

# Лабораторная работа №3: Рекурсивные структуры данных. Списки

## Цель работы

## Основные теоретические положения

Пролог позволяет определить и использовать рекурсивные типы данных. Примерами рекурсивных типов данных служат списки и деревья. Список – это объект данных, содержащий конечное число других объектов (элементов списка). Список, содержащий числа 1, 2 и 3, записывается следующим образом: [1, 2, 3].

Для объявления списка используется следующее описание домена:

```
domains
integerlist=integer*
```

Список является рекурсивным составным объектом, он состоит из двух частей:

- головы списка – первого элемента списка;
- хвоста списка – списка, включающего все последующие элементы.

Пусть имеется список [1| [2, 3]]. Тогда головой списка будет элемент 1, а хвостом [2, 3].

Так как список имеет рекурсивную составную структуру, для работы со списками используется рекурсия.

*Пример 1.* Вывод элементов списка.

```
domains
integerlist=integer*
predicates
printlist (integerlist)
clauses
printlist ([]):- !. % Для пустого списка ничего не делать
printlist ([H|T]):- write (H), nl, printlist (T). % Для непустого списка
                                                    % отделить голову,
                                                    % вывести ее,
                                                    % продолжить вывод для
                                                    % хвоста списка
```

*Пример 2.* Необходимо преобразовать список, элементами которого являются целые числа, инвертируя знак элементов списка, то есть положительные числа преобразовать в отрицательные, отрицательные в положительные, для нулевых значений никаких действий не предпринимать.

Придется рассмотреть два случая – для непустого и пустого списков. Преобразование пустого списка дать в результате также пустой список. Если же список не пуст, то следует рекурсивно выполнять отделение головы списка, ее обработку и рассматривать полученный результат как голову списка-результата.

```
domains
intlist=integer*
predicates
inverting (intlist, intlist)
processing (integer, integer)
clauses
% обработка пустого списка дает, в результате, тоже пустой список
inverting ([ ], [ ]):- !.
% если список непустой, отделить голову, обработать ее,
% и добавить в качестве головы списка-результата
inverting ([H | T], [Inv_H | Inv_T]):-
processing (H, Inv_H), inverting (T, Inv_T).
% предикат processing выполняет действия по обработке элемента списка в
% зависимости от его знака, предикат имеет два предложения,
% так как нужно рассмотреть два варианта: ненулевое и нулевое значения
processing (, ):- !.
processing (H, Inv_H):- Inv_H=-H.
goal
inverting ([-2, -1, , 1, 2], Inv_List), write("Inv_List=", Inv_List).
```

Результат работы программы:

```
Inv_List=[2, 1, , -1, -2]
```

## Постановка задачи

## Порядок выполнения работы

## Варианты заданий

## Содержание отчёта

From:  
<http://se.moevm.info/> - se.moevm.info

Permanent link:  
[http://se.moevm.info/doku.php/courses:knowledge\\_base\\_and\\_expert\\_system:lab3?rev=1567035045](http://se.moevm.info/doku.php/courses:knowledge_base_and_expert_system:lab3?rev=1567035045)

Last update: 2022/12/10 09:08



