**Федеральное агентство по образованию**

**ГОУ ВПО Нижегородский государственный технический университет им. Р.Е. Алексеева**

**Институт радиоэлектроники и информационных технологий**

**Кафедра “Вычислительные системы и**

**Технологии”**

**Функциональное и логическое программирование**

**Отчёт по лабораторной работе №3**

Студент группы 21-ПО

Малинок С.М. ­­­­­\_\_\_\_\_\_\_

Провел преподаватель кафедры ВСТ

Тарасов А.В.\_\_\_\_\_\_\_

Г. Нижний Новгород 2024

Задание

Имеется список, элементы которого – непустые бинарные деревья. Для каждого элемента списка найти глубину дерева, из полученных результатов сформировать список (без использования стандартного предиката findall). Для полученного списка выполнить циклический сдвиг вправо на заданное число элементов.

Код программы

% Определение предиката для нахождения глубины бинарного дерева

depth(empty, 0). % Глубина пустого дерева равна 0

depth(tree(\_, Left, Right), Depth) :- % Глубина непустого дерева

depth(Left, LeftDepth), % Находим глубину левого поддерева

depth(Right, RightDepth), % Находим глубину правого поддерева

Depth is max(LeftDepth, RightDepth) + 1. % Глубина текущего дерева - максимальная из глубин поддеревьев плюс 1

% Предикат для нахождения глубины каждого дерева в списке

depth\_list([], []). % Базовый случай: для пустого списка возвращаем пустой список глубин

depth\_list([Tree|Trees], [Depth|Depths]) :- % Находим глубину для каждого дерева в списке

depth(Tree, Depth), % Находим глубину текущего дерева

depth\_list(Trees, Depths). % Рекурсивно обрабатываем оставшиеся деревья в списке

% Предикат для циклического сдвига списка вправо на один элемент

shift\_right([Last|Rest], Shifted) :-

append(Rest, [Last], Shifted).

% Предикат для циклического сдвига списка вправо на заданное количество раз

shift\_right\_n(Times, List, Shifted) :-

length(List, Length),

Shift is Times mod Length, % Учитываем случаи, когда сдвиг больше длины списка

shift\_right\_n\_times(Shift, List, Shifted).

shift\_right\_n\_times(0, List, List) :- !.

shift\_right\_n\_times(N, List, Shifted) :-

shift\_right(List, ShiftedTemp),

NextN is N - 1,

shift\_right\_n\_times(NextN, ShiftedTemp, Shifted).

Результат работы

