Министерство науки и высшего образования РФ

Севастопольский государственный университет

Кафедра информатики и управления в технических системах

ОТЧЕТ

по лабораторной работе №6

ТИПОВЫЕ ОПЕРАЦИИ С БИНАРНЫМИ ДЕРЕВЬЯМИ

по дисциплине «Логическое программирование»

Выполнил:

Студент группы ИВТ/б 22-о

Черняев Н.Г.

Проверил:

Ткаченко К.С.

г. Севастополь 2019

**Цель работы**

Научиться описывать рекурсивные типы данных, выработать навыки представления структур в виде дерева и реализации типовых процедур с деревьями.

**Вариант № 23(3, 7, 4)**

Задание:

Решить задачи согласно варианту.

* Вариант 3 - Определите отношение **линеаризация( Дерево, Список)** соответствующее "выстраиванию" всех вершин дерева в список.
* Вариант 4 - Определите отношение **максэлемент( Д, Элемент)** таким образом, чтобы переменная Элемент приняла значение наибольшего из элементов, хранящихся в дереве Д.
* Вариант 7 - Определите отношение **ordered(Tree)**, выполненное, если дерево Tree является упорядоченным деревом целых чисел, то есть число, стоящее в любой вершине дерева больше любого элемента в левом поддереве и меньше любого элемента в правом поддереве.

**Ход работы**

**Задание по варианту 3.**

Определим отношение **линеаризация( Дерево, Список)** соответствующее "выстраиванию" всех вершин дерева в список.

Текст программы:

implement main   
    open core   
constants   
    className = "main".   
    classVersion = "".  
domains  
    treetype=tree(integer, treetype, treetype); nil().  
    intl=integer\*.  
class predicates   
    append :(intl,intl,intl) procedure(i,i,o).  
    tree2list :(treetype,intl) procedure(i,o).  
clauses  
    classInfo(className, classVersion).  
      
    append([],L2,R):- R=L2,!.   
    append([H1|[]],L2,R):-R = [H1|L2],!.   
    append([H1|T1],L2,R):-append(T1,L2,R1), R=[H1|R1],!.   
    append(\_,\_,[]).   
   
    tree2list(nil,[]).  
    tree2list(tree(N,L,R),W) :- tree2list(L,LL), tree2list(R,RR), append(LL,[N],LN), append(LN,RR,W).  
clauses  
    run():-   
        console::init(),tree2list(tree(5,tree(3,tree(2,nil,nil),tree(4,nil,nil)),tree(7,tree(6,nil,nil),tree(8,nil,nil))),Z),stdio::write(Z),stdio::nl,!.  
end implement main   
goal  
    mainExe::run(main::run).

Результат выполнения программы:



**Задание по варианту 4.**

Определим отношение **максэлемент( Д, Элемент)** таким образом, чтобы переменная Элемент приняла значение наибольшего из элементов, хранящихся в дереве Д.

Текст программы:

implement main   
    open core   
constants   
    className = "main".   
    classVersion = "".  
domains  
    treetype=tree(integer, treetype, treetype); nil().  
class predicates  
     max2 :(integer,integer,integer) nondeterm(i,i,o).  
     max3 :(integer,integer,integer,integer) nondeterm(i,i,i,o).  
     max :(treetype,integer) nondeterm(i,o).

clauses

max2(M1,M2,M1) :- M1>=M2.  
    max2(M1,M2,M2) :- M2>=M1.  
   
    max3(M1,M2,M3,M1) :- M1>=M2, M1>=M3.  
    max3(M1,M2,M3,M2) :- M2>=M1, M2>=M3.  
    max3(M1,M2,M3,M3) :- M3>=M2, M3>=M1.  
      
    max(tree(N,nil,nil),N).  
    max(tree(N,L,nil),M) :- max(L,M1), max2(N,M1,M).  
    max(tree(N,nil,R),M) :- max(R,M1), max2(N,M1,M).  
    max(tree(N,L,R),M)   :- max(L,M1), max(R,M2), max3(M1,M2,N,M).

clauses  
    run():-  
       console::init(),max(tree(5,tree(3,tree(2,nil,nil),tree(4,nil,nil)),tree(7,tree(6,nil,nil),tree(8,nil,nil))),M), stdio::write(M), stdio::nl,!.  
   run():-  
        console::write("End").  
end implement main   
goal   
    mainExe::run(main::run).

Результат выполнения программы:



**Задание по варианту 7.**

Определим отношение **ordered(Tree)**, выполненное, если дерево Tree является упорядоченным деревом целых чисел, то есть число, стоящее в любой вершине дерева больше любого элемента в левом поддереве и меньше любого элемента в правом поддереве.

Текст программы:

implement main   
    open core   
constants   
    className = "main".   
    classVersion = "".  
domains  
    treetype=tree(integer, treetype, treetype); nil().  
class predicates  
    order :(treetype) determ(i).  
    order :(treetype,integer,integer) determ(i,i,i).  
    left :(treetype,integer) determ(i,o).  
    right :(treetype,integer) determ(i,o).  
clauses   
    classInfo(className, classVersion).  
      
    order(nil):-!.  
    order(tree(H,nil,RightTree)):-right(RightTree,R),order(RightTree,H,R),!.  
    order(tree(H,LeftTree,nil)):-left(LeftTree,L),order(LeftTree,L,H),!.  
    order(tree(H,LeftTree,RightTree)):-right(RightTree,R),left(LeftTree,L),  
    order(LeftTree,L,H),order(RightTree,H,R).  
    
    left(tree(H,nil,\_),H1):-!,H1=H-1.  
    left(tree(\_,LeftTree,\_),L):-left(LeftTree,L).  
    
    right(tree(H,\_,nil),H1):-!,H1=H+1.  
    right(tree(\_,\_,RightTree),R):-right(RightTree,R).  
    
    order(nil,\_,\_).  
    order(tree(H,LeftTree,RightTree),A,B):-H>A,H<B,  
    order(LeftTree,A,H),order(RightTree,H,B).  
      
clauses  
    run():-  
        console::init(),  Tree =tree(5,tree(3,tree(2,nil,nil),tree(4,nil,nil)),tree(7,tree(6,nil,nil),tree(8,nil,nil))),   
  order(Tree),!,stdio::write("Дерево упорядоченное");stdio::write("Дерево не упорядоченное"),!.  
end implement main   
goal   
    mainExe::run(main::run).

Результат выполнения программы:



**Выводы**

В данной лабораторной работе были изучены рекурсивные типы данных, выработаны навыки представления структур в виде дерева и реализации типовых процедур с деревьями.