

Лабораторная работа №4: Рекурсивные структуры данных. Деревья

Цель работы

Основные теоретические положения

Деревья, также как и списки, являются рекурсивным типом данных. Дерево – это структура данных, которая может быть разделена на корень дерева, левое и правое поддеревья. Так как левое и правое поддеревья в свою очередь являются деревьями, структура рекурсивна. Кроме того, дерево является еще и составным объектом данных.

Дерево, которое имеет только два поддерева, называется двоичным или бинарным. В том случае, если для каждого корня дерева выполняется условие, при котором значение, находящееся в корне дерева меньше значения, находящегося в корне левого поддерева и больше значения, находящегося в корне правого поддерева, двоичное дерево называется упорядоченным.

В Visual Prolog можно определить дерево следующим образом:

DOMAINS

```
treetype=tree(integer, treetype, treetype); empty()
```

Такое определение говорит о том, что дерево является составным объектом, состоящим из трех составных частей: корня, принадлежащего домену `integer` и двух поддеревьев, принадлежащих домену `treetype`, так как именно этот домен и описывает структуру данных типа дерево. Так как дерево является составным объектом, его составные части объединяет функтор `tree`. Кроме того, дерево может находиться в двух состояниях: быть непустым (иметь хотя бы один корень) или пустым (не иметь ни одного корня). Пустое дерево описывается функтором `empty` без параметров. Если у функтора нет параметров, пустые скобки можно не указывать и записывать только имя функтора. Имена функторов `tree` и `empty`, домена `treetype` выбраны произвольно. Такое определение позволяет записать следующую структуру данных:

```
tree(5,
    tree(3,
        tree(6, empty, empty),
        tree(4, empty, empty)),
    tree(10,
        tree(2, empty, empty),
        tree(8, empty, empty)))
```

Одной из наиболее частых операций с деревом является обход узлов дерева и выполнение некоторых действий с ними. Например, вывод значений всех корней дерева. Способ решения этой задачи можно описать следующим образом:

1. если дерево пустое, корня в дереве нет, нет и значения корня для вывода, не выполнять никаких действий;
2. если дерево непустое, то разделить дерево на корень, левое и правое поддеревья, выполнить вывод значения, находящегося в корне дерева, затем обработать левое и правое поддеревья.

Каждое из условий в описании задачи соответствует предложению в программе Visual Prolog.

DOMAINS

```
treetype = tree(integer, treetype, treetype); empty()
```

PREDICATES

```
print_tree(treetype)
```

CLAUSES

```
% дерево пусто, поэтому никакие действия не выполняются
```

```
print_tree(empty): -!.
```

```
% дерево непусто, поэтому дерево разбивается на три составные части,
```

```
% сначала выводится корень, затем обрабатываются левое и правое
```

```
% поддеревья
```

```
print_tree(tree(Root, Left, Right)) :- write(Root), nl,  
                                       print_tree(Left),  
                                       print_tree(Right).
```

OAL

```
print_tree(tree(5,  
               tree(3,  
                   tree(6, empty, empty),  
                   tree(4, empty, empty)),  
               tree(10,  
                   tree(2, empty, empty),  
                   tree(8, empty, empty)))).
```

Следует отметить, что в большинстве случаев рекурсия, используемая при работе с деревьями, хвостовой не является, так приходится обрабатывать левое и правое поддеревья, что дает две рекурсивные цели в одном предложении и, соответственно, не выполняется первое правило хвостовой рекурсии - рекурсивный вызов должен быть последней целью в хвостовой части правила вывода.

Постановка задачи

Порядок выполнения работы

Варианты заданий

Содержание отчёта

From:

<http://se.moevm.info/> - **se.moevm.info**

Permanent link:

http://se.moevm.info/doku.php/courses:knowledge_base_and_expert_system:lab4?rev=1567034456 

Last update: **2022/12/10 09:08**