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»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №6 ПО ДИСЦИПЛИНЕ ТИПЫ И СТРУКТУРЫ ДАННЫХ

Студент **Звягин Даниил Олегович**

Группа **ИУ7-33Б**

Название предприятия **НУК ИУ МГТУ им. Н. Э. Баумана**

Студент _____ **Звягин Д.О.**

Преподаватель _____ **Барышникова М. Ю.**

Преподаватель _____ **Никульшина Т.А.**

Оценка _____

2023 г.

Условие задачи

Рассмотреть применение двоичных деревьев, реализовать основные операции над деревьями: обход деревьев, включение, исключение и поиск узлов.

Техническое задание

Построить двоичное дерево поиска из чисел, содержащихся в файле. Вывести его на экран в виде дерева. Определить количество узлов на каждом уровне дерева. Добавить число в дерево и в файл, сравнить время добавления числа в каждую из структур.

Входные данные:

Пользователю будет предложено меню.
Пункты меню:

0. Выход
1. Вывести дерево на экран
2. Выбрать файл для работы
3. Считать дерево из выбранного файла
4. Добавить число в дерево и в файл
5. Посчитать количество узлов на каждом уровне дерева
6. Вывести среднее время добавление числа в дерево
7. Удалить число из дерева

Выходные данные:

В зависимости от пункта меню: Изображение дерева на экране (в формате .png, сгенерированного с помощью graphviz); Информация об ошибках; Замеры времени по добавлению элементов в разные структуры;
Изображения открываются с помощью утилиты sxiv, т.к. это приложение, которое я использую для отображения изображений, команда может быть заменена на любой обработчик в 377 строке файла my_tree.c

Возможные аварийные ситуации:

Нехватка памяти

Способ обращения к программе

Собрать программу с помощью make release (/debug);
Запустить программу с помощью файла app.exe;

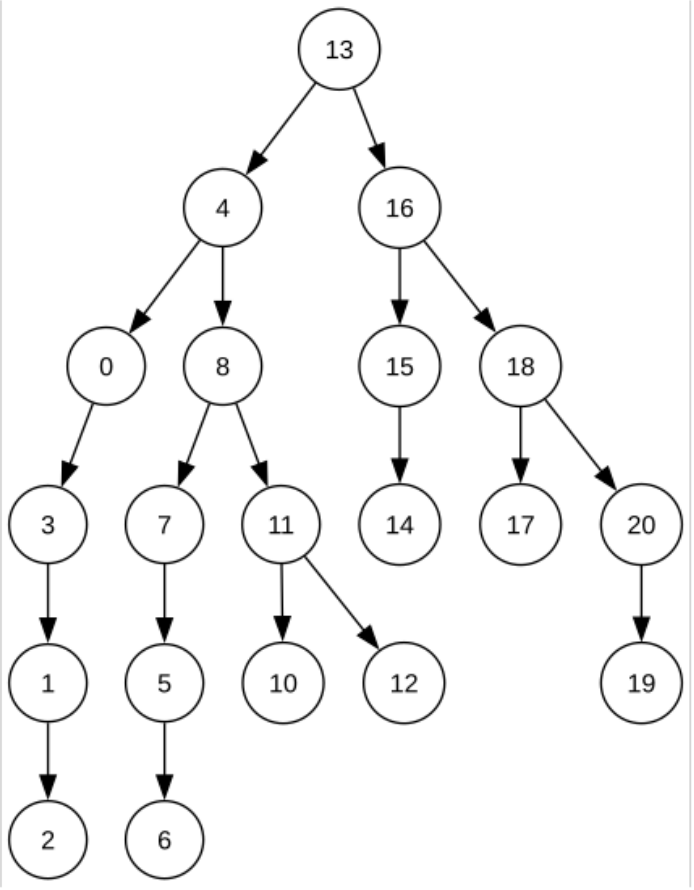
Структуры данных

// перечисление пунктов меню
```
enum menu
```

```
{
	EXIT,
	SHOW_TREE,
	CHOOSE_FILE,
	PARSE_FILE,
	ADD_NUMBER,
	COUNT_NODES_ON_LEVELS,
	AVG_ADD_TIME,
};

// перечисление ошибок при работе с деревом
enum tree_errors
{
	TREE_NO_MEMORY = 1,
	TREE_BAD_DOT,
};

// тип данных узла дерева
typedef struct tree_node *tree_node_t;
struct tree_node
{
	void *data;
	size_t count;
	struct tree_node *parent;
	struct tree_node *left;
	struct tree_node *right;
};

// структура данных дерева (оформлено в видео АТД)
typedef struct tree *tree_t;
struct tree
{
	tree_node_t root;
	compataror_t cmp;
	const char *format;
};
```

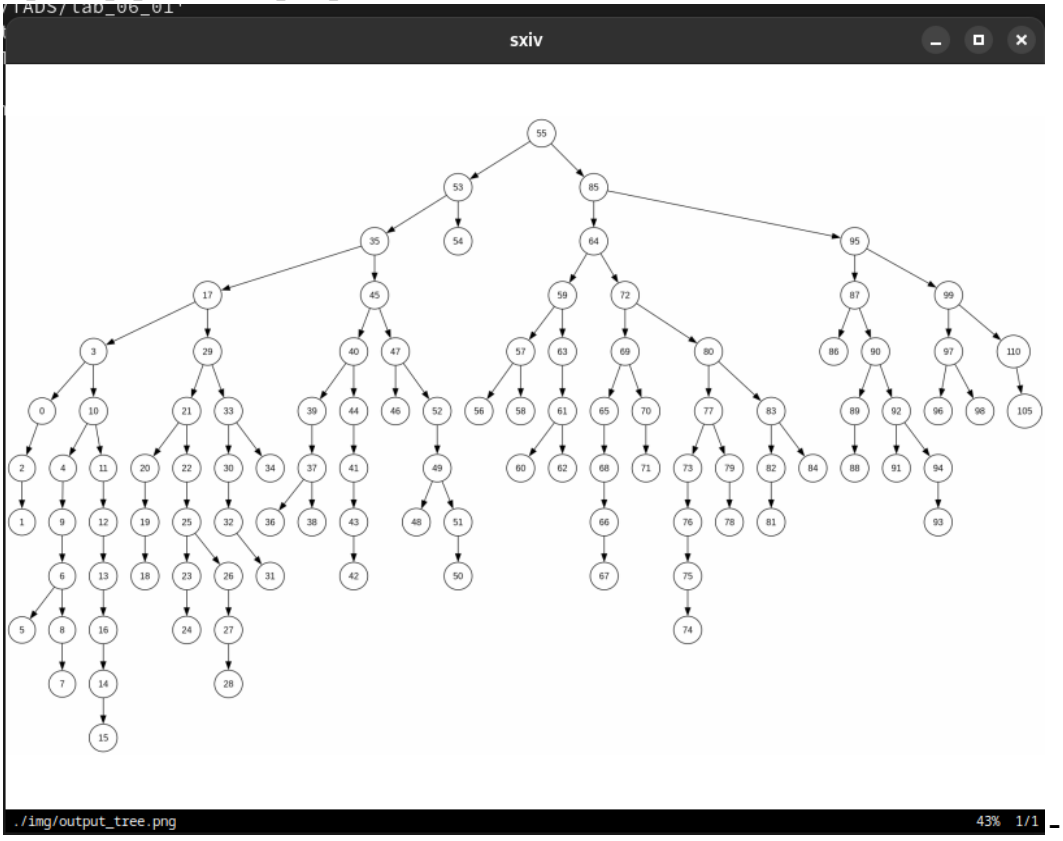
Пример работы

Обход дерева:
Я использую обход в глубину влево (префиксно), то есть для дерева на рисунке ниже порядок будет следующим:



13, 4, 0, 3, 1, 2, 8, 7, 5, 6, 11, 10, 12, 16, 15, 14, 18, 17, 20, 19

Пример работы программы:



```
3: Считать дерево из файла
4: Добавить число в дерево
5: Посчитать количество узлов дерева на каждом уровне
6: Вывести среднее время добавления числа в дерево
>2
Введите название файла
>test100pos.txt
Текущий файл: test100pos.txt
Меню:
0: Выход
1: Вывести дерево на экран
2: Выбрать файл для работы
3: Считать дерево из файла
4: Добавить число в дерево
5: Посчитать количество узлов дерева на каждом уровне
6: Вывести среднее время добавления числа в дерево
>3
Всего считано 102 чисел
Текущий файл: test100pos.txt
Меню:
0: Выход
1: Вывести дерево на экран
2: Выбрать файл для работы
3: Считать дерево из файла
4: Добавить число в дерево
5: Посчитать количество узлов дерева на каждом уровне
6: Вывести среднее время добавления числа в дерево
>1
Текущий файл: test100pos.txt
Меню:
0: Выход
1: Вывести дерево на экран
2: Выбрать файл для работы
3: Считать дерево из файла
4: Добавить число в дерево
5: Посчитать количество узлов дерева на каждом уровне
6: Вывести среднее время добавления числа в дерево
>5
На слое 0 : 1
На слое 1 : 2
На слое 2 : 4
На слое 3 : 6
На слое 4 : 12
На слое 5 : 20
На слое 6 : 21
На слое 7 : 16
На слое 8 : 10
На слое 9 : 6
На слое 10 : 3
На слое 11 : 1
```

Замеры

Замерять скорость добавления числа в файл отдельно не имеет особого смысла, так как запись числа в конец файла - это процедура, которая не зависит от каких-либо факторов. На моей машине добавление числа в файл занимает примерно 3100 нс

Добавление числа в дерево с другой стороны зависит от размера самого дерева и его содержимого (приходится искать место, в которое можно поставить число)
Я составил небольшую таблицу. N - количество узлов в дереве
t средн - среднее время добавления чисел от минимального до максимального в графе. Значения меньше, т.к. для многих чисел по настоящему записывать их в дерево не требуется - достаточно лишь найти их
Третий столбец - я добавляю несколько чисел в дерево и пытаюсь найти такое, чтобы время добавления было наибольшим

N	t (нс средн)	t (нс плохой)
10	127	762
50	144	964
100	196	1160
500	178	1809
1000	187	2903
5000	204	3061
10000	568	2896

Выводы по проделанной работе

Деревья - тип данных, который удобно применять при хранении иерархических данных или в случаях, когда нужен быстрый поиск элементов, так как в бинарном дереве он происходит примерно с той же сложностью, что и бинарный поиск (особенно если дерево сбалансировать). Но при использовании деревьев следует учитывать, что добавление данных в эту структуру занимает определённое время и вычислительные мощности, так как сначала программе

требуется определить место для очередного узла, а затем добавить его в дерево. В сбалансированном дереве пришлось бы также производить перестройку структуры, что занимало бы ещё большее время.
В сравнении с добавлением элемента в файл, добавление в дерево менее предсказуемо по времени и может увеличиваться с разрастанием самого дерева.

Контрольные вопросы

1. Что такое дерево? Как выделяется память под представление деревьев?
Дерево – это нелинейная структура данных, используемая для представления иерархических связей, имеющих отношение «один ко многим».
Каждый узел дерева содержит в себе указатели на узлы - потомки, а также хранящиеся в нём данные (и может содержать указатель на родителя, что может облегчить процесс удаления узла)

2. Какие бывают типы деревьев?
Двоичное дерево - иерархическая структура данных, в которой каждый узел имеет не более двух потомков
АВЛ-дерево — сбалансированное по высоте двоичное дерево поиска: для каждой его вершины высота её двух поддеревьев различается не более чем на 1.
Красно-чёрное дерево — один из видов самобалансирующихся двоичных деревьев поиска, гарантирующих логарифмический рост высоты дерева от числа узлов и позволяющее быстро выполнять основные операции дерева поиска: добавление, удаление и поиск узла.

3. Какие стандартные операции возможны над деревьями?
Обход дерева, включение, исключение и поиск узлов

4. Что такое дерево двоичного поиска?

Двоичное дерево, для каждого узла которого сохраняется условие: Левый потомок меньше родителя, правый потомок больше родителя