1 Scop

Insusirea notiunilor teoretice privind sistemele Lindenmayer precum si construirea unei aplicatii care sa implementeze aceste notiuni.

2 Studiu necesar

2.1 N. Tandareanu - "Introducere in programarea logica. Limbajul Prolog"

2.2 N. Tandareanu - Note de curs

3 Mod de realizare al lucrarii

3.1 Enuntarea problemei de rezolvat

3.2 Modul de rezolvare al problemei (algoritm, cod, exemple de date)

4 Exemplu

4.1 Enuntul problemei

4.1.1 Construiti un un sistem Lindenmayer care sa implementeze urmatoarea regula

4.1.2 - L -> L + L - L - L + L

4.1.3 Implementati acest sistem in limbajul Prolog

4.1.4 Codul Turbo Prolog

```
domains
    actiune=latmare;latmic;adaug;scad actiuni=actiune* database
    /*
        punctele de plecare
    */
        pointplec(actiuni)
    /*
        actiunile generatorului imaginii
    */
        lowact(actiuni)
        raport_tot(integer)
        predicates
    /*
        valorile de plecare
    */
    firsttime(integer,integer,integer,integer,integer)
    desen(integer,actiuni)
    /*
        se va construi o figura
    */
    job(integer,actiune)
    /*
        transformarea unei laturi
    */
    transf(actiuni,actiuni,actiuni)
    /*
        adaugarea unei noi laturi
    */
    adauga(actiuni,actiuni,actiuni)
    raport(actiuni,integer,integer,integer)
    act_raport mareste(integer,integer,integer)
    marime_noua(integer,integer)
    /*
        se va trasa figura nou obtinuta
    */
    picture(integer,integer,integer,actiuni)
    /*
        acest predicat execua scopul programului
    */
    actiune inpoint clauses
```

```
firsttime(20000,90,0,3000,5000).
desen(_,[]):-!. desen(D,[C|RestLc]):-job(D,C),desen(D,RestLc).
job(D,latmare):-!,forward(D). job(D,latmic):-!,penup,forward(D),pendown.
job(_,adaug):-!,firsttime(_,Delta,_,_,_),left(Delta).
job(_,scad):-!,firsttime(_,Delta,_,_,_),right(Delta).
transf([],[],_):-!.
transf([latmare|RestLc],Lcnew,Generator):-!,
transf(RestLc,RestLcnew,Generator),
adauga(Generator,RestLcnew,Lcnew).
transf([O|RestLc],[O|RestLcnew],Generator):-
transf(RestLc,RestLcnew,Generator).
adauga([],Lc2,Lc2):-!.
adauga([C|RestLc1],Lc2,[C|RestLc3]):-adauga(RestLc1,Lc2,RestLc3).
raport([],_,NrComF,NrComF):-!.
raport([adaug|RestGen],Abatere,NrComF1,NrComF):-!,
firsttime(_,Delta,_,_,_),
Abatere1=(Abatere+Delta) mod 360,
raport(RestGen,Abatere1,NrComF1,NrComF).
raport([scad|RestGen],Abatere,NrComF1,NrComF):-!,
firsttime(_,Delta,_,_,_),
Abatere1=(Abatere-Delta) mod 360,
raport(RestGen,Abatere1,NrComF1,NrComF).
raport([_|RestGen],Abatere,NrComF1,NrComF):-
mareste(Abatere,NrComF1,NrComF2),
raport(RestGen,Abatere,NrComF2,NrComF).
act_raport:-lowact(G),
raport(G,0,0,NrComF), assert(raport_tot(NrComF)).
mareste(0,NrComF1,NrComF2):-!,NrComF2=NrComF1+1.
mareste(_,NrComF1,NrComF1).
marime_noua(D,Dnew):-raport_tot(NrComF),
Dnew=D/NrComF.
picture(_,Nc,Nc,_):-!. picture(D,Nc,N,Lc):-firsttime(_,_,Alfa,X,Y),
penpos(X,Y,Alfa),
Cul=Nc+1,pencolor(Cul), pendown,
desen(D,Lc),marime_noua(D,Dnew),lowact(G),
transf(Lc,Lcnew,G), readchar(_), clearviewport, Nc1=Nc+1,
picture(Dnew,Nc1,N,Lcnew).
inpoint:-%assert(firsttime(15000,90,0,10000,10000)),
assert(pointplec([latmare])),
assert(lowact([latmare,adaug,latmare,scad,latmare,scad,latmare,
adaug,latmare])), act_raport, graphics(2,0,0),
initgraph(0,0,_,_,"C:\\soft\\prolog\\bgi").
actiune:-inpoint,pointplec(Lc),picture(22000,0,6,Lc),closegraph.
goal
```

actiune

5 Exercitii

5.1 In vederea aprofundarii cunostiintelor expuse se recomanda sa se efectueze urmatoarele

5.1.1 o implementare bazata pe alta lege de transformare

5.1.2 modificarea modului de apel a generatorului(per latura)