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 № 19

Дисциплина Функциональное и логическое программирование

Тема _Обработка 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 способы организации, представления и обработки списков в программах на Prolog, методы создания эффективных рекурсивных программ обработки списков и порядок их реализации.

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 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!

Вопрос:…

№ 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…	…	…	Комментарий, вывод…
…	…	…	…

Текст программы

```
domains
	list = integer*.
	length = integer.
	sum = integer
	index = integer.

predicates
	list_length(list, length).
	list_sum(list, sum).
	list_sum_odd(list, sum).
	list_sum_odd(list, sum, index).

clauses
	list_length([], 0) :- !.
	list_length([_|Tail], Length) :-
		list_length(Tail, TailLength),
		Length = TailLength + 1.

	list_sum([], 0) :- !.
	list_sum([Head|Tail], Sum) :-
		list_sum(Tail, TailSum),
		Sum = TailSum + Head.

	list_sum_odd(List, Sum) :- list_sum_odd(List, Sum, 0).
	list_sum_odd([], 0, _) :- !.
	list_sum_odd([_|Tail], Sum, Index) :-
		Index mod 2 = 0,
		NextIndex = Index + 1,
		list_sum_odd(Tail, Sum, NextIndex).
	list_sum_odd([Head|Tail], Sum, Index) :-
		Index mod 2 = 1,
		NextIndex = Index + 1,
		list_sum_odd(Tail, TailSum, NextIndex),
		Sum = TailSum + Head.

goal
	list_length([6, 8, 2], Length);
	list_sum([1, 3, 5, 2, 4, 6], Sum);
	list_sum_odd([1, 2, 3, 4, 5], SumOdd).
```

Примеры работы программы

1. Поиск длины списка [6, 8, 2].

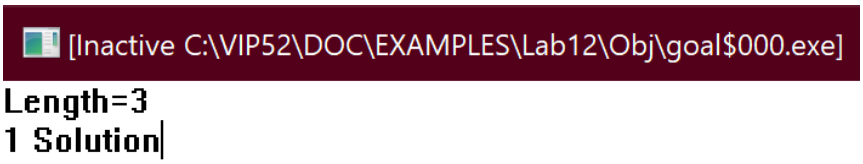

Рисунок 1 – длина списка [6, 8, 2]

2. Поиск суммы элементов списка [1, 3, 5, 2, 4, 6].

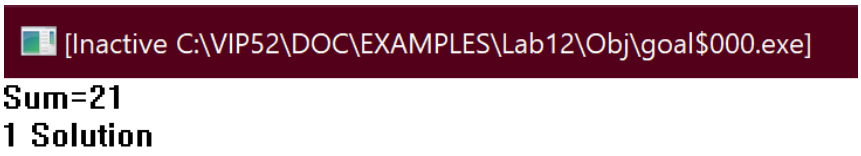

Рисунок 2 – сумма элементов списка [1, 3, 5, 2, 4, 6]

3. Поиск суммы элементов списка [1, 2, 3, 4, 5], стоящих на нечетных местах.

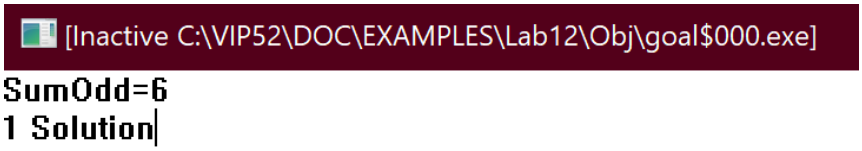

Рисунок 3 – сумма элементов списка [1, 2, 3, 4, 5], стоящих на нечетных местах

Задание с таблицей

```
list_sum([1, 3, 5, 2, 4, 6], Sum).
```

№ 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	list_sum([1, 3, 5, 2, 4, 6], Sum)	T1 = list_sum([1, 3, 5, 2, 4, 6], Sum). T2 = list_length([], 0). Неудача, разные функторы.	Прямой ход к следующему предложению.
2	list_sum([1, 3, 5, 2, 4, 6], Sum)	T1 = list_sum([1, 3, 5, 2, 4, 6], Sum). T2 = list_length([_|Tail], Length). Неудача, разные функторы.	Прямой ход к следующему предложению.
3	list_sum([1, 3, 5, 2, 4, 6], Sum)	T1 = list_sum([1, 3, 5, 2, 4, 6], Sum). T2 = list_sum([], 0). Неудача, разные функторы.	Прямой ход к следующему предложению.
4	list_sum([1, 3, 5, 2, 4, 6], Sum)	T1 = list_sum([1, 3, 5, 2, 4, 6], Sum). T2 = list_sum([Head|Tail], Sum). Успех, подстановка {Sum = Sum,	Прямой ход к следующему предложению.

		Head = 1, Tail = [3, 5, 2, 4, 6]}. Результат list_sum([1, 3, 5, 2, 4, 6], Sum), list_sum([Head|Tail], Sum).	
4	list_sum(Tail, TailSum), Sum = TailSum + Head.	Подстановка {Head = 3, Tail = [5, 2, 4, 6], Sum = TailSum} list_sum([5, 2, 4, 6], TailSum)	Прямой ход к следующему предложению.
5	list_sum(Tail, TailSum), Sum = TailSum + Head, Sum = TailSum + Head	Подстановка {Head = 5, Tail = [2, 4, 6], Sum = TailSum} list_sum([2, 4, 6], TailSum)	Прямой ход к следующему предложению.
6	list_sum(Tail, TailSum), Sum = TailSum + Head, Sum = TailSum + Head, Sum = TailSum + Head	Подстановка {Head = 2, Tail = [4, 6], Sum = TailSum} list_sum([4, 6], TailSum)	Прямой ход к следующему предложению.
7	list_sum(Tail, TailSum), Sum = TailSum + Head, Sum = TailSum + Head, Sum = TailSum + Head, Sum = TailSum + Head	Подстановка {Head = 4, Tail = [6], Sum = TailSum} list_sum([6], TailSum)	Прямой ход к следующему предложению.
8	list_sum(Tail, TailSum), Sum = TailSum + Head, Sum = TailSum + Head, Sum = TailSum + Head, Sum = TailSum + Head, Sum = TailSum + Head.	Подстановка {Head = 6, Tail = [], Sum = TailSum} list_sum([], TailSum)	Прямой ход к следующему предложению.
9	list_sum(Tail, TailSum), Sum = TailSum + Head, Sum = TailSum + Head, Sum = TailSum + Head, Sum = TailSum + Head, Sum = TailSum + Head, Sum = TailSum + Head.	Сравнение [] и [] list_sum([], TailSum) list_sum([], 0). Подстановка {Head = [], Tail = [], Sum = TailSum}	Прямой ход к следующему предложению.
10	list_sum([], 0), Sum = TailSum + Head, Sum = TailSum + Head, Sum = TailSum + Head, Sum = TailSum + Head, Sum = TailSum + Head, Sum = TailSum + Head.	Подстановка {TailSum = 0}	Прямой ход к следующему предложению.
	Sum = TailSum + Head, Sum = TailSum + Head, Sum = TailSum + Head,	Подстановка {Sum = 1}	Прямой ход к следующему предложению.

	Sum = TailSum + Head, Sum = TailSum + Head, Sum = TailSum + Head.		
	Sum = TailSum + Head, Sum = TailSum + Head, Sum = TailSum + Head, Sum = TailSum + Head, Sum = TailSum + Head.	Подстановка {Sum = 4}	Прямой ход к следующему предложению.
	Sum = TailSum + Head, Sum = TailSum + Head, Sum = TailSum + Head, Sum = TailSum + Head.	Подстановка {Sum = 9}	Прямой ход к следующему предложению.
	Sum = TailSum + Head, Sum = TailSum + Head, Sum = TailSum + Head.	Подстановка {Sum = 11}	Прямой ход к следующему предложению.
	Sum = TailSum + Head, Sum = TailSum + Head.	Подстановка {Sum = 15}	Прямой ход к следующему предложению.
	Sum = TailSum + Head.	Подстановка {Sum = 21}	Прямой ход к следующему предложению.
11		**Результат**: Res = 21.	Обратный ход.

Ответы на вопросы

1) **Что такое рекурсия? Как организуется хвостовая рекурсия в Prolog? Как организовать выход из рекурсии 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. Организовать выход из рекурсии можно при помощи отсечения.

2) **Какое первое состояние резольвенты?**

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

3) **В каких пределах программы уникальны переменные?**

Переменные уникальны в пределах предложения, т.е. в рамках предложения одно и то же имя принадлежит одной и той же переменной. Исключение – анонимные переменные (обозначаются символом нижнего подчеркивания «_») – каждая такая переменная является отдельной сущностью и применяется, когда ее значение неважно для данного предложения.

4) **В какой момент, и каким способом системе удается получить доступ к голове списка?**

Системе удается это сделать в момент подстановки переменной, с помощью конструкции [Head|Tail], где Head – голова списка, а Tail – хвост.

5) **Каково назначение использования алгоритма унификации?**

Назначение алгоритма унификации заключается в попарном сопоставлении термов и попытке построить для них общий пример.

Унификация может завершаться успехом или тупиковой ситуацией (неудачей).

6) **Каков результат использования алгоритма унификации?**

Унификация может завершаться успехом или тупиковой ситуацией (неудачей).

7) **Как формируется новое состояние резольвенты?**

Изменение резольвенты происходит в 2 этапа:

1) из стека выбирается подцель (верхняя, т.к. стек) и для нее выполняется редукция, т.е. замена подцели на тело найденного правила;

2) к полученной конъюнкции целей применяется подстановка (наибольший общий унификатор выбранной цели и заголовка сопоставленного с этой целью правила).

8) **Как применяется подстановка, полученная с помощью алгоритма унификации – как глубоко?**

При успешной унификации применяется подстановка. Переменные связываются со значениями.

9) **В каких случаях запускается механизм отката?**

Механизм отката запускается в 2 случаях:

- Если алгоритм попал в тупиковую ситуацию.
- Если резольвента не пуста и решение найдено, но в базе знание остались не отмеченные предложения.

10) **Когда останавливается работа системы? Как это определяется на формальном уровне?**

Работа системы останавливается, если резольвента пуста.