|  |  |  |
| --- | --- | --- |
|  | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования FPMI_ngtu_neti_rgb_polya«Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра теоретической и прикладной информатики | | |
| Практическое задание № 5 | | |
| по дисциплине «Введение в искусственный интеллект и логическое программирование» | | |
| Место для ввода текста. | | |
|  | | |
|  | Место для ввода текста. |  |
| Группа ПМ-13 | Вострецова екатерина |
| Вариант 4 | ИСАКИН ДАНИИЛ |
|  |  |
|  |  |
|  |  |
| Преподаватели | Авдеенко татьяна владимировна |
|  | целебровская марина юрьевна |
| Новосибирск, 2024 | | |

1. **Цель работы**

Изучение логических программ, задающих отношения над объектами ре-курсивных типов и приобретение навыков самостоятельной разработки таких программ.

1. **Задание для исследования**

4. Обход бинарного дерева [1, программа 3.27, с.57];

- сверху вниз;

- слева направо;

- снизу вверх.

Программа без отсечения

domains

exe=tree(symbol,exe,exe);nil

list=symbol\*

predicates

nondeterm append(list,list,list)

nondeterm pre\_order(exe,list)

nondeterm in\_order(exe,list)

nondeterm post\_order(exe,list)

clauses

append ([],Ys,Ys).

append ([X|Xs],Ys,[X|Zs]):- append (Xs,Ys,Zs).

%up to down

pre\_order(tree(X,L,R),Xs):-

pre\_order(L,Ls),

pre\_order(R,Rs),

append ([X|Ls],Rs,Xs).

pre\_order(\_,[]).

%left to right

in\_order(tree(X,L,R),Xs):-

in\_order(L,Ls),

in\_order(R,Rs),

append (Ls,[X|Rs],Xs).

in\_order(\_,[]).

%down to up

post\_order(tree(X,L,R),Xs):-

post\_order(L,Ls),

post\_order(R,Rs),

append (RS,[X],Rsl),

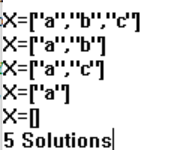
append (Ls,Rsl,Xs).

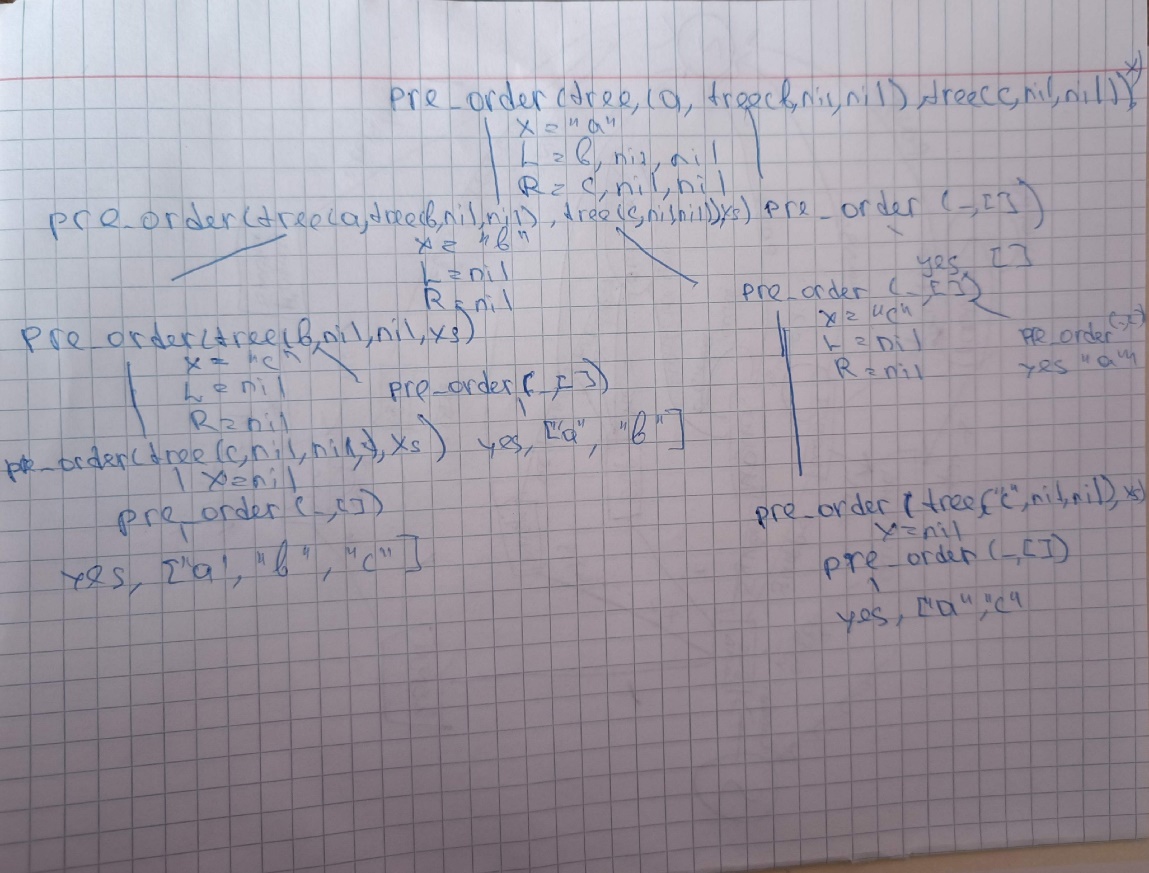
post\_order(\_,[]).

goal

pre\_order(tree("a",tree("b",nil,nil),tree("c",nil,nil)),X).

Результат работы:





Программа с отсечением

domains

exe=tree(symbol,exe,exe);nil

list=symbol\*

predicates

nondeterm append (list,list,list)

nondeterm pre\_order(exe,list)

nondeterm in\_order(exe,list)

nondeterm post\_order(exe,list)

clauses

append ([],Ys,Ys).

append ([X|Xs],Ys,[X|Zs]):- append (Xs,Ys,Zs).

%up to down

pre\_order(tree(X,L,R),Xs):-

pre\_order(L,Ls),

pre\_order(R,Rs),

append ([X|Ls],Rs,Xs),

!.

pre\_order(\_,[]).

%left to right

in\_order(tree(X,L,R),Xs):-

in\_order(L,Ls),

in\_order(R,Rs),

append (Ls,[X|Rs],Xs),

!.

in\_order(\_,[]).

%down to up

post\_order(tree(X,L,R),Xs):-

post\_order(L,Ls),

post\_order(R,Rs),

append (RS,[X],Rsl),

append (Ls,Rsl,Xs),

!.

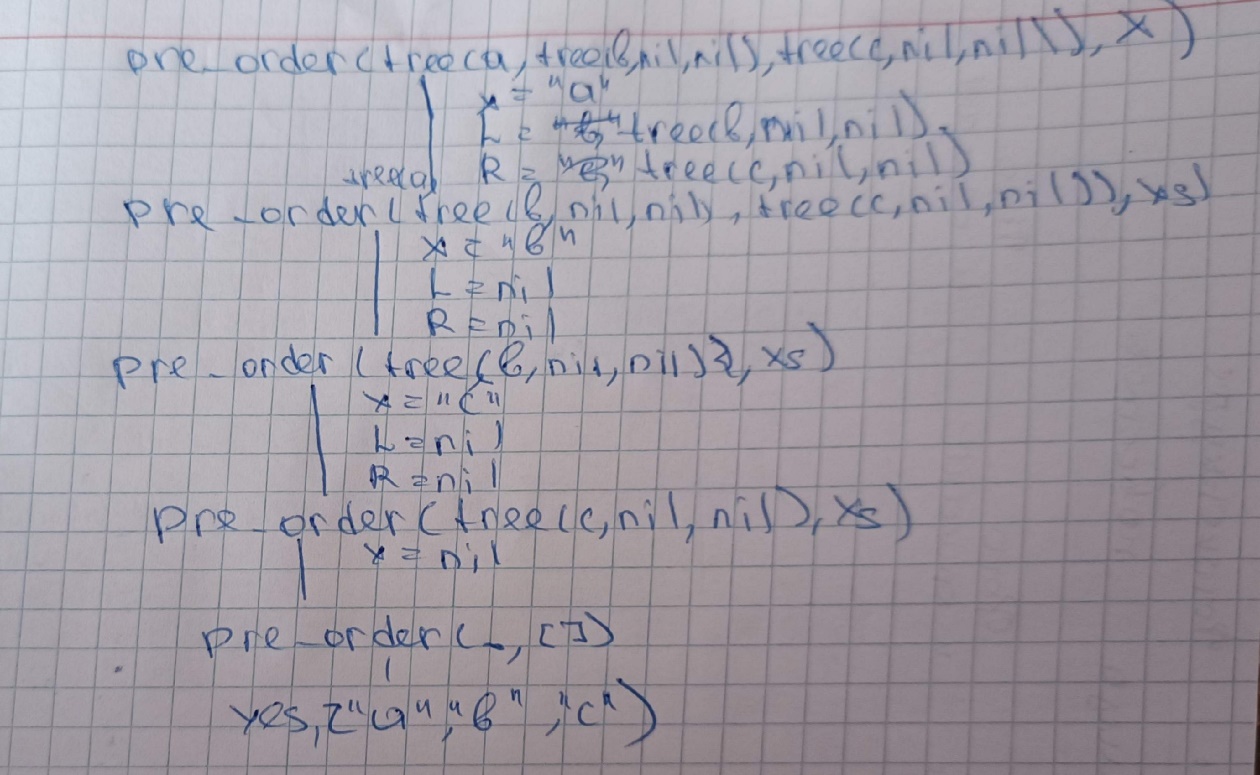
post\_order(\_,[]).

goal

pre\_order(tree("a",tree("b",nil,nil),tree("c",nil,nil)),X).

Результат работы программы





Так как результат работы изменился, то такое отсечение называется «красным»

Теперь попробуем сделать отсечение таким образом, чтобы оно стало «зеленым»

domains

exe=tree(symbol,exe,exe);nil

list=symbol\*

predicates

nondeterm append (list,list,list)

nondeterm pre\_order(exe,list)

nondeterm in\_order(exe,list)

nondeterm post\_order(exe,list)

clauses

append ([],Ys,Ys).

append ([X|Xs],Ys,[X|Zs]):- append (Xs,Ys,Zs).

%up to down

pre\_order(tree(X,L,R),Xs):-

pre\_order(L,Ls),

pre\_order(R,Rs),

append ([X|Ls],Rs,Xs).

pre\_order(\_,[]):-!.

%left to right

in\_order(tree(X,L,R),Xs):-

in\_order(L,Ls),

in\_order(R,Rs),

append (Ls,[X|Rs],Xs).

in\_order(\_,[]):-!.

%down to up

post\_order(tree(X,L,R),Xs):-

post\_order(L,Ls),

post\_order(R,Rs),

append (RS,[X],Rsl),

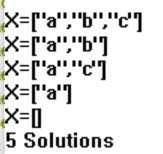
append(Ls,Rsl,Xs).

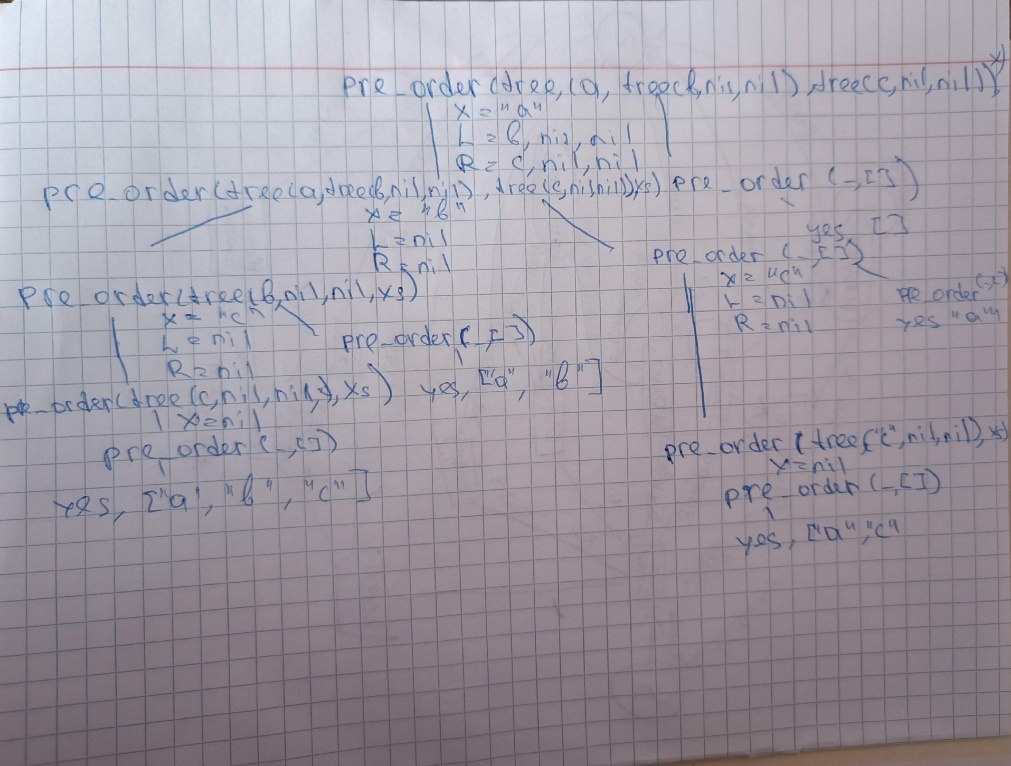
post\_order(\_,[]):-!.

goal

pre\_order(tree("a",tree("b",nil,nil),tree("c",nil,nil)),X).

Результат работы:





Так как добавления отсечения в данном случае не повлияло на результат без отсечений, то такое отсечение называется «зеленым».

**Результаты работы программы**

**pre\_order(tree("a",tree("b",nil,nil),tree("c",nil,nil)),X).**

****

**in\_order(tree("a",tree("b",nil,nil),tree("c",nil,nil)),X).**

****

**post\_order(tree("a",tree("b",nil,nil),tree("c",nil,nil)),X).**

****

1. **Индивидуальное задание**

Напишите программу, которая преобразует логическую формулу F1 в логическую формулу F2 внесением всех операторов отрицания внутрь конъюнкций и дизъюнкций.

Текст программы

domains

pr=var(symbol);true;false;con(pr,pr);diz(pr,pr);no(pr)

predicates

nondeterm conversion(pr,pr)

nondeterm find(pr)

nondeterm result(pr,pr)

nondeterm check(pr,pr)

clauses

result(X,Z):-conversion(X,Y),find(Y),result(Y,Z),!.

result(X,Z):-conversion(X,Y),check(Y,Z),!.

find(no(con(\_,\_))).

find(no(diz(\_,\_))).

find(diz(X,Y)):- find(X);find(Y).

find(con(X,Y)):- find(X);find(Y).

conversion(no(con(X,Y)),diz(no(A),no(B))):-conversion(X,A),conversion(Y,B),!.

conversion(no(diz(X,Y)),con(no(A),no(B))):-conversion(X,A),conversion(Y,B),!.

conversion(no(no(X)),Y):-conversion(X,Y).

conversion(no(X),no(X)).

conversion(diz(X,Y),diz(C,D)):-conversion(X,C),conversion(Y,D),!.

conversion(con(X,Y),con(C,D)):-conversion(X,C),conversion(Y,D),!.

conversion(X,X).

check(con(X,Y),con(A,B)):-check(X,A),check(Y,B).

check(diz(X,Y),diz(A,B)):-check(X,A),check(Y,B).

check(no(true),false).

check(no(false),true).

check(no(X),no(X)).

goal

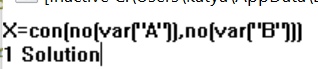
result(no(con(var("A"),var("B"))),X).

Тесты

result(no(con(var("A"),var("B"))),X).



result(no(diz(var("A"),var("B"))),X).



result(diz(no(con(var("X"),var("Y"))),no(diz(var("C"),var("D")))),Result).

