**ФГБОУ ВО   
Уфимский университет науки и технологий**

**Кафедра ВМиК**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 100 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 90 |  |  |  |  |  |  |  |  |  |  |  |  |
| 80 |  |  |  |  |  |  |  |  |  |  |  |  |
| 70 |  |  |  |  |  |  |  |  |  |  |  |  |
| 60 |  |  |  |  |  |  |  |  |  |  |  |  |
| 50 |  |  |  |  |  |  |  |  |  |  |  |  |
| 40 |  |  |  |  |  |  |  |  |  |  |  |  |
| 30 |  |  |  |  |  |  |  |  |  |  |  |  |
| 20 |  |  |  |  |  |  |  |  |  |  |  |  |
| 10 |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |

Построение поискового дерева

**ОТЧЕТ**

**по лабораторной работе**

**по** Структурам и алгоритмам

компьютерной обработки данных

(*наименование дисциплины*)

|  |
| --- |
| Лабораторная работа 3 (Вар. 25) |
| (обозначение документа) |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Группа |  |  | Фамилия, И., О. | Подпись | Дата | Оценка |
| номер |  |
|  |  |
| Студент | | | Студент\_ФИО |  |  |  |
| Преподаватель | | | Преподаватель\_ФИО |  |  |  |
| Принял | | |  |  |  |  |

**Уфа 2024 г****.**

Содержание

[1 Цель работы 3](#_Toc181657351)

[2 Практическая часть 4](#_Toc181657352)

[2.1 Задание на лабораторную работу 4](#_Toc181657353)

[2.2 Входные и выходные данные 4](#_Toc181657354)

[2.3 Порядок построения хеш-таблицы 4](#_Toc181657355)

[3 Вывод 9](#_Toc181657356)

# Цель работы

Целью работы является построение поискового дерева, содержащего заданную последовательность случайных элементов.

# Практическая часть

## Задание на лабораторную работу

Сгенерировать 26 двузначных неповторяющихся чисел (элементов). Вывести их на экран. Построить и вывести в консоль поисковое дерево и прямой порядок его прохождения.

## Входные и выходные данные

Входные данные:

* size\_sample – количество двузначных элементов;
* sample[size\_sample] – массив сгенерированных элементов;

Выходные данные:

* tree\_output – функция, выводящая построенное дерево в консоль;
* tree\_traversal – функция, выводящая прямой порядок прохождения дерева.

## Порядок построения хеш-таблицы

Для построения дерева необходимо следовать алгоритму, который представлен ниже.

Сначала необходимо сгенерировать случайную последовательность неповторяющихся элементов:

* запустим цикл «while», в котором в «r» применением комбинации с использованием функции «rand()» будем записывать случайное значение (рисунок 2.1);

Изображение выглядит как текст, Шрифт, линия, снимок экрана

Автоматически созданное описание

Рисунок 2.1 – Генерация случайного значения

* далее вложенным циклом «for» проходим по массиву сгенерированных элементов, и если полученный на i-ом шаге элемент «r» не встретился в массиве, то добавляем его (рисунок 2.2).

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 2.2 – Проверка уникальности

На рисунке 2.3 приведен пример сгенерированных уникальных неповторяющихся элементов.

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок 2.3 – Пример сгенерированных элементов

Теперь необходимо построить дерево:

* для начала создадим структуру – узел дерева, как показано на рисунке 2.4;

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок 2.4 – Структура – узел дерева

* затем с помощью цикла «for» проходим по массиву сгенерированных чисел (рисунок 2.5);

Изображение выглядит как текст, Шрифт, снимок экрана, линия

Автоматически созданное описание

Рисунок 2.5 – Перебор сгенерированных чисел

чтобы каждое из них [чисел] передать в функцию «tree\_construction» для установления ссылок следующим образом: если очередной элемент меньше вершины, то продолжаем поиск свободной позиции в левом поддереве, иначе в правом (рисунок 2.6).

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 2.6 – Функция «tree\_construction»

Далее выведем дерево в консоль, для чего применим функцию «tree\_output», работающую следующим образом:

* на первой итерации проверяем вершину дерева, а на последующих те узлы, которые находятся в указателях предыдущих; показано на рисунке 2.7;

Изображение выглядит как текст, Шрифт, линия, снимок экрана

Автоматически созданное описание

Рисунок 2.7 – Выбор узла

* затем, расставляя отступы, равные значению узла, выводим проверяемый узел в консоль (рисунок 2.8);

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 2.8 – Расстановка отступов

* наконец добавляем в массив узлы, которые находятся в левом и правом указателе проверяемого в данный момент узла, как изображено на рисунке 2.9.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 2.9 – Добавление узлов в массив

На рисунке 2.10 представлен пример вывода дерева в консоль.

Изображение выглядит как снимок экрана, линия, диаграмма

Автоматически созданное описание

Рисунок 2.10 – Выведенное в консоль дерево

В конце выполним прямое прохождения следующим образом (рисунок 2.11):

* сначала выводим значение очередного узла;
* затем проходим по его левому поддереву;
* наконец – по его правому поддереву.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 2.11 – Прямое прохождение дерева

# Вывод

В ходе лабораторной работы мы построили поисковое дерево, которое содержит заданную последовательность случайных элементов.