Повторение

Сортировка строк файла

Отсортировать строки в файле.

- Определить количество строк и длину самой большой строки в файле (используйте посимвольное чтение из файла).
- Выделить память под массив строк необходимого размера.
- Вместо того, чтобы открывать/закрывать файл, можно переместить указатель позиции файла на начало файла с помощью функции `fseek(<указатель на файл>, <номер символа>, SEEK_SET)`. SEEK_SET означает, что номер символа отсчитывается от начала файла. Например, следующая команда устанавливает файловый указатель на начало файла.

  ```
  fseek(fp, 0, SEEK_SET);
  ```

- Считать все строки в выделенный массив. Чтобы считать строку целиком можно использовать функцию fscanf() со следующими параметрами:

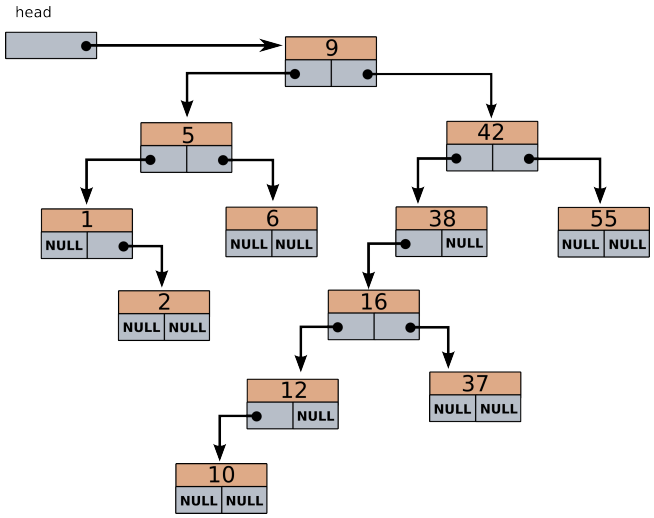
  ```
  fscanf(fp, "%[^\n]", str);
  ```

- Лексиграфичеси отсортировать массив строк используя функции qsort() и strcmp() из стандартной библиотеки.
- Записать все строки в файл “output.txt”.
- Добавить поддержку передачи имен входного и выходных файлов через аргументы командной строки(argc, argv).

Бинарное дерево поиска (BST)

```c
struct node
{
    int value;
    struct node* left;
    struct node* right;
}
typedef struct node Node;

void print_tree(Node* ptr)
{
    if (ptr != NULL)
    {
        printf("%d ", ptr->value);
        print_tree(ptr->left);
        print_tree(ptr->right);
    }
}
```



Бинарное дерево поиска это особый вид дерева, у которого для любого узла `ptr` выполняется следующее:

- все значения левого поддерева меньше чем `ptr->val`
- все значения правого поддерева больше или равны `ptr->val`

Задачи на бинарное дерево поиска

1. Написать функцию `void binarytree_insert(Node** p_head, int val)`, которая добавляет элемент в бинарное дерево. Используйте цикл while для определения места добавления нового элемента.
2. Написать функцию `void binarytree_insert_rec(Node** p_head, int val)`, которая добавляет элемент в бинарное дерево. Используйте рекурсию.
3. Написать функцию `int binarytree_depth(Node* head)`, вычисляющую глубину бинарного дерева. Подсказка: напишите дополнительную рекурсивную функцию `int calculate_binarytree_depth(Node* ptr, int current_depth)` и вызовите её из `binarytree_depth`.
4. Решить задачу **data_tree_flat** из контрольной 2015/2016.
5. Написать функцию `Node* binarytree_find(Node* head, int val)`, которая ищет элемент в бинарном дереве и возвращает указатель на этот элемент. Если такого элемента нет, то функция должна вернуть `Null`.
6. Написать функцию `Node* binarytree_destroy(Node* head)`, которая освобождает всю выделенную память.