**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 |
| МО-325Б |  |
|  |  |
| Студент | | | Шарыгин М.С. |  |  |  |
| Преподаватель | | | Верхотурова Г.Н. |  |  |  |
| Принял | | |  |  |  |  |

**Уфа 2024 г****.**

Содержание

[1 Цель работы 3](#_Toc185321340)

[2 Практическая часть 4](#_Toc185321341)

[2.1 Задание на лабораторную работу 4](#_Toc185321342)

[2.2 Определение поискового дерева 4](#_Toc185321343)

[2.3 Алгоритм построения дерева 4](#_Toc185321344)

[2.4 Алгоритм прохождения 5](#_Toc185321345)

[2.5 Входные и выходные данные 5](#_Toc185321346)

[2.6 Реализация построения дерева 5](#_Toc185321347)

[2.7 Вывод дерева 8](#_Toc185321348)

[2.8 Прохождение дерева 10](#_Toc185321349)

[3 Вывод 12](#_Toc185321350)

# 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, а также прямой и обратный порядки его прохождения.

## Определение поискового дерева

Если дерево организовано так, что для каждой вершины справедливо утверждение, что все ключи левого поддерева меньше, а правого – больше, то такое дерево называется поисковым. В таком дереве легко найти произвольный элемент, для этого достаточно, начав с корня, двигаться к левому или правому поддереву на основании лишь одного сравнения со значением текущей вершины.

## Алгоритм построения дерева

Ниже приведен алгоритм построения поискового дерева:

* первый элемент – это вершина дерева;
* следующие элементы размещаются по правилу:
  + если очередной элемент меньше вершины, то продолжаем поиск в левом поддереве:
    - если в левом поддереве вершины нет элемента, то текущий размещается на этом месте;
    - иначе элемент, расположенный в левом поддереве, считаем за вершину, после чего снова проводим аналогичную проверку до тех пор, пока не разместим текущий элемент в дереве;
  + иначе если элемент больше или равен вершине, то продолжаем поиск в правом поддереве способом, аналогичным тому, который описан для левого поддерева.

## Алгоритм прохождения

Алгоритм прямого прохождения дерева:

* алгоритм начинает свою работу с корня дерева, который становится текущим узлом;
* сначала выводится значение текущего узла;
* затем выводятся значения узлов, находящихся слева от текущего узла;
* в конце выводятся значения справа.

## Входные и выходные данные

Входные данные:

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

Выходные данные:

* Дерево, каждый узел которого представлен структурой Knot (данная структура приведена на рисунке 2.1).

Изображение выглядит как текст, Шрифт, снимок экрана, линия

Автоматически созданное описание

Рисунок 2.1 – Структура – узел дерева

## Реализация построения дерева

Для построения дерева необходимо следовать алгоритму, описанному в пункте 2.3.

Сначала создадим массив из 26 узлов (рисунок 2.2).



Рисунок 2.2 – Массив узлов дерева

Затем необходимо сгенерировать случайную последовательность неповторяющихся элементов и расположить их в узлах:

* запустим цикл «while», в котором в «r» применением комбинации с использованием функции «rand()» будем записывать случайное значение (рисунок 2.3);

Изображение выглядит как текст, Шрифт, линия, снимок экрана

Автоматически созданное описание

Рисунок 2.3 – 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 в поле «value» очередного узла (рисунок 2.4).

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 2.4 – Проверка уникальности

На рисунке 2.5 приведен пример сгенерированных уникальных неповторяющихся элементов.

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок 2.5 – Пример сгенерированных элементов

Теперь необходимо построить дерево:

* с помощью цикла «for» проходим по массиву узлов, начиная со второго[[1]](#footnote-1), в котором содержатся сгенерированные числа;
* каждое из них [чисел] передаём в функцию «tree\_construction» (рисунок 2.6) для установления ссылок следующим образом:

Изображение выглядит как текст, Шрифт, снимок экрана, линия

Автоматически созданное описание

Рисунок 2.6 – Перебор сгенерированных чисел

* + если очередной элемент меньше вершины, то продолжаем поиск свободной позиции в левом поддереве:
    - если левый указатель вершины не инициализирован, то добавляем в него ссылку на проверяемый на данном шаге итерации узел;
    - иначе, если левый указатель вершины не пустой, продолжаем поиск, считая за вершину тот узел, на который и указывает левый указатель;
  + иначе, продолжаем поиск свободной позиции в правом поддереве аналогичным образом (рисунок 2.7).

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 2.7 – Функция «tree\_construction»

## Вывод дерева

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, как приведено на рисунке 2.8;

Изображение выглядит как текст, Шрифт, линия, снимок экрана

Автоматически созданное описание

Рисунок 2.8 – Выбор узла

* затем, расставляя отступы, равные значению узла, и стрелки выводим проверяемый узел в консоль (рисунок 2.9);

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 2.9 – Расстановка отступов

* наконец добавляем в массив узлы, которые находятся в левом и правом указателе проверяемого в данный момент узла, как изображено на рисунке 2.10.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 2.10 – Добавление узлов в массив

На рисунке 2.11 представлен пример вывода дерева в консоль.

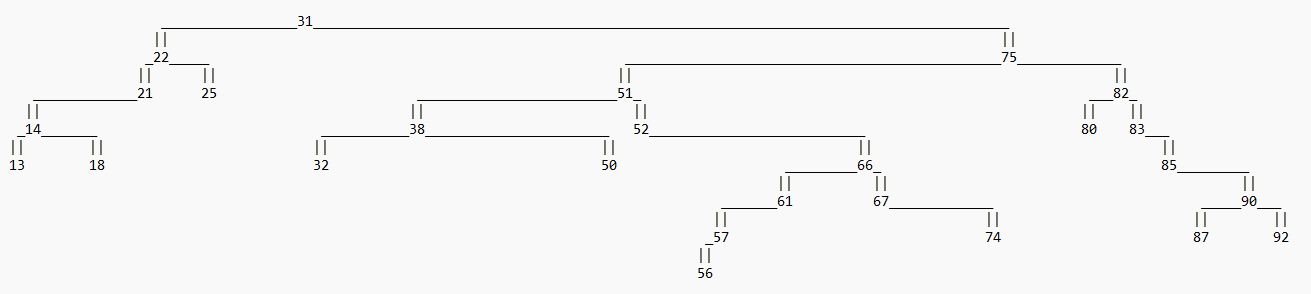


Рисунок 2.11 – Выведенное в консоль дерево

## Прохождение дерева

В конце выполним прохождения дерева следующими способами:

* прямой порядок (рисунок 2.12):
  + 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 2.12 – Прямое прохождение дерева

* обратный порядок (рисунок 2.13):
  + сначала заходим в левое поддерево очередного узла;
  + затем – в правое поддерево:
  + наконец, когда у очередной вершины нет ни левого, ни правого узла – потомка, выводим значение этой вершины в консоль.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 2.13 – Обратное прохождение дерева

На рисунке 2.14 изображены последовательности, получаемые при прямом и обратном прохождении дерева.

Изображение выглядит как текст, Шрифт, снимок экрана, линия

Автоматически созданное описание

Рисунок 2.14 – Пример прямого и обратного прохождения дерева

# Вывод

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

1. Второй узел – это узел, который занимает ячейку с индексом 1 в массиве узлов. [↑](#footnote-ref-1)