Tipuri structurate de date definite de utilizator

Problema 1. Jucarii

Scrieti o aplicatie care il ajuta pe Mos Craciun sa tina evidenta:

- **Jucariilor: idJucarie, denumireJucarie, dimensiuneJucarie**
- **Copiilor: idCopil, numecopil, adresaCopil**
- **Cadourilor: idCadou, idJucarie, idCopil, an**

Si va gestiona o lista de jucarii, o lista de copii si o lista de cadouri cu urmatoarele functionalitati:

- **Adauga, elimina, actualizare jucarie/copil**
- **Cautare jucarie/copil**
- **Creeaza statistici**
    1. **Jucariile cu dimensiunea mai mare decat o dimensiune precizata;**
    2. **Cea mai mica jucarie de un anumit fel (denumire);**
    3. **Toate cadourile pentru un copil dat**
    4. **Toti copiii care primesc aceeasi jucarie data**
    5. **Lista cadourilor dintr-un an dat.**
    6. **Toate jucariile ce sunt livrate la o adresa data.**

Analiza

A. Identificarea entitatilor

- **Jucarie:**
    - **idJucarie: intreg;**
    - **denumireJucarie: string**
    - **dimensiuneJucarie: real;**

- **SirJucarii:**
    - **sir: Jucarie [];**
    - **n: intreg;**

- **Copil:**
    - **idCopil: intreg;**
    - **numeCopil: string**
    - **adresaCopil: string;**

- **SirCopii:**
    - **sir: Copil [];**
    - **n: intreg;**

B. Identificarea si Specificarea subalgoritmilor

Subalgoritmul **citireDinFisier(sJ, sCo, sCa)**:
Descriere: citeste din fisier sirul de jucarii, sirul de copii si sirul de cadouri;
Date: -
Rezultate: sirul de jucarii sJ, sirul de copii sCo, sirul de cadouri sCa;

Sublagoritmul **afisareJucarii(sJ):**
Descriere: afiseaza sirul de jucarii;
Date: un sir de jucarii;
Rezultate: se afiseaza atributele tuturor jucariilor din sir

Sublagoritmul **afisareCopii(sCo):**
Descriere: afiseaza sirul de copii;
Date: un sir de copii;
Rezultate: se afiseaza atributele tuturor copiilor din sir

Sublagoritmul **afisareCadouri(sCa):**
Descriere: afiseaza sirul de cadouri;
Date: un sir de cadouri;
Rezultate: se afiseaza atributele tuturor cadourilor din sir

Functia **listaJucariiDimensiuneMaiMare (jucarii, D):**
Descriere: returneaza o lista cu jucarii a caror dimensiune este mai mare decat cea data;
Date: jucarii ∈ SirJucarii, D: real;
Rezultate: listaJucarii ∈ SirJucarii;
daca (∀ jucarie ∈ jucarii, jucarie.dimensiune > D)
- atunci listaJucarii ’= listaJucarii + jucarie;

Functia **ceaMaiMicaJucarieDenumire (jucarii, denumire):**
Descriere: determina jucaria cu denumriea precizata care are cea mai mica dimensiune;
Date: jucarii ∈ SirJucarii, denumire: string;
Rezultate: jucarie sau NIL
daca (∃ jucarie ∈ jucarii, jucarie.denumire = denumire, jucarie.dimensiune este minima)
- atunci returneaza jucarie
- altfel returneaza NIL;

Sublagoritmul **afiseazaOJucarie(oJucarie):**
Descriere: afiseaza caracteristicile unei jucarii;
Date: o jucarie;
Rezultate: se afiseaza atributele jucariei

Functia **existaCopil(sCo, idCo):**
Descriere: determina daca un copil (identificat prin id) se afla intr-un sir de copii;
Date: un sir de copii si un id de copil
Rezultate: *True*, daca copilul este gasit si *False*, altfel

Functia **toateCadourilePentruUnCopilDat(sC, idCo, sCo):**
Descriere: determina toate cadourile unui copil dat;
Date: un sir de cadouri, un id de copil si un sir de copii;
Rezultate: un sir de cadouri (aferente copilului)

Functia **getCopilCuIdDat(sCo, idCo):**
Descriere: determina dintr-un sir de copii copilul cu un id dat

Date: un sir de copii si un id de copil
Rezultate: copilul cautat

Functia **getJucarieCuIdDat(sJ, idJ):**
Descriere: determina dintr-un sir de jucarii jucaria cu un id dat
Date: un sir de jucarii si un id de jucarie
Rezultate: jucaria cautata

Sublagoritmul **afisareCadouriPentruUnCopil (sC, sJ, sCo):**
Descriere: afiseaza cadourile unui copil
Date: un sir de cadouri, un sir de jucarii, un sir de copii
Rezultate: se afiseaza cadourile unui copil

Functia **existaJucarie(sJ, idJ):**
Descriere: determina daca o jucarie (identificata prin id) se afla intr-un sir de jucarii;
Date: un sir de jucarii si un id de jucarie
Rezultate: *True*, daca cjucaria este gasita si *False*, altfel

Functia **totiCopiiiCarePrimescAceeasiJucarie(sC, idJ, sJ):**
Descriere: determina toate cadourile care contin aceeasi jucarie
Date: un sir de cadouri, un id de jucarie si un sir de jucarii
Rezultate: un sir de cadouri care contin jucaria precizata

Sublagoritmul **afisareCopiiPentruOJucarie (sC, sJ, sCo):**
Descriere: afiseaza copiii care au primit acelasi cadou
Date: un sir de cadouri, un sir de jucarii, un sir de copii
Rezultate: se afiseaza copiii care au primit acelasi cadou

Functia **listaCadourilorDinAnDat (sC, anDat):**
Descriere: determina toate cadourile care au fost oferite intr-un an dat
Date: un sir de cadouri, un an
Rezultate: un sir de cadouri din anul precizat

Sublagoritmul **afisareCadouriPentruAnDat(sC, sJ, sCo):**
Descriere: afiseaza cadourile dintr-un anumit an
Date: un sir de cadouri, un sir de jucarii, un sir de copii
Rezultate: se afiseaza cadourile

Exemple

1. Lista cu jucariilor cu dimensiunea mai amre decat D		
Exemplu	**Date de intrare**	**Date de iesire**
1.	*Sirul de jucarii:* • (1, “minge”, 7.5); • (2, “camion”, 8.5); • (3, “papusa”, 5); • (4, “tamburina”, 15); • (5, “trenulet”, 40); • (6, “cub magic”, 5); • (7, “masina”, 10); • (8, “camion”, 6); • (9, “calut”, 15); • (10, “papusa”, 10). *Dimensiunea D:* 9.00	Lista jucariilor cu dimensiune mai mare decat 9.00 este: • (4, “tamburina”, 15); • (5, “trenulet”, 40); • (7, “masina”, 10); • (9, “calut”, 15); • (10, “papusa”, 10);
2.	*Sirul de jucarii:* • (1, “minge”, 7.5); • (2, “camion”, 8.5); • (3, “papusa”, 5); • (6, “cub magic”, 5); • (8, “camion”, 6); *Dimesiunea D:* 9.00	Lista jucariilor cu dimensiune mai mare decat 9.00 este: Nu exista jucarii cu dimensiunea mai mare decat 9.00.

Implementare

```
program AppEvidentaCadouriMosCraciun;
type
  jucarie = record
    idJ   : integer;
    denuJ : string[20];
    dimJ  : Double;
  end;

sirJucarii = record
  nE   : integer;
  sirE : array [0..99] of jucarie;
end;

copil = record
  idC   : integer;
  numeC : string;
  adrC  : string;
end;

sirCopii = record
  nE: integer;
  sirE: array [0..Pred(100)] of copil;
end;

{*sir de cadouri*}
type cadou = record
       idCa: integer;
       idJ: integer;
       idCo: integer;
       an: integer;
end;

sirCadouri = record
       nE: integer;
       sirE: array [0..Pred(100)] of cadou;
end;

function citireStringDinFisier(var myfile: TEXT) : string;
var chRead : char;
    citit : string;
begin
  citit := '';
  Read(myfile, chRead); {spatiu}
  Read(myfile, chRead); {primul caracter din string}
  while(chRead <> ' ') and not eoln(myfile) do
  begin
    citit := citit + chRead;
    Read(myfile, chRead);
  end;
  citireStringDinFisier := citit;
end;
```

```
procedure citireDinFisier(var sJ: sirJucarii;  var sCo: sirCopii;  var sCa:
sirCadouri);
var  myfile: TEXT;
     nJ : integer;
     i : integer;
     nCo : integer;
     nCa : integer;
begin
  Assign(myfile, 'CadouriMosCraciun.txt');
  Reset(myfile);
  Readln(myfile, nJ);
  sJ.nE:= 0;
  for i:= 0 to nJ -1 do
  begin
    Read(myfile, sJ.sirE[sJ.nE].idJ);
    sj.sirE[sJ.nE].denuJ := citireStringDinFisier(myfile);
    Readln(myfile, sj.sirE[sJ.nE].dimJ);
    inc(sJ.nE);
  end;
  Readln(myfile,nCo);
  sCo.nE:= 0;
  for i:=0 to nCo -1 do
  begin
    Read(myfile, sCo.sirE[sCo.nE].idC);
    sCo.sirE[sCo.nE].numeC := citireStringDinFisier(myfile);
    sCo.sirE[sCo.nE].adrC := citireStringDinFisier(myfile);
    inc(sCo.nE);
  end;

  Readln(myfile,nCa);
  sCa.nE:= 0;
  for i:=0 to nCa -1 do
  begin
    Read(myfile, sCa.sirE[sCa.nE].idJ);
    Read(myfile, sCa.sirE[sCa.nE].idCo);
    Read(myfile, sCa.sirE[sCa.nE].an);
    inc(sCa.nE);
  end;

  Close(myfile)
end;

procedure afisareJucarii(sJ: sirJucarii);
var i : integer;
begin
  WriteLn('Jucariile lui Mos Craciun sunt: ');
  for i:=0 to Pred(sJ.nE) do
    Writeln(sJ.sirE[i].idJ, ', ', sJ.sirE[i].denuJ, ', ', sJ.sirE[i].dimJ);
end;

procedure afiseazaOJucarie(oJucarie: jucarie);
begin
  Writeln(oJucarie.idJ , ', ' , oJucarie.denuJ , ', ' , oJucarie.dimJ);
end;
```

```
procedure afisareCopii(sC: sirCopii);
var i : integer;
begin
  Writeln('Copiii sunt: ');
  for  i:=0 to sC.nE -1 do
    Writeln(sC.sirE[i].idC,', ',sC.sirE[i].numeC<<', ',sC.sirE[i].adrC);
end;

procedure afiseazaUnCadou(unCadou: cadou);
begin
  Writeln(unCadou.idCa , ', ',unCadou.idJ , ', ' , unCadou.idCo , ', ' , unCadou.an );
end;

procedure afisareCadouri(sC: sirCadouri);
var i : integer;
begin
  Writeln('Cadourile sunt: ');
  Writeln('idCa ' , ', ' , 'idJ ' , ', ' , 'idCo ' , ', ' , 'an ' );
  for i:=0 to sC.nE -1 do
    Writeln(sC.sirE[i].idCa, ', ',sC.sirE[i].idJ,', ',sC.sirE[i].idCo,', ',
                    sC.sirE[i].an );
end;

function listaJucariiDimensiuneMaiMare(sJ: sirJucarii;  dim: real): sirJucarii;
var sJDim: sirJucarii;
    i : integer;
begin
  sJDim.nE:= 0;
  for i:=0 to sJ.nE -1 do
    if sJ.sirE[i].dimJ>dim then
    begin
      sJDim.sirE[sJDim.nE]:= sJ.sirE[i];
      inc(sJDim.nE);
    end;
  listaJucariiDimensiuneMaiMare := sJDim;
end;

function ceaMaiMicaJucarieDenumire(sJ: sirJucarii;  const den : string): jucarie;
var oJucarie: jucarie;
    i : integer;
begin
  oJucarie.denuJ:= '';
  oJucarie.idJ:= -1;
  oJucarie.dimJ:= 1000;
  for i:=0 to sJ.nE -1 do
    if den=sJ.sirE[i].denuJ then
          begin
            if sJ.sirE[i].dimJ<oJucarie.dimJ then
            begin
              oJucarie:= sJ.sirE[i];
            end;
          end;
  ceaMaiMicaJucarieDenumire := oJucarie;
end;
```

```
procedure afiseazaOJucarie(oJucarie: jucarie);
begin
  Writeln(oJucarie.idJ , ', ' , oJucarie.denuJ , ', ' , oJucarie.dimJ);
end;

function existaCopil(sCo: sirCopii;  idCo: integer): boolean;
var gasit: boolean;
    i: integer;
begin
  gasit:=false;
  i:=0;
  while (i<sCo.nE) and( not gasit) do
        begin
          if sCo.sirE[i].idC=idCo then
            gasit:= true;
          inc(i);
        end;
  existaCopil := gasit;
end;

function toateCadourilePentruUnCopilDat(sC: sirCadouri;  idCo: integer;  sCo:
sirCopii): sirCadouri;
var  sCUnCopil: sirCadouri;
     i : integer;
begin
  sCUnCopil.nE:= 0;
  if not existaCopil(sCo,idCo) then
    Writeln('Nu exista copilul in lista de copii! ')
  else begin
    for i:=0 to sC.nE -1 do
      if sC.sirE[i].idCo=idCo then
            begin
              sCUnCopil.sirE[sCUnCopil.nE].idCa:= sC.sirE[i].idCa;
              sCUnCopil.sirE[sCUnCopil.nE].idCo:= sC.sirE[i].idCo;
              sCUnCopil.sirE[sCUnCopil.nE].idJ:= sC.sirE[i].idJ;
              sCUnCopil.sirE[sCUnCopil.nE].an:= sC.sirE[i].an;
              inc(sCUnCopil.nE);
            end;
  end;
  toateCadourilePentruUnCopilDat:= sCUnCopil;
end;

function getCopilCuIdDat(sCo: sirCopii;  idCo: integer): copil;
var c: copil;
    i: integer;
    gasit: boolean;
begin
  i:=0;
  gasit:=false;
  while (i<sCo.nE)and (not gasit) do
    if sCo.sirE[i].idC=idCo then
    begin
      c:= sCo.sirE[i];
      gasit:= true;
```

```
    end else
     inc(i);
  getCopilCuIdDat := c;
end;

function getJucariaCuIdDat(sJ: sirJucarii;  idJ: integer): jucarie;
var  j: jucarie;
     i: integer;
     gasit: boolean;
begin
  i:=0;
  gasit:=false;
  while (i<sJ.nE) and (not gasit) do
    if sJ.sirE[i].idJ=idJ then
          begin
            j:= sJ.sirE[i];
            gasit:= true;
          end
    else
      inc(i);
  getJucariaCuIdDat := j;
end;

procedure afisareCadouriPentruUnCopil(sC: sirCadouri;  sJ: sirJucarii;  sCo: sirCopii);
var idCopil: integer;
    unC: copil;
    idJucarie: integer;
    oJ: jucarie;
    i : INTEGER;
begin
  if sC.nE>=1 then
  begin
    idCopil:=sC.sirE[0].idCo;
    unC:=getCopilCuIdDat(sCo,idCopil);
    Writeln('Copilul ' , unC.numeC , ' a primit urmatoarele cadouri: ' );
  end;
  for i:=0 to sC.nE-1 do
  begin
    idJucarie:=sC.sirE[i].idJ;
    oJ:=getJucariaCuIdDat(sJ,idJucarie);
    Writeln('Jucaria ' , oJ.denuJ , ', dimensiune ' , oJ.dimJ , ', in anul ' ,
                    sC.sirE[i].an , '.');
  end;
end;

function existaJucarie(sJ: sirJucarii;  idJ: integer): boolean;
var gasit: boolean;
    i: integer;
begin
  gasit:=false;
  i:=0;
  while (i<sJ.nE)and not gasit do
  begin
    if sJ.sirE[i].idJ=idJ then
```

```
      gasit:= true;
    inc(i);
  end;
  existaJucarie := gasit;
end;

function totiCopiiiCarePrimescAceeasiJucarie(sC: sirCadouri;  idJ: integer;  sJ:
sirJucarii): sirCadouri;
var sCaAceeasiJucarie: sirCadouri;
    i : integer;
begin
  sCaAceeasiJucarie.nE:= 0;
  if not existaJucarie(sJ,idJ) then
    Writeln('Nu exista jucaria in lista de jucarii! ')
  else
  begin
    for i:=0 to sC.nE -1 do
      if sC.sirE[i].idJ=idJ then
      begin
        sCaAceeasiJucarie.sirE[sCaAceeasiJucarie.nE].idCa:= sC.sirE[i].idCa;
        sCaAceeasiJucarie.sirE[sCaAceeasiJucarie.nE].idCo:= sC.sirE[i].idCo;
        sCaAceeasiJucarie.sirE[sCaAceeasiJucarie.nE].idJ:= sC.sirE[i].idJ;
        sCaAceeasiJucarie.sirE[sCaAceeasiJucarie.nE].an:= sC.sirE[i].an;
        inc(sCaAceeasiJucarie.nE);
      end;
  end;
 totiCopiiiCarePrimescAceeasiJucarie := sCaAceeasiJucarie;
end;

procedure afisareCopiiPentruOJucarie(sC: sirCadouri;  sJ: sirJucarii;  sCo: sirCopii);
var  idJucarie: integer;
     oJ: jucarie;
     idCopil: integer;
     unC: copil;
     i : integer;
begin
  if sC.nE>=1 then begin
    idJucarie:=sC.sirE[0].idJ;
    oJ:=getJucariaCuIdDat(sJ,idJucarie);
    Writeln('Jucaria ' , oJ.denuJ , ' a fost primita de copiii: ' );
  end;
  for i :=0 to sC.nE -1 do
  begin
    idCopil := sC.sirE[i].idCo;
    unC := getCopilCuIdDat(sCo,idCopil);
    Writeln('Copilul ' , unC.numeC , ', adresa ' , unC.adrC , ', in anul ' ,
sC.sirE[i].an , '.');
  end;
end;

function listaCadourilorDinAnDat(sC: sirCadouri;  anDat: integer): sirCadouri;
var sCaAnDat: sirCadouri;
    i : integer;
begin
```

```
  sCaAnDat.nE:= 0;
  for i:=0 to Pred(sC.nE) do
    if sC.sirE[i].an=anDat then
    begin
      sCaAnDat.sirE[sCaAnDat.nE].idCa:= sC.sirE[i].idCa;
      sCaAnDat.sirE[sCaAnDat.nE].idCo:= sC.sirE[i].idCo;
      sCaAnDat.sirE[sCaAnDat.nE].idJ:= sC.sirE[i].idJ;
      sCaAnDat.sirE[sCaAnDat.nE].an:= sC.sirE[i].an;
      inc(sCaAnDat.nE);
    end;
  listaCadourilorDinAnDat := sCaAnDat;
end;

procedure afisareCadouriPentruAnDat(sC: sirCadouri;  sJ: sirJucarii;  sCo: sirCopii);
var idJucarie: integer;
    oJ: jucarie;
    idCopil: integer;
    unC: copil;
    i : integer;
begin
  for i:=0 to sC.nE-1 do
  begin
    idJucarie:=sC.sirE[i].idJ;
    oJ:=getJucariaCuIdDat(sJ,idJucarie);
    idCopil:=sC.sirE[i].idCo;
    unC:=getCopilCuIdDat(sCo,idCopil);
    Writeln('Jucaria ' , oJ.denuJ , ' cu dim ' , oJ.dimJ , ' oferita copilului ' ,
unC.numeC , ', adresa ' , unC.adrC , ', in anul ' , sC.sirE[i].an , '.');
  end;
end;


var
  mySirJucarii: sirJucarii;
  mySirCopii: sirCopii;
  mySirCadouri: sirCadouri;
  dim: real;
  resultatSirJucariiDim: sirJucarii;
  denumire: string[20];
  ceaMaiMicaDenumire: jucarie;
  copilSirCadouri: sirCadouri;
  idCopilDat: integer;
  jucarieSirCadouri: sirCadouri;
  idJucarieData: integer;
  cadouriAnDat: sirCadouri;
  anDat: integer;
begin
  citireDinFisier(mySirJucarii,mySirCopii,mySirCadouri);
  afisareJucarii(mySirJucarii);
  afisareCopii(mySirCopii);
  afisareCadouri(mySirCadouri);

  Write('Dati dimensiunea pentru cautare jucarii de dimensiune mai mare: ');
  Readln(dim);
```

```
  resultatSirJucariiDim:=listaJucariiDimeniuneMaiMare(mySirJucarii,dim);
  afisareJucarii(resultatSirJucariiDim);
  Readln;

  Write('Dati denumirea jucariei: ');
  Readln(denumire);
  ceaMaiMicaDenumire:=ceaMaiMicaJucarieDenumire(mySirJucarii,denumire);
  afiseazaOJucarie(ceaMaiMicaDenumire);
  Readln;

  {*Toate cadourile unui copil dat*}

  Write('Dati un id copil de determinat toate cadourile: ');
  Readln(idCopilDat);
  copilSirCadouri:= toateCadourilePentruUnCopilDat(mySirCadouri,idCopilDat,mySirCopii);
  Writeln('Cadourile pentru copilul cerut: ');
  afisareCadouriPentruUnCopil(copilSirCadouri,mySirJucarii,mySirCopii);
  Readln;

  {*Toti copiii care au primit acelasi cadou*}

  Write('Dati un id jucarie de determinat toti copiii: ');
  Readln(idJucarieData);
  jucarieSirCadouri:= totiCopiiiCarePrimescAceeasiJucarie(mySirCadouri,idJucarieData,
            mySirJucarii);
  Writeln('Copiii pentru jucaria data: ');
  afisareCopiiPentruOJucarie(jucarieSirCadouri,mySirJucarii,mySirCopii);
  Readln;

  {*Toate cadourile care au fost oferite intr-un an dat*}

  Write('Dati un an pentru care se doreste lista de jucarii/copii: ');
  Readln(anDat);
  cadouriAnDat:= listaCadourilorDinAnDat(mySirCadouri,anDat);
  Writeln('Cadourile din anul ' , anDat );
  afisareCadouriPentruAnDat(cadouriAnDat,mySirJucarii,mySirCopii);
  Readln;
end.
```

Problema 2. Emotii

Scrieti o aplicatie care gestioneaza emotional persoanele dintr-o incapare:

- **Emotie (expresie gura): fericit, trist, surprins, nervos**
- **Persoana care exprima o emotie**
- **Sir de persoane**

Si va gestiona o lista de persoane cu urmatoarele functionalitati:

a) Sa se determine (iterativ si recursiv) din fiecare tip de emotie numarul de persoane care exprima emotia (frecventa emotiei).
b) Sa se determine emotia predominanta din incapere (emotia cu frecventa cea mai mare).
c) Sa se determine toate subsirurile cu o proprietate data ("fericit" are la stanga si la dreapta lui emotie "trist") astfel incat distanta dintre proprietati sa fie mai mica decat o valoare data.

Alte functionalitati (Tema):

d) Sa se determine daca exista persoane vecine care au emotii distincte.

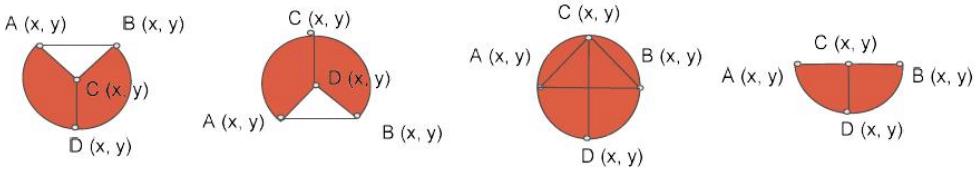
e) Sa se insereze intre oricare doua persoane triste, o persoana vesela (aceeasi persoana).

f) Sa se trimita la terapie persoanele nervoase (eliminare din sir a persoanele nervoase).

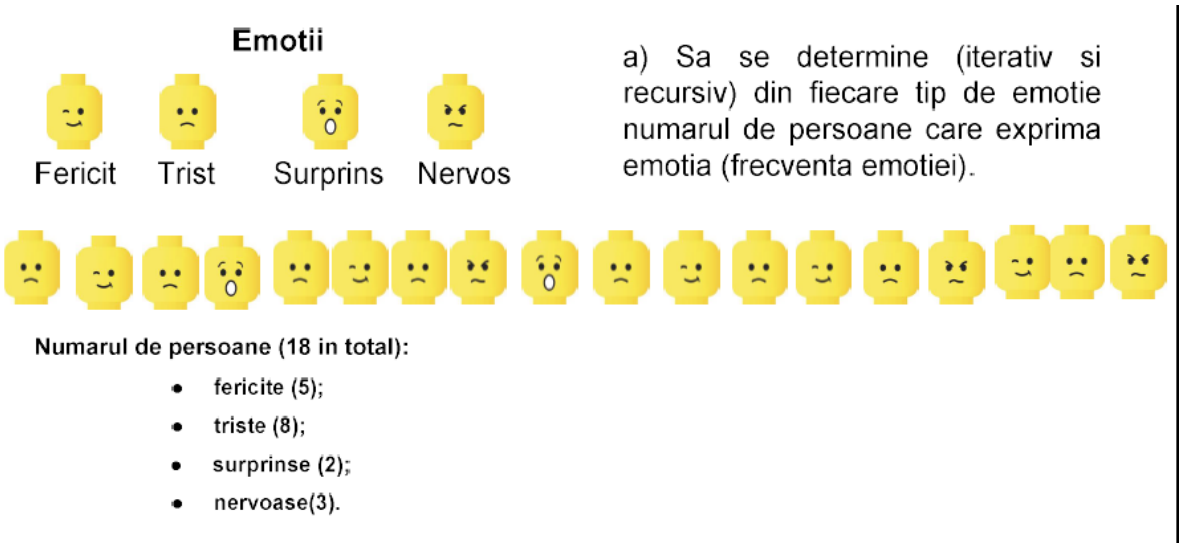
g) Sa se aranjeze persoanele astfel încât cele triste sa nu fie vecine (sau cele nervoase sa nu fie langa cele triste).

Analiza

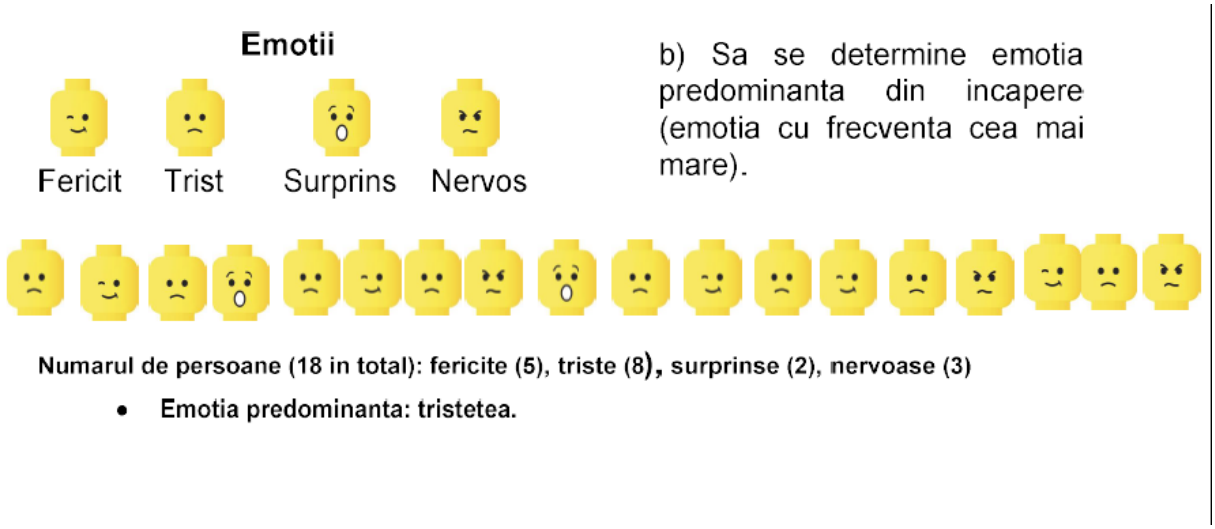
Tipuri de emotii



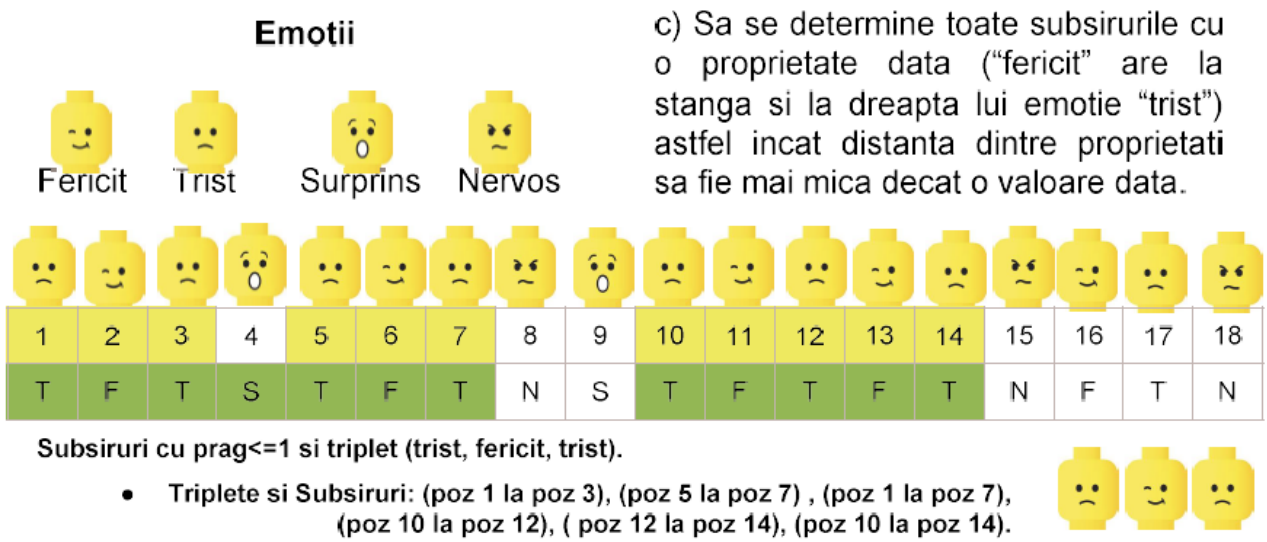
Exemplu pt cerinta a



Exemplu pentru cerinta b



Exemplu pentru cerinta c



Proiectare

A. Identificarea entitatilor:

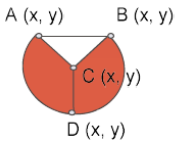
- **Punct:**
  - **x,y: intreg;**
- **Emotie**
  - **A, B, C, D: Punct;**
- **SirEmotii:**
  - **nE:intreg**
  - **sirE:Emotie[ ]**
- **PozSubsir**
  - **pS, pF:intreg**
- **SirPozSubsir**
  - **nE:intreg**
  - **sirPozSubsir: PozSubsir[ ]**

B. Descrierea operatiilor pentru entitati

- citireDinFisierEmotii
- afisareEmotiiPersoane
  - afisareOEmotie
- numarPersoaneFiecareEmotie
  - detEmotieOPersoana
  - emotieFericit
  - emotieSurprins
  - emotieNervos
  - emotieTrist
  - emotieDeBaza
- afisareSubsiruri
  - subsiruriProprietate
  - cautaTriplet
  - adaugaSubSirNou

C. Identificarea si specificarea subalgoritmilor

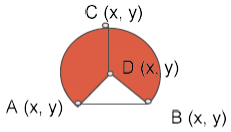
Functia **emotieDeBaza(e):**
Descriere: verifica daca emotia indica o stare valida
Date : o emotie e
Rezultate: true, daca e este o emotie valida, altfel false



Particularitate Emotie Fericit:

- **C.y < A.y**
- **D.y < C.y**

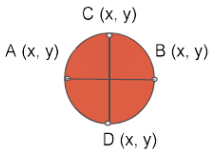
Functia **emotieFericit(e)**
Descriere: verifica daca o emotie indica starea “Fericit”
Date: o emotie e
Rezultate: true, daca (e este o emotie valida si are particularitatile “Fericit”), altfel false



Particularitate Emotie Trist:

- **C.y > A.y**
- **D.y > A.y**
- **D.y < C.y**

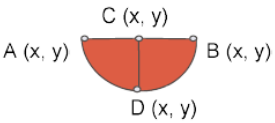
Functia **emotieTrist(e)**
Descriere: verifica daca o emotie indica starea “Trist”
Date: o emotie e
Rezultate: true, daca (e este o emotie valida si are particularitatile “Trist”), altfel false



Particularitate Emotie Surprins:

- **C.y > A.y**
- **D.y < C.y**

Functia **emotieSurprins(e)**
Descriere: verifica daca o emotie indica starea “Surprins”
Date: o emotie e
Rezultate: true, daca (e este o emotie valida si are particularitatile “Surprins”), altfel false



Particularitate Emotie Nervos:

- **C.y = A.y**
- **D.y < C.y**

Functia **emotieNervos(e)**
Descriere: verifica daca o emotie indica starea “Nervos”
Date: o emotie e
Rezultate: true, daca (e este o emotie valida si are particularitatile “Nervos”), altfel false

Subalgoritmul **citireDinFisierEmotii(sEmo)**
Descriere: citeste din fisier un sir de emotii
Date: -
Rezultate : sirul de emotii citit

Functia **numarPersoaneFiecareEmotie(sEmo, tipEmotieDeDeterminat)**
Descriere: : determina numarul de persoane din sir care au o emotie data
Date: un sir de emotii si emotia
Rezultate : numarul persoanelor cu emotia data

Subalgoritmul **determinaEmotiePredominanta(sEmo, maximPredominant, strEmotieDominanta)**
Descriere: `determina emotia predominanta din sirul de emotii`
Date: un sir de emotii sEmo,
Rezultate : frecventa emotiei predominante (maximPredominant) si denumirea acesteia (strEmotieDominanta este din { “fericit”, “trist” “surprins”, “nervos” }); se afiseaza si frecventa fiecarei emotii

Subalgoritmul **cautaTriplet(sEmo; poz; pS; pF)**
Descriere: determina o secventa de emotii incepsi cu pozitia data, care indeplineste proprietatea ceruta “fericit” intre doua persoane “triste”;
Date: sEmo este de tip sirEmotii, pozitie: intreg;
Rezultate: pS este pozitia de inceput, pF este pozitia de sfarsit a unei secvente din emotii care indeplineste proprietatea ceruta (“fericit” intre doua persoane “triste”)

Subalgoritmul **adaugaSubSirNou(mySirPozSubsir; pSNoua; pFNoua);**
Descriere: retine pozitia de inceput si de sfarsit a unui nou subsir care indeplineste proprietatea ceruta “fericit” intre doua persoane “triste”;
Date: mySirPozSubsir ∈ sirPozSubSir, pS, pF: intreg;
Rezultate: mySirPozSubsir’ = mySirPozSubsir + (pS, pF)

Subalgoritmul **subsiruriProprietate(sEmo; prag; mySirPozSubsir);**
Descriere: determina subsirurile care indeplinesc proprietatea “fericit” intre doua persoane “triste” din sirul cu emotii, pentru un prag dat; prag = indica un numar maxim de pozitii acceptat intre doua secvente de emotii determinate, astfel incat acestea sa formeze un singur subsir; in caz contrar secventele formeaza subsiruri distincte;
Date: un sir de emotii sEmo, prag - intreg, subSiruri ∈ SirPozSubSir;
Rezultate: mySirPozSubsir contine toate subsirurile de emotii care respecta proprietatea data, cu pragul maxim dat;

Subalgoritmul **afisareSubsiruri(mySirPozSubsir)**
Descriere: afiseaza toate subsirurile care indeplinesc proprietatea ceruta “fericit” intre doua persoane “triste”, din sirul cu emotii, pentru un prag dat;
Date: emotii este de tip SirEmotii, mySirPozSubsir este de tip sirPozSubSir
Rezultate : se afiseaza subsirurile de emotii care indeplinesc proprietatea si pragul dat

Implementare

```
program EmotiiPersoanePascal;

type

punct = record
  x : real;
  y: real;
end;

emotie = record
  a: punct;
  b: punct;
  c: punct;
  d: punct;
end;

sirEmotii = record
  nE: integer;
  sirE: array [0..99] of emotie;
end;

pozSubsir = record
  pS: integer;
  pF: integer;
end;

sirPozSubsir = record
  nE: integer;
  sirSubsir: array [0..99] of pozSubsir;
end;

procedure citireDinFisierEmotii(var sEmo: sirEmotii);
var
  myfile: TEXT;
  nP : integer;
  nErr : integer;
  i : integer;
begin
  Assign(myfile, 'EmotiiPersoane.txt');
  Reset(myfile);

  ReadLn(myfile, nP);
  sEmo.nE:= 0;
  for i:=0 to nP -1 do begin
    Read(myfile, sEmo.sirE[sEmo.nE].a.x);
    Read(myfile, sEmo.sirE[sEmo.nE].a.y);
    Read(myfile, sEmo.sirE[sEmo.nE].b.x);
    Read(myfile, sEmo.sirE[sEmo.nE].b.y);
    Read(myfile, sEmo.sirE[sEmo.nE].c.x);
    Read(myfile, sEmo.sirE[sEmo.nE].c.y);
    Read(myfile, sEmo.sirE[sEmo.nE].d.x);
    Readln(myfile, sEmo.sirE[sEmo.nE].d.y);
    inc(sEmo.nE);
  end;
  Close(myfile);
end;
```

```
function numarPersoaneFiecareEmotie(sEmo: sirEmotii; const tipEmotieDeDeterminat :
string): integer;
var
  nrPersoaneOEmotie : integer;
  tipEmotie : string;
  i : integer;
begin
  nrPersoaneOEmotie:=0;
  for i:=0 to sEmo.nE-1 do
  begin
    tipEmotie := detEmotieOPersoana(sEmo.sirE[i]);
    if tipEmotie = tipEmotieDeDeterminat then
    begin
      inc(nrPersoaneOEmotie);
    end;
  end;
  numarPersoaneFiecareEmotie := nrPersoaneOEmotie;
end;

function numarPersoaneFiecareEmotieRecursiv(sEmo: sirEmotii;  const
tipEmotieDeDeterminat : string; i : integer): integer;
var
  tipEmotie: string;
begin
  if i=-1 then
  begin
    numarPersoaneFiecareEmotieRecursiv := 0;
  end else begin
    tipEmotie := detEmotieOPersoana(sEmo.sirE[i]);
    if tipEmotie = tipEmotieDeDeterminat then
    begin
      numarPersoaneFiecareEmotieRecursiv := 1 +
numarPersoaneFiecareEmotieRecursiv(sEmo,tipEmotieDeDeterminat,i-1);
    end else begin
      numarPersoaneFiecareEmotieRecursiv :=
numarPersoaneFiecareEmotieRecursiv(sEmo,tipEmotieDeDeterminat,i-1);
    end;
  end;
end;

procedure determinaEmotiePredominanta(sEmo: sirEmotii;  var maximPredominant: integer;
var strEmotieDominanta : string);
var
  nrFericite: integer;
  nrFericiteRec: integer;
  i: integer;
  nrTriste: integer;
  nrSurprinse: integer;
  nrNervoase: integer;

begin
  nrFericite := numarPersoaneFiecareEmotie(sEmo,'fericit');
  Writeln('Numar de persoane fericite: ',nrFericite);
  ReadLn;

  (*RECURSIV *)
  i := sEmo.nE;
  nrFericiteRec := numarPersoaneFiecareEmotieRecursiv(sEmo,'fericit',i-1);
  Writeln('Numar de persoane fericite (***RECURSIV***): ', nrFericiteRec);
  ReadLn;
```

```
  nrTriste:= numarPersoaneFiecareEmotie(sEmo,'trist');
  Writeln('Numar de persoane triste: ', nrTriste);
  ReadLn;

  nrSurprinse:= numarPersoaneFiecareEmotie(sEmo,'surprins');
  Writeln('Numar de persoane surprins: ', nrSurprinse);
  ReadLn;

  nrNervoase:= numarPersoaneFiecareEmotie(sEmo,'nervos');
  Writeln('Numar de persoane nevoase: ', nrNervoase);
  ReadLn;

  strEmotieDominanta:= 'fericit';
  if maximPredominant<nrTriste then
  begin
    maximPredominant:= nrTriste;
    strEmotieDominanta:= 'trist';
  end else if maximPredominant<nrSurprinse then
  begin
    maximPredominant:= nrSurprinse;
    strEmotieDominanta:= 'surprins';
  end else if maximPredominant<nrNervoase then
  begin
    maximPredominant:= nrNervoase;
    strEmotieDominanta:= 'nervos';
  end;
end;

function emotieTrist(e: emotie): Boolean;
begin
  emotieTrist:=false;
  if (emotieDeBaza(e))and(e.c.y>e.a.y)and(e.d.y>e.a.y)and(e.d.y<e.c.y) then
    emotieTrist:= true;
end;

function emotieFericit(e: emotie): Boolean;
begin
  emotieFericit:=false;
  if (emotieDeBaza(e))and(e.c.y<e.a.y)and(e.d.y<e.c.y) then
    emotieFericit:= true;
end;

procedure cautaTriplet(sEmo: sirEmotii;  poz: integer;  var pS: integer;  var pF:
integer);
var
 gasitT: Boolean;
begin
  pS:= -1;
  pF:= -1; (*cod de eroare cand nu gasesc triplet*)
  gasitT:=false;
  while (not gasitT) and (poz < sEmo.nE) do begin
    while (poz<sEmo.nE) and not emotieTrist(sEmo.sirE[poz]) do
      inc(poz); (*daca am gasit pozitie trist si mai am cel putin 2 pozitii de
                  verificat*)
      if (poz<sEmo.nE)and((poz+2)<sEmo.nE) then
      begin
        if emotieFericit(sEmo.sirE[poz+1])and emotieTrist(sEmo.sirE[poz+2]) then
        begin
          pS:= poz;
```

```
          pF:= poz+2;
          gasitT:= true;
        end else
         inc(poz);
      end else
       inc(poz);
  end; (*while gasitT*)
end;

procedure adaugaSubSirNou(var mySirPozSubsir: sirPozSubsir;  pSNoua: integer;  pFNoua:
integer);
begin
  mySirPozSubsir.sirSubsir[mySirPozSubsir.nE].pS:= pSNoua;
  mySirPozSubsir.sirSubsir[mySirPozSubsir.nE].pF:= pFNoua;
  inc(mySirPozSubsir.nE);
end;

function subsiruriProprietate(sEmo: sirEmotii;  prag: integer;  var mySirPozSubsir:
sirPozSubsir): sirPozSubsir;
var
  pS: integer;
  pF: integer;
  pS2: integer;
  pF2: integer;
  existaTriplet2: Boolean;
  i: integer;
begin
  mySirPozSubsir.nE:= 0;
  pS:=-1;
  pF:=-1;
  pS2:=-1;
  pF2:=-1;
  existaTriplet2:=false;
  i:=0;
  cautaTriplet(sEmo,i,pS,pF);
  adaugaSubSirNou(mySirPozSubsir,pS,pF);
  if (pS<>-1) and (pF<>-1) then begin
    cautaTriplet(sEmo,pF,pS2,pF2);
    adaugaSubSirNou(mySirPozSubsir,pS2,pF2);
    (*caut incepand de la pF *)
    if (pS2<>-1) and (pF2<>-1) then existaTriplet2:= true;
    while (existaTriplet2) do
    begin
      if (pS2-pF-1)<=prag then
      begin  (*distanta <=prag ==> subsecventa mai lunga devine cea contopita din secv1
                    si secv2*)
        pS:= pS;
        pF:= pF2;
        adaugaSubSirNou(mySirPozSubsir,pS,pF);
      end else begin
        (*prag prea mare ==> prima subsecenta devine a doua subsecventa *)
        (* si continuu cautare de dupa secventa 2*)
        pS:= pS2;
        pF:= pF2;
        adaugaSubSirNou(mySirPozSubsir,pS,pF);
      end;
      cautaTriplet(sEmo,pF,pS2,pF2);
      (*caut incepand de la pF  (poate fi 'trist')   *)
      if (pS2<>-1)and(pF2<>-1) then
      begin
```

```
        existaTriplet2:= true;
        adaugaSubSirNou(mySirPozSubsir,pS2,pF2);
      end else
        existaTriplet2:= false;
    end;
  end;

  subsiruriProprietate := mySirPozSubsir;
end;

procedure afisareSubsiruri(mySirPozSubsir: sirPozSubsir);
var
  j : integer;
begin
  for j:=0 to Pred(mySirPozSubsir.nE) do
    Writeln(mySirPozSubsir.sirSubsir[j].pS, ' , ', mySirPozSubsir.sirSubsir[j].pF);
end;


var
  mySirEmo: sirEmotii;
  maximDominat: integer;
  strEmotieDominanta: string;
  prag: integer;
  mySirPozSubsir: sirPozSubsir;

begin
  citireDinFisierEmotii(mySirEmo);
  maximDominat:=-1;
  determinaEmotiePredominanta(mySirEmo,maximDominat,strEmotieDominanta);
  Writeln('Emotia predominanta este:', strEmotieDominanta, ' cu numar maxim ',
                  maximDominat);
  prag:=1;
  subsiruriProprietate(mySirEmo,prag,mySirPozSubsir);
  afisareSubsiruri(mySirPozSubsir);
  readln;
end.
```

Exemple

22 1 6 3 6 2 8 2 7 1 7 3 7 2 6 2 5 1 6 3 6 2 8 2 7 1 6 3 6 2 7 2 5 1 6 3 6 2 8 2 7 1 7 3 7 2 6 2 5 1 6 3 6 2 8 2 7 1 7 3 7 2 6 2 5 1 6 3 6 2 8 2 7 1 7 3 7 2 6 2 5 1 6 3 6 2 7 2 5 1 6 3 6 2 8 2 7 1 6 3 6 2 8 2 7 1 7 3 7 2 6 2 5 1 6 3 6 2 8 2 7 1 6 3 6 2 6 2 5 1 6 3 6 2 8 2 7 1 7 3 7 2 6 2 5 1 6 3 6 2 8 2 7 1 6 3 6 2 8 2 7 1 7 3 7 2 6 2 5 1 6 3 6 2 8 2 7	a) trist: 12 fericit: 7 surprins: 2 nervos: 1 b) emotia predominanta: trist frecventa = 12 c) 2 subsiruri 1 6 3 6 2 8 2 7 1 7 3 7 2 6 2 5 1 6 3 6 2 8 2 7 1 6 3 6 2 7 2 5 1 6 3 6 2 8 2 7 1 7 3 7 2 6 2 5 1 6 3 6 2 8 2 7 1 7 3 7 2 6 2 5 1 6 3 6 2 8 2 7 Si 1 6 3 6 2 8 2 7 1 7 3 7 2 6 2 5 1 6 3 6 2 8 2 7 1 6 3 6 2 6 2 5 1 6 3 6 2 8 2 7 1 7 3 7 2 6 2 5 1 6 3 6 2 8 2 7 1 6 3 6 2 8 2 7 1 7 3 7 2 6 2 5 1 6 3 6 2 8 2 7