1. Министерство образования и науки Российской Федерации
2. Санкт-Петербургский Политехнический Университет Петра Великого
3. —
4. **Институт компьютерных наук и кибербезопасности**

**ЛАБОРАТОРНАЯ РАБОТА № 4**

1. «Поиск шестнадцатеричных чисел»
2. по дисциплине «Структуры данных»
3. Выполнил
4. студент гр. 5131001/30003 Шевчук Н.Е.

1. Преподаватель Семьянов П. В.

Санкт-Петербург

2024

Оглавление

[1. Цель работы 3](#_Toc168056431)

[2. Теоретическая справка 3](#_Toc168056432)

[3. Ход работы 4](#_Toc168056433)

[4. Результаты работы 5](#_Toc168056436)

[5. Вывод 6](#_Toc168056437)

# Цель работы

Определить наиболее оптимальную структуру данных для поиска шестнадцатеричных чисел (0xN) во входном файле на языке C.

# Теоретическая справка

Для выполнения поставленной задачи была выбрана структура данных – префиксное дерево («дерево-словарь»). Эта структура эффективна для задач, связанных с автозаполнением, словарями и индексированием текста. Кроме того префиксное дерево имеет интуитивно понятную структуру. По потреблению памяти такое дерево часто выигрывает у хэш-таблиц и сбалансированных деревьев. Это связано с тем что у множества ключей в таком дереве совпадают префиксы, и вместе с ними память, которую они используют.

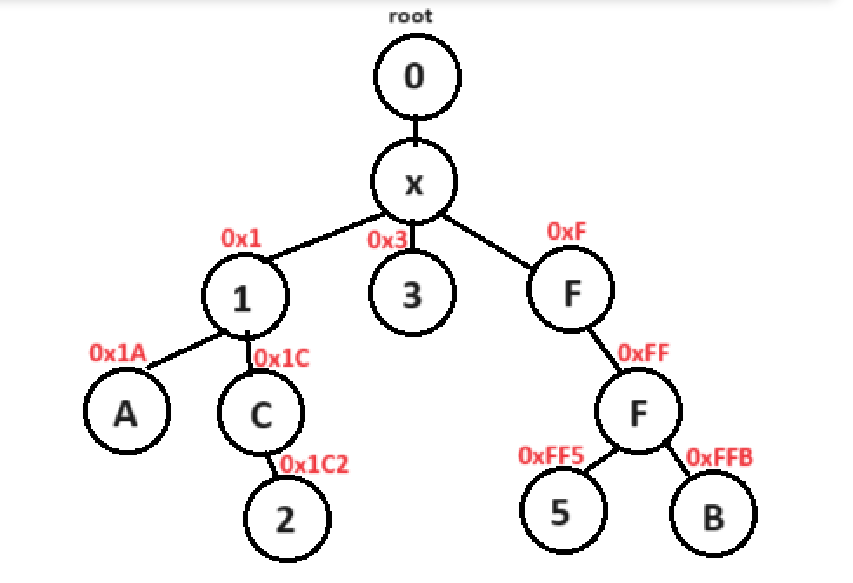


Рисунок 1. Структура используемого дерева

Каждая вершина дерева – элемент структуры *Node*, которая содержит:

* *char\* element* – указатель на строку (динамический массив char), хранящую полное представление данного шестнадцатеричного узла (0xFF) используется для хранения самого элемента, если вершина – конец строки;
* *unsigned long long elementSize* – размер данной строки;
* *char key* – ключ – символ текущей вершины (F);
* *unsigned long long counter* – счетчик элемента вершины;
* *struct Node\*\* segments* – массив указателей на вершины-потомки;
* *unsigned long long segmentsNumber* – текущее количество вершин-потомков;
* *unsigned long long segmentsSize* – максимальное количество вершин-потомков, под которые выделена память.

Дерево создается как структура с корнем *Node \*root*, таким образом, что каждый путь от корня к листу представляет собой одну из строк (шестнадцатеричное число), добавленную в дерево.

# Ход работы

Программа построчно считывает данные из входного файла, записывая их в динамический массив типа char, проверяет является ли входная строка шестнадцатеричным числом с помощью функции *isSixtet (char\* word)*, и при положительном результате добавляет его в дерево с помощью следующих функций:

* *createNode* - создает и инициализирует новую вершину дерева с указанным размером массива указателей на потомков, возвращает указатель на созданную вершину;
* *deleteNode* – рекурсивно удаляет вершину и ее потомков, освобождая выделенную на них память;
* *setElement* – устанавливает элемент в вершину, копируя в после element строку, являющуюся шестнадцатеричным числом;
* *setKey/getKey* – устанавливает/возвращает ключ вершины;
* *setCounter/getCounter* – устанавливает/возвращает счетчик вершины;
* *insertKey* – вставляет в вершину новый ключ, создавая новую вершину-потомка, если это необходимо;
* *containsOn* – проверяет, существует ли вершина с заданным ключом, возвращает индекс этой вершины;
* *insert* – вызывает функцию *add* с входным параметром – корнем дерева;
* *add –* рекурсивно добавляет строку в дерево, проходясь по ее символам и добавляя их в соответствующие по ключам вершины. Увеличивает счетчик вершины, если строка добавлена полностью;
* *createThree –* создает и инициализирует дерево с корнем;
* *deleteThree –* удаляет дерево с помощью рекурсивной функции deleteNode;
* *print –* рекурсивно выводит все строки – шестнадцатеричные числа, хранящиеся в дереве и их счетчики с помощью функции *printBranches*;
* *printBranches –* рекурсивно обходит все вершины дерева и выводит строки и счетчики с помощью функции *printThis*;
* *printThis –* выводит строку – шестнадцатеричное число и ее счетчик.

По завершению добавления элементов в дерево, программа выводит его, удаляет дерево, освобождая всю выделенную память, и закрывает входной файл.



# Результаты работы

Программа правильно считывает и обрабатывает данные входного файла на языке C, выводя на экран шестнадцатеричные числа, встреченные в файле и их количество.

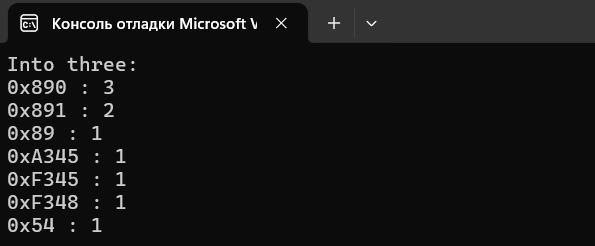


Рисунок 2. Результат работы программы

# Вывод

По результатам лабораторной работы была реализована программа, находящая шестнадцатеричные числа и подсчитывающая количество их вхождений, во входном файле на языке С. Были освоены навыки работы с такой структурой данных, как префиксные деревья.