Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное

образовательное учреждение высшего образования

«Уфимский университет науки и технологий»

Кафедра ВМиК

Отчет к лабораторной работе №4

По дисциплине «Логическое программирование»

Выполнил:

Студентка группы ПРО-331Б:

 Семенов Л.А.

Проверил:

Старший преподаватель кафедры ВМиК

Шакирзянов А. А.

Уфа 2024

**Цель работы**:

Ознакомиться с основными механизмами работы Пролога.

**Задание**:

Реализовать на Прологе задачи поиска совершенных чисел, генерации перестановок и сочетаний, наивной и бинарной сортировки.

**Ход работы**

**1 задача**: написать программу, которая бы находила все совершенные числа.

**Код программы:**

ints(0).

ints(X) :- ints(Y), X is Y + 1.

perfect\_number(X) :- ints(X), Y is X - 1, calculatesum\_divisors\_till(Sum, X, Y), Sum = X.

calculatesum\_divisors\_till(0, \_NumberToDivide, 0).

calculatesum\_divisors\_till(Sum, NumberToDivide, Till) :- Till > 0,

Rem is NumberToDivide mod Till, Rem = 0, Ts is Till - 1,

calculatesum\_divisors\_till(SumPrev, NumberToDivide, Ts),

Sum is SumPrev + Till.

calculatesum\_divisors\_till(Sum, NumberToDivide, Till) :- Till > 0,

Rem is NumberToDivide mod Till, Rem > 0, Ts is Till - 1,

calculatesum\_divisors\_till(Sum, NumberToDivide, Ts).

% perfect\_number(X), X > 6.

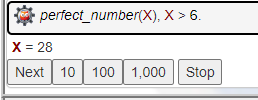
% определение натуральных чисел

% Совершенное число - это 1) натуральное число 2) сумма делителей равна числу

% Проверка суммы делителей 1-й аргумент Сумма, 2-й - число для которого ищем делители,

% 3-е - число до которого ищем делители

**Результат работы**



**2 задача**: перестановки.

**Код программы:**

perm([], []).

perm(Source, [Element|Tail]) :-

member\_list\_exclude(Element, Source, SourceExcluded),

perm(SourceExcluded, Tail).

member\_list\_exclude(X, [X|L], L).

member\_list\_exclude(X, [Y|L], [Y|Ls]) :- member\_list\_exclude(X, L, Ls).

% perm([1, 2, 3, 4], P), write(P), nl, fail.

% Если исходный список пустой, то существует одна перестановка - пустой список

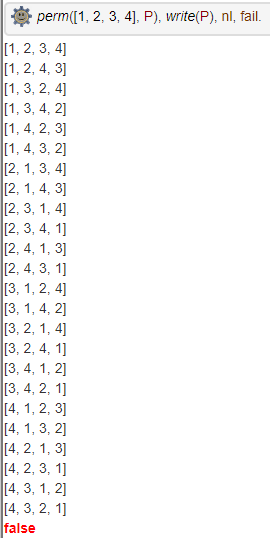
% Генерация перестановок

% Проверка того, что элемент содержится в списке, а 2-й список является списком без элемента

% Название предиката member\_list\_exclude соответствует порядку аргументов:

% 1-й - элемент, 2-й - список, 3-й - список без элементов

**Результат работы**



**3 задача**: написать программу, которая бы генерировала сочетания.

**Код программы:**

% Предикат member/2, проверяющий принадлежность элемента к списку

member(X, [X|\_]). % будет истинным, если X равен первому элементу списка

member(X, [\_|L]) :- member(X, L). % это тело правила, которое указывает на рекурсивное применение предиката member/2 к хвосту списка L.

comb([], []).

comb([X|List], [X|Tail]) :- comb(List, Tail).

comb([\_|List], Tail) :- comb(List, Tail).

%comb([1,2,3,4],C),write(C),nl,fail.

%сочетания

% Предикат comb/2 для генерации сочетаний

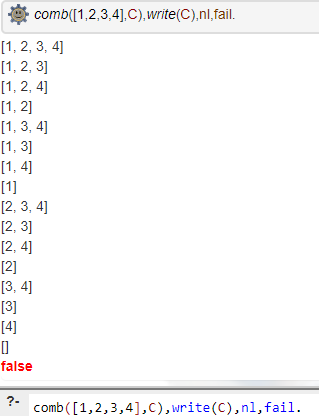
% Вариант 2: сочетание является правильным сочетанием хвоста списка,

% то есть 1-й элемент исходного списка не содержится в сочетании

% Вариант 2: сочетание является правильным сочетанием хвоста списка,

% то есть 1-й элемент исходного списка не содержится в сочетании

**Результат работы**



**4 задача**: написать программу, которая бы реализовывала наивную сортировку.

**Код программы:**

sort\_n([], []).

sort\_n(List, [Min|SortRest]) :- min\_list\_exclude(Min, List, Exclude), sort\_n(Exclude, SortRest).

min\_list\_exclude(M, [M], []).

min\_list\_exclude(Min, [M|L], ExcludeRes) :- min\_list\_exclude(Ms, L, Exclude),

find\_result(result(M, L), result(Ms, [M|Exclude]), result(Min, ExcludeRes)).

find\_result(result(M, L), result(Ms, \_), result(M, L)):- M < Ms.

find\_result(result(M, \_), result(Ms, Exclude), result(Ms, Exclude)):- Ms =< M.

%запуск кода

%sort\_n([13,1,24,16,5,10,21,6,9], SortedList)

%наивная

% здесь Min - это минимальный элемент списка List, а SortRest - список, который будет содержать

% оставшиеся элементы List, отсортированные в порядке возрастания.

% Рекурсивно исключаем минимальное число, если в списке одно число исключаем его

% если в списке только один элемент M, то минимальным элементом является он сам (M),

% и в результате исключений остаётся пустой список.

% если в списке более одного элемента ([M|L]), то минимальный элемент (Min) может быть

% либо головой списка (M), либо где-то в хвосте (Ms), который находится путем рекурсивного

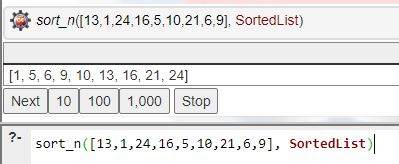
% вызова min\_list\_exclude/3.

% После этого используется предикат find\_result/3 для определения, какое из двух

% значений (M или Ms) будет минимальным.

%% Дополнительный предикат для определения пары с минимальным ключом

**Результат работы**



**5 задача**: сортировка с использованием бинарного дерева.

**Код программы:**

sort\_tree([], nil).

sort\_tree([X|L], Tree) :-

sort\_tree(L, LTree),

plus(X, LTree, Tree),

writeln(Tree). % Display the tree after each iteration

plus(X, nil, tree(X, nil, nil)).

plus(X, tree(O, L, R), tree(O, ResL, R)) :-

O >= X,

plus(X, L, ResL).

plus(X, tree(O, L, R), tree(O, L, ResR)) :-

O < X,

plus(X, R, ResR).

sort\_t(X, Y) :-

sort\_tree(X, Tree),

tree\_list(Tree, Y).

append\_list([], L, L).

append\_list([X|L], R, [X|T]) :-

append\_list(L, R, T).

tree\_list(nil, []).

tree\_list(tree(X, L, R), List) :-

tree\_list(L, ListL),

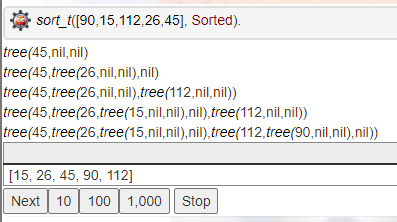
tree\_list(R, ListR),

append\_list(ListL, [X|ListR], List).

% Example usage

% ?- sort\_t([90,15,112,26,45], Sorted).

**Результат работы**



Задача 6: Быстрая сортировка с пошаговым решением.

quicksort([], []).

quicksort([Pivot|T], Sorted) :-

partition(Pivot, T, Left, Right),

quicksort(Left, SortedLeft),

quicksort(Right, SortedRight),

append(SortedLeft, [Pivot|SortedRight], Sorted),

write(Sorted), nl.

partition(\_, [], [], []).

partition(Pivot, [H|T], [H|Left], Right) :-

H =< Pivot,

partition(Pivot, T, Left, Right).

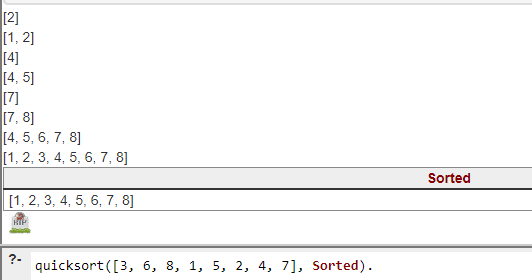
partition(Pivot, [H|T], Left, [H|Right]) :-

H > Pivot,

partition(Pivot, T, Left, Right).

% Example usage:

% quicksort([3, 6, 8, 1, 5, 2, 4, 7], Sorted).



**Вывод:**

В ходе выполнения лабораторной работы были получены навыки для работы на Прологе и реализованы задачи поиска совершенных чисел, генерации перестановок и сочетаний, наивной, бинарной сортировки и быстрой сортировки.