МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский технологический университет»
МИРЭА

Подлежит возврату
№ 0012

ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ ПРОЛОГ

ЧАСТЬ 3

Методические указания
по выполнению лабораторных работ
для студентов, обучающихся по направлениям подготовки
09.03.04 «Программная инженерия»
Профиль «Системная и программная инженерия»,
09.03.01 «Информатика и вычислительная техника»
Профиль «Математическое обеспечение вычислительной техники и автоматизированных систем»

МОСКВА 2015

Составители: В.А. Смольянинова

В методических указаниях излагаются рекомендации и методический материал для выполнения лабораторных работ по дисциплинам «Функциональное и логическое программирование», «Системно-программные основы искусственного интеллекта», «Системы искусственного интеллекта» (в зависимости от учебного плана).

Материал предназначен для студентов дневного, вечернего и заочного отделений и может быть использован для самостоятельной работы.

ЛАБОРАТОРНАЯ РАБОТА 3

ОПЕРАЦИИ НАД СПИСКАМИ. АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ

Цель работы: Изучение представления и обработки списков

Подготовка к выполнению работы

1. Повторите разделы курса:
   - представление списков;
   - операции над списками;
   - арифметика в Прологе.

2. Изучите следующие вопросы:
   - представление списков в Прологе;
   - отношение принадлежности к списку;
   - конкатенация списков;
   - добавление элемента в список;
   - удаление элемента из списка.

3. Выполните задания к каждому примеру и в соответствии с вариантом задания и собственной предметной областью, напишите текст программы, определяющей требуемое отношение над списками.

Содержание отчета

1. Программа.
2. Цели-вопросы
3. Результаты выполнения программы.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

1.Представление списков

Список - последовательность из произвольного числа элементов. Список является основной структурой данных в Прологе. Элементы списка разделяются запятыми и заключаются в квадратные скобки. Любой список представляет собой:

- либо пустой список (атом []);

- либо непустой список - структуру, состоящую из двух частей:

*- первый элемент - **голова** (Head) списка;*

*- второй элемент - **хвост** (Tail) списка*.

В общем случае голова списка может быть любым объектом языка Пролог, а хвост - обязательно должен быть списком. Поскольку хвост - список, то он либо пуст, либо имеет свои собственные голову и хвост.

Для повышения наглядности программ в Прологе предусматриваются специальные средства для списковой нотации, позволяющие представлять списки в виде

[Элемент1, Элемент2,...]

или

[Голова | Хвост]

или

[Элемент1, Элемент2,... | Остальные].

Здесь знак | используется для отделения начала списка от конца.

2.Операции над списками

2.1.Принадлежность к списку (member)

Отношение принадлежности записывается в виде двух предложений:

```
member(X,[X|Tail]).
member(X,[_|Tail]) :-
```

```
member(X,Tail).
```

Оно основано на следующих соображениях:

- либо X - голова списка,
- либо X - принадлежит хвосту этого списка.

2.2.Сцепление (конкатенация) списков (conc)

Обозначается через *conc(L1,L2,L3*).

Здесь L1 и L2 - два списка, L3- список, получаемый при их сцеплении.

Определение отношения conc содержит два случая:

- если первый аргумент - пуст, то второй и третий аргументы представляют собой один и тот же список:

```
conc([],L,L).
```

- если первый аргумент отношения conc не пуст, то он имеет голову и хвост - [X|L1]. Результат сцепления - список [X|L3], где L3 - получен после сцепления списков L1 и L2:

```
conc([X|L1],L2,[X|L3]) :-
                conc(L1,L2,L3).
```

Примеры сцепления заданных списков.

```
Goal: conc([a,b,c],[x,y,z],L).
   L = [a,b,c,x,y,z]

Goal: conc([a,[b,c],d],[a,[],b],L).
   L = [a,[b,c],d,a,[],b]
```

Программу для conc можно применить "в обратном направлении" - для разбиения списка на две части.Например:

```
Goal: conc(L1,L2,[a,b,c]).

  L1=[]
  L2=[a,b,c]

  L1=[a]
  L2=[b,c]
```

```
L1=[a,b]
L2=[c]

L1=[a,b,c]
L2=[]
```

Используя conc можно определить отношение принадлежности следующим эквивалентным способом:

```
member(X,L) :-
      conc(_,[X|_],L).
```

Здесь символом "_" обозначены анонимные переменные (переменные, встречающиеся в предложении только по одному разу).

2.3.Добавление элемента (append)

Наиболее простой способ добавления элемента в список - вставить его в начало так, чтобы он стал его новой головой. Процедура добавления определяется в форме факта

```
append(X,L,[X|L]).
```

2.4. Удаление элемента (remove)

Имеем два случая:
- если X - голова списка, то результат удаления - хвост списка;
- если X - в хвосте списка, то его нужно удалить оттуда.

В результате отношение remove определяется так:

```
remove(X,[X|Tail],Tail).

remove(X,[Y|Tail],[Y|Tail1]) :-
          remove(X,Tail,Tail1).
```

Если в списке встречается несколько вхождений элемента X, то remove сможет исключить их все при помощи возвратов.

3.Арифметические действия

Турбо-Пролог располагает двумя числовыми типами доменов: целыми и действительными числами. Четыре основные арифметические операции - это сложение, вычитание, умножение и деление. Для их реализации в Турбо- Прологе используются предикаты.

ПРИМЕРЫ ПРОГРАММ

Пример 1. Поиск нужного элемента в списке

```
domains
   i=integer
   s=symbol
   i_list=i*  /* определение типа "список целых
   чисел" */
   s_list=s*  /* определение типа "список атомов"
   */

predicates
   /* предикат member определяется над списками
   двух типов */
   member(i,i_list)
   member(s,s_list)

clauses
   member(Head,[Head|_]).
   member(Head,[_|Tail]):-
         member(Head,Tail).
```

Пример 2. Определение суммы элементов списка

```
domains
   i=integer

   i_list=i*

predicates
   sum_list(i_list,i)

clauses
```

```
   /* Если список пуст, то сумма его элементов
   равна нулю */
   sum_list([],0).
   /* Иначе найти сумму элементов хвоста списка
         и прибавить к ним голову */
   sum_list([H|T],Sum):-
              sum_list(T,Sum1),
              Sum=H+Sum1.
```

Задайте 4 цели-вопроса к задаче.

Пример 3. Реализация арифметики

```
domains
   i=integer
   r=real
predicates
   add(i,i,i)
   sub(i,i,i)
   mul(i,i,i)
   div(i,i,i)
   fadd(r,r,r)
   fsub(r,r,r)
   fmul(r,r,r)
   fdiv(r,r,r)

clauses

   add(X,Y,Z) :-
         Z=X+Y.
         sub(X,Y,Z):-
         Z=X-Y.
         mul(X,Y,Z):-
         Z=X*Y.
         div(X,Y,Z):-

         Z=X/Y.

          fadd(X,Y,Z) :-
         Z=X+Y.
```

```
fsub(X,Y,Z):-
Z=X-Y.
 fmul(X,Y,Z):-
Z=X*Y.
fdiv(X,Y,Z):-
Z=X/Y.
```

Задайте несколько целей-вопросов. Используйте также операции сравнения, такие как = , < , > , <= , >= , <> .

ВАРИАНТЫ ЗАДАНИЙ

1. Определение максимального элемента в списке.
2. Определение числа элементов в списке.
3. Определение произведения элементов списка.
4. Исключение из списка отрицательных элементов.
5. Сортировка элементов списка по возрастанию.
6. Даны два списка, имеющие ненулевое пересечение. Построить список, включающий все элементы указанных двух списков без повторений.
7. Определить отношение

   Обращение(Список, Обращенный список),которое располагает элементы списка в обратном порядке.

8. Определить отношение

   перевод(Список1, Список2) для перевода списка чисел от 0 до 9 в список соответствующих слов.

9. Определить отношение

   разбиение_списка(Список, Список1, Список2)

так, чтобы оно распределяло элементы списка между двумя списками Список1 и Список2, длины которых отличаются друг от друга не более чем на единицу.

10. Определить отношение

пересечение(Список1, Список2, Список3),

где элементы списка Список3 являются общими для списков Список1 и Список2.

11. Определить отношение

разность(Список1, Список2, Список3),

где элементы списка Список3 принадлежат Списку1, но не принадлежат Списку2.

12. Определить отношение

element_mult(List1,List2,List3),

в котором элементы списка List3 равны произведениям соответствующих элементов списков List1 и List.

13. Используя отношение conc, напишите цель, соответствующую вычеркиванию трех последних элементов списка L. Результат - новый список L1. Указание : L - сцепление L1 и трех элементного списка.

14. Напишите последовательность целей для порождения списка L2, получающегося из списка L вычеркиванием его трех первых и трех последних элементов.