**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра МО ЭВМ**

**ОТЧЕТ**

**по лабораторной работе № 2**

**по дисциплине «Логическое программирование»**

**Тема: РЕКУРСИЯ И СТРУКТУРЫ ДАННЫХ**

|  |  |  |
| --- | --- | --- |
| Студент гр. 0304 |  | Козиков А.Е. |
| Студент гр. 0304 |  | Жиглов Д.С. |
| Студент гр. 0304 |  | Докучаев Р.Д. |
| Преподаватель |  | Родионов С.В. |

Санкт-Петербург

2024

## Цели работы

Целью работы является изучение особенностей реализации рекурсии на языке Пролог, освоение принципов решения типовых логических программ.

Для достижения поставленной цели требуется решить следующие задачи:

1) Изучить теоретический материал.

2) Создать правила в соответствии с вариантом задания и общей формулировкой задачи (п.3).

3) Проверить выполнение программы.

4) Составить отчет о выполнении работы.

5) Представить на проверку файл отчета и файл текста программы на языке GNU Prolog, решающей поставленные задачи.

Номер варианта и текст варианта задания должны быть представлены в форме комментариев в тексте программы. Номер группы и номер варианта должны присутствовать в имени файла с текстом программы.

## Постановка задачи

Необходимо реализовать выполнение задания с номером варианта, равным номеру бригады (для каждого варианта - по две задачи, одна – из Задания 1, вторая – из Задания 2).

Рекомендуется во всех заданиях использовать рекурсивную обработку списка, с разделением его элементов на голову и хвост; можно определять/использовать вспомогательные предикаты.

Вариант 4.

Задание 1. Осуществить перевод числа из десятичной системы счисления в двоичную; результат представить в виде списка двоичных цифр, читаемых слева-направо

?- binary(10, X).

X = [0,1,0,1]

Задание 2. Реализуйте предикат, возвращающий количество листьев заданного бинарного дерева (т.е. узлов, не имеющих потомков).

?- countLeafs(tr(5, tr(4, nil, nil), tr(8, tr(6, tr(3, nil, nil), nil), tr(9, nil, nil))), X).

X = 3.

## Выполнение работы

*Правило для перевода десятичного числа в двоичную форму.*

Для перевода десятичного числа в двоичную систему, необходимо это число делить на 2 и сохранять остаток от деления, а в конце объединить остатки и развернуть их в списке в другую сторону.

Для этого нам нужно реализовать правило и задать 2 базовых случая. 1 и 2 базовые случаи – двоичные представления чисел 0 и 1 состоят из цифр 0 и 1 соответственно.



Рисунок 1. Терминальные состояния

В самом же рекурсивном правиле необходимо написать проверку, является ли число больше 1. После задаются переменные для остатка и частного от деления, и прописывается рекурсивный вызов для остатка от деления. В конце правила все остатки склеиваются, а так как остатки получены в ходе работы рекурсии, список уже получается перевернутым.

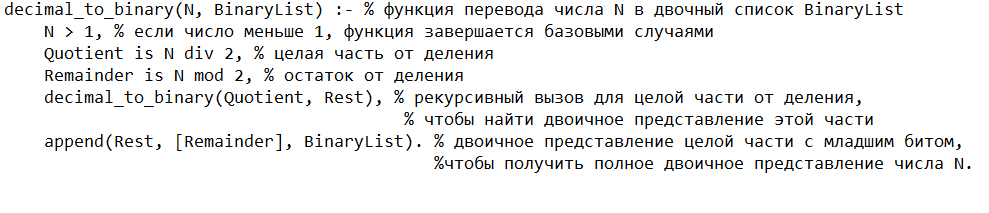


Рисунок 2. Правило для перевода двоичного числа в десятичное

Попробуем запустить написанные правила для 3 чисел и проверим, корректно ли отрабатывает функция (см. рис. 3).

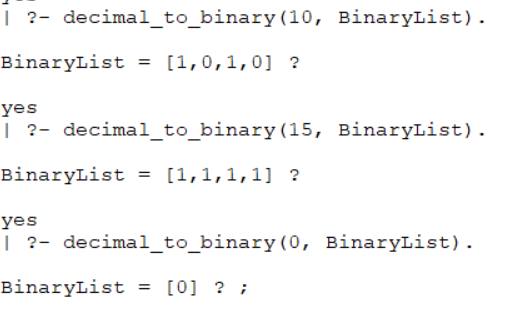


Рисунок 3. Результат работы правила для перевода числа в двоичную форму .

Как можно заметить, все вызовы вывели корректные результаты.

*Правило для определения количества листьев в дереве .*

Листьями в дереве являются элементы дерева, на которые не ссылаются другие элементы.

Бинарное дерево— иерархическая структура данных, в которой каждый узел имеет не более двух потомков

Пример дерева с 3 листьями приведен ниже(см. рис. 6)

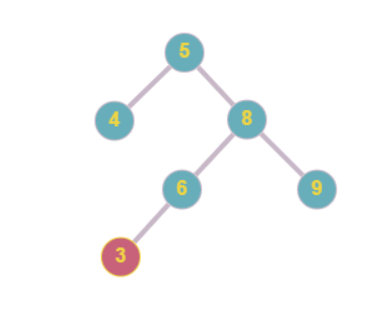


Рисунок 4. Бинарное дерево с 3 листьями

Для подсчета количества листьев в дереве нужно реализовать 3 правила. 1 базовое правило для пустого дерева, и 1 правило для узла, у которого 2 потомка равняются ‘nil’, в таком случае узел считается листом.

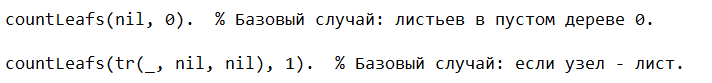


Рисунок 5. 2 базовых правила для бинарного дерева

3 правило является рекурсивным, и применяется, когда узел не является листом и имеет поддеревья. Он вызывает countLeafs для левого и правого поддеревьев, чтобы рекурсивно подсчитать количество листьев в них, а затем суммирует результаты.

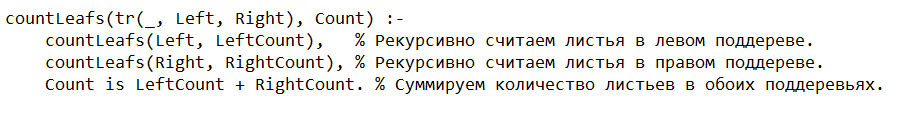


Рисунок 6. Рекурсивное правило для подсчета количества листьев

Проверим бинарные деревья, приведенные на данных рисунках.

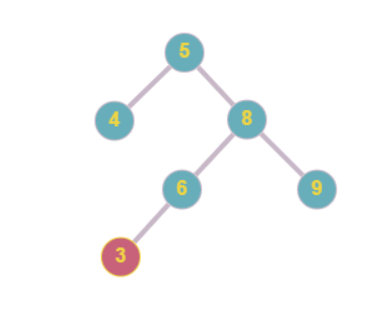


Рисунок 7. Бинарное дерево с 3 листьями

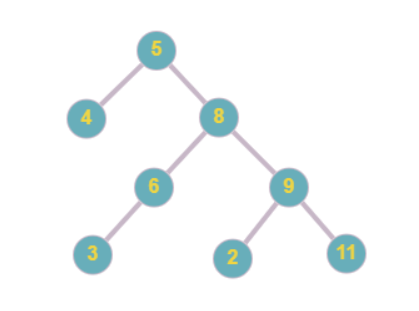


Рисунок 8. Бинарное дерево с 4 листьями

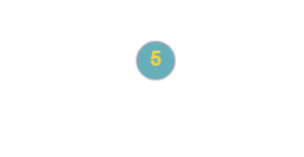


Рисунок 9. Бинарное дерево с 1 листом

Далее следует вызов функции для проверки работы алгоритма.

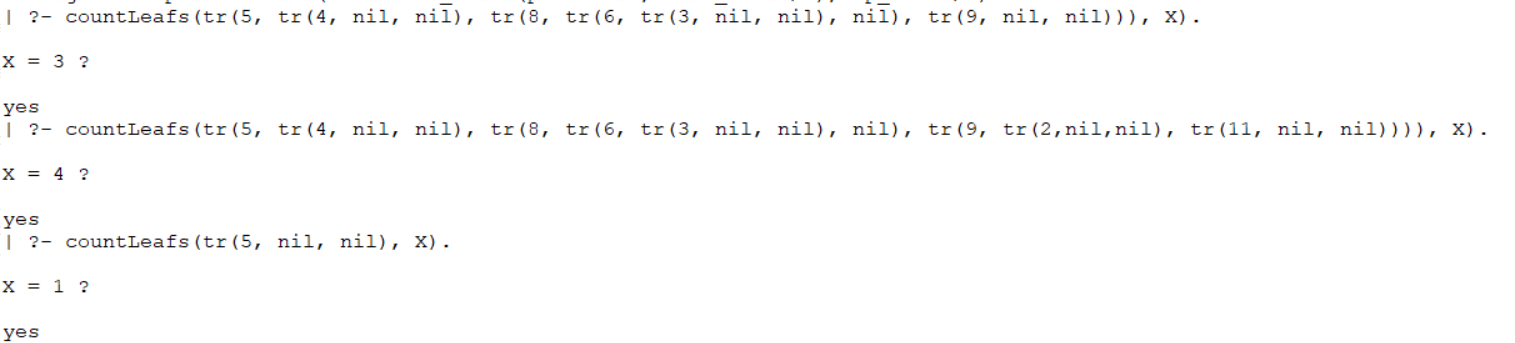


Рисунок 10. Результат работы правила для определения количества листьев.

Как можно заметить, все вызовы вывели корректные результаты.

**Сложности**

1. Работать в условиях, когда доступна только рекурсия для манипуляции со списком, ощущается непривычно.

2. Непривычно работать с деревьями на логическом языке программирования.

3. Для оптимальной работы со списками в прологе необходимо детально понимать рекурсию, её раскрутку и как будут выполняться операции при её обратном ходе.

**Роли в работе**

Козиков А. - Реализовал Задание №1.

Жиглов Д. - Реализовал Задание №2.

Докучаев Р. - Написал тестовые сценарии и протестировал обе программы, подготовил отчет по проделанной работе.

**Выводы**

В ходе выполнения работы были изучены особенности реализации рекурсии на языке Пролог, освоены принципы решения типовых логических программ и реализованы две программы на языке Пролог для решения двух поставленных задач: перевод числа из десятичной системы в двоичную и определение количества листьев в бинарном дереве.