Лабораторная работа N 2

Тема: Использование структур данных для поиска на деревьях и графах. Алгоритм поиска в глубину

Цель работы: Изучение структур данных и алгоритмов для поиска в глубину на деревьях и графах, а также различных способов обработки структур данных списков и матрицы.

Структуры данных:

Списки

Каждый тип списков определяет множество конечных последовательностей элементов, имеющий заданный базисный тип. Число элементов списка, называемое его длиной, в разных списках одного типа может быть различным. Список, не имеющий элементов, называется пустым. Для списка определены понятия начального, конечного и текущего элементов, а также следующего и предыдущего элементов по отношению к текущему. Текущий элемент - это тот единственный элемент списка, который в данный доступен для обработки.

Очередь

Очередь используется для реализации такой дисциплины обработки элементов списка, при которой элементы списка удаляются из него в порядке их включения в список (т.е. по принципу "Первым пришел - первым ушел").

Набор базисных операций над списками, являющимися очередями, состоит из четырех операций:

1) Создание пустой очереди;
2) Проверка очереди на пустоту;
3) Выборки первого элемента из очереди с одновременным его удалением;
4) Занесения некоторого значения базисного типа в качестве нового последнего элемента очереди.

Стек

Стек используется для реализации такой дисциплины обработки элементов списка, при которой элементы списка удаляются из него в порядке, обратном порядку их занесения в стек (т.е. по принципу "Последним пришел - первым ушел").

Набор базисных операций над списками, являющимися стеками, состоит из пяти операций:

1) Создание пустого стека;
2) Проверка стека на пустоту;
3,4) Выборки последнего элемента из стека без удаления его из стека или с его удалением;

5) Занесения некоторого значения базисного типа в качестве нового последнего элемента стека.

Деревья

Над некоторым базисным типом определяют множество структур, каждая из которых состоит из объекта базисного типа, называемого вершиной или корнем данного дерева, и некоторого списка элементов из определяемого множества, называемых поддеревьями данного дерева. Дерево, в котором список поддеревьев пуст, называется тривиальным, а корень тривиального дерева - листом дерева. Корень дерева называется отцом вершин, являющихся корнями поддеревьев; а эти вершины называются сыновьями корня дерева, причем корень первого поддерева является старшим сыном, а корень каждого следующего поддерева в списке называется братом корня предыдущего поддерева.

Набор базисных операций для деревьев состоит из следующих операций : Создание тривиального дерева по элементу базисного типа и выборки или замены корня дерева или списка его поддеревьев, а также всех операций (для списка поддеревьев), которые являются базисными для списка.

Обход дерева и графа в глубину

При обходе дерева в глубину (известном также как прохождение в прямом порядке) вершины дерева посещаются в соответствии со следующей рекурсивной процедурой: сначала посетить корень дерева q, затем если корень рассматриваемого дерева не является листом, то для каждого сына p корня q рекурсивно обратиться к процедуре обхода в глубину, для того чтобы обойти все поддеревья с корнями p в порядке упорядоченности вершин p как сыновей корня q. Если использовать стек S для хранения текущего пути по дереву, т.е. пути, который начинается в корне дерева и заканчивается в вершине, посещаемой в данный момент, то можно построить нерекурсивный алгоритм для обхода дерева в глубину [3, 4]:

```
Посетить корень дерева и поместить его в пустой стек S;
WHILE стек S не является пустым DO
BEGIN
        пусть p - вершина, находящаяся на верху стека S;
        IF сыновья вершины p еще не посещались
        THEN посетить  старшего сына вершины  p и поместить  его в
                стек S
        ELSE BEGIN
                удалить вершину p из стека S
                IF p  имеет братьев THEN  посетить брата вершины  p и
                        поместить его в стек S
        END
END
```

Алгоритм обхода дерева в глубину можно модифицировать таким образом, чтобы его можно было использовать для систематического обхода всех вершин произвольного графа [4]. В приведенной ниже формулировке алгоритма поиска в глубину на графе предполагается, во-первых, что зафиксирован некоторый линейный порядок на множестве всех вершин графа, и, во-вторых, что множество вершин, смежных со всякой вершиной графа, также линейно упорядочено:

```
WHILE имеется хотя бы одна не посещенная вершина DO
BEGIN
    пусть p - первая (т.е. минимальная) из не посещенных вершин;
    посетить вершину p и поместить ее в пустой стек S;
    WHILE стек S не пуст DO
    BEGIN
        пусть p - вершина, находящаяся на верхушке стека S;
        IF у вершины p есть не посещенные смежные вершины
        THEN BEGIN
            пусть q - первая не посещенная вершина из вершин,
            смежных вершине p;
            пройти по ребру (p,q), посетить вершину q и
            поместить ее в стек S;
        END
        ELSE удалить вершину p из стека S
    END
END
```

В результате работы алгоритма, пройденные ребра графа образуют вместе с посещенными вершинами одно или несколько деревьев. Если приписать пройденным ребрам ориентацию в соответствии с тем направлением, в каком они проходят при выполнении алгоритма, то мы получим совокупность корневых деревьев, причем их корнями будут служить все те вершины, которые в процессе работы алгоритма помещались в пустой стек.

В том случае, когда мы имеем дело с произвольным связанным графом, вершины которого не имеют линейной упорядоченности, порядок прохода вершин не имеет значения. Поэтому предлагается другой алгоритм, не обеспечивающий строгого порядка прохода вершин и более широко использующий возможности стека, что приводит к увеличению быстродействия программ, построенных на его основе. В частности, этот алгоритм в рекурсивном исполнении широко используется в программах, выполняющих глобальный поиск по подкаталогам на дисках (например, в антивирусах).

```
Поместить в стек исходную вершину графа и пометить ее;
WHILE стек не пуст DO
 BEGIN
        извлечь вершину из стека;
       IF есть непомеченные вершины и смежные с данной
                THEN пометить их и загрузить в стек;
 END
```

Задание для лабораторной работы № 2:

1. Основываясь на одном из алгоритмов, разработать процедуры, выполняющие обход графа и дерева в глубину.

2. Используя разработанные процедуры, создать для заданного варианта **графа** программу, выполняющую следующие функции:

- ввод графа в виде списка смежности;
- обход введенного графа в глубину (стартовая вершина задается преподавателем)
- выдачу номеров пройденных вершин на дисплей;

3. Используя разработанные процедуры, создать для заданного варианта **дерева** программу, выполняющую следующие функции:

- ввод дерева в виде списка смежности;
- обход введенного дерева в глубину;
- выдачу номеров пройденных вершин на дисплей.

Контрольные вопросы:

1. Что представляют собой структуры данных: списки, очереди, стеки, деревья?

2. По какому принципу организована обработка элементов в очередях и стеках ?

3. Что такое обход дерева и графа в глубину?

4. Какие структуры данных используются в алгоритмах поиска в глубину ?

5. Для чего используется алгоритм поиска в глубину?