***Лабораторная работа 7.***

**Основные операции с деревьями**

***Определение.*** Двоичным (бинарным) деревом называется упорядоченное дерево степени 2 (*лпд, ппд* — левое и правое поддерево).

***Определение.*** Деревья со степенью больше двух называются сильноветвящимися.

**Алгоритмы обхода двоичного дерева**

Алгоритм обхода дерева — алгоритм, в котором доступ к каждой вершине дерева осуществляется один и только один раз.

Обход двоичного дерева сводится к выполнению трёх действий:

1. обработка корня (К);
2. обход левого поддерева (Л);
3. обход правого поддерева (П).

В зависимости от очерёдности этих действий различают следующие способы обхода:

1. «сверху вниз» (префиксный) = КЛП;
2. «снизу вверх» (постфиксный) = ЛПК;
3. «слева направо» (инфиксный) = ЛКП.

*Замечание 1.* Рассмотренный порядок обхода является левым. Можно ввести правый порядок обхода поддеревьев (КПЛ, ПЛК, ПКЛ).

*Замечание 2.* Алгоритм обхода не зависит от вида обработки вершин. Он является общим для алгоритмов, выполняющих различную обработку.

*Пример.*

\*

+ –

g \* d /

b c e f

1. КЛП: \*(+(g**,**\*(b, c)), –(d, /(e, f))) (префиксный обход: запятая — разделитель поддеревьев);
2. ЛПК: ((g, (b,c)\*)+, (d, (e,f)/)–)\* — постфиксный обход;
3. ЛКП: (g+(b\*c))\*(d– (e/f)) — инфиксный обход; корень — разделитель поддеревьев).

**Алгоритм печати дерева**

Пусть  — двоичное дерево. Рекурсивный алгоритм печати дерева можно представить в форме:

печать\_дерева;

*начало*

*если* not(дерево\_пусто ) *то*

*начало*

печать\_корня (чтение\_корня ); {печать корня}

печать\_дерева (левое\_поддерево ); {рекурсия}

печать\_дерева (правое\_поддерево );

*конец;*

*конец;*

*Замечание.* При разработке алгоритма печати дерева использован алгоритм обхода дерева «сверху вниз» (КЛП).

**Двоичное дерево поиска**

Пусть для элементов типа T установлено отношение порядка. Если множество таких элементов организовать в структуру, являющуюся двоичным деревом, у которого в левом поддереве содержатся элементы с ключом, меньшим, чем ключ узла, а в правом — с ключом, большим, чем ключ узла, то такое дерево называется двоичным деревом поиска.

*Пример*. Пусть данные поступают из файла в следующем порядке: 7,6,8,1,3,5,4,2,9,12,10. Двоичное дерево поиска из этих элементов будет иметь вид:

7

6 8

1 9

3 12

2 5 10

4

**Включение элемента в двоичное дерево поиска**

**Алгоритм включения элемента в двоичное дерево поиска можно представить следующим образом.**

***программа* включение: ДДТ (*аргументы* t:T; : ДДТ);**

***начало***

включение *если* дерево\_пусто() *то*

построение (t, ПУСТО, ПУСТО)

*иначе*

*начало* *если* t< чтение\_корня()

*то*

построение ( чтение\_корня (),

включение(t, лпд()), ппд())

*иначе*

построение (корень(),

лпд(), включение( t, ппд()))

*конец*

*конец*

*Замечание.* Элемент t проходит по дереву, начиная от корня, поворачивая вправо или влево от каждой вершины в зависимости от значения, пока не попадает на пустое место. На это место он и записывается.

Программа построения двоичного дерева поиска из элементов 3, 1, 4, 2, 5, 8 имеет вид:

*начало*

 создание\_дерева;

включение(3, );

включение(1, );

включение(4, );

включение(2, );

включение(5, );

включение(8, );

*конец*

**Удаление из двоичного дерева поиска**

Пусть задано двоичное дерево поиска. Надо удалить из него вершину с указанным значением ключа, не нарушая упорядоченности. Возможные варианты:

1). Удаляемая вершина — лист. Удаление происходит без перестройки дерева (соответствующий указатель получает значение ПУСТО).

2). Удаляемый элемент имеет одного потомка. В цепочке указателей производится обход удалённого элемента.

3). Удаляемый элемент имеет двух потомков. В этом случае возможны два варианта перестройки дерева. При этом удаляемую вершину заменяют либо на самый правый элемент её левого поддерева (самый большой в поддереве), либо на самый левый элемент её правого поддерева (самый маленький в поддереве).

**Физическое представление двоичного дерева**

Используются следующие основные способы представления двоичного дерева:

1. Цепное с использованием указателей.
2. Сплошное с использованием указателей.
3. Сплошное без использования указателей.

а). Для цепного представления каждый элемент должен содержать два указателя на левое и правое поддерево:

Type ptree= ^elem;

elem= record

info: <тип данных>;

left, right: ptree

end;

б). Сплошное представление с использованием указателей требует описать массив данных для записи ключей и связанных с этим массивом двух массивов указателей на левые и правые поддеревья. Необходимы также две переменные АДРЕС\_ВХОДА и СТЕК, которые будут указывать на начало структуры данных и вершину стека свободных элементов.

Этот стек можно представить в одном из массивов указателей на поддеревья. Если нумерация элементов массивов начинается с 1, то удобно считать, что указатель=0 соответствует структуре данных ПУСТО.

*Пример*.

К 5

И 7 У 8

В 10 Л 2 Ч 1

Г 4

Представленное дерево является двоичным деревом поиска (порядок алфавитный). Рядом с узлами дерева указаны номера элементов массива данных, в которых находятся соответствующие ключи.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 10 | В |  | 10 | 0 |  | 10 | 4 |
| 9 |  |  | 9 |  |  | 9 | 0 |
| 8 | У |  | 8 | 2 |  | 8 | 1 |
| 7 | И |  | 7 | 10 |  | 7 | 0 |
| 6 |  |  | 6 |  |  | 6 | 9 |
| 5 | К |  | 5 | 7 |  | 5 | 8 |
| 4 | Г |  | 4 | 0 |  | 4 | 0 |
| 3 |  |  | 3 |  |  | 3 | 6 |
| 2 | Л |  | 2 | 0 |  | 2 | 0 |
| 1 | Ч |  | 1 | 0 |  | 1 | 0 |

Данные ЛПД ППД

Здесь АДРЕС\_ВХОДА = 5; СТЕК = 3.

в). Сплошное представление без использования указателей (в одном массиве) удобно для так называемых полных деревьев, то есть таких, у которых все вершины заняты, кроме, может быть, нескольких «правых» в ряду последнего уровня. Для отображения на массив последовательно записываются элементы каждого уровня слева направо.

*Пример*.

1 уровень A

2 уровень B C

3 уровень D E F

*Замечание.* Здесь уровни для удобства считаются с 1, а не с 0, как в классическом варианте.

На одномерный массив это дерево отображается следующим образом:

1 2 3 4 5 6 7 8

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| A | B | C | D | E | F |  |  |

а). Потомки элемента с номером  имеют номера  и .

б). Элемент  уровня  имеет индекс .

*Замечание*. В рассмотренном представлении неявно использован алгоритм обхода «по уровням», отличающийся от рассмотренных ранее способов обхода.

**Поиск ключа в двоичном дереве поиска**

Алгоритм поиска, возвращающий поддерево, корнем которого является ключ поиска , можно представить в форме:

*программа* ПОИСК :ДДТ (*арг* x:T; : ДДТ);

найден  *false*;

*пока* (дерево\_пусто()) (найден)

*начало*

*если* = чтение\_корня() *то* найден=*true*

*иначе*

*начало*

*если*  < чтение\_корня()

*то*   ПОИСК(, *лпд*())

*иначе*  ПОИСК(, *ппд*())

*конец;*

*конец;*

ПОИСК ;

**ПОИСК С БАРЬЕРОМ В ДВОИЧНОМ ДЕРЕВЕ ПОИСКА**

##### Если установить барьер, то есть добавить элемент, информационное поле которого совпадает с ключом поиска, то можно упростить условие окончания цикла поиска (не надо проверять, что очередное поддерево не пусто).

Барьер можно рассматривать как общее представление всех внешних вершин. Иногда барьер называют якорем, а структуру с барьером — «гамаком».

Очевидно, что при наличии барьера ключ поиска всегда будет найден. Для определения результата поиска следует после окончания цикла поиска проверить, с каким указателем найден ключ. Если найден барьер, то результат поиска отрицательный.

##### ЗАДАНИЕ

**1). Разработать подпрограмму построения двоичного дерева поиска с указанными элементами. Предусмотреть возможность ввода данных с клавиатуры или из предварительно приготовленного файла.**

**2). Разработать подпрограмму поиска элемента в двоичном дереве поиска без барьера и с возможностью установки барьера. Установить счётчики числа сравнений в процессе поиска.**

**3). Разработать подпрограмму добавления элемента в двоичное дерево поиска.**

**4). Разработать подпрограмму удаления элемента из двоичного дерева поиска.**