**EЗИК ЗА ФУНКЦИОНАЛНО ПРОГРАМИРАНЕ - SCHEME**

***Семинарно упражнение 1***

Задача: Да се напише програма, която въвежда цяло число n и пресмята стойността на израза:

10n +1 - 10n

ln x <=> ех Взаимнообратни

ln =x

ex = (exp x)

ln x = (log x)

xy = -> (exp (\* y (log x)))

->(exp (\*(/ 1 3)(log 17)))

**Програма: (define( izpaz n)**

**(define t (exp (\* n (log 10))))**

**(-(+ t 1)t))**

Извикване на програмата: (izraz 10)

(izraz -5)

Задача: Да се напише програма, която изчислява по въведена стойност на х стойността на функцията:

f(x)=

Синтаксис на if

(if (<условие>)<следствие><алтернатива>)

= -> равно

not -> различно

(not (= a 0)) -> and , or

**Програма: (define (f x)**

**(if (<= x 0 ) 0**

**(if(> x 1 ) (\* x x x x)**

**x)))**

Извикване на програмата: (f 5); (f 0)

Задача: Да се напише програма, която намира най - голямото от три дадени числа

**Програма: (define (max a b c )**

**(if (and (> a b)(> a c)) a**

**(if (> b c) b c )))**

(define (max5 a b c d e)

(max (max a b c) d e))

(define (max9 a b c d e f j x y)

(max (max (max (max a b c ) d e )f j ) x y))

(define (max 9.1 a b c d e f j x y)

(max (max a b c)(max d e f)(max j x y))

Задача: По дадени три страни на триъгълник да се определи лицето

-> Хиронова формула

p =

**Програма: (define (lice a b c)**

**(define p (/(+ a b c)2))**

**(sqrt(\* p )(- p a)(- p b)(- p c)))**

Задача: Да се напише функция, която по дадени координати (x, y) определя дали точката с посочените координати принадлежи на защрихваната област

y

x

-5 <=x<=-2

2<=x<=5

-5<=y<=-2

2<=y<=5

|OM|=

2<=<=5

**Програма: (define (oblast x y)**

**(if (and(<=(sqrt(+(\* x x)(\* y y)))5**

**(>=(sqrt(+(\* x x)(\* y y)))2)) #t #f))**

#t - true (истина)

#f - false (лъжа)

* Оператор cond

(cond (<условие 1><изрази 1>)

(<условие 2><изрази 2>)

---------------------------------

(<условие n>< изрази n>)

(else <изрази n+1>))

* Локално присвояване - оператор let

(let ((<var 1><израз 1>

(<var 2><израз 2>)

-------------------------

(<var n><израз n>))

<тяло>)

Със стойностите var1, var2, ..., се изпълнява тялото на оператора.

Задача: Да се напише функция, която въвежда коефициентите a, b и c на уравнението

и намира решенията му.

- c = (- 0 c)

**Програма: (define (ur a b c)**

**(if (= a 0)**

**(if (= b 0)**

**(if (= c 0) "Всяко х е решение"**

**"Няма решение")**

**' (x = , (/(- 0 c) b)))**

**(let((d(-(\* b b)(\* 4 a c))))**

**(cond((> d 0) ' X1 = , (/(+(- 0 b)(sqrt d))(\* 2 a ))**

**X2=, (/(-(- 0 b)(sqrt d))(\* 2 a ))))**

**((= d 0) ' (X1 = X2 =, (/(- 0 b)(\* 2 a))))**

**(else "Няма реални корени")))))**

***Семинарно упражнение 2 . Рекурсия***

Задача: Да се напише програма, която въвежда естествено число n и и пресмята n!

n! = 1, ако n = 0

n! = n\*(n-1)! , ако n>0

4! = 1.2.3.4 = 24

**Програма: (define (n1! n)**

**(if ( = n 0) 1**

**(\* n (n1! (- n 1)))))**

**Програма:**

**(define (it\_n! n r i)**

**(if (> i n) r**

**(it\_n! n (\* r i)(+ i 1))))**

Извикване : (it\_n! 4 1 1) -> 24

n r i

4 1 1

4 1 2

4 2 3

4 6 4

4 24 5

**Програма (разгърнат вариант на предходната): (define (n! n)**

**(define (it\_n! n r i)**

**(if (> i n) r**

**(it\_n! n (\* r i)(+ i 1))))**

**(it\_n! n 1 1))**

**Програма: (define (it\_n! n r)**

**(if (> n 0) r**

**(it\_n! (- n 1)(\* r n))))**

**(it\_n! n 1))**

Задача: Да се напише програма, която изчислява по зададено естествено число n , n - тото число на Фибоначи

fib 1 = fib 2 = 1

fib n = fib n-1+ fib n-2

**Програма: (define (fib n)**

**(if (or(= n 1)(= n 2)) 1**

**(+ (fib (- n 1))(fib (- n 2)))))**

Задача: Да се напише функция, която въвежда две естествени числа и намира НОД

**Програма: (define (nod a b)**

**(if (= a b) a**

**(if (> a b) (nod (- a b) b)**

**(nod (a (- b a)))))**

Задача: Да се напише функция, която по въведено естевствено число n, пресмята сумата

S(n)=

**Програма: (define (f n)**

**(if (= n 1) 1**

**(+(f (- n 1))(/1 n))))**

Задача: Да се напише функция, която намира суматаот цифрите на дадено естествено число.

**Програма: (define (sum n)**

**(if (= n 0) 0**

**(+(sum (quotient n 10))(remainder n 10)))**

quotient - целочислено деление

remainder - остатък от целочислено деление

Задача: Да се напише функция, кояо обръща дадено естествено число.

**Програма: (define (it\_rev n r)**

**(if (= n 0) r**

**(it\_rev (quotientn 10)(+ ( remainder n 10)(\* r 10)))))**

**(define (rev n)**

**(it\_rev n 0))**

Извикване : (rev 1657)

Задача: Да се напише програма, която в даден интервал от цели числа [а,b) намира броя на числата в интервала, чиято сума от цифрите е равна на 13.

**Програма: (define (br13 a b)**

**(if (> a b) 0**

**(if (= (sum a) 13) (+ 1 (br13 (+ a 1) b))**

**(br13 (+ a 1) b ))))**

(if (= (remainder a 7) 0)) -> числа кратни на 7

(+ a (br13 (+ a 1) b)) -> не броим, а сумираме самите числа

Задача: Да се напише функция, която отпечатва целите числа в даден интервал (a, b)

**Програма: (define ( print a b)**

**(cond ((<= a b) (display a)(display " ")**

**(print (+ a 1) b))**

**(else (display " "))))**

***Семинарно упражнение 3. Списъци***

(a b 3 17)

() - празен списък

* Оператор cons

(cons 5 '()) -> (5)

(cons 5 ' (15 17 16 2)) -> (5 15 17 16 2)

* Оператор car - вземане на първия елемент на списък

(car '(6 4 11 3)) -> 6

* Оператор cdr - всички елементи без първия

(cdr '(11 2 8 3)) -> (11 2 8 3)

* Оператор Null

(null? l) = #t , ако L е празен списък

(null? l) = #f, ако L не е празен и има елементи

* Оператор List

(list? l) =#t , ако L e списък

(list? l) =#f, ако L не е списък

Задача: Да се напише функция, която преброява елементите на даден списък.

(11 2 4 6 8) ->5

**Програма:(define (len l)**

**(if (null? l) 0**

**(+ 1 (len (cdr l)))))**

Задача: Да се напише функция, която намира к- тия елемент в даден списък

**Програма: (define (uti k l)**

**(if (= k l) (car l)**

**(uti (- k 1)(cdr l))))**

(if (> k (len l))) "Списъка е по - къс"

Задача: Да се напише програма, която намира мин. елемент на даден числов списък

**Програма: (define(min\_el l)**

**(if (null? (cdr l)) (car l)**

**(if (> (car l) (min\_el (cdr l))) (min\_el (cdr l)) (car l))))**

**(define (min\_abs l)**

**(if (null? (cdr l)) (car l)**

**(if (<(abs (car l)(abs (min\_abs(cdr l)))) (car l))**

**(min\_abs (cdr l)))))**

Задача: Да се напише програма, която намира последния елемент на списък

**Програма: (define (last l)**

**(if (null? (cdr l)) (car l)**

**(last (cdr l ))))**

**Програма: (define posl l)**

**(uti (len l) l))**

Задача: Да се напише програма, която пресмята сумата на синусите на числа в даден интервал.

**Програма: (define (sum a b)**

**(if (> a b) 0**

**(+ (sin a) (sum (+ a 1) b))))**

Задача: Да се напише програма, която намира сумата на числата кратни на 3 в даден интервал (a,b)

**Програма: (define (int a b)**

**(if (> a b) 0**

**(if (= (remainder a 3) 0)**

**(+ a (int ((+ a 1) b ))**

**(int (+ a 1) b))))**

**Програма: (define (sum2 n)**

**(if (= n 1) 1**

**(+ (sum2 (- n 1 ))(/ 1 (\* n n)))))**

Задача: Повтарящи се цифри в число

**Програма: (define (ima? k n)**

**(if (= n 0) #f**

**(if (= (remainder n 10) k) #t**

**(ima? k (quotient n 10)))))**

**(define (povt n)**

**(if (< n 10) #f**

**(if (ima? remainder n 10) (quotient n 10)) #t**

**(povt (quotient n 10)))))**

Задача: Да се намери средноаритметичното на числа в даден интервал

**Програма: (define (sum3 a b)**

**(if (> a b) 0**

**(+ a (sum3 (+ a 1) b ))))**

**(define (ave a b)**

**(/(sum3 a b)(+ ( - b a ) 1)))**

Задача: Да се напише програма, която изстрива първото срещане на елемнт х от даден числов списък

**Програма : (define (del1 l x)**

**(if (null? l) l**

**(if (= (car l) x) (cdr l)**

**(cons (car l)(del1 (cdr l) x)))))**

Задача: Да се напише програма, която изстрива всяко срещане на даден елемент х в числов списък

**Програма: (define (delall x)**

**(if (null? l) l \*\***

**(if (= (car l) x) (delall (cdr l) x)**

**(cons (car l)(delall (cdr l) x)))))**

**\*\*В дълбочина -> (if (list? (car l))**

**(cons (delall (car l) x)(delall (cdr l) x))**

Задача: Обръщане на списък в дълбочина

**Програма: (define (rev l)**

**(if (null? l) l**

**(if (list? (car l)) (cons (rev (cdr l))(rev (car l)))**

**(cons(rev(cdr l)) (list (car l))))))**

Задача: Да се напише функция, която подрежда даден числов списък във възходящ ред

**Програма: (define (sort l)**

**(if (null? l) (cdr l)) (car l)**

**(cons (min\_el l) (sort (del1 l (min\_el l))))))**

**(define (sort1 l )**

**(rev (sort l)))**

Задача: Да се напише функция, която замества всяко срещане на елемент х с у в даден списък

**Програма: (define (ins l x y)**

**(if (null? l) l**

**(if(list? (car l)) (cons (ins (car l) x y) (ins (cdr l) x y))**

**(if (= (car l) x ) (cons y (ins (cdr l) x y))**

**(cons (car l)(ins (cdr l) x y ))))))**

Задача: Да се напише функция, която проверява има ли го елемента х в даден списък

**Програма: (define (is\_in l x )**

**(if (null? l ) l**

**(if (list? (car l)) (or(is\_in (car l) x) (is\_in (cdr l) x))**

**(if (= (car l) x ) #t**

**(is\_in (cdr l) x )))))**

**ЕЗИК ЗА ЛОГИЧЕСКО ПРОГРАМИРАНЕ - PROLOG**

***Семинарно упражнение 4***

Програмата е изградена от две части:

* Факти - Безусловно вярно твърдение

den(petak). -> клауза

предикат аргументи на отношението

den (X).

Х - променлива

* Правила - Условно вярно твърдение

uspeh(X):- X>= 3.

глава ако тяло на правилото

Въпроси:

?- uspeh(2.75).

No.

@

?- uspeh(3.50).

Yes.

\*\* - повдигане на степен

// - целочислено деление

10//3 = 3

10 mod 3 = 1

|X| -> abs(X)

sin(X)

cos(X)

ln|x|= log(X)

= sqrt(X)

ex = exp(X)

=:= - равно

=\= - неравно

is - сравнение

?- 5 is 10//2

Yes.

?- X is 5\*3, write(X).

15 Yes.

, - конюнкция

; - дезюнкция

?- X=10, X is 5\*3.

No.

Задача: Да се напише програма, която по даден радиус намира лицето на кръг.

**Програма:**

**lice(R, S):-S is 3.14 \* R \*\* 2.**

Въпроси: ?- lice(2,20).

No.

?- lice(4, S).

Yes.

?- lice(4,S), write (S).

50 Yes.

Задача: Да се напише програма, която по зададено Х пресмята функцията

**Програма: f(X,Y):- X < 1, !, Y is 0.**

**f(X,0):- X < 1.**

**f(X,Y):- X > 2, !, Y is 1 / X.**

**f(X,Y):-Y is X \* X - X + 4.**

? - f(1,Y), write (Y).

Ако P, то Q иначе R

C++: if (P) Q else R

Prolog : Q:- P,!.

R.

! - cut -> предикат, който не позволява връщане наляво от него.

Задача: Да се напише програма, която по зададени три страни на триъгълник изчислява лицето му.

-> Хиронова формула

p =

**Програма: per(A,B,C,P):- P is (A + B + C) / 2**

**lice(A,B,C, S):- per(A,B,C,P), S is sqrt(P \* (P - A) \* (P - B) \* (P - C)).**

? - lice(3,4,5,S), write(S).

6 Yes.

Задача: Да се напише програма, която по зададени коефициенти A,B и C решава уравнението .

**Програма: kv\_ur (0,0,0, R):-R="Всяко Х".**

**kv\_ur (A,B,C,R):- A =:= 0, B =:= 0, C =:= 0, R="Всяко Х".**

**kv\_ur(0,0,C,R):-C =\= 0, R= "Няма решение".**

**kv\_ur(0,B,C,R):-B =\= 0, R is (0 - C) / B.**

**kv\_ur(A,B,C,R):- D is B\*B - 4\*A\*C,**

**(D>0, X1 is ((0 - B + sqrt (D)) / (2 \* A), R=[X1,X2];**

**D =:= 0, R is (0 - B) / (2 \* A);**

**D < 0, R ="Няма реални корени").**

Задача: Да се напише програма, която по зададени естествено число n, изчислява n!

n! = 1, ако n = 0

n! = n\*(n-1)! , ако n>0

4! = 1.2.3.4 = 24

**Програма: factoriel(N,R):-N=:=0, R is 1.**

**factoriel (0,1).**

**factoriel (N,R):- N1 is N - 1, factoriel(N1, R1), R is N \* R1.**

?- factoriel (5,X), write(X).

120 Yes.

Задача: Да се напише програма, която намира сумата от цифрите на дадено естествено число.

**Програма: suma(0,0).**

**suma(N,S):-N1 is N mod 10, N2 is N // 10, sum(N2, R1), S is N1 + R1.**

Задача: Да се напише програма, която по зададено естествено число намира обърнатото.

**Програма: br\_cif(N,1):- N < 10, !.**

**br\_cif(N,B):- N1 is N // 10, br\_cif(N1, B1), B is B1 + 1.**

**ob\_chislo(N,N):- N < 10.**

**ob\_chislo(N,R):- N1 is N // 10, ob\_chislo(N1,R1), br\_cif (N,K), N2 is N mod 10,**

**R is N2 \* 10 \*\* (K-1) + R1.**

Задача: Да се напише програма, която проверява дали има цифра К в естествеото число N.

**Програма: imali (N,K):- N<10, !, K =:=N.**

**imali(N,K):- K is N mod 10,!.**

**imali(N,K):- N1 is N//10, imali(N1,K).**

Задача: Да се напише програма, която в даден интервал (a,b) преброява кратните на 7 елементи.

**Програма: br\_int(A,B, Br):- B > A, Br is 0.**

**br\_int(A,B, Br):- 0 is A mod 7, !, A1 is A + 1, br\_int(A1,B,Br1), Br is Br1+ 1.**

**br\_int(A,B,Br):- A1 is A + 1, br\_int(A1,B, Br).**

?- br\_int(113, 148, Br), write(Br).

Задача: Да се напише програма, която въвежда имената на три пилона и броя на дисковете и извежда реда на преместване на дисковете от първия на третия пилон.

**Програма: hanoy(A,B,C,1):- write(A), write("->"), write(C), nl.**

**hanoy(A,B,C,N):- N1 is N-1, hanoy(A,C,B,N1), write(A), write(" to "), write(C), nl,**

**hanoy(B,A,C, N1).**

fail - Прекъсва изпълнението и извежда No.

***Семинарно упражнениние 5. Списъци в пролог.***

[1,2, 56, b] - > списъка може да не е от еднотипни елементи

[H|T] - списък с глава H и опашка T

Опашката задължително е списък.

[] - празен списък

[1,2,5] [1,2,X] - при Х= 5 се унифицират

[Y,2,5] [1,2,X] - при Y =1 и X = 5 се унифицират

Задача: Да се напише програма, която увеличава два пъти всеки елемент на даден числов списък.

[2,0,-4,6,8] ->[4,0,-8, 12,16]

**Програма: uv2([],[]).**

**uv2([H|T],[H1|T1]):-H1 is H \* 2, uv2(T,T1).**

Задача: Да се напише програма, която намира дължината на даден списък.

**Програма: len([], 0).**

**len([H|T], R):- len(T,R1), R is 1 + R1.**

Ако 1 заместим с H получаваме сумата на елементите на списъка.

Задача: Да се напише програма, която слива два дадени списъка

**Програма: append([], L,L).**

**append([H|T], L,[H|T1]):-append(T,L,T1).**

?- append([2,0,-4,6,8], [5,8,3,1], R), write (R).

?- append(X,Y, [2,0,-4,6,8,5,7]), write (X), nl, write(Y), nl, fail.

? - append(X, [-3|Y], [5,8,11,-3,6,0,2]), write (X), nl, write(Y).

Задача: Да се напише програма, която номира последния елемент на даден списък.

**Програма: last (L,X):- append(\_,[X], L).**

?- last ([5,2,1,3], X), write(X).

Задача: Да се напише програма, която проверява принадлежи ли елемент на даден списък.

**imali(X,L):-append(\_,[X|\_],L).**

**imali(X,[]):-!,fail.**

**imali(X, [H|T]):- H=:=X,!.**

**imali(X,[H|T]):- imali(X,T).**

Задача: Да се напише програма, която разбива даден числов списък на подсписък на положителните и подсписък на отрицателните числа.

**Програма: razbivane([],[],[]).**

**razbivane([H|T],[H|T1], T2):- H > 0, razbivane(T,T1,T2).**

**razbivane([H|T], T1, [H|T2]):- H < 0, razbivane(T,T1,T2).**

**razbivane([H|T], T1,T2):-razbivane(T,T1,T2).**

Задача: Да се напише програма, която намира минималния елемент на числов списък.

**Програма: min\_el([X], X).**

**min\_el([H|T],L):- min\_el(T, L1), H < L1, !, L is H.**

**min\_el([H|T],L):- min\_el(T,L1), L is L1.**

Задача: Да се напише програма, която обръща елементите на даден списък

**Програма: rev([], []).**

**rev([H|T], L):-rev(T,T1), append(T1, [H], L).**

list(L):-L=[\_|\_].

допълнителна клауза за дълбочина: rev([H|T], L):-list(H)!, rev(H,H1), rev(T,T1), append(T1, H1, L).

Задача: Да се напише програма, която изстрива всяко срещане на елемент в даден списък.

**Програма: delall(\_,[],[]).**

**delall(X,[X|T], T1):-delall(X,T,T1).**

**delall(X,[H|T],[H|T1]):-delall(X,T,T1).**

**delall(X,[H|T], [H1|T1]):-list(H),!, delall(X,H,H1), delall(X,T,T1).**

Задача: Да се напише програма, която заменя всяко срещане на елемент X с елемент Y.

**Програма: zam(X,Y,[],[]).**

**zam(X,Y,[X|T], [Y|T1]):-zam(X,Y,T,T1).**

**zam(X,Y,[H|T],[H|T1]):-zam(X,Y,T,T1).**

**zam(X,Y,[H|T], [H1|T1]):-list(H),!, zam(X,Y,H,H1), zam(X,Y,T,T1).**

Задача: Да се напише програма, която намира сумата от простите числа, елементи на даден числов списък.

**Програма: sum\_pr([],0).**

**sum\_pr([H|T], R):- prosto(H), !, sum\_pr(T,R1), R is R1 + H.**

**sum\_pr([H|T], R):- sum\_pr(T,R).**

Промяна за създаване на списък от простите:

sum\_pr([],[]).

append([H], R1,R)

**prosto(N):- N1 is N//2, not(del(N,2,N1)).**

**del(N,A,B):- A > B, !, fail.**

**del(N,A,B):- 0 is N mod A,!.**

**del(N,A,B):- A1 is A + 1, del(N, A1,B).**

Задача: Да се напише програма, която сортира даден числов списък във възходящ ред.

**Програма: sort\_vm([],[]).**

**sort\_vm([H|T], L):-sort\_vm(T,T1), insert(H,T1,L).**

**insert(A, [H|T], [H|T1]):-H < A, insert(A,T,T1).**

**insert(A, L, [A|L]).**

Задача: Да се напише програма, която изстрива първото срещане на елемент в даден списък.

**Програма: del1(\_,[],[]).**

**del1(X,[X|T], T).**

**del1(X,[H|T],[H|T1]):-del1(X,T,T1).**