



**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 ПРОГРАММНАЯ ИНЖЕНЕРИЯ (ИУ7)**

**НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.03.04 Программная инженерия**

**О Т Ч Е Т**

**По лабораторной работе № 6**

**Название:** Обработка деревьев

**Дисциплина:** Типы и структуры данных

Студент ИУ7-32Б

Беляк С.С.

Преподаватель

Барышникова М.Ю.

Москва, 2023

## **Описание условия задачи**

Построить двоичное дерево поиска, в вершинах которого находятся слова из текстового файла. Вывести его на экран в виде дерева. Определить количество вершин дерева, содержащих слова, начинающиеся на указанную букву. Выделить эти вершины цветом. Сравнить время поиска начинающихся на указанную букву слов в дереве и в файле.

## **Описание задачи, реализуемой в программе**

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

## **Описание ТЗ.**

### **Описание исходных данных**

**Пользователь выбирает опцию из меню, представленного в консоли.**

Возможные варианты выбора:

#### **Меню:**

0. Выход
1. Считать дерево из файла
2. Добавить узел
3. Удалить узел
4. Вывести дерево
5. Найти узел
6. Найти и выделить слова на заданную букву
7. Оценка эффективности

Пользователь может выбрать соответствующую опцию, введя число от 0 до 7 в консоли.

### **Имеются следующие ограничения:**

#### **Ограничение по памяти:**

Необходимо обеспечить эффективное использование динамической памяти и предотвратить утечки.

#### **Ограничение по времени:**

Время выполнения программы может зависеть от множества факторов, включая объем обрабатываемых данных.

#### **Ограничения на работу с файлами:**

Ограничения на открытие и считывание файла могут привести к ошибкам, если файл не существует, недоступен для чтения или содержит данные, не соответствующие ожидаемому формату.

### **Ограничения бинарного дерева:**

Реализация бинарного дерева может сталкиваться с ограничениями при обработке больших объемов данных или несбалансированных деревьев, что может привести к неэффективной работе.

### **Ограничение по количеству операций ввода-вывода:**

Программа взаимодействует с пользователем через стандартный ввод/вывод. Необходимо предусмотреть обработку ошибок при вводе и выводе данных.

### **Ограничения по вводу параметров:**

Пользователь должен обеспечить корректность ввода данных. Программа должна корректно обрабатывать ввод пользователя и предоставлять информацию о возможных ошибках.

## **0. Описание результатов программы**

### **1. Выход:**

- Программа завершает выполнение.

### **2. Считать дерево из файла:**

- Пользователь вводит имя файла.
- Если файл успешно открывается, существующее дерево уничтожается, и данные считываются из файла для построения нового бинарного дерева.

### **3. Добавить узел:**

- Пользователь вводит слово.
- Слово вставляется в бинарное дерево.

### **4. Удалить узел:**

- Пользователь вводит слово.
- Слово удаляется из бинарного дерева.

### **5. Вывести дерево:**

- Если дерево не пусто, происходит отрисовка дерева на экране. Каждый узел выводится и, возможно, с указанием цвета.

### **6. Найти узел:**

- Пользователь вводит слово.
- Производится поиск слова в бинарном дереве.
- Если слово найдено, выводится поддерево, начиная с найденного узла.

### **7. Найти и выделить слова на заданную букву:**

- Пользователь вводит букву.
- Производится поиск всех узлов, содержащих введенную букву.
- Найденные узлы окрашиваются для выделения.

## 8. Оценка эффективности:

- Пользователь вводит имя файла.
- Программа выполняет тесты для оценки эффективности выполнения операций, таких как вставка, удаление и поиск, и выводит результаты в файл.

## 9. Способ обращения к программе

Способ обращения к программе пользователем происходит через исполняемый файл app.exe.

## 3. Описание возможных аварийных ситуаций и ошибок пользователя

### CHOICE\_ERROR (Ошибка выбора)

Эта ошибка возникает, когда пользователь вводит некорректное значение пункта меню программы. Например, если пользователь вводит символ или значение, которое не соответствует доступным опциям в меню. Пользователю будет предложено ввести корректное значение из доступных опций.

### ALLOCATE (Ошибка выделения памяти)

В случае ошибки выделения памяти, программа сообщит об этом пользователю.

### OVERFLOW (Ошибка переполнения)

Возникает, когда система не может выделить достаточно памяти. В случае ошибки переполнения памяти, программа сообщит об этом пользователю.

## 5.Описание внутренних структур данных

Структура в программе определена для представления узлов бинарного дерева.

```
typedef struct Node
{
    char *data;
    struct Node *left;
    struct Node *right;
    bool colour;
} Node;
```

**char \*data** представляет собой указатель на строку, которая хранит данные, связанные с узлом.

**struct Node\* left** является указателем на левое поддерево текущего узла. Если узел не имеет левого потомка, то **left** содержит значение **NULL**.

**struct Node\* right** представляет собой указатель на правое поддерево текущего узла. Если узел не имеет правого потомка, то **right** содержит значение **NULL**.

**Bool colour** представляет цвет узла. В данной структуре, оно используется для обозначения цвета узла.

## 6. Алгоритм программы

### Общий алгоритм программы:

1. Инициализация параметров.
2. Основной цикл программы:
3. Программа входит в главный цикл, который ожидает ввода пользователя и предоставляет меню с различными опциями.
4. В зависимости от выбора пользователя, программа выполняет следующие действия:
  - Считать дерево из файла:  
Ввод имени файла. Открытие файла. Очистка текущего дерева. Считывание данных из файла и вставка их в дерево.
  - Добавить узел:  
Ввод слова. Вставка нового узла с этим словом в дерево.
  - Удалить узел:  
Ввод слова. Удаление узла с этим словом из дерева.
  - Вывести дерево:  
Если дерево не пусто, отрисовка дерева на экране.
  - Найти узел:  
Ввод слова. Поиск узла с этим словом в дереве. Вывод поддерева, начиная с найденного узла.
  - Найти и выделить слова на заданную букву:  
Ввод символа. Поиск всех узлов, содержащих этот символ. Выделение найденных узлов.
  - Оценка эффективности:  
Ввод имени файла. Запуск тестов для оценки эффективности операций.
5. Возврат в главное меню:
6. При завершении программы, программа освобождает выделенную память для структуры данных, завершает работу и закрывает все ресурсы.

Выведем исходное дерево из файла:

```
4
|
|   |   |   /woald
|   |   | /undreamed
|   |   | \pharmacophobia
|   |   | /peneplain
|   |   | nonsignificative
|   |   | /nonclaim
|   |   | \governessdom
|   |   | \euthyneural
|   |   |   | /echinite
|   |   |   | \child
```

Выберем букву с, слово child стало окрашено.

```
4
| | | /woald
| | | /undreamed
| | | \pharmacophobia
| | | /peneplain
| | | nonsignificative
| | | /nonclaim
| | | \governessdom
| | | \euthyneural
| | | | /echinite
| | | | \child
```

Попробуем удалить слово child:

```
4
| | | /woald
| | | /undreamed
| | | \pharmacophobia
| | | /peneplain
| | | nonsignificative
| | | /nonclaim
| | | \governessdom
| | | \euthyneural
| | | | \echinite
```

Слово удалено.

Попробуем добавить слово new\_word в дерево:

```
Введите слово: new word
Выберите пункт меню:
Пункт Действие
-----
0 Выход
1 Считать дерево из файла
2 Добавить узел
3 Удалить узел
4 Вывести дерево
5 Найти узел
6 Найти и выделить слова на заданную букву
7 Оценка эффективности
-----
4
| | | /woald
| | | /undreamed
| | | | \pharmacophobia
| | | | /peneplain
| | | | nonsignificative
| | | | /nonclaim
| | | | \new word
| | | | \governessdom
| | | | \euthyneural
| | | | | /echinite
| | | | | \child
```

Слово успешно добавлено.

Из файла слова выводятся в результате постфиксного обхода:

```

| | | /woald
| | | /undreamed
| | | | \pharmacophobia
| | | | /peneplain
| | | | nonsignificative
| | | | /nonclaim
| | | | \governessdom
| | | | \euthyneural
| | | | | /echinite
| | | | | \child
child echinite euthyneural governessdom nonclaim nonsignificative peneplain pharmacophobia undreamed woald
```

### Вывод по алгоритму работающей программы:

Программа предоставляет интерфейс для взаимодействия с бинарным деревом поиска. Пользователь может выполнить различные операции, такие как считывание дерева из файла, добавление и удаление узлов, поиск узлов по словам, вывод дерева, а также выделение слов, содержащих заданную букву. Дополнительно, предусмотрена возможность проведения тестов для оценки эффективности операций.

## Временная эффективность

Сравним время поиска начинающихся на указанную букву слов в дереве и в файле. Найдем среднее время функций поиска при 1000 итерациях:

**Количество слов 10:**

Тип	Время (нс)
Дерево	55
Файл	1473

**Разность производительности = 1418 нс. Реализация с помощью файла дольше на 2578 %**

**Количество слов 100:**

Тип	Время (нс)
Дерево	91
Файл	5857

**Разность производительности = 5766 нс**

**Реализация с помощью файла дольше на 6336 %**

**Количество слов 1000**

Тип	Время (нс)
Дерево	542
Файл	90202

**Разность производительности = 89660 нс**

**Реализация с помощью файла дольше на 16542 %**

**Количество слов 10000:**

Тип	Время (нс)
Дерево	8725
Файл	608018

**Разность производительности = 599293 нс**

**Реализация с помощью файла дольше на 6868%**

Программа, использующая бинарное дерево, демонстрирует высокую эффективность по сравнению с операциями над файлом для небольших объемов данных (10 слов).

С увеличением объема данных (до 10000 слов), эффективность остается высокой, что свидетельствует о хорошей производительности алгоритма на больших объемах данных.

Использование бинарного дерева для хранения и поиска данных позволяет эффективно обрабатывать запросы и операции в сравнении с операциями ввода/вывода через файл

## Вывод

В ходе выполнения лабораторной работы была разработана программа для работы с бинарным деревом поиска, предоставляющая пользователю удобный интерфейс для взаимодействия с данными. В результате тестирования эффективности программы на различных объемах данных (10, 100, 1000, 10000 слов), были получены временные показатели для операций с бинарным деревом и операциями с файлом.

Согласно результатам тестов, программа, использующая бинарное дерево, демонстрирует высокую временную эффективность. Эффективность программы, измеренная в процентах, подтверждает, что использование бинарного дерева поиска позволяет существенно ускорить операции по сравнению с файлом.

В целом, разработанная программа успешно решает поставленную задачу и предоставляет эффективные средства для работы с бинарным деревом поиска.

## Ответы на контрольные вопросы

### 1. Что такое дерево? Как выделяется память под представление деревьев?

Дерево - структура данных, состоящая из узлов, связанных между собой рёбрами. Один из узлов называется корнем, остальные разделяются на узлы и листья. Узлы, соединенные ребрами, образуют поддеревья. Память под представление деревьев обычно выделяется динамически. Каждый узел дерева содержит информацию и указатели на своих потомков (или нулевые указатели, если потомков нет). Для каждого узла память выделяется отдельно при добавлении новых узлов.

### 2. Какие бывают типы деревьев?

**Дерево двоичное:** Каждый узел имеет не более двух потомков.

**Дерево двоичного поиска:** Узлы упорядочены так, что для каждого узла все узлы в его левом поддереве меньше его, а в правом — больше.

**N-арное дерево:** Каждый узел может иметь произвольное количество потомков.

**Распределенное дерево:** используется в распределенных вычислениях и сетевых структурах.

**AVL-дерево, красно-черное дерево:** Сбалансированные бинарные деревья для эффективного поиска.

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

Стандартные операции над деревьями включают:

**Добавление узла:** Вставка нового узла в дерево.

**Удаление узла:** Удаление существующего узла из дерева.



**Поиск узла:** Нахождение узла с определенным значением.

**Обход дерева:** Посещение всех узлов дерева в определенном порядке (прямой, обратный, симметричный).

**Вывод дерева в виде строки:** Представление дерева в текстовой или графической форме.

**Изменение данных узла:** Обновление значений в существующем узле.

#### **4. Что такое дерево двоичного поиска?**

Дерево двоичного поиска - бинарное дерево, в котором каждый узел имеет не более двух потомков. При этом для каждого узла выполнено следующее свойство: все узлы в левом поддереве меньше текущего узла, а все узлы в правом поддереве больше текущего узла. Это свойство делает дерево двоичного поиска эффективной структурой данных для поиска, вставки и удаления элементов, так как оно обеспечивает логарифмическую сложность этих операций в среднем случае.