|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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** \_Формирование и модификация списков 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: Т1=Т2 и каков **результат** (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).



Рисунок 1

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

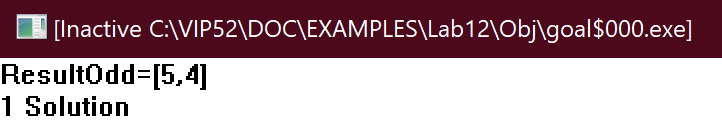


Рисунок 2

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



Рисунок 3

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



Рисунок 4

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

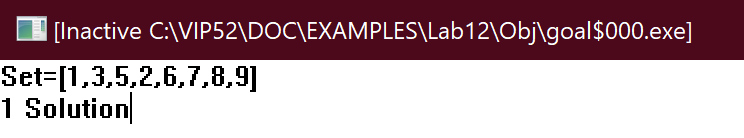


Рисунок 5

**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: Т1=Т2 и каков **результат** (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]  Res = [Head|TailRes] | Сравнение 4 > 3, успех | Прямой ход к следующему предложению. |
| 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 |  | **Результат:** 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.

1. **Какое первое состояние резольвенты?**

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

1. **Каким способом можно разделить список на части, какие, требования к частям?**

В Prolog для разделения списка на части используется специальный символ «|». Разделение происходит на голову и хвост.

1. **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.

1. **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.

1. **Когда останавливается работа системы? Как это определяется на формальном уровне?**

Работа системы останавливается, если резольвента пуста.