визуализация бинарного дерева в виде png картинки*

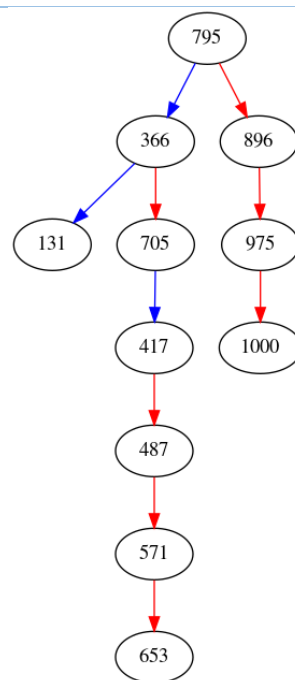
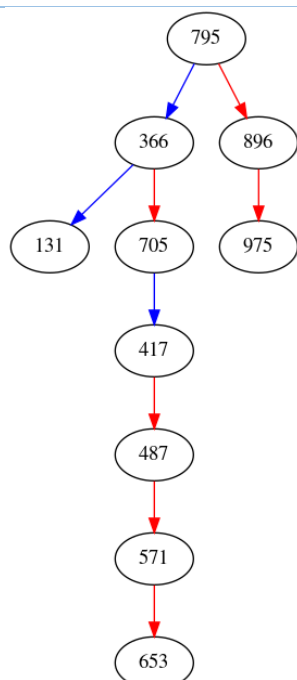
При визуализации бинарного дерева в виде png картинки элементы правого поддерева соединены дугами красного цвета, а элементы левого поддерева соединены дугами синего цвета.

Операция	Исходный вид дерева	Дерево после изменений
----------	---------------------	------------------------

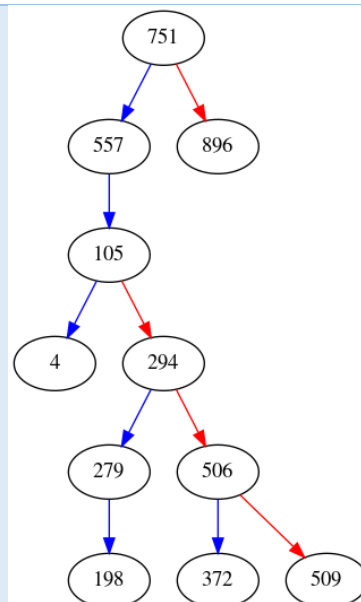
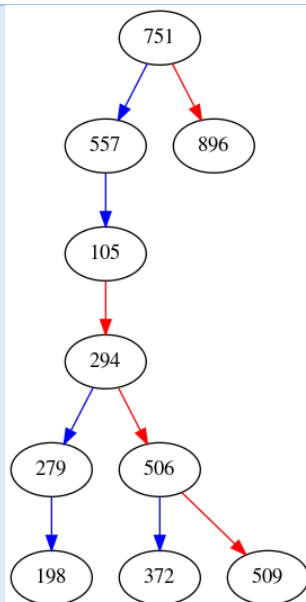


<p><b>Удаление элемента с номером 289</b></p> <p>У этой вершины есть два потомка. В данном случае будет найден самый левый (минимальный) элемент в правом поддереве и поставлен на место удаляемого элемента. Найденным элементом будет <b>368</b></p>		
<p><b>Удаление элемента с номером 906</b></p> <p>У этой вершины всего один потомок с номером <b>641</b>, поэтому именно эта вершина встанет на место удаляемого элемента</p>		
<p><b>Удаление элемента с номером 159</b></p> <p>У этой вершины нет потомков, поэтому она будет просто удалена из дерева</p>		

**Добавление вершины  
в дерево со значением  
1000**



**Добавление вершины  
в дерево со значением  
4**



## 4.2. Ограничения на входные данные

Попытка удаления из пустого дерева обрабатывается программой, выдавая соответствующие сообщения, но не завершая программу с ненулевым кодом возврата.

При вводе любых не валидных данных программа выдает соответствующее сообщение и завершает работу с ненулевым кодом возврата.

## 5. Оценка эффективности работы алгоритмов

Результаты измерения времени добавления данных в файл и в бинарное дерево при разных количествах вершин. Для каждого количества вершин было проведено 5 измерений и взято среднее из временных интервалов.

Сравнение время добавления чисел в дерево и в файл  
(используя массив)

Количество элементов	Добавление элемента (мкс)	
	Дерево	Файл
1000	0.800000	643.000000
3000	1.600000	722.000000
5000	1.800000	934.000000
8000	2.000000	1189.000000
10000	2.800000	1706.600000

Сравнение время добавления чисел в дерево и в файл  
(без использования массива)

Количество элементов	Добавление элемента (мкс)	
	Дерево	Файл
1000	0.400000	4.600000
3000	1.300000	4.600000
5000	1.700000	4.800000
8000	2.000000	4.700000
10000	2.000000	5.100000

## 6. Выводы

Как видно из таблицы выше, время добавления при небольшом количестве элементов в дереве и в файле (~1000) примерно в 600-650 раз быстрее, а при большом количестве элементов (~5000-10000) примерно 590-600 раз быстрее при добавлении в дерево, нежели в файл.

Это ожидаемый результат, поскольку при добавлении числа в файл сначала считывается содержимое этого файла в массив целых чисел, потом к нему в конец дописывается добавленный элемент, а затем новый массив вновь записывается в файл. То есть запись числа в файл – это линейная сложность  $O(n)$ , в то время как поиск подходящего узла в бинарном дереве – это сложность  $O(\log(n))$ , где  $n$  – число вершин.

Если же при записи данных в файл не использовать массив (а это можно сделать, поскольку при добавлении новых данных в файл они записываются в конец файла), а просто, открыв файл в режиме добавления данных, записать новое число, можно увидеть, что время добавления данных в файл значительно уменьшилось (в среднем  $\sim 200$  раз), и не росло с увеличением числа записанных элементов в файле. Но время добавления данных в файл все также проигрывает времени добавления данных в файл (время добавления данных в файл примерно в 2-4 раза медленнее, чем добавление данных в дерево). Вероятно, это связано с тем, что каждое добавление новых данных в файл сопровождается открытием файла, записью и закрытием файла. Но сложность алгоритма добавления данных файл в таком случае  $O(1)$ .

## 7. Набор тестов

### Позитивные тесты

<b>№ Теста</b>	<b>Входные данные</b>	<b>Выходные данные</b>	<b>Результат</b>
<b>01</b>	Номер пункта меню 1 Файл с данными существует и не пустой	Сообщение об успешности считывания данных из файла в дерево	Ожидание следующего действия.
<b>02</b>	Номер пункта 2 Пользователь ввел валидное число	Сообщение об успешном добавлении данных в дерево	Ожидание следующего действия.
<b>03</b>	Номер пункта 3 Пользователь ввел валидное число	Сообщение об успешном добавлении данных в файл	Ожидание следующего действия.
<b>04</b>	Номер пункта 4 Дерево не пустое Удаляемый элемент существует	Сообщение об успешном удалении данных из дерева	Ожидание следующего действия.
<b>05</b>	Номер пункта 4 Дерево пустое	Сообщение о том, что дерево пустое	Ожидание следующего действия.
<b>06</b>	Номер пункта 4 Дерево не пустое Удаляемый не элемент существует	Сообщение о том, что в дереве нет удаляемого элемента	Ожидание следующего действия.
<b>07</b>	Номер пункта 5 Дерево не пустое	Вывод бинарного дерева на экран	Ожидание следующего действия.
<b>08</b>	Номер пункта 5 Дерево пустое	Сообщение о том, что дерево пустое	Ожидание следующего действия.
<b>09</b>	Номер пункта 6 Дерево не пустое	Создание png изображения дерева	Ожидание следующего действия.
<b>10</b>	Номер пункта 6 Дерево пустое	Сообщение о том, что дерево пустое	Ожидание следующего действия.
<b>10</b>	Номер пункта 7 Дерево пустое	Сообщение о том, что дерево пустое	Ожидание следующего действия.
<b>11</b>	Номер пункта 7 Дерево не пустое	Вывод количества узлов на каждом из уровней текущего графа	Ожидание следующего действия.
<b>12</b>	Номер пункта 8	Вывод сравнения времени добавления чисел в дерево и в файл	Ожидание следующего действия.
<b>13</b>	Номер пункта 9 Дерево ранее не было очищено	Сообщение об успешном очищении дерева	Ожидание следующего действия.
<b>14</b>	Номер пункта 9 Дерево очищено	Сообщение о том, что дерево пустое	Ожидание следующего действия.

## Негативные тесты

№ Теста	Входные данные	Выходные данные	Результат
01	Номер пункта меню 100	Сообщение о том, что введен неверный номер пункта меню	Завершение программы с ненулевым кодом возврата
02	Номер пункта равен 2 или 3 Элемент дерева: папввывы	Сообщение о том, что введен некорректный элемент дерева	Завершение программы с ненулевым кодом возврата

## 8. Ответы на контрольные вопросы

### 8.1. Что такое дерево?

Дерево — это широко используемая структура данных, который представляет собой иерархическую древовидную структуру с набором связанных узлов. Каждый узел в дереве может быть связан со многими дочерними узлами (в зависимости от типа дерева), но должен быть связан только с одним родительским узлом, за исключением корневого узла, у которого нет родительского узла.

### 8.2. Как выделяется память под представление деревьев?

Память выделяется для каждой вершины дерева отдельно.

### 8.3. Какие бывают типы деревьев?

Двоичное дерево, Дерево двоичного поиска, АВЛ-дерево, В-дерево, Красно-черное дерево.

### 8.4. Какие стандартные операции возможны над деревьями?

Поиск узла, добавление узла, удаление узла.

## 8.5. Что такое дерево двоичного поиска?

Двоичное дерево поиска — двоичное дерево, для которого выполняются следующие дополнительные условия (свойства дерева поиска):

- оба поддерева — левое и правое — являются двоичными деревьями поиска;
- у всех узлов *левого* поддерева произвольного узла  $X$  значения ключей данных *меньше*, нежели значение ключа данных самого узла  $X$ ;
- у всех узлов *правого* поддерева произвольного узла  $X$  значения ключей данных *больше либо равны*, нежели значение ключа данных самого узла  $X$ .