Лабораторная работа 2. Сжатое префиксное дерево (Radix tree).

Постановка задачи

Реализовать структуру данных «*Radix tree*». Реализовать функции:

```
struct rtree *rtree_create()
struct rtree *rtree_insert(struct rtree *root,
                           char *key, char value)
struct rtree *rtree_lookup(struct rtree *root, char *key)
struct rtree *rtree_delete(struct rtree *root, char *key)
void rtree_print(struct rtree *root, int level)
```

Если лабораторная работа сдается после 26.10, количество баллов за лабораторную снижается на 2, после 02.11. – на 5.

Экспериментальное исследование

- В качестве ключа использовать строку (*char**), в качестве значения — целое число (*uint32_t*).
- Длина слова случайная, от 6 до 30 символов, возможен вариант генерации слов с помощью кода, пример кода на языке *Python* для генерации текстового файла со словами можно найти в приложении (*README.pdf*), как и рекомендации по более точному измерению времени.
- Заполнить таблицу 1 результатами измерения времени функции *rtree_lookup*, ключ для поиска случайный.
- Чтение из файла занимает большое количество времени. Можно один прочитать файл и записать все слова в массив, после чего использовать массив для заполнения сжатого префиксного дерева определенным количеством элементов.

Таблица 1. Результаты экспериментов

#	Количество элементов	Время выполнения функции *rtree_lookup*, с
1	10000	
2	20000	
3	30000	
...	...	
10	100000	

Контрольные вопросы

- Что такое префиксное дерево?
- Объяснить принцип работы префиксных деревьев.
- Описать различные подходы к хранению списков дочерних узлов в префиксном дереве.
- Что такое сжатое префиксное дерево (*radix tree*)? Чем отличается от обычного префиксного дерева?
- Вычислительная сложность префиксного дерева и сжатого префиксного дерева (*radix tree*).
- Объяснить и продемонстрировать алгоритм добавления ключа *radix tree*.
- Объяснить и продемонстрировать алгоритм удаления ключа из *radix tree*.
- Провести сравнительный анализ вычислительной сложности операций префиксного дерева, сбалансированного дерева поиска (*red-black tree*) и хеш-таблицы при хранении данных со строковыми ключами.
- Описать принцип работы вариаций префиксного дерева: *bitwise tree, suffix tree.*
- Анализ вычислительной сложности функций сжатого префиксного дерева.
- Область применения префиксных деревьев.