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»

Отчет
по лабораторной работе № 10

Дисциплина: Функциональное и логическое программирование

Студент: Платонова Ольга

Группа: ИУ7-65Б

Преподаватели: Толпинская Н. Б.
Строганов Ю. В.

Москва, 2021 г.

Задание

Используя хвостовую рекурсию, разработать эффективную программу, позволяющую:

1. Сформировать список из элементов числового списка, больших заданного значения;
2. Сформировать список из элементов, стоящих на нечетных позициях исходного списка;
3. Удалить заданный элемент из списка;
4. Преобразовать список в множество.

Листинг

```
domains
        list = integer*.

predicates
        formBigger(list, integer, list).

clauses
        formBigger([], _, []) :- !.
        formBigger([H | T], N, [H | Ta]) :- H > N, formBigger(T, N, Ta).
        formBigger([H | T], N, Lst) :- H <= N, formBigger(T, N, Lst).

goal
        formBigger([1, 2, 3, 1, 5, 1], 2, Res).
```

```
[Inactive C:\VIP52\DOC\EXAMPLES\TestGoal\Obj\goal$000.exe]
Res=[3,5]
1 Solution
```

```
domains
        list = integer*.

predicates
        formOdd(list, list).

clauses
        formOdd([], []) :- !.
        formOdd([_ | []], []) :- !.
        formOdd([_, H | T], [H | Ta]) :- formOdd(T, Ta).

goal
        formOdd([1, 2, 3, 4, 5, 6], Res).
```

```
[Inactive C:\VIP52\DOC\EXAMPLES\TestGoal\Obj\goal$000.exe]
Res=[2,4,6]
1 Solution
```

```
domains
        list = integer*.

predicates
        deleteElem(list, integer, list).

clauses
        deleteElem([], _, []) :- !.
        deleteElem([Elem | T], Elem, Res) :- deleteElem(T, Elem, Res), !.
        deleteElem([H | T], Elem, [H | Ta]) :- deleteElem(T, Elem, Ta).

goal
        deleteElem([1, 2, 3, 4, 3], 3, Res).
```

```
[Inactive C:\VIP52\DOC\EXAMPLES\TestGoal\Obj\goal$000.exe]
Res=[1,2,4]
1 Solution
```

```
domains
        list = integer*.

predicates
        deleteElem(list, integer, list).
        formSet(list, list).

clauses
        deleteElem([], _, []) :- !.
        deleteElem([Elem | T], Elem, Res) :- deleteElem(T, Elem, Res), !.
        deleteElem([H | T], Elem, [H | Ta]) :- deleteElem(T, Elem, Ta).

        formSet([], []) :- !.
        formSet([H | T], [H | Ta]) :- deleteElem(T, H, Res), formSet(Res, Ta).

goal
        %deleteElem([1, 2, 3, 4, 3], 3, Res).
        formSet([1, 2, 3, 2, 1, 3, 4], Res).
```

```
[Inactive C:\VIP52\DOC\EXAMPLES\TestGoal\Obj\goal$000.exe]
Res=[1,2,3,4]
1 Solution
```

1. formBigger([1, 2, 3, 4], 2, Res).

№ шага	Состояние резольвенты и вывод	Для каких термов запускается алгоритм унификации и каков результат	Дальнейшие действия
0	formBigger([1, 2, 3, 4], 2, Res).		
1	formBigger([1, 2, 3, 4], 2, Res).	T1 = formBigger([1, 2, 3, 4], 2, Res). T2 = formBigger([], _, []). Неудача. Не унифицируемые.	Переход к следующему заголовку БЗ

2	formBigger([1, 2, 3, 4], 2, Res).	T1 = formBigger([1, 2, 3, 4], 2, Res). T2 = formBigger([H | T], N, [H | Ta]). Успех. Унифицируемые. Подстановка: {[H | T] = [1, 2, 3, 4], N = 2, [H | Ta] = Res}	Удаляется из стека: formBigger([1, 2, 3, 4], 2, Res). Связываются переменные: H = 1, T = [2, 3, 4], N = 2, Res = [1|Ta]
3	1 > 2, formBigger([2, 3, 4], 2, Ta).	1 > 2. Неверно.	Удаляется из стека: 1 > 2, formBigger([2, 3, 4], 2, Ta). Развязываются переменные: H, T, N, Res Откат.
4	formBigger([1, 2, 3, 4], 2, Res).	T1 = formBigger([1, 2, 3, 4], 2, Res). T2 = formBigger([H | T], N, Lst). Успех. Унифицируемые. Подстановка: {[H | T] = [1, 2, 3, 4], N = 2, Lst = Res}	Удаляется из стека: formBigger([1, 2, 3, 4], 2, Res). Связываются переменные: H = 1, T = [2, 3, 4], N = 2
5	1 <= 2, formBigger([2, 3, 4], 2, Lst).	1 <= 2. Верно.	Удаляется из стека: 1 <= 2.
6	formBigger([2, 3, 4], 2, Lst).	T1 = formBigger([2, 3, 4], 2, Lst). T2 = formBigger([], _, []). Неудача. Не унифицируемые.	Переход к следующему заголовку БЗ
7	formBigger([2, 3, 4], 2, Lst).	T1 = formBigger([2, 3, 4], 2, Lst). T2 = formBigger([H | T], N, [H | Ta]). Успех. Унифицируемые. Подстановка: {[H | T] = [2, 3, 4], N = 2, [H | Ta] = Lst}	Удаляется из стека: formBigger([2, 3, 4], 2, Lst). Связываются переменные: H1 = 2, T1 = [3, 4], N1 = 2, Lst = [2|Ta]
8	2 > 2, formBigger([3, 4], 2, Ta).	2 > 2. Неверно.	Удаляется из стека: 2 > 2, formBigger([3, 4], 2, Ta). Развязываются переменные: H1, T1, N1, Lst Откат.
9	formBigger([2, 3, 4], 2, Lst).	T1 = formBigger([2, 3, 4], 2, Lst). T2 = formBigger([H | T], N, Lst). Успех. Унифицируемые.	Удаляется из стека: formBigger([2, 3, 4], 2, Lst). Связываются переменные: H1 = 2, T1 = [3, 4], N1 = 2, Lst = Lst1

		Подстановка: {[H | T] = [2, 3, 4], N = 2, Lst = Lst}	
10	2 <= 2, formBigger([3, 4], 2, Lst1).	2 <= 2. Верно.	Удаляется из стека: 2 <= 2.
11	formBigger([3, 4], 2, Lst1).	T1 = formBigger([3, 4], 2, Lst1). T2 = formBigger([], _, []). Неудача. Не унифицируемые.	Переход к следующему заголовку БЗ
12	formBigger([3, 4], 2, Lst1).	T1 = formBigger([3, 4], 2, Lst1). T2 = formBigger([H | T], N, [H | Ta]). Успех. Унифицируемые. Подстановка: {[H | T] = [3, 4], N = 2, [H | Ta] = Lst1}	Удаляется из стека: formBigger([3, 4], 2, Lst1). Связываются переменные: H2 = 3, T2 = [4], N2 = 2, Lst1 = [3|Ta]
13	3 > 2, formBigger([4], 2, Ta).	3 > 2. Верно.	Удаляется из стека: 3 > 2.
14	formBigger([4], 2, Ta).	T1 = formBigger([4], 2, Ta). T2 = formBigger([], _, []). Неудача. Не унифицируемые.	Переход к следующему заголовку БЗ
15	formBigger([4], 2, Ta).	T1 = formBigger([4], 2, Ta). T2 = formBigger([H | T], N, [H | Ta]). Успех. Унифицируемые. Подстановка: {[H | T] = [4], N = 2, [H | Ta] = Ta}	Удаляется из стека: formBigger([4], 2, Ta). Связываются переменные: H3 = 4, T3 = [], N3 = 2, Ta = [4 | Ta1]
16	4 > 2, formBigger([], 2, Ta1).	4 > 2. Верно.	Удаляется из стека: 4 > 2.
17	formBigger([], 2, Ta1).	T1 = formBigger([], 2, Ta1). T2 = formBigger([], _, []). Успех. Унифицируемые. Подстановка: {[] = [], _ = 2, [] = Ta1}	Удаляется из стека: formBigger([], 2, Ta1). Связываются переменные: Ta1 = []
18	!.	!. Истина.	Удаляется из стека: !.

19	Резольвента пуста.		Выводится Res = [3, 4] Развязываются переменные: Ta1 Откат.
20	!.	!. Завершение процедуры.	Удаляется из стека: !. Развязываются переменные: H3, T3, N3, Ta, H2, T2, N2, Lst1, H1, T1, N1, Lst, H, T, N, Res
21	Резольвента пуста.		Завершение работы программы.

Вывод

Эффективность работы системы может быть достигнута за счет хвостовой рекурсии и использования отсечения (уменьшения количества унификаций) в тех случаях, когда заведомо известна единственность ответа на вопрос.

1. *Как организуется хвостовая рекурсия в Prolog?*

   Для организации хвостовой рекурсии необходимо, чтобы рекурсивный вызов был последним в теле рекурсивного правила, и не оставалось других точек выбора.

2. *Какое первое состояние резольвенты?*

   Исходная резольвента содержит вопрос.

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

Список можно разделить на части в результате унификации со списком вида [H | T], где H – голова, T – хвост. Голова (начало) – несколько (не менее 1) элемента через запятую. Хвост (остаток) – один список.

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

   С помощью унификации со списком [A, B | T] и [A, B, C | T].

5. *Как формируется новое состояние резольвенты?*

   Резольвента меняется в 2 этапа:

   a. Редукция (замена вопроса на тело правила, заголовок которого был успешно унифицирован);
   b. Применение подстановки.

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

   Когда резольвента пуста и все указатели находятся в конце БЗ.