

Министерство образования Республики Беларусь

Учреждение образования  
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет информационных технологий и управления

Кафедра информационных технологий автоматизированных систем

Отчёт  
по практической работе №2  
«Обработка Списков»  
по дисциплине «Экспертные Системы»

Вариант 2

Выполнил:  
студент гр. 820601  
Шведов А. Р.

Проверила:  
Т. В. Тиханович

Минск 2022

# **1 ЦЕЛЬ РАБОТЫ**

Целью данной работы является изучение механизмов и принципов работы списков в языке Пролог.

## 2 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

### 2.1 Списки в языке Пролог

В функциональных и логических языках списки используются чрезвычайно часто, они позволяют сохранить набор данных произвольной длины. В общем случае список представляет собой абстрактный тип данных, задающий набор значений. В этой статье под списком понимается «связный список», являющийся одной из возможных реализаций абстрактных списков.

Связный список – структура данных, состоящая из узлов. Узел содержит данные и ссылку (указатель, связку) на один или два соседних узла. Списки языка *Prolog* являются односвязными, т.е. каждый узел содержит лишь одну ссылку. В языке *Prolog* программист не сталкивается с явной работой с указателями в узлах, однако ему нужно иметь общее представление о списках, т.к. являясь основной структурой данных в функциональных и логических языках, они обладают рядом существенных отличий от массивов, используемых в императивных языках. В частности, элемент данных может быть очень быстро добавлен или удален из начала односвязного списка. Однако операция произвольного доступа (обращения к  $n$ -ному элементу) в списках выполняется гораздо дольше чем в массивах, т.к. требует  $n$  операций перехода по ссылкам.

При работе с односвязными списками необходимо выделять первый узел (называемый головой списка), остальные узлы (составляющие хвост списка) можно получить передвигаясь по указателям вплоть до последнего узла. Хвост списка является таким же списком, как и исходный, поэтому обрабатывается аналогичным образом (рекурсивно).

При использовании списков в Пролог необходимо объявить тип списка разделе *domains*:

*domains*

*ilist = integer\**

В дальнейшем, при объявлении функций обработки списков в разделе *predicates* необходимо использовать объявленный тип данных. Функция обработки списков вещественных чисел не будет обрабатывать список целых – это не очень удобно, однако позволяет выявлять несоответствие типов на этапе трансляции и генерировать более оптимальный код компилятору.

При работе со списками всегда (или обычно) применяется рекурсия. Рекурсия – это раскрытие предиката через самого себя.

## 3 ПРАКТИЧЕСКАЯ ЧАСТЬ

### 3.1 Ход работы

Согласно заданию, напомним программу по определению количества элементов в списке.

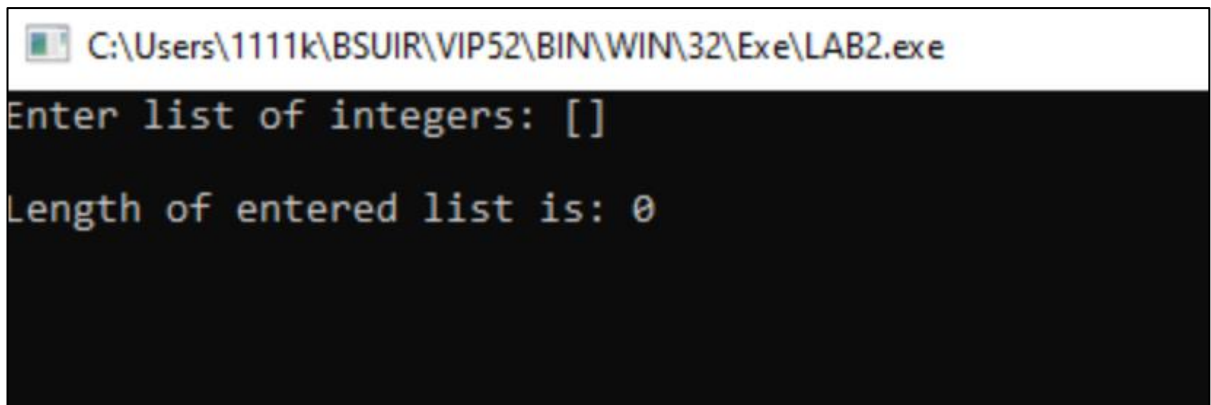
В разделе *domains* был объявлен тип списка целых чисел *intList*. В разделе *predicates* были объявлены предикаты *result*, *listLength(intList, integer)* и *listLength(intList, integer, integer)*.

Целевым предикатом является предикат-правило ***result***. Он вызывает стандартный предикат *write()* для вывода на экран сообщения о необходимости ввести список целых чисел. Далее с помощью конъюнкции с стандартным предикатом *readterm()* в переменную *LIST* типа *intList* считывается введенный пользователем список. Далее с помощью конъюнкции вызывается предикат *listLength(intList, integer)*, с передачей значения переменной *LIST* и переменной *Length*. После этого путём конъюнкции со стандартным предикатом *write()* на экран выводится значение переменной *Length*, которое было изменено правилом предиката *listLength(intList, integer)*. Далее программа ожидает ввода любого символа пользователем с помощью конъюнкции со стандартным предикатом *readln()*.

Предикат ***listLength([\_|Tail], Counter, Length)*** принимает список целых чисел 1-м аргументом, счётчик 2-м и переменную для занесения конечного значения длины списка 3-м аргументом. При передаче предикату списка, содержащего хотя бы один элемент, выполняется правило, которое присваивает переменной *NewCounter* значение переданной переменной *Counter*, увеличенное на 1, а также с помощью конъюнкции вызывает предикат *listLength([\_|Tail], Counter, Length)* рекурсивно с передачей хвоста переданной списка, значения переменной *NewCounter* и переданной переменной *Length*. При передаче данному предикату списка, не содержащего элементов, выполняется правило, которое присваивает переданной переменной *Length* значение переданной переменной *Counter*.

**Предикат *listLength(List, Length)*** принимает список целых чисел 1-м аргументом и переменную для занесения конечного значения длины списка 2-м аргументом. Данный предикат используется для вызова предиката *listLength([\_|Tail], Counter, Length)* с начальным значением переменной *Counter* равным нулю.

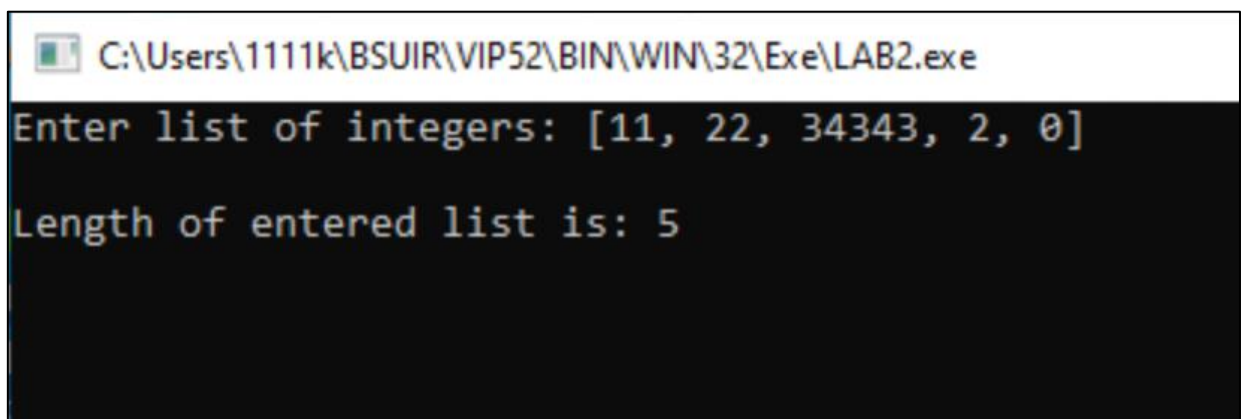
Пример исполнения программы при вводе пустого списка представлен на рисунке 1.



```
C:\Users\1111k\BSUIR\VIP52\BIN\WIN\32\Exe\LAB2.exe
Enter list of integers: []
Length of entered list is: 0
```

Рисунок 1 – Результат выполнения программы при вводе пользователем пустого списка

Пример исполнения программы при вводе списка, содержащего элементы, представлен на рисунке 2.



```
C:\Users\1111k\BSUIR\VIP52\BIN\WIN\32\Exe\LAB2.exe
Enter list of integers: [11, 22, 34343, 2, 0]
Length of entered list is: 5
```

Рисунок 2 – Результат выполнения программы при вводе пользователем списка, содержащего элементы

Ниже представлен код данной программы:

```
include "lab2.inc"
```

```
domains
```

*intList = integer\**

*predicates*

*result*

*listLength(intList, integer)*

*listLength(intList, integer, integer)*

*clauses*

*result:-*

*write("Enter list of integers: "),*  
*readterm(intList, LIST), nl,*  
*listLength(LIST, Length),*  
*write("Length of entered list is: ", Length),*  
*readln(\_).*

*listLength(List, Length):-*

*listLength(List, 0, Length).*

*listLength([], Counter, Length):- Length = Counter.*

*listLength([\_Tail], Counter, Length) :-*

*NewCounter = Counter + 1,*  
*listLength(Tail, NewCounter, Length).*

*goal*

*result.*

## **ВЫВОД**

В ходе выполнения данной лабораторной работы я изучил механизмы и принципы работы списков в языке Пролог, а также приобрёл практические навыки их использования.