



Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное  
учреждение  
высшего образования  
«Московский государственный технический университет  
имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)

---

ФАКУЛЬТЕТ «Информатика и системы управления»

---

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

---

**Лабораторная работа № 20**

Дисциплина Функциональное и логическое программирование

Тема Рекурсия на Prolog

Студент Ильясов И. М.

Группа ИУ7-63Б

Оценка (баллы) \_\_\_\_\_

Преподаватель Толпинская Н. Б., Строганов Ю. В.

Москва, 2020 г.

**Цель работы** – изучить рекурсивные способы организации программ на Prolog, методы формирования эффективных рекурсивных программ и порядок реализации таких программ.

**Задачи работы:**

- приобрести навыки эффективного декларативного описания предметной области с использованием фактов и правил;
- изучить порядок использования фактов и правил в программе на Prolog, принципы и особенности сопоставления и отождествления термов, на основе механизма унификации. Способ формирования и изменения резольвенты. Порядок формирования ответа.

**Задание лабораторной работы**

Используя хвостовую рекурсию, разработать, комментируя аргументы, эффективную программу, позволяющую:

1. Сформировать список из элементов числового списка, больших заданного значения;
2. Сформировать список из элементов, стоящих на нечетных позициях исходного списка (нумерация от 0);
3. Удалить заданный элемент из списка (один или все вхождения);
4. Преобразовать список в множество (можно использовать ранее разработанные процедуры).

Убедиться в правильности результатов

Для одного из вариантов **ВОПРОСА** и **1-ого задания** составить таблицу, отражающую конкретный порядок работы системы:

Т.к. резольвента хранится в виде стека, то состояние резольвенты требуется отображать в столбик: вершина – сверху! Новый шаг надо начинать с нового состояния резольвенты! Для каждого запуска алгоритма унификации, требуется указать № выбранного правила и соответствующий вывод: успех или нет –и почему.

**Вопрос:...**

№ шага	Состояние резольвенты, и вывод: дальнейшие действия (почему?)	Для каких термов запускается алгоритм унификации: $T1=T2$ и каков <b>результат</b> (и подстановка)	Дальнейшие действия: прямой ход или откат (почему и к чему приводит?)
1 ...	...	...	Комментарий, вывод...
...	...	...	...

## Текст программы

```
domains
    list = integer*.
    num = integer.
    index = integer.

predicates
    get_more_than(list, num, list).
    get_odd(list, list).
    get_odd(list, list, index).
    remove(list, num, list).
    remove_all(list, num, list).
    set(list, list).

clauses
    get_more_than([], _, []) :- !.
    get_more_than([Head|Tail], Num, Res) :-
        Head > Num, get_more_than(Tail, Num, TailRes),
        Res = [Head|TailRes].
    get_more_than([Head|Tail], Num, Res) :-
        Head <= Num, get_more_than(Tail, Num, Res).

    get_odd(List, Res) :- get_odd(List, Res, 0).
    get_odd([], [], _) :- !.
    get_odd([_|Tail], Res, Index) :-
        Index mod 2 = 0, NextIndex = Index + 1,
        get_odd(Tail, Res, NextIndex).
    get_odd([Head|Tail], Res, Index) :-
        Index mod 2 = 1, NextIndex = Index + 1,
        get_odd(Tail, TailRes, NextIndex),
        Res = [Head|TailRes].

    remove([], _, []) :- !.
    remove([Head|Tail], Num, Res) :-
        Head = Num, Res = Tail, !.
    remove([Head|Tail], Num, Res) :-
        Head <> Num, remove(Tail, Num, TailRes),
        Res = [Head|TailRes].

    remove_all([], _, []) :- !.
    remove_all([Head|Tail], Num, Res) :-
        Head = Num, remove_all(Tail, Num, Res).
    remove_all([Head|Tail], Num, Res) :-
        Head <> Num, remove_all(Tail, Num, TailRes),
        Res = [Head|TailRes].

    set([], []) :- !.
    set([Head|Tail], Set) :-
        remove_all(Tail, Head, NextTail),
        set(NextTail, TailSet),
        Set = [Head|TailSet].

goal
    get_more_than([1, 5, 2, 4, 6, 8, 9, 8], 4, ResultMore);
    get_odd([3, 5, 2, 4, 6], ResultOdd);
    remove([1, 3, 2, 5, 4], 3, ResultFirst);
    remove_all([1, 4, 2, 2, 4], 4, ResultAll);
    set([1, 3, 5, 2, 1, 2, 5, 6, 6, 7, 8, 9, 2], Set).
```

## Примеры работы программы

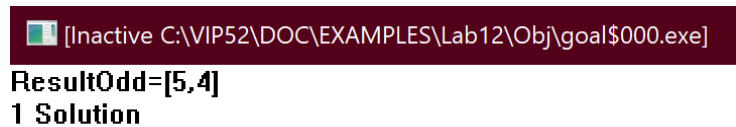
1. Список из элементов числового списка, больших заданного значений ([1, 5, 2, 4, 6, 8, 9, 8], больше, чем 4).



```
[Inactive C:\VIP52\DOC\EXAMPLES\Lab12\Obj\goal$000.exe]  
ResultMore=[5,6,8,9,8]  
1 Solution|
```

Рисунок 1

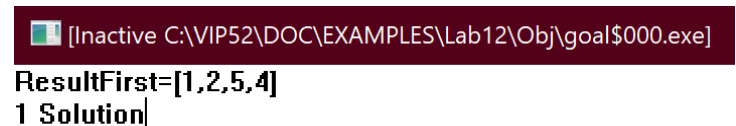
2. Список из элементов числового списка, стоящих на нечетных позициях исходного списка (нумерация от 0) – [3, 5, 2, 4, 6].



```
[Inactive C:\VIP52\DOC\EXAMPLES\Lab12\Obj\goal$000.exe]  
ResultOdd=[5,4]  
1 Solution
```

Рисунок 2

3. Удалить заданный элемент из списка (первое вхождение) – [1, 3, 2, 5, 4], удалить 3.



```
[Inactive C:\VIP52\DOC\EXAMPLES\Lab12\Obj\goal$000.exe]  
ResultFirst=[1,2,5,4]  
1 Solution|
```

Рисунок 3


4. Удалить заданный элемент из списка (все вхождения) – [1, 4, 2, 2, 4], удалить 4.



```
[Inactive C:\VIP52\DOC\EXAMPLES\Lab12\Obj\goal$000.exe]  
ResultAll=[1,2,2]  
1 Solution|
```

Рисунок 4

5. Преобразовать список в множество – [1, 3, 5, 2, 1, 2, 5, 6, 6, 7, 8, 9, 2].



```
[Inactive C:\VIP52\DOC\EXAMPLES\Lab12\Obj\goal$000.exe]  
Set=[1,3,5,2,6,7,8,9]  
1 Solution|
```

Рисунок 5

## Задание с таблицей

get\_more\_than([1, 5, 2, 4], 3, ResultMore).

№ шага	Состояние резольвенты, и вывод: дальнейшие действия (почему?)	Для каких термов запускается алгоритм унификации: T1=T2 и каков <b>результат</b> (и подстановка)	Дальнейшие действия: прямой ход или откат (почему и к чему приводит?)
1	get_more_than([1, 5, 2, 4], 3, ResultMore).	Успех, подстановка {Head = 1, Tail = [5, 2, 4], Num = 3, Res = ResultMore} get_more_than([1, 5, 2, 4], 3, ResultMore) get_more_than([Head Tail], Num, Res)	Прямой ход к следующему предложению.
2	Head > Num get_more_than(Tail, Num, TailRes) Res = [Head TailRes]	Сравнение 1 > 3, неудача	Обратный ход
3	get_more_than([1, 5, 2, 4], 3, ResultMore).	Успех, подстановка {Head = 1, Tail = [5, 2, 4], Num = 3, Res = ResultMore} get_more_than([1, 5, 2, 4], 3, ResultMore) get_more_than([Head Tail], Num, Res)	Прямой ход к следующему предложению.
4	Head <= Num get_more_than(Tail, Num, Res)	Сравнение 1 <= 3, успех	Прямой ход к следующему предложению.
5	get_more_than(Tail, Num, Res)	Успех, подстановка {Head = 5, Tail = [2, 4], Num = 3, Res = Res} get_more_than([5, 2, 4], 3, ResultMore) get_more_than([Head Tail], Num, Res)	Прямой ход к следующему предложению.
6	Head > Num get_more_than(Tail, Num, TailRes) Res = [Head TailRes]	Сравнение 5 > 3, успех	Прямой ход к следующему предложению.
7	get_more_than(Tail, Num, TailRes) Res = [Head TailRes]	Успех, подстановка {Head = 2, Tail = [4], Res = TailRes, Num = 3}	Прямой ход к следующему предложению.
8	Head > Num get_more_than(Tail, Num, TailRes) Res = [Head TailRes]	Сравнение 4 > 3, успех	Прямой ход к следующему предложению.

	Res = [Head TailRes]		
9	get_more_than(Tail, Num, TailRes) Res = [Head TailRes] Res = [Head TailRes]	Успех, подстановка {Head = 4, Tail = [], Res = TailRes, Num = 3}	Прямой ход к следующему предложению.
10	! Res = [Head TailRes] Res = [Head TailRes]	Отсечение	Прямой ход к следующему предложению.
11	Res = [Head TailRes] Res = [Head TailRes]	Успех, подстановка {Res = [4]}	Прямой ход к следующему предложению.
12	Res = [Head TailRes]	Успех, подстановка {Res = [5, 4]}	
13		<b>Результат:</b> MoreResult = [5, 4]	Обратный ход
14	get_more_than(Tail, Num, Res)	Успех, подстановка {Tail = [4], Res = Res} get_more_than([4], 3, ResultMore) get_more_than([Head Tail], Num, Res)	Прямой ход к следующему предложению.
15	Head <= Num get_more_than(Tail, Num, Res)	4 <= 3	Обратный ход
16	get_more_than(Tail, Num, Res)	Успех, подстановка {Tail = [5, 2, 4], Res = Res} get_more_than([5, 2, 4], 3, ResultMore) get_more_than([Head Tail], Num, Res)	Прямой ход к следующему предложению
17	Head <= Num get_more_than(Tail, Num, Res)	5 <= 3	Обратный ход
	...		

## Ответы на вопросы

### 1) Как организуется хвостовая рекурсия в Prolog?

Рекурсия – это ссылка на определяемый объект во время его определения. В Prolog рекурсия организуется правилом, в котором есть обращение к тому же правилу.

### 2) Какое первое состояние резольвенты?

Если задан простой вопрос, то сначала он попадает в резольвенту. То есть первое состояние – вопрос.

**3) Каким способом можно разделить список на части, какие, требования к частям?**

В Prolog для деления списка на части используется специальный символ «|».

Разделение происходит на голову и хвост.

**4) Как выделить за один шаг первые два подряд идущих элемента списка? Как выделить 1-й и 3-й элемент за один шаг?**

$[A, B|_]$  – выделение двух подряд идущих элементов списка.

$[A, _, C|_]$  – выделение первого и третьего элементов списка.

**5) Как формируется новое состояние резольвенты?**

Преобразование резольвенты происходит с помощью редукции (замена цели телом того правила, заголовок которого унифицируется с вопросом). Новое состояние резольвенты формируется в два шага:

- В текущем состоянии резольвенты одна из целей выбирается и для нее выполняется редукция.
- К полученному новому состоянию резольвенты применяется подстановка.

**6) Когда останавливается работа системы? Как это определяется на формальном уровне?**

Работа системы останавливается, если резольвента пуста.