Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики» (СибГУТИ)

Кафедра прикладной математики и кибернетики

Лабораторная работа № 5

по дисциплине «Функциональное и логическое программирование» Бригада № 2

Работу выполнили:

студенты группы ИП-814

Иванов Константин

Краснов Илья

Якунин Андрей

Работу проверил: ассистент кафедры ПМиК

Пащенко А.А.

Новосибирск 2020 г.

**Содержание:**

1. Задание……………………………….…………………………..………3
2. Листинг программы……………………..………………………………4
3. Демонстрация работы программы…………..………………………....7

# Задание

1. Написать предикат, который печатает все нечётные числа из диапазона в порядке убывания. Границы диапазона вводятся с клавиатуры в процессе работы предиката.

2. Написать предикат, который находит числа Фибоначчи по их номерам, которые в цикле вводятся с клавиатуры. Запрос номера и нахождение соответствующего числа Фибоначчи должно осуществляться до тех пор, пока не будет введено отрицательное число.

Циклический ввод организовать с помощью предиката repeat.

Числа Фибоначчи определяются по следующим формулам:

F(0)=1, F(1)=1, F(i)=F(i-2)+F(i-1) (i=2, 3, 4, ...).

3. Написать предикат, который разбивает числовой список по двум числам, вводимым с клавиатуры на три списка: меньше меньшего введенного числа, от меньшего введенного числа до большего введенного числа, больше большего введенного числа. Список и два числа вводятся с клавиатуры в процессе работы предиката.

Например: [3,7,1,-3,5,8,0,9,2], 8, 3 –> [1,-3,0,2], [3,7,5,8], [9].

4. Написать предикат, который формирует список из наиболее часто встречающихся элементов списка. Список вводится с клавиатуры в процессе работы предиката. Встроенные предикаты поиска максимума и сортировки не использовать!

Например: [0,3,5,7,1,5,3,0,3,3,5,7,0,5,0] –> [0,3,5].

1. **Листинг программы**

fib(1,0,1).

fib(D,M,N):-

D > 0,

D1 is D-1,

fib(D1,M1,N1),

N is M1,

M is M1 + N1.

second(M,N):-

repeat,

write("Введите номер элемента: "),

read(D),

(D > 0 -> fib(D,M,N), write("M = "), writeln(M); write("")),

is\_it\_time(D), !.

is\_it\_time(A):-

A < 0, write("Выход").

recurs(L,R):-

L >= R,

L mod 2 =\= 0 -> format("~a ", L),

L1 is L - 1,

recurs(L1 ,R);

L >= R,

L1 is L - 1,

recurs(L1 ,R).

input:-

write("left border"),

read(L),

write("right border"),

read(R),

format("[ ~a:~a ]~n",[L, R]),

recurs(L,R).

third:-

write("Введите список: "),

read(List),

write("Введите первое число: "),

read(B),

write("Введите второе число: "),

read(C),

L1 = [],

L2 = [],

L3 = [],

(B > C, B1 is C, C1 is B; B1 is B, C1 is C),

first(List, B1, C1, L1, L2, L3).

first(List, B, C, L1, L2, L3):-

List = [Head|\_],

(Head > C, append([Head], L2, L22); L22 = L2),

(Head < B, append([Head], L1, L11); L11 = L1),

(Head =< C, Head >= B, append([Head], L3, L33); L33 = L3),

select(Head, List, List2),

(List2 \== [], first(List2, B, C, L11, L22, L33),!;

reverse(L11, L111),

reverse(L22, L222),

reverse(L33, L333),

write(L111),

write(' '),

write(L333),

write(' '),

write(L222)).

fourth:-

write("Введите список: "),

read(List),

in\_sort(List, L1),

L2 = [],

S is 1,

sch(L1, L1, S, L2).

in\_sort([ ],[ ]).

in\_sort([X|Tail], Sort\_list):-

in\_sort(Tail, Sort\_tail),

insert(X, Sort\_tail, Sort\_list).

insert(X,[Y|Sort\_list],[Y|Sort\_list1]):-

X@>Y,!,

insert(X, Sort\_list, Sort\_list1).

insert(X, Sort\_list, [X|Sort\_list]).

sch(L,[Head|Tail], S, L2):-

Tail = [H|\_],

(Head =:= H, S1 is S+1, L22 = L2; append([S], L2, L22), S1 is 1),

(Tail \== [], sch(L, Tail, S1, L22),!;

(Head =:= H, S2 is S1 + 1; S2 is S1),

append([S2], L22, L222),f(L, L222)).

del(L, LIST, List, L1):-

List = [Head|Tail],

append([Head], L1, L2),

delete(List, Head, List2),

(Tail \== [], del(L, LIST, List2, L2),!;f2(L, L2)).

find([Head|Tail], Del, Max, LRes):-

Del = [H|T],

(Head =:= Max, append([H], LRes, LRes1); LRes1 = LRes),

(Tail \== [], find(Tail, T, Max, LRes1),!; write(LRes1)).

f(List, L):-

L1 = [],

del(L, List, List, L1).

f2(L,Del):-

find\_max(L, Max),

LRes = [],

find(L, Del, Max, LRes).

find\_max([Max],Max):-!.

find\_max([Head|Tail],Max):-

find\_max(Tail,Max1),

Max1 > Head, !, Max = Max1; Max = Head.

# Демонстрация работы программы

# 

# *Рис. 1 Выполнение 1 задачи*

# 

# *Рис. 2 Выполнение 2 задачи*

# 

# *Рис. 3 Выполнение 3 задачи*

# 

# *Рис. 4 Выполнение 4 задачи*