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»

Отчёт

по лабораторной работе №8

Название: Формирование и модификация списков на Prolog.

Дисциплина: Функциональное и логическое программирование.

Студент	ИУ7-64Б		Л.Е.Тартыков
	(Группа)	(Подпись, дата)	(И.О. Фамилия)
Преподаватель			Н.Б.Толпинская
		(Подпись, дата)	(И.О. Фамилия)
Преподаватель			Ю.В.Строганов
		(Подпись, дата)	(И.О. Фамилия)

Москва, 2022

1 Практические задания

1.1 Задание

Используя хвостовую рекурсию, разработать эффективную программу, (комментируя назначение аргументов), 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-ого задания составить таблицу, отражающую конкретный порядок работы системы.

Код программы для пункта 1 представлен на листинге 1.1.

Листинг 1.1 – Код программы (пункт 1)

```
domains
    list = integer*
predicates
    form_list_greater(list, integer, list).
clauses
    form_list_greater([Head|Tail], N, [Head|List]) :-
    Head > N, !,
    form_list_greater(Tail, N, List).
    form_list_greater([_|Tail], N, List) :- form_list_greater(Tail, N, List).
    form_list_greater([], _, []).
goal
    %Сформировать список из элементов числового списка,
    %больших заданного значения
    form_list_greater([1, 2, 3, 4, 5], 2, ResList).
```

Код программы для пункта 2 представлен на листинге 1.2.

Листинг 1.2 – Код программы (пункт 2)

```
domains
    list = integer*
predicates
    form_odd(list, list).
clauses
    form_odd([_, Head|Tail], [Head|ResList]) :- form_odd(Tail, ResList).
    form_odd([], []).
goal
    %Сформировать список из элементов, стоящих на
    %нечетных позициях исходного списка (нумерация от 0);

    form_odd([1, 2, 3, 4, 5], ResList).
```

Код программы для пункта 3-4 представлен на листинге 1.3.

Листинг 1.3 – Код программы (пункт 3)

```
domains
    list = integer*.
predicates
    del_elem(list, integer, list).
    form_set(list, list).
clauses
    del_elem([Head|Tail], N, [Head|ResList]) :- Head <> N, !,
    del_elem(Tail, N, ResList).
    del_elem([_|Tail], N, ResList) :- del_elem(Tail, N, ResList).
    del_elem([], _, []).

    form_set([Head|Tail], [Head|ResList]) :- del_elem(Tail, Head, Temp), !,
    form_set(Temp, ResList).
    form_set([], []).
goal
    %Удалить заданный элемент из списка (один или все вхождения);
    %del_elem([1, 2, 3, 1, 4, 1, 6, 1], 1, ResList).

    %Преобразовать список в множество (можно использовать ранее разработанные
    %процедуры).

    form_set([1, 2, 3, 2, 3, 4], ResList).
```

Ниже на рисунках 1.1 1.2 приведена таблица порядка поиска ответа для нахождения суммы элементов списка:

№ 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	form_list_greater([1, 2, 3, 4, 5], 2, ResList)	form_list_greater([1, 2, 3, 4, 5], 2, ResList) = form_list_greater([Head|Tail], N, [Head|List]) Результат: унификация успешна Подстановка: {Head = 1, Tail = [2, 3, 4, 5], N = 2}	Прямой ход. Переход к телу правила. Редукция и подстановка в резольвенту.
2	1 > 2 ! form_list_greater([2, 3, 4, 5], 2, List)	1 > 2 Результат: нет	Обратный ход. Восстановление предыдущего состояния резольвенты (шаг 1). Реконкретизация переменных.
3	form_list_greater([1, 2, 3, 4, 5], 2, ResList)	form_list_greater([1, 2, 3, 4, 5], 2, ResList) = form_list_greater([Head|Tail], N, [Head|List]) Результат: унификация успешна. Подстановка: {Tail = [2, 3, 4, 5], N = 2}	Прямой ход. Переход к телу правила. Редукция и подстановка в резольвенту.
4	form_list_greater([2, 3, 4, 5], 2, List)	form_list_greater([2, 3, 4, 5], 2, List) = form_list_greater ([Head|Tail], Old_sum, Sum) Результат: унификация успешна. Подстановка: {Head = 2, Tail = [3, 4, 5]}	Прямой ход. Переход к телу правила. Редукция и подстановка в резольвенту.
5	2 > 2 ! form_list_greater([2, 3, 4, 5], 2, List)	2 > 2 Результат: нет	Обратный ход. Восстановление предыдущего состояния резольвенты (шаг 4). Реконкретизация переменных.
6	form_list_greater([2, 3, 4, 5], 2, List)	form_list_greater([2, 3, 4, 5], 2, List) = form_list_greater([_|Tail], N, List) Результат: унификация успешна. Подстановка: {Tail = [3, 4, 5], N = 2}	Прямой ход. Переход к телу правила. Редукция и подстановка в резольвенту.
7	form_list_greater([3, 4, 5], 2, List)	form_list_greater([3, 4, 5], 2, List) = form_list_greater([Head|Tail], N, [Head|List]) Результат: унификация успешна. Подстановка: {Head = 3, Tail = [4, 5]}	Прямой ход. Переход к телу правила. Редукция и подстановка в резольвенту.
8	3 > 2 ! form_list_greater([3, 4, 5], 2, List)	3 > 2 Результат: да	Прямой ход. Переход к следующей цели в резольвенте.
9	! form_list_greater([3, 4, 5], 2, List)	! Результат: да.	Прямой ход. Переход к следующей цели в резольвенте.
10	form_list_greater([3, 4, 5], 2, List)	form_list_greater([3, 4, 5], 2, List) = form_list_greater([Head|Tail], N, [Head|List]) Результат: унификация успешна. Подстановка: {Head = 3, Tail = [4, 5]}	Прямой ход. Переход к телу правила. Редукция и подстановка в резольвенту.
…			

Рисунок 1.1 – Таблица порядка поиска ответов для нахождения суммы элементов списка.

№ 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?)
13	form_list_greater([5], 2, List)	form_list_greater([5], 2, List) = form_list_greater([Head|Tail], N, [Head|List]) Результат: унификация успешна. Подстановка: {Head = 5, Tail = []}	Обратный ход (резольвента пуста, БЗ просмотрена вся). Восстановление состояния резольвенты (шаг 27).
14	5 > 2 ! form_list_greater([], 2, List)	5 > 2 Результат: да	Прямой ход. Переход к следующей цели в резольвенте.
15	! form_list_greater([], 2, List)	! Результат: да.	Прямой ход. Переход к следующей цели в резольвенте.
…			
18	form_list_greater([], 2, List)	form_list_greater([], 2, List) = form_list_greater([], _, []) Результат: унификация успешна. Подстановка: {List = []}	Обратный ход. Реконкретизация переменных. Восстановление состояния резольвенты (шаг 15).
19	! form_list_greater([], 2, List)	! Результат: неудача.	Запрет выполнения form_list_greater([], 2, List). Обратный ход. Восстановление состояния резольвенты (шаг 13).
20	form_list_greater([5], 2, List)	Резольвента пуста (после прямого хода). Подстановка: {Head = 5, List = [5])	Обратный ход. Реконкретизация переменных. Восстановление состояния резольвенты (шаг 15).
…			
22	form_list_greater([4, 5], 2, List)	Резольвента пуста (после прямого хода). Подстановка: {Head = 4, List = [4, 5])	Обратный ход. Реконкретизация переменных. Восстановление состояния резольвенты.
…			
24	form_list_greater([3, 4, 5], 2, List)	Резольвента пуста (после прямого хода). Подстановка: {Head = 3, List = [3, 4, 5])	Обратный ход. Реконкретизация переменных. Восстановление состояния резольвенты (шаг 10). Сохранение подстановки {List = [3, 4, 5]} в памяти.
…			Обратный ход. Реконкретизация переменных. Восстановление состояния резольвенты (шаг 1).
…	form_list_greater([1, 2, 3, 4, 5], 2, ResList)	form_list_greater([1, 2, 3, 4, 5], 2, ResList) = form_list_greater([], _, []) Результат: унификация неуспешна	Обратный ход. Резольвента пуста. Все метки спущены. Вывод подстановки {ResList = [3, 4, 5]} на экран.

Рисунок 1.2 – Таблица порядка поиска ответов для нахождения суммы элементов списка (продолжение).

2 Контрольные вопросы

1. **Что такое рекурсия? Как организуется хвостовая рекурсия в Prolog? Как организовать выход из рекурсии?**
   Рекурия – ссылка на описываемый объект при описании объекта. Хвостовая рекурсия организовывается следующим образом: сначала выполняются необходимые вычисления, и последним шагом такой «функции» является вызов того же самого объекта. При этом вычисления собираются по мере выхода из рекурсии. Для того, чтобы система не выполняла лишних действий и при этом правильно отрабатывала, необходимо ставить условия выхода из рекурсии вначале.

2. **Какое первое состояние резольвенты?**
   В резольвенте изначально хранится конъюнкция вопросов.

3. **В каких пределах программы переменные уникальны?**
   Именованные переменные уникальны в пределах одного предложения, анонимные переменные – уникальные всегда.

4. **В какой момент, и каким способом системе удается получить доступ к голове списка?**
   Система использует разделитель (если указано это явно), которая делит исходный список на «голову» и «хвост». Например, такая конструкция языка [Head|Tail] позволит конкретизировать переменной Head голову списка.

5. **Каково назначение использования алгоритма унификации?**
   Назначение алгоритма унификации – подбор знаний.

6. **Как формируется новое состояние резольвенты?**
   Резольвента меняется в два этапа.

   (a) Новое состояние приобретается в результате алгоритма редукции.

   (b) К полученному состоянию применятся подстановка.

7. **Как применяется подстановка, полученная с помощью алгоритма унификации?**
   Подстановка применяется путем конкретизации переменных.

8. **В каких случаях применяется механизм отката?**
   Механизм отката применяется в случае тупиковой ситуации – в ситуации, когда нельзя перейти и данного состояния в новое, и при этом резольвента непуста.

9. **Когда останавливается работа системы? Как это определяется на формальном уровне?**
   Работа системы останавливается, когда ей [системе] не удалось подобрать факт для ответа на вопрос. На формальном уровне резольвента должна быть пуста , а также просмотрена вся БЗ.