Лабораторная работа №6

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 подпрограмму, определить количество узлов дерева на каждом уровне. Добавить число в дерево и в файл. Сравнить время добавление в указанные структуры.

T3

Исходные данные и результаты

Входные данные: текстовый файл с целыми числами;

Результат работы:

- вывод дерева поиска в понятном для человека виде;
- вывод дерева по алгоритму DFS;
- время добавления целых чисел в файл и дерево;

Описание задачи, реализуемой программой

Целые числа считываются из файла и одновременно добавляются в дерево. По желанию пользователя печатаются контейнеры в приятном для человека виде. В операции поиска измеряется, добавления и удаления замеряется время.

Возможные аварийные ситуации и ошибки пользователя

- Некорректный формат данных во входном файле;
- Ошибки выделения памяти.

Описание внутренних структур данных

```
      struct TreeNode {
      // вершина дерева

      tree_el_t data;
      // данные

      int n;
      // порядковый номер

      struct TreeNode* left;
      // указатель на "левого" ребёнка

      struct TreeNode* right;
      // указатель на "правого"

      ребёнка
      struct TreeNode* parent;
      // указатель на родителя

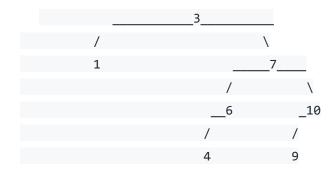
      };
      struct Tree {
      // дерево

      struct tree_node* root;
      // указатель на корень

      int n;
      // количество вершин
```

Тесты

Вывод дерева



DFS pre-order: 3 1 7 6 4 10 9
DFS in-order: 1 3 4 6 7 9 10
DFS post-order: 1 4 6 9 10 7 3

Добавление

```
Enter item: 128

Amount of time taken to do action element on Tree: 3794 (ns)

Amount of time taken to do action element on File: 97667 (ns)
```

Удаление

```
Enter item: 128
```

Amount of time taken to do action element on Tree: 2301 (ns)

Поиск

```
Enter item: 128
```

Amount of time taken to do action element on Tree: 940 (ns)

Amount of time taken to do action element on File: 78866 (ns)

Время работы

Добавление элемента в структуру

Структ ура	Время (тики)
Дерево	138744
Файл	3978944

Добавление элемента в дерево производится примерно в 28 раз быстрее.

Выводы

Структура дерево позволяет наглядно представлять данные (например, числовые) и работать с ними быстрее, чем при использовании файла. При этом без дополнительной балансировки может возникнуть ситуация, когда дерево будет представлять линейный список, что негативно скажется на скорости работы с его элементами.

Контрольные вопросы

1. Что такое дерево? Дерево – это нелинейная структура данных, используемая для представления иерархических связей, имеющих отношение «один ко многим».

- 2. Как выделяется память под представление деревьев? Память выделяется динамически (как под элементы списка).
- 3. Какие бывают типы деревьев? Деревья отличаются по количеству потомков вершины.
- 4. Какие стандартные операции возможны над деревьями? Добавление, удаление, поиск элемента, балансировка дерева.
- 5. Что такое дерево двоичного поиска? Основные операции с деревьями: обход дерева, поиск по дереву, включение в дерево, исключение из дерева.