**ЛАБОРАТОРНАЯ РАБОТА №7**

**«Динамические структуры данных. Бинарные деревья»**

**Цель работы**: изучить понятие, формирование, особенности доступа к данным и работы с памятью в бинарных деревьях, научиться решать задачи с использованием *рекурсивных функций* и алгоритмов обхода бинарных деревьев в языке C.

**Теоретический материал**

*Дерево* является одним из важнейших и интересных частных случаев графа. *Древовидная модель* оказывается довольно эффективной для представления динамических данных с целью быстрого *поиска информации*.

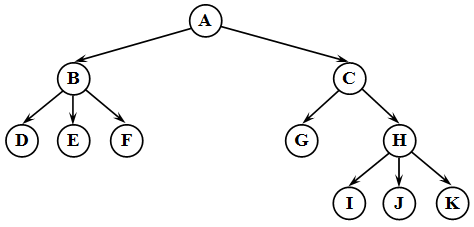
Деревья являются одними из наиболее широко распространенных структур данных в *информатике* и программировании, которые представляют собой иерархические структуры в виде набора связанных узлов.

**Дерево** – это *структура данных*, представляющая собой совокупность элементов и отношений, образующих иерархическую структуру этих элементов ( рис.1). Каждый элемент дерева называется вершиной (узлом) дерева. Вершины дерева соединены направленными дугами, которые называют ветвями дерева. Начальный узел дерева называют корнем дерева, ему соответствует нулевой уровень. Листьями дерева называют вершины, в которые входит одна *ветвь* и не выходит ни одной ветви.

Каждое *дерево* обладает следующими свойствами:

1. существует узел, в который не входит ни одной дуги (корень);
2. в каждую вершину, кроме корня, входит одна дуга.

Деревья особенно часто используют на практике при изображении различных иерархий. Например, популярны генеалогические деревья.



**Рис. 1.**Дерево

Все вершины, в которые входят ветви, исходящие из одной общей вершины, называются потомками, а сама *вершина* – предком. Для каждого предка может быть выделено несколько. Уровень потомка на единицу превосходит уровень его предка. *Корень дерева* не имеет предка, а *листья дерева* не имеют потомков.

Высота (глубина) дерева определяется количеством уровней, на которых располагаются его вершины. *Высота* пустого дерева равна нулю, *высота дерева* из одного корня – единице. На первом уровне дерева может быть только одна *вершина* – *корень дерева*, на втором – потомки корня дерева, на третьем – потомки потомков корня дерева и т.д.

**Поддерево** – часть древообразной структуры данных, которая может быть представлена в виде отдельного дерева.

Степенью вершины в дереве называется количество дуг, которое из нее выходит. Степень дерева равна *максимальной степени*вершины, входящей в *дерево*. При этом листьями в дереве являются вершины, имеющие степень нуль. *По* величине степени дерева различают два типа деревьев:

* двоичные – степень дерева не более двух;
* сильноветвящиеся – степень дерева произвольная.

**Упорядоченное дерево** – это *дерево*, у которого ветви, исходящие из каждой вершины, упорядочены *по* определенному критерию.

Деревья являются рекурсивными структурами, так как каждое *поддерево* также является деревом. Таким образом, *дерево* можно определить как рекурсивную структуру, в которой каждый элемент является:

* либо пустой структурой;
* либо элементом, с которым связано конечное число поддеревьев.

Действия с рекурсивными структурами удобнее всего описываются с помощью рекурсивных алгоритмов.

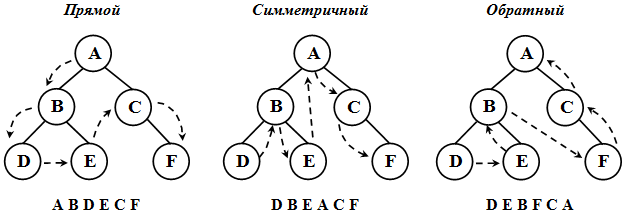
Списочное *представление деревьев* основано на элементах, соответствующих вершинам дерева. Каждый элемент имеет *поле*данных и два поля указателей: *указатель* на начало списка потомков вершины и *указатель* на следующий элемент в списке потомков текущего уровня. При таком способе представления дерева обязательно следует сохранять *указатель* на вершину, являющуюся *корнем дерева*.

Для того, чтобы выполнить определенную операцию над всеми вершинами дерева необходимо все его вершины просмотреть. Такая задача называется обходом дерева.

**Обход дерева** – это упорядоченная последовательность вершин дерева, в которой каждая *вершина* встречается только один раз.

При обходе все вершины дерева должны посещаться в определенном порядке. Существует несколько способов обхода всех вершин дерева. Выделим три наиболее часто используемых способа *обхода дерева* ( рис. 2):

* прямой;
* симметричный;
* обратный.



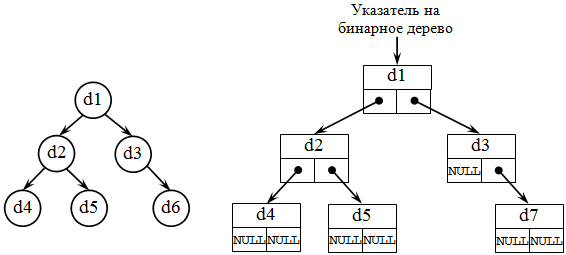
**Рис. 2.**Обходы деревьев

Существует большое многообразие древовидных структур данных. Выделим самые распространенные из них: бинарные (двоичные) деревья, красно-черные деревья, В-деревья, *АВЛ-деревья*, матричные деревья, смешанные деревья и т.д.

**Бинарные деревья**

*Бинарные деревья* являются деревьями со степенью не более двух.

**Бинарное (двоичное) дерево** – это *динамическая структура данных*, представляющее собой *дерево*, в котором каждая *вершина* имеет не более двух потомков ( рис.3). Таким образом, *бинарное дерево* состоит из элементов, каждый из которых содержит информационное *поле* и не более двух ссылок на различные бинарные поддеревья. На каждый элемент дерева имеется ровно одна *ссылка*.

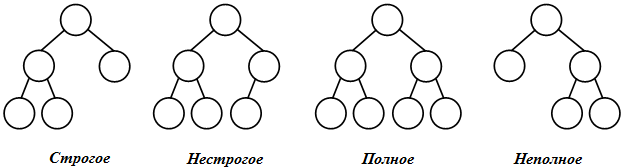


**Рис. 3.**Бинарное дерево и его организация

Каждая *вершина* *бинарного дерева* является структурой, состоящей из четырех видов полей. Содержимым этих полей будут соответственно:

* информационное поле (ключ вершины);
* служебное поле (их может быть несколько или ни одного);
* указатель на *левое поддерево*;
* указатель на правое *поддерево*.

*По* *степени вершин* *бинарные деревья* делятся на ([рис. 4](http://www.intuit.ru/studies/courses/648/504/lecture/6557?page=1#image.31.4)):



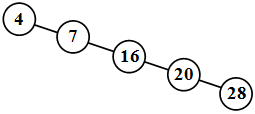
**Рис. 4.**

* строгие – вершины дерева имеют степень ноль (у листьев) или два (у узлов);
* нестрогие – вершины дерева имеют степень ноль (у листьев), один или два (у узлов).

В общем случае у *бинарного дерева* на k -м уровне может быть до 2k-1 вершин. *Бинарное дерево* называется полным, если оно содержит только полностью заполненные уровни. В противном случае оно является неполным.

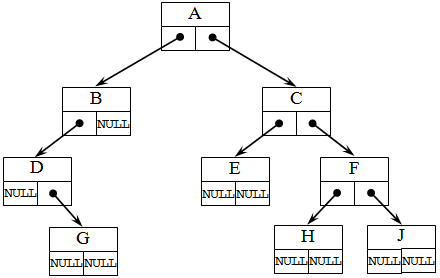
*Дерево* называется сбалансированным, если длины всех путей от корня к внешним вершинам равны между собой. *Дерево* называется почти сбалансированным, если длины всевозможных путей от корня к внешним вершинам отличаются не более, чем на единицу.

*Бинарное дерево* может представлять собой *пустое множество*. *Бинарное дерево* может выродиться в *список* (рис. 5).



**Рис. 5.**Список как частный случай бинарного дерева

Структура дерева отражается во входном потоке данных так: каждой вводимой пустой связи соответствует условный символ, например, '\*' (звездочка). При этом сначала описываются левые потомки, затем, правые. Для структуры *бинарного дерева*, представленного на следующем рисунке 6,  *входной* *поток* имеет вид: ABD\*G\*\*\*CE\*\*FH\*\*J\*\*.



**Рис. 6.**Адресация в бинарном дереве

*Бинарные деревья* могут применяться для поиска данных в специально построенных деревьях (*базы данных*), сортировки данных, вычислений *арифметических выражений*, кодирования (метод Хаффмана) и т.д.

Описание *бинарного дерева* выглядит следующим образом:

**struct** имя\_типа {

информационное поле;

[служебное поле;]

адрес левого поддерева;

адрес правого поддерева;

};

где информационное поле – это *поле* любого ранее объявленного или стандартного типа;

адрес левого (правого) *поддерева* – это *указатель* на *объект* того же типа, что и определяемая структура, в него записывается *адрес* следующего элемента левого (правого) *поддерева*.

Например:

struct point {

int data;//информационное поле

int count; //служебное поле

point \*left;//адрес левого поддерева

point \*right;//адрес правого поддерева

};

Основными операциями, осуществляемыми с бинарными деревьями, являются:

* создание *бинарного дерева*;
* печать *бинарного дерева*;
* обход *бинарного дерева*;
* вставка элемента в *бинарное дерево*;
* удаление элемента из *бинарного дерева*;
* проверка пустоты *бинарного дерева*;
* удаление *бинарного дерева*.

Для описания алгоритмов этих основных операций используется следующее объявление:

struct BinaryTree{

int Data; //поле данных

BinaryTree\* Left; //указатель на левый потомок

BinaryTree\* Right; /указатель на правый потомок

};

. . . . . . . . . .

BinaryTree\* BTree = NULL;

**Задания к лабораторной работе**.

При выполнении лабораторной работы для каждого задания требуется написать программу на языке С, в которой выполнено формирование бинарных деревьев в соответствии с постановкой задачи, ввод данных элементов деревьев с учетом типа информационного поля, их обработка и *вывод* на экран в указанном формате. Для хранения данных бинарных деревьев следует использовать ресурсы динамической памяти. Ввод данных осуществляется с клавиатуры с учетом требований к входным данным, содержащихся в постановке задачи. Ограничениями на *входные данные* являются максимальный размер строковых данных, диапазоны числовых типов полей структуры и допустимый размер области динамической памяти в языке С.

Выполните приведенные ниже задания.

1. Реализуйте программу, в которой выполняются все основные операции с бинарным деревом.
2. Найдите количество четных элементов *бинарного дерева*. Укажите эти элементы и их уровни.
3. Найдите сумму элементов *сбалансированного дерева*, находящихся на уровне k.
4. Оператор мобильной связи организовал базу данных *абонентов*, содержащую сведения о телефонах, их владельцах и используемых тарифах, в виде *бинарного дерева*. Составьте программу, которая:
   * обеспечивает начальное формирование базы данных в виде *бинарного дерева*;
   * производит вывод всей базы данных;
   * производит поиск владельца по номеру телефона;
   * выводит наиболее востребованный тариф (по наибольшему числу *абонентов*).

**Указания к выполнению работы**.

Выполнение работы следует начать с решения задачи 1, реализовав алгоритмы основных операций над бинарным деревом. Каждое из заданий 2, 3 и 4 необходимо решить в соответствии с изученными методами и реализованными алгоритмами формирования, вывода и обработки данных бинарных деревьев в языке С. Обработку бинарных деревьев следует выполнить на основе базовых алгоритмов: *поиск* *по* дереву, вставка элемента в *дерево*, балансировка дерева, *удаление элемента* из дерева, удаление всего дерева.

**Контрольные вопросы**

1. С чем связана популярность использования деревьев в программировании?
2. Можно ли список отнести к деревьям? Ответ обоснуйте.
3. Какие данные содержат адресные поля элемента *бинарного дерева*?
4. Может ли *бинарное дерево* быть строгим и неполным? Ответ обоснуйте.
5. Может ли *бинарное дерево* быть нестрогим и полным? Ответ обоснуйте.
6. Каким может быть почти сбалансированное *бинарное дерево*: полным, неполным, строгим, нестрогим? Ответ обоснуйте.
7. Куда может быть добавлен элемент в *бинарное дерево* в зависимости от его вида (полное, неполное, строгое, нестрогое)? Вид дерева при этом должен сохраниться.
8. Куда может быть добавлен элемент в сбалансированное *бинарное дерево*? Вид дерева при этом должен сохраниться.
9. Чем отличаются, с точки зрения реализации алгоритма, прямой, симметричный и *обратный обходы* *бинарного дерева*?