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 № 11 по курсу «Функциональное и логическое программирование»

Тема Рекурсия на Prolog

Студент Виноградов А. О.

Группа ИУ7-66Б

Оценка (баллы)

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

Москва — 2023 г.

1 Практические задания

Введение

Цель работы:изучить рекурсивные способы организации программ на Prolog, методы формирования эффективных рекурсивных программ обработки списков и порядок их реализации.

Задачи работы: приобрести навыки использования списков на Prolog, эффективного способа их обрвботки, организации и прядка работы соответствующих программ. Изучить особенность использования переменных при обработке списков. Способ формирования и изменения резольвенты в этом случае и порядок формирования ответа.

Постановка задачи

Используя хвостовую рекурсию, разработать (комментируя назначение аргументов) эффективную программу , 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);

4. Сформировать список из элементов числового списка, больших заданного значения;

5. Удалить заданный элемент из списка (один или все вхождения).

6. Объединить два списка.

Убедиться в правильности результатов. Для одного из вариантов ВОПРОСА уметь составить таблицу, отражающую конкретный порядок работы системы.

1.1 Код программы

Листинг 1.1 – Код программы

```
domains
    int = integer.
    list = int*.

predicates
    len(list, int)
    len(list, int, int)
    sum(list, int)
    sum(list, int, int)
    sum_odd(list, int)
    sum_odd(list, int, int)
    bigger_list(list, int, list)
    delete_elem_all(list, int, list)
    delete_elem_once(list, int, list)
    list_cnt(list,list,list)

clauses
    len(List, N):-
        len(List, N, 0).
    len([], N, N).
    len([_|T], N, I):-
        I1 = I + 1,
        len(T, N, I1).

    sum(List, N):-
        sum(List, N, 0).
    sum([], N, N).
    sum([H|T], N, Sum):-
        Tmp = Sum + H,
        sum(T, N, Tmp).

    sum_odd(List, N):-
        sum_odd(List, N, 0).
    sum_odd([], N, N).
    sum_odd([_|[]], N, N).
    sum_odd([_, Odd|T], N, Sum):-
        Tmp = Sum + Odd,
        sum_odd(T, N, Tmp).
```

Листинг 1.2 – Код программы (продолжение)

```
    bigger_list([], _,[]):-!.
    bigger_list([H|T], Num, Res):-
        H <= Num,
        bigger_list(T, Num, Res).
    bigger_list([H|T], Num, [H|Res]):-
        H > Num,
        bigger_list(T, Num, Res).

    delete_elem_all([], _,[]):-!.
    delete_elem_all([Num|T], Num, Res):- %equal
        !,delete_elem_all(T, Num, Res).
    delete_elem_all([H|T], Num, [H|Res]):- %not equal
        delete_elem_all(T, Num, Res).

    delete_elem_once([], _,[]):-!.
    delete_elem_once([Num|T], Num, T):-!. %equal
        %!,delete_elem_once(T, Num, Res).
    delete_elem_once([H|T], Num, [H|Res]):- %not equal
        delete_elem_once(T, Num, Res).

    list_cnt([], L, L):-!.
    list_cnt([H|T], L, [H|Result]):-
        list_cnt(T, L, Result).

goal
    %len([1,2,3], L).
    %sum([1,2,3,4,5], Sum).
    %sum_odd([0,1,2,3,4,5,6,7], Sum).
    %bigger_list([0,1,2,3,4,5,6,7], 3, List).
    %delete_elem_all([0,1,3,4,3,5], 3, List).
    %delete_elem_once([0,3,1,3,3], 3, List).
    list_cnt([1,2,3],[4,5,6], List).
```

№ шага	Состояние резольвенты, и вывод: дальнейшие действия (почему?)	Для каких термов запускается алгоритм унификации: T1=T2 и каков результат (и подстановка)	Дальнейшие действия: прямой ход или откат (почему и к чему приводит?)
1	len([1,2,3], L]	T1 = len([1,2,3],L) T2 = len(List, N) Попытка унификации. Унификация успешна. Подстановка: {List = [1,2,3], N = L}	Прямой ход
2	len([1,2,3],L, 0)	T1 = len([1,2,3],L, 0) T2 = len(List, N) Попытка унификации. Унификация не успешна.	Переход к следующему предложению
	len([1,2,3],L, 0)	T1 = len([1,2,3],L, 0) T2 = len([], N, N) Попытка унификации. Унификация не успешна.	Переход к следующему предложению
	len([1,2,3],L, 0)	T1 = len([1,2,3],L, 0) T2 = len([_|T], N, I) Попытка унификации. Унификация успешна. Подстановка: {T = [2,3], N = L, I = 0}	Прямой ход
3	I1 = 0 + 1 len([2,3], L, I1)	I1 = 1	Прямой ход
4	len([2,3], L,1)	T1 = len([2,3], L,1) T2 = len(List, N) Попытка унификации. Унификация не успешна.	Переход к следующему предложению
	len([2,3], L,1)	T1 = len([2,3], L,1) T2 = len([], N, N) Попытка унификации. Унификация не успешна.	Переход к следующему предложению
	len([2,3], L,1)	T1 = len([2,3], L,1) T2 = len([_|T], N, I) Попытка унификации. Унификация успешна. Подстановка: {T=3, N = L, I = 1}	Прямой ход
5	I1 = 1 + 1 len([3], L, I1)	I1 = 2	Прямой ход
6	len([3], L, 2)	T1 = len([3], L,2) T2 = len([_|T], N, I) Попытка унификации. Унификация успешна. Подстановка: {T = [], N = L, I = 2)	Переход к следующему предложению

7	I1 = 2 + 1 len([], L, I1)	I1 = 3	Прямой ход
8	len([], L, 3)	T1 = len([], L,3) T2 = len([], N, N) Попытка унификации. Унификация успешна. Подстановка: {N = 3, L = 3)	Найдено решение. Обратный ход
9	len([], L, 3)	T1 = len([], L,3) T2 = len([_|T], N, I) Унификация неуспешна. Закончен перебор возможных вариантов.	Завершение работы. Вывод результата.