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 лабораторным работам № 18-20 по курсу "Функциональное и логическое программирование"

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

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!

```
domains
  num = integer

predicates
  fact(num, num)
  rfact(num, num, num)

  fib(num, num)
  rfib(num, num, num, num)

clauses
  rfact(N, Res, Acc) :- N > 1, !, Nn = N - 1, Tacc = Acc * N, rfact(Nn, Res, Tacc).
  rfact(_, Res, Acc) :- Res = Acc, !.
  fact(N, Res) :- rfact(N, Res, 1), !.

  rfib(N, A, B, Res) :- N > 1, !, Na = B, Nb = B + A, Nn = N - 1, rfib(Nn, Na, Nb, Res).
  rfib(_, _, B, Res) :- Res = B, !.
  fib(N, Res) :- Nn = N - 1, rfib(Nn, 1, 1, Res), !.

goal
  %fact(3, Res).
  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.

```
domains
  elem = integer
  intlist = elem*

predicates
  rlength(integer, integer, intlist)
  length(integer, intlist)

  rsum(integer, integer, intlist)
  sum(integer, intlist)

  roddsum(integer, integer, intlist)
  oddsum(integer, intlist)

  append(intlist, intlist, intlist).

clauses
  rlength(Res, Len, [_ | T]) :- Nlen = Len + 1, rlength(Res, Nlen, T), !.
  rlength(Res, Len, []) :- Res = Len, !.
  length(Res, List) :- rlength(Res, 0, List), !.

  rsum(Res, Sum, [H | T]) :- Nsum = Sum + H, rsum(Res, Nsum, T), !.
  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.

```
domains
  elem = integer
  intlist = elem*

predicates
  only_more(intlist, integer, intlist)
  only_odd(intlist, intlist)
  delete_all(intlist, integer, intlist)
  delete_one(intlist, integer, intlist)
  to_set(intlist, intlist).

clauses
  only_more([H | T], Num, [H | Res]) :- H > Num, only_more(T, Num, Res), !.
  only_more([_ | T], Num, Res) :- only_more(T, Num, Res), !.
  only_more([], _, []), !.

  only_odd([_, H | T], [H | Res]) :- only_odd(T, Res), !.
  only_odd([], []), !.
```

```
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, Res).
33  %only_odd([1, 2, 3, 4, 5, 6], Res).
34  %delete_all([1, 2, 3, 1, 2, 3, 1, 2, 3], 1, Res).
35  %delete_one([1, 2, 3, 1, 2, 3, 1, 2, 3], 1, Res).
36  to_set([1, 2, 3, 1, 2, 3, 1, 2, 3], Res).
```

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*).

```
domains
  surname = string
  city, street = string
  house, flat = integer
  phone = string
  address = addr(city, street, house, flat)
  mark = string
  color = string
  price = integer
  bank = string
  id, amount = integer
  name = string
  ind_property = building(name, price);
    region(name, price);
    water_transport(mark, color, price);
    car(mark, color, price).
  props = ind_property*

predicates
  phone(surname, phone, address)
  bank_depositor(surname, bank, id, amount)
  owner(surname, ind_property)

  get_price(ind_property, price)
  money_amount(surname, price)
  collect_properties(surname, props, props)
  collect_properties(surname, props)
  not_exist(ind_property, props)
  sum_props(props, price)
  sum_props(props, price, price)

clauses
  phone("Perestoronin", "+79999999999", addr("Moscow", "Lesnaya", 12, 2)).
  phone("Romanov", "+71111111111", addr("Moscow", "Lesnaya", 13, 87)).
  phone("Nitenko", "+73333333333", addr("Ekaterinburg", "Kamennaya", 13, 87)).
  phone("Yacuba", "+66666666666", addr("Moscow", "Wall-street", 123, 87)).

```

```
owner("Nitenko", car("bmw", "green", 1000)).
owner("Nitenko", region("empty field", 1000)).
owner("Nitenko", building("Moscow center", 1000)).
owner("Romanov", car("bmw", "green", 1000)).
owner("Romanov", region("rublevka", 10000)).
owner("Romanov", building("mini-village", 20000)).
owner("Romanov", water_transport("bmw", "red", 10000)).
owner("Yacuba", car("golfR", "black", 20000)).
owner("Yacuba", building("tiktok", 200000)).
owner("Perestoronin", car("mercedes", "yellow", 30000)).
owner("Perestoronin", building("tent", 10)).
owner("Sukocheva", car("Mercedes", "pink", 1111)).
owner("Sukocheva", car("BMW", "lightblue", 2222)).
owner("Sukocheva", car("Porshe", "black", 3333)).
owner("Sukocheva", region("rublevka", 2000)).
owner("Sukocheva", region("dacha garden", 100)).
owner("Sukocheva", building("university", 200000)).
owner("Sukocheva", building("house", 110000)).
owner("Sukocheva", building("dacha", 100000)).
owner("Sukocheva", water_transport("Yachta", "white", 1000000)).

bank_depositor("Nitenko", "Sber", 22, 1000).
bank_depositor("Yacuba", "Sber", 33, 10000).
bank_depositor("Yacuba", "Alfa", 44, 20000).
bank_depositor("Romanov", "Sper", 238, 10).
bank_depositor("Perestoronin", "Maze", 1, 10000).

not_exist(H, [H | _]) :- !, 1 = 2.
not_exist(Prop, [_ | T]) :- not_exist(Prop, T).
not_exist(_, []).

collect_properties(Surname, Acc, Res) :- owner(Surname, Prop), not_exist(Prop, Acc),
    !, collect_properties(Surname, [Prop | Acc], Res).
collect_properties(_, Acc, Res) :- Res = Acc.
collect_properties(Surname, Res) :- collect_properties(Surname, [], Res).

get_price(building(_, Price), Price).
get_price(region(_, Price), Price).
get_price(water_transport(_, _, Price), Price).
get_price(car(_, _, Price), Price).

sum_props([Prop | T], Acc, Res) :- get_price(Prop, Price), Nacc = Acc + Price,
    sum_props(T, Nacc, Res).
sum_props([], Acc, Res) :- Res = Acc.
sum_props(Props, Res) :- sum_props(Props, 0, Res).

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
```