Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по домашнему заданию.

Выполнил:
студент группы ИУ5-32Б
Носков Алексей

Проверил:
преподаватель каф. ИУ5
Гапанюк Юрий
Евгеньевич

Москва, 2024 г.

Язык программирования Prolog

Prolog — это логический языки программирования, основанный на предикатной логике. Он используется для решения задач, связанных с искусственным интеллектом, и отлично подходит для обработки естественного языка и разработки экспертных систем. Основные возможности языка Prolog включают работу с фактами, правилами и запросами.

Prolog (от англ. Programming in Logic) — это декларативный язык программирования, основанный на логике предикатов. Он широко используется в области искусственного интеллекта, особенно для задач, связанных с обработкой естественного языка, экспертными системами и базами данных. В этой статье мы рассмотрим основные аспекты Prolog, включая его синтаксис, идеи, логику и возможности.

Синтаксис Prolog

Основные элементы

Факты: Факты представляют собой базовые утверждения, которые являются истинными.

```
% Факт: Сократ — человек.
человек(сократ).

% Факт: Сократ — философ.
философ(сократ).
```

Правила: Правила определяют отношения между фактами.

```
% Правило: Если кто-то философ, то он мудрый.
мудрый(X) :- философ(X).
```

Запросы: Запросы используются для поиска информации в базе знаний.

```
% Запрос: Кто мудрый?
?- мудрый(X).
```

Переменные и условия

Переменные в Prolog начинаются с большой буквы или символа подчеркивания.

```
% Правило: Если X — человек и X — философ, то X — мудрый.
мудрый(X) :- человек(X), философ(X).
```

Списки

Списки в Prolog записываются в квадратных скобках.

% Список: [голова|хвост]
список([1, 2, 3]).

Идеи и Логика Prolog. Декларативность

Prolog — декларативный язык, что означает, что программист описывает, что должно быть сделано, а не как это должно быть сделано. Это позволяет сосредоточиться на логике задачи, а не на деталях реализации.
Рекурсия

Рекурсия — это основной механизм для выражения повторяющихся задач в Prolog.

% Правило: Факториал числа N.
факториал(0, 1).
*факториал(N, F) :- N > 0, N1 is N - 1, факториал(N1, F1), F is N * F1.*

Унификация

Унификация — это процесс сопоставления термов, который используется для выполнения запросов и применения правил.

% Правило: Если X — человек, то X — смертен.
смертен(X) :- человек(X).

Возможности Prolog. Обработка естественного языка

Prolog широко используется для обработки естественного языка благодаря своей способности работать с символьными данными и сложными структурами.

% Правило: Если X — существительное, то X — слово.
слово(X) :- существительное(X).

Экспертные системы

Prolog идеально подходит для создания экспертных систем, где знания представлены в виде правил и фактов.

% Правило: Если у пациента жар и кашель, то у пациента грипп.
грипп(Пациент) :- жар(Пациент), кашель(Пациент).

Базы данных

Prolog можно использовать для создания и запроса баз данных, где данные представлены в виде фактов.

```
% Факт: Иван — студент.
студент(иван).

% Запрос: Кто студент?
?- студент(X).
```

Пример программы на Prolog

Рассмотрим пример программы, которая определяет семейные отношения.

```
% Факты: Родители и дети.
родитель(иван, петр).
родитель(мария, петр).
родитель(петр, alex).

% Правило: Если X — родитель Y, то Y — ребенок X.
ребенок(Y, X) :- родитель(X, Y).

% Правило: Если X — родитель Y и Y — родитель Z, то X — дедушка/бабушка Z.
дедушка_бабушка(X, Z) :- родитель(X, Y), родитель(Y, Z).

% Запрос: Кто дедушка/бабушка alex?
?- дедушка_бабушка(X, alex).
```

Решение конкретных задач на Prolog.

Настоящее домашнее задание представляет собой не только теоретическое описание возможностей языка парадигмы логического программирования, но и решение трёх конкретных алгоритмических задач.

Задача 1. Шахматные клетки

Условие задачи

На вход программе подаётся две шахматных клетки в формате стандартной шахматной нотации (например, a1 — левый нижний угол доски, если смотреть со стороны белых). Требуется определить, одного ли цвета эти две клетки. В случае положительного исхода напечатать «YES», в противном случае - «NO».

Текст программы

```
:- prompt(_, '').

spaces(" \s\n\t").

read_input(X1, Y1, X2, Y2) :-
    read_string(current_input, _, Input),
    spaces(S),
    split_string(Input, S, S, Words),
    Words = [Str],
    string_codes(Str, Codes),
    Codes = [A1, Y1, A2, Y2],
    X1 is A1 - 0'a + 1,
    X2 is A2 - 0'a + 1.

result(0, "YES").
result(1, "NO").

:-
    read_input(X1, Y1, X2, Y2),
    A is (X1 + Y1 + X2 + Y2) mod 2,
    result(A, Answer),
    write(Answer), nl, halt.
```

Вывод программы

```
a1b1
NO

a1b2
YES

a1e4
NO

e4e5
NO

e5h7
NO
```

Задача 2. Пузырьковая сортировка

Условие задачи

На вход подаётся массив целых чисел. Нужно пошагово отсортировать его алгоритмом bubble-sort.

Текст программы

```
:- prompt(_, '').

bubble([], [], 0).
bubble([X], [X], 0).
bubble([X, Y | Tail], [Y | Local], 1) :- Y < X, bubble([X | Tail], Local, _).
bubble([X, Y | Tail], [X | Local], Z) :- X =< Y, bubble([Y | Tail], Local, Z).

elements_write([]) :- !.
elements_write([Head | Tail]) :- write(Head), write(' '), elements_write(Tail).

sort(L) :- bubble(L, _, T), T =:= 0, !.
sort(L) :- bubble(L, Local, T), T =:= 1, elements_write(Local), nl, sort(Local).

spaces(" \n\t").

:-
    spaces(S),
    read_string(user_input, _, Input),
    split_string(Input, S, S, Words),
    findall(N, (member(W, Words), number_string(N, W)), Numbers),
    sort(Numbers),
    halt.
```

Вывод программы

```
3 2 4 5 1
2 3 4 1 5
2 3 1 4 5
2 1 3 4 5
1 2 3 4 5

5 4 3 2 1
4 3 2 1 5
3 2 1 4 5
2 1 3 4 5
1 2 3 4 5

34 9 -1 100500 43 42 573 5 6 7 4 3 1
```

9 -1 34 43 42 573 5 6 7 4 3 1 100500
-1 9 34 42 43 5 6 7 4 3 1 573 100500
-1 9 34 42 5 6 7 4 3 1 43 573 100500
-1 9 34 5 6 7 4 3 1 42 43 573 100500
-1 9 5 6 7 4 3 1 34 42 43 573 100500
-1 5 6 7 4 3 1 9 34 42 43 573 100500
-1 5 6 4 3 1 7 9 34 42 43 573 100500
-1 5 4 3 1 6 7 9 34 42 43 573 100500
-1 4 3 1 5 6 7 9 34 42 43 573 100500
-1 3 1 4 5 6 7 9 34 42 43 573 100500
-1 1 3 4 5 6 7 9 34 42 43 573 100500

Задача 3. Печать квадратов из звёздочек

Условие задачи

На вход подаётся два натуральных числа m и n. Требуется напечатать в консоль *n* «полых» квадратов длины *m*, составленных из звёздочек, разделённых пробелами.

Текст программы

```
:- prompt(_, '').

spaces(" \n\t").

% repeat_string(S, -1, "") :- !.
repeat_string(_, 0, "") :- !.
repeat_string(S, N, Result) :- N1 is N-1, repeat_string(S, N1, Local), string_concat(S, Local, Result).

write_string_n_times(_, 0) :- !.
write_string_n_times(S, N) :-
    N1 is N-1,
    writeln(S),
    write_string_n_times(S, N1).

:-
    spaces(S),
    read_string(user_input, _, Input),
    split_string(Input, S, S, Words),
    findall(N, (member(W, Words), number_string(N, W)), Numbers),
    [N, K] = Numbers,
    repeat_string("*", N, Local),
    string_concat(Local, " ", Local2),
    repeat_string(Local2, K, Border),
```

```
    (N > 1 ->
        N2 is N-2,
        repeat_string(" ", N2, Spaces),
        string_concat("*", Spaces, Local3),
        string_concat(Local3, "* ", Local4),
        repeat_string(Local4, K, Inner),

        writeln(Border),
        write_string_n_times(Inner, N2),
        writeln(Border);
        repeat_string("* ", K, Result),
        writeln(Result)
    ),
    halt.
```

Вывод программы

```
1 1
*

2 3
** ** **
** ** **

3 2
*** ***
*  *  *
*** ***

5 1
*****
*   *
*   *
*   *
*****

1 5
* * * * *

7 7
******* ******* ******* ******* ******* ******* *******
*       * *       * *       * *       * *       * *    *
*       * *       * *       * *       * *       * *    *
*       * *       * *       * *       * *       * *    *
```

```
*       * *       * *       * *       * *       * *       * *    *
*       * *       * *       * *       * *       * *       * *    *
******* ******* ******* ******* ******* ******* *******
```

Заключение

Prolog — мощный язык программирования, который предоставляет уникальные возможности для работы с логическими задачами. Его декларативный подход, рекурсия и унификация делают его идеальным инструментом для решения сложных проблем в области искусственного интеллекта и обработки данных.