Министерство науки и образования РФ
Федеральное государственное автономное образовательное
учреждение высшего профессионального образования
«Санкт-Петербургский государственный электротехнический
университет «ЛЭТИ» им. В. И. Ульянова (Ленина)»
(СПбГЭТУ «ЛЭТИ»)

Кафедра вычислительной техники

Отчёт
по лабораторной работе № 2
по дисциплине «Интеллектуальные системы»
Тема: «Рекурсивные структуры данных (списки)»

Выполнили студенты гр. 8307 Николаев Д.Е.
Репин С.А.
Такшеев А.Д.

Преподаватель Родионов С.В.

Санкт-Петербург
2022г.

Задание:

№ варианта	Операции
3	3, 9, 15

Операция 3
Удаление из списка всех элементов, равных 0.
Например: список [1, 0, 2, 0, 0, 3] преобразуется в список [1, 2, 3].

Операция 9
Подсчет количества элементов списка без какого-либо указываемого элемента.
Например: в списке [1, 2, 1, 3, 1] два элемента без учета единиц.

Операция 15
NEXT(A,B,Z) – элемент *B* следует за *A* в списке *Z*.

Ход работы:

Операция 3

Код программы:

```
domains
    intlist=integer*
predicates
    nondeterm list_exclude_zero(intlist, intlist)
clauses
    list_exclude_zero([], []).
    %если элемент не равен 0, то записываем, иначе пропускаем
    list_exclude_zero([H|T], [H|R_T]) :- H <> 0, list_exclude_zero(T, R_T).
    list_exclude_zero([0|T], R) :- list_exclude_zero(T, R).

goal
    list_exclude_zero([1, 0, 2, 0, 0, 3, 0], Res).
    %list_exclude_zero(Res, [1, 2, 3]).
    %list_exclude_zero([1, 0, A, 3], [1, 2, 3]).
    %list_exclude_zero([1, A, 2, C, D, B], [1, 2, 3]).
```

Результаты работы:

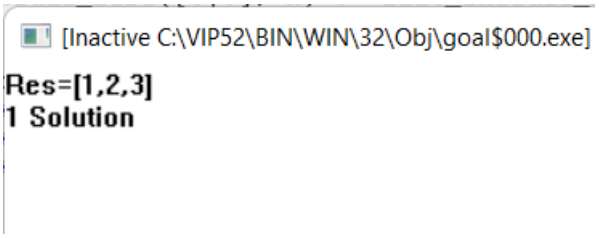

Рис. 1.1 – Результат выполнения цели list_exclude_zero([1, 0, 2, 0, 0, 3, 0], Res).

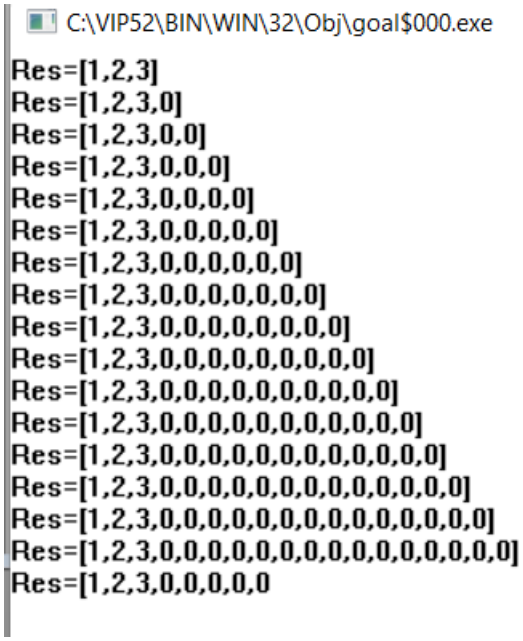

Рис. 1.2 - Результат выполнения цели list_exclude_zero(Res, [1, 2, 3]).

В результате выполнения получается бесконечный набор решений, которые отличаются количеством 0.

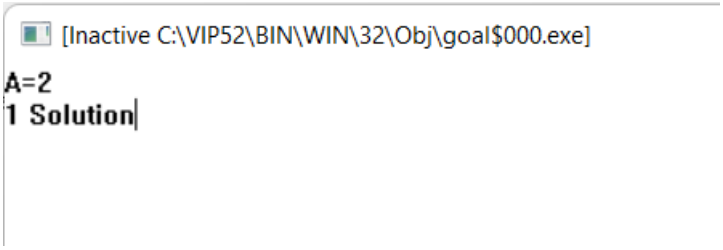

Рис. 1.3 – Результат выполнения цели list_exclude_zero([1, 0, A, 3], [1, 2, 3]).

В результате получается вывод возможного второго элемента списка чтобы в результате получался список [1,2,3].

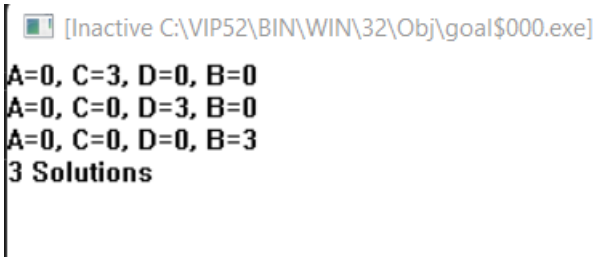

Рис. 1.4 – Результат выполнения цели list_exclude_zero([1, A, 2, C, D, B], [1, 2, 3]).

В результате выводятся все возможные варианты переменных A, C, D, B, чтобы в результате получался список [1,2,3].

Трасса для цели list_exclude_zero([1, 0, 2, 0, 0, 3, 0], Res).:

Steps	Call Stack	Variables
1	_PROLOG_Goal()	Res = _
2	List_execute_zero([1,0,2,0,0,3,0], _)	-
3		H = _ T =_ R_T = _
4	List_execute_zero([0,2,0,0,3,0], _)	-
5	List_execute_zero([2,0,0,3,0], _)	T =_ R_T = _
6	List_execute_zero([0,0,3,0], _)	T =_ R_T =
7	List_execute_zero([0,3,0], _)	T =_ R_T =
8	List_execute_zero([3,0], _)	T =_ R_T =
9	List_execute_zero([0], _)	T =_ R_T =[3]
10	List_execute_zero([0], _)	T =_ R_T =[2,3]
11	List_execute_zero([0], _)	Res = [1,2,3]

Операция 9

Код программы:

```
domains
    intlist=integer*
predicates
    nondeterm list_len_exl(integer, intlist, integer)
clauses
    list_len_exl(_, [], 0).
      %если элемент списка H не равняется элементу H, то увеличиваем счетчик
      %иначе пропускаем
    list_len_exl(X, [X|T], L) :- list_len_exl(X, T, L).
    list_len_exl(X, [H|T], L) :- list_len_exl(X, T, Nl), L = Nl + 1, X <> H.

goal
    list_len_exl(1, [1, 2, 1, 3, 1], Len).
```

Результат работы:

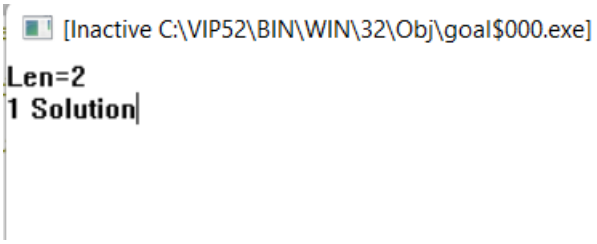

Рис. 2.1 – Результат выполнения цели list_len_exl(1, [1, 2, 1, 3, 1], Len).

Трасса для цели list_len_exl(1, [1, 2, 1, 3, 1], Len).

Steps	Call Stack	Variables
1	_PROLOG_Goal()	Len = _
2	list_len_exl(1, [1, 2, 1, 3, 1], _).	X = 1 T = _ L = _
3	list_len_exl(1, [2, 1, 3, 1], _).	X = 1 H = _ T = _ L = _ N1 = _
4	list_len_exl(1, [1, 3, 1], _).	-
5	list_len_exl(1, [3, 1], _).	
6	list_len_exl(1, [1], _).	X = 1 T = _ L = 1 H = _
7	list_len_exl(1, [1], _).	X = 1 T = _ L = 2
8	list_len_exl(1, [1], _).	Len = 2

Операция 15

Код программы:

```
domains
    intlist=integer*
predicates
    nondeterm list_next(integer, integer, intlist)
```

```
clauses
    list_next(A, B, []) :- fail.
      %когда нашли следующий элемент после A - закончили программу
    list_next(A, B, [A|[B|T]]) :- !.
    list_next(A, B, [H|T]) :- list_next(A, B, T).

goal
    list_next(5, 6, [1, 4, 5, 6, 0]). % yes
    %list_next(5, 6, [1, 4, 5, 5, 6, 0]). % yes
    %list_next(5, 6, [1, 4, 5, 0, 6, 5]). % no
    %list_next(5, 6, [5, 6, 5, 0, 6]). % yes

    %list_next(6, B, [1, 4, 5, 6, 0]). % yes, B=0
    %list_next(A, B, [1, 4, 5, 6, 0]). % yes, A=1, B=4
    %list_next(A, 7, [1, 4, 5, 6, 0]). % no
```

Результат работы:

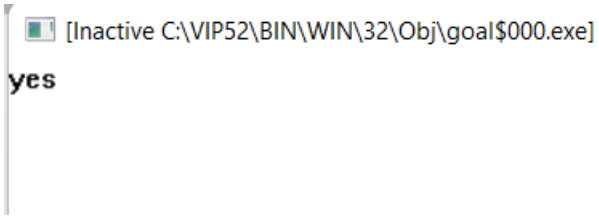

Рис. 3.1 – Результат выполнения цели list_next(5, 6, [1, 4, 5, 6, 0])

В списке действительно после 5 идёт элемент 6, поэтому программа выдала yes.

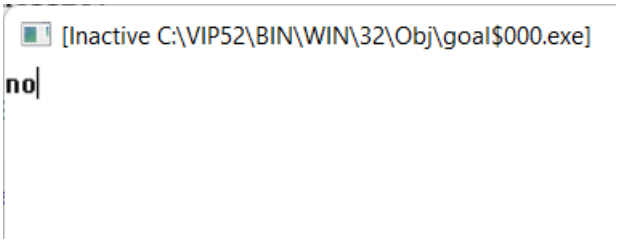

Рис. 3.2 – Результат выполнения цели list_next(5, 6, [1, 4, 5, 0, 6, 5])

В списке нет подпоследовательности 5, 6, поэтому программа выдаёт no.

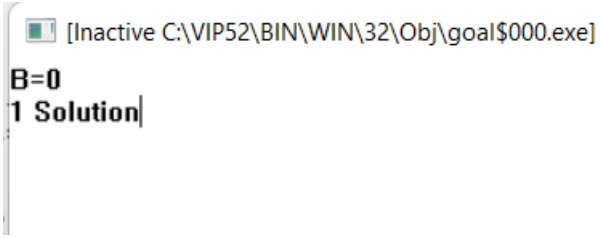

Рис 3.3 – Результат выполнения цели list_next(6, B, [1, 4, 5, 6, 0]).

Программа нашла первый элемент, который стоит после 6 в списке.

Трасса для цели list_next(5, 6, [1, 4, 5, 6, 0]).

Steps	Call Stack	Variables
1	_PROLOG_Goal()	_
2	list_next(5, 6, [1, 4, 5, 6, 0]).	A = 5 B = 6 T = _ H = _
3	PROLOG_Goal()	-

Вывод:
В ходе выполнения лабораторной работы мы рассмотрели способы работы с рекурсивными структурами данных, а именно списками в Prolog. Репин Степан совместно с Николаевым Дмитрием сформировали три программы, реализующие заданные операции над списками в соответствии с вариантом задания. Такшеев Артем, после их тестирования сформировал отчет указав результаты работы этих программ и их трассы. При выполнении лабораторной работы с трудностями не столкнулись