



КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Тема Рекурсия и обработка списков в языке Prolog

Студент Пересторонин П.Г.

Группа ИУ7-63Б

Оценка

Преподаватель Толпинская Н. Б.

Москва — 2021 г.

Оглавление

1	Лабораторная работа №18	2
2	Лабораторная работа №19	3
3	Лабораторная работа №20	5

1 Лабораторная работа №18

Задание: используя хвостовую рекурсию, разработать программу, позволяющую найти:

1. $n!$;
2. n -е число Фибоначчи.

Убедиться в правильности результатов.

Для одного из вариантов ВОПРОСА и каждого 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!

```
1 domains
2   num = integer
3
4 predicates
5   fact(num, num)
6   rfact(num, num, num)
7
8   fib(num, num)
9   rfib(num, num, num, num)
10
11 clauses
12   rfact(N, Res, Acc) :- N > 1, !, Nn = N - 1, Tacc = Acc * N, rfact(Nn, Res, Tacc).
13   rfact(_, Res, Acc) :- Res = Acc, !.
14   fact(N, Res) :- rfact(N, Res, 1), !.
15
16   rfib(N, A, B, Res) :- N > 2, !, Na = B, Nb = B + A, Nn = N - 1, rfib(Nn, Na, Nb, Res).
17   rfib(_, _, B, Res) :- Res = B, !.
18   fib(N, Res) :- rfib(N, 1, 1, Res), !.
19
20 goal
21   %fact(3, Res).
22   fib(6, Res).
```

Таблицы представлены на отдельных листах и приложены к отчету.

2 Лабораторная работа №19

Задание: используя хвостовую рекурсию, разработать эффективную программу (комментируя назначение аргументов), позволяющую:

1. Найти длину списка (по верхнему уровню);
2. Найти сумму элементов числового списка;
3. Найти сумму элементов числового списка, стоящих на нечетных позициях исходного списка (нумерация от 0);

Убедиться в правильности результатов.

Для одного из вариантов ВОПРОСА и одного из заданий 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 дальнейшие действия – и почему.

```
1 domains
2   elem = integer
3   intlist = elem*
4
5 predicates
6   rlength(integer, integer, intlist)
7   length(integer, intlist)
8
9   rsum(integer, integer, intlist)
10  sum(integer, intlist)
11
12  roddsum(integer, integer, intlist)
13  oddsum(integer, intlist)
14
15  append(intlist, intlist, intlist).
16
17 clauses
18  rlength(Res, Len, [_ | T]) :- Nlen = Len + 1, !, rlength(Res, Nlen, T).
19  rlength(Res, Len, []) :- Res = Len, !.
20  length(Res, List) :- rlength(Res, 0, List), !.
21
22  rsum(Res, Sum, [H | T]) :- Nsum = Sum + H, !, rsum(Res, Nsum, T).
23  rsum(Res, Sum, []) :- Res = Sum, !.
```

```

24 sum(Res, List) :- rsum(Res, 0, List), !.
25
26 roddsum(Res, Sum, [_ , H | T]) :- Nsum = Sum + H, !, roddsum(Res, Nsum, T).
27 roddsum(Res, Sum, []) :- Res = Sum, !.
28 oddsum(Res, List) :- roddsum(Res, 0, List), !.
29
30 append([], L2, L2) :- !.
31 append([H | T], L2, [H | T3]) :- append(T, L2, T3), !.
32
33 goal
34 %length(Res, [1, 2, 3, 4]).
35 %sum(Res, [1, 2, 3, 4]).
36 %oddsum(Res, [1, 2, 3, 4]).
37 append([1, 2, 3], [4, 5, 6], Res).

```

Таблицы представлены на отдельных листах и приложены к отчету.

3 Лабораторная работа №20

Задание: 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-го 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.

```
1 domains
2   elem = integer
3   intlist = elem*
4
5 predicates
6   only_odd(intlist, intlist)
7   only_more(intlist, integer, intlist)
8   delete_all(intlist, integer, intlist)
9   delete_one(intlist, integer, intlist)
10  to_set(intlist, intlist).
11
12 clauses
13   only_odd([_, H | T], [H | Res]) :- !, only_odd(T, Res).
14   only_odd([], []) :- !.
15
16   only_more([H | T], Num, [H | Res]) :- H > Num, !, only_more(T, Num, Res).
17   only_more([_ | T], Num, Res) :- only_more(T, Num, Res), !.
18   only_more([], _, []) :- !.
```

```

19
20 delete_all([H | T], Num, [H | Res]) :- H <> Num, !, delete_all(T, Num, Res).
21 delete_all([_ | T], Num, Res) :- delete_all(T, Num, Res), !.
22 delete_all([], _, []) :- !.
23
24 delete_one([H | T], Num, T) :- H = Num, !.
25 delete_one([H | T], Num, [H | Res]) :- delete_one(T, Num, Res), !.
26 delete_one([], _, []) :- !.
27
28 to_set([H | T], [H | Res]) :- delete_all(T, H, Nt), !, to_set(Nt, Res).
29 to_set([], []) :- !.
30
31 goal
32 %only_more([1, 2, 3, 4, 5, 6], 3, Result).
33 %only_odd([1, 2, 3, 4, 5, 6], Result).
34 %delete_all([1, 2, 3, 1, 2, 3, 1, 2, 3], 1, Result).
35 %delete_one([1, 2, 3, 1, 2, 3, 1, 2, 3], 1, Result).
36 to_set([1, 2, 3, 1, 2, 3, 1, 2, 3], Result).

```

Таблицы представлены на отдельных листах и приложены к отчету.

Дополнительное задание

Задание: преобразовать программу из 15 лабораторной работы (в базе данных содержится информация о собственности людей с использованием вариантных доменов) таким образом, чтобы можно было считать суммарную стоимость всех объектов собственности человека без ограничения на количество объектов конкретного типа (*не более 1 объекта конкретного типа собственности*).

```
1 domains
2   surname = string
3   city, street = string
4   house, flat = integer
5   phone = string
6   address = addr(city, street, house, flat)
7   mark = string
8   color = string
9   price = integer
10  bank = string
11  id, amount = integer
12  name = string
13  ind_property = building(name, price);
14    region(name, price);
15    water_transport(mark, color, price);
16    car(mark, color, price).
17  props = ind_property*
18
19 predicates
20  phone(surname, phone, address)
21  bank_depositor(surname, bank, id, amount)
22  owner(surname, ind_property)
23
24  get_price(ind_property, price)
25  money_amount(surname, price)
26  collect_properties(surname, props, props)
27  collect_properties(surname, props)
28  not_exist(ind_property, props)
29  sum_props(props, price)
30  sum_props(props, price, price)
31
32 clauses
33  phone("Perestoronin", "+79999999999", addr("Moscow", "Lesnaya", 12, 2)).
34  phone("Romanov", "+7111111111", addr("Moscow", "Lesnaya", 13, 87)).
35  phone("Nitenko", "+7333333333", addr("Ekaterinburg", "Kamennaya", 13, 87)).
36  phone("Yacuba", "+6666666666", addr("Moscow", "Wall-street", 123, 87)).
37
```



```

38 owner("Nitenko", car("bmw", "green", 1000)).
39 owner("Nitenko", region("empty_field", 1000)).
40 owner("Nitenko", building("Moscow_center", 1000)).
41 owner("Romanov", car("bmw", "green", 1000)).
42 owner("Romanov", region("rublevka", 10000)).
43 owner("Romanov", building("mini-village", 20000)).
44 owner("Romanov", water_transport("bmw", "red", 10000)).
45 owner("Yacuba", car("golfR", "black", 20000)).
46 owner("Yacuba", building("tiktok", 200000)).
47 owner("Perestoronin", car("mercedes", "yellow", 30000)).
48 owner("Perestoronin", building("tent", 10)).
49 owner("Sukocheva", car("Mercedes", "pink", 1111)).
50 owner("Sukocheva", car("BMW", "lightblue", 2222)).
51 owner("Sukocheva", car("Porsche", "black", 3333)).
52 owner("Sukocheva", region("rublevka", 2000)).
53 owner("Sukocheva", region("dacha_garden", 100)).
54 owner("Sukocheva", building("university", 200000)).
55 owner("Sukocheva", building("house", 110000)).
56 owner("Sukocheva", building("dacha", 100000)).
57 owner("Sukocheva", water_transport("Yachta", "white", 1000000)).
58
59 bank_depositor("Nitenko", "Sber", 22, 1000).
60 bank_depositor("Yacuba", "Sber", 33, 10000).
61 bank_depositor("Yacuba", "Alfa", 44, 20000).
62 bank_depositor("Romanov", "Sber", 238, 10).
63 bank_depositor("Perestoronin", "Maze", 1, 10000).
64
65 not_exist(H, [H | _]) :- !, 1 = 2.
66 not_exist(Prop, [_ | T]) :- not_exist(Prop, T).
67 not_exist(_, []).
68
69 collect_properties(Surname, Acc, Res) :- owner(Surname, Prop), not_exist(Prop, Acc),
    !, collect_properties(Surname, [Prop | Acc], Res).
70 collect_properties(_, Acc, Res) :- Res = Acc.
71 collect_properties(Surname, Res) :- collect_properties(Surname, [], Res).
72
73 get_price(building(_, Price), Price).
74 get_price(region(_, Price), Price).
75 get_price(water_transport(_, _, Price), Price).
76 get_price(car(_, _, Price), Price).
77
78 sum_props([Prop | T], Acc, Res) :- get_price(Prop, Price), Nacc = Acc + Price,
    sum_props(T, Nacc, Res).
79 sum_props([], Acc, Res) :- Res = Acc.
80 sum_props(Props, Res) :- sum_props(Props, 0, Res).
81
82 money_amount(Surname, Res) :- collect_properties(Surname, [], Props), sum_props(Props,
    Res).

```

```
83 goal
84   %collect_properties("Nitenko", [], Props).
85   money_amount("Sukocheva", Res). %1418766
```