29 februarie 2020

Conf. Dr. Andreea Vescan
Lect. Dr. Radu Găceanu

Probleme tip grilă

1. Care dintre urmatoarele secvente determina suma elementelor pare dintr-un sir x cu n numere naturale?

Varianta Pascal

a)
```
S:=0
For i:=n+1 downto 2 do
      If not odd(x[i-1]) then s:=s+x[i-1]
```

b)
```
S:=0
For i:=1 to n+1 do
      If odd(x[i]) then s:=s+x[i]
```

c)
```
S:=0
For i:=1 to n+1 do
      If not odd(x[i]) then s:=s+x[i]
```

d)
```
S:=x[1]
For i:=2 to n do
      If x[i] mod 2=0 then s:=s+x[i]
```

Varianta C

a)
```
S=0;
for (i=n+1;i>=2;i--)
      If (x[i-1]%2==0)    S+=x[i-1];
```

b)
```
S=0;
for (i=1;i<=n+1;i++)
      If (x[i]%2==0)      S+=x[i];
```

c)
```
S=0;
for (i=1;i<=n+1;i++)
      If (!(x[i]%2))      S+=x[i];
```

d)
```
S=x[1];
for (i=2;i<=n;i++)
      If (x[i]%2==0)      S+=x[i];
```

2. Ce realizează următorul program?

Varianta Pascal
```
e:=0;
for i:=1 to n do
begin
      a=i;
```

29 februarie 2020 Conf. Dr. Andreea Vescan
Lect. Dr. Radu Găceanu

```
      if x[i]<a then e:=e+1;
end;
writeln(e);

Varianta C
e=0;
for(i=1;i<=n;i++){
      a=i;
      if (x[i]<a) e++;
}
printf("%d \n",e);
```

a) Afiseaza numarul de aparitii a elementului maxim din sir.
b) Afiseaza pozitia elementului maxim din sir.
c) Afiseaza numarul de numere din sir cu proprietatea ca elementul de pe pozitie este mai mic decat pozitia.
d) Afiseaza numarul de numere din sir cu proprietatea ca elementele sunt mai mici decat numarul a dat.

3. Care este rezultatul urmatorului program pentru n= 7 si x = [3,4,2,1,4,5,6]?

```
Varianta Pascal
e:=0;
for i:=1 to n do
begin
      if (x[i]<i) and (x[i] mod 2 ==0) then e:=e+1;
end;
writeln(e);

Varianta C
e=0;
for(i=1;i<=n;i++){
      if ((x[i]<i)&&(x[i]%2==0)) e++;
}
printf("%d \n",e);
```

a) **3** b) 2 c) 0 d) 4

29 februarie 2020

Conf. Dr. Andreea Vescan
Lect. Dr. Radu Găceanu

Problema 1

Figuri geometrice

Enunț

Se citeste un sir de n elemente, fiecare element fiind o figura geometrica (cer, patrat, romb).

Se cere:

a) Sa se determine (iterativ si recursiv) din fiecare tip de figura geometrica, numarul de astfel de figuri.
b) Sa se determine figura predominanta.
c) Sa se determine: toate subsecventele cu proprietatea (cerc, patrat, romb).
d) Sa se elimine toate tripletele identificate la punctul c).
e) **Problema**: Sa se determine daca dintr-o configuratie data de figuri geometrice, aplicand operatiile de la c) si d) putem ajunge la sirul vid.

Tema (sau de rezolvat in clasa daca mai ramane timp):

1) Sa se determine daca sunt pozitii vecine cu aceeasi figura data si cate sunt.
2) Sa se insereze intre oricare doua cercuri vecine, un patrat.

Exemplu
Codificare (1=cerc, 2=patrat, 3=romb).

nF=5 [1,1,2,3,1]

a) 3 cercuri, 1 patrat, 1 romb
b) Figura predominanta: cerc
c) Subsecventa [1,2,3]
d) Eliminare [1,2,3]➔[1,1]

nF=15 [1,1,2,3,1,2,3,2,1,2,3,3,1,2,3]

a) 5 cercuri, 5 patrat, 5 romb
b) Figura predominanta: cerc
c) Subsecventa [1,2,3] pozitia 2, pozitia 5,pozitia 9, pozitia 13
d) Eliminare [1,2,3]➔[1,2,3]➔[]

nF=15 [1,1,2,3,1,2,3,2,1,2,3,2,1,2,3]

a) 5 cercuri, 6 patrat, 4 romb
b) Figura predominanta: patrat
c) Subsecventa [1,2,3] pozitia 2, pozitia 5,pozitia 9, pozitia 13

29 februarie 2020 Conf. Dr. Andreea Vescan
Lect. Dr. Radu Găceanu

d) Eliminare [1,2,3]➔[1,2,2]

Analiza

Pentru a putea reutiliza metodele de la cerintele c) si d) pentru cerinta e), se proiecteaza subalgoritmi pentru:

- Eliminarea unei pozitii date din sir
- Determinarea primei subsecvente de 3 elemente incepand cu o pozitie data.

Specificarea functiilor

Subalgoritmul **citireFiguri (vF,nF)**:
Descriere: Citeste numarul de figure si figurile
Date:
Rezultate: nF – numarul de figuri, sF – vectorul de figuri

Subalgoritmul **afisareFiguri(vF,nF)**:
Descriere: Se afiseaza figurile din vestorul sF
Date: nF – numarul de figuri, vF – vectorul de figure
Rezultate:

Functia **numarDeFiguriDeUnFel(vF, nF, tF)**
Descriere: Se afiseaza figurile din vestorul sF
Date:
Rezultate: nF – numarul de figuri, vF – vectorul de figuri

Functia **figuraPredominanta**(sF, nF)
Descriere: Se determina figura predominanta
Date: sF, nF
Rezultate: nP – figura predominanta

Subalgoritmul **procedure cautaTriplet(sF,nF,pos, pS,pF)**
Descriere: Se determina primul triplet incepand cu pozitia pos
Date: sF, nF, pos
Rezultate: pS, pF – pozitiile Start si Final ale tripletului gasit.

Subalgoritmul cautaToateTripletele(sF, nF, sPS, sPF, nSF)
Descriere: Se determina toate tripletele.
Date: sF, nF
Rezultate: sPS- pozitiile de Start ale tripletelor gasite, sPF-pozitiile Final ale tripletelor gasite, nSF – numarul de triplete cu pozitiileStart si Final

Subalgoritmul **eliminarePozitieData**(sF, nF,p);
Descriere: Se elimina din sirul de elemente, elementul de pe pozitia p
Date: sF, nF, p
Rezultate: sF,nF – sirul modificat dupa eliminarea elementului de pe pozitia p

29 februarie 2020 Conf. Dr. Andreea Vescan
Lect. Dr. Radu Găceanu

Subalgoritmul **eliminareTriplet**(sF,nF,ps,pf);
Descriere: Se elimina din sirul de elemente, elementele intre pozitiile ps si pf
Date: sF, nF, ps,pf
Rezultate: sF,nF – sirul modificat dupa eliminarea elementelor intre pozitiile ps, pf

Subalgoritmul **jocEliminaTripletePanaLaSirVid**_Versiune1(sF,nF);
Descriere: Se elimina din sirul de elemente, tripletele, inclusive cele formate dupa eliminarea unora intermediare
Date: sF, nF
Rezultate: sF,nF – sirul modificat dupa eliminarea tripletelor, inclusive a celor formate dupa eliminari intermediare

Implementare
Varianta Pascal

```
program Hello;

type  sirE= array [0..99] of integer;

procedure citireFiguri(var sF: sirE; var nF:integer);
var
i : integer;
begin
Writeln('Dati numarul de figuri:');
Readln(nF);
writeln('numarul de figuri citite este=',nF);
writeln('dati figurile');
for i:=1 to nF do
  begin
    writeln('figura sF[',i,']=');
   Readln(sF[i]);
  end;
end;
procedure afisareFiguri(sF: sirE; nF:integer);
var
i : integer;
begin
    writeln;
   for i:=1 to nF do
   begin
      write('sF[',i,']=');
```

29 februarie 2020 Conf. Dr. Andreea Vescan
Lect. Dr. Radu Găceanu

```
      writeln(sF[i]);
   end;
end;

function numarDeFiguriDeUnFel(sF:sirE; nF:integer; tipF:integer):integer;
var i:integer;
   nFTip:integer;
begin

    nFTip:=0;
    for i:=1 to nF do
       if (sF[i]=tipF) then
          nFTip := nFTip+1;
    numarDeFiguriDeUnFel := nFTip;
end;

function numarDeFiguriDeUnFelRecursiv(sF:sirE; nF:integer; tipF:integer; i:integer):integer;
begin
    if (i=0) then
      numarDeFiguriDeUnFelRecursiv:=0
    else
       if (sF[i]=tipF) then
          numarDeFiguriDeUnFelRecursiv := numarDeFiguriDeUnFelRecursiv(sF, nF, tipF, i-1) +1
       else
          numarDeFiguriDeUnFelRecursiv := numarDeFiguriDeUnFelRecursiv(sF, nF, tipF, i-1);
end;
function figuraPredominanta(sF:sirE; nF:integer):integer;
var nC,nP,nR:integer;
   nPredominant, rezultat:integer;
begin

    nC := numarDeFiguriDeUnFel(sF, nF, 1);
    nP := numarDeFiguriDeUnFel(sF, nF, 2);
    nR := numarDeFiguriDeUnFel(sF, nF, 3);
    nPredominant := nC;
    rezultat :=1;
    if (nPredominant <nP) then
      begin
            nPredominant := nP;
            rezultat:=2
```

29 februarie 2020 Conf. Dr. Andreea Vescan
Lect. Dr. Radu Găceanu

```
      end;
      if (nPredominant <nR) then
        begin
           nPredominant := nR;
           rezultat:=3;
        end;
    figuraPredominanta :=rezultat;
end;

procedure cautaTriplet(sF:sirE; nF: integer; 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 < nF) do
   begin
       while ((poz<nF) and (not (sF[poz]=1))) do
           inc(poz);
       (*daca am gasit pozitie cerc si mai am cel putin 2 pozitii de verificat*)
       if (poz<nF)and((poz+2)<=nF) then
       begin
            if (sF[poz+1]=2)and (sF[poz+2]=3) then
              begin
                 pS:= poz;
                 pF:= poz+2;
                 gasitT:= true;
                // writeln('gasit tripletul ps=',ps,' si pf=',pf);
              end
             else
                 inc(poz);
       end
       else
            inc(poz);
   end; (*while gasitT*)
end;

procedure cautaToateTripletele(sF:sirE; nF: integer; var sPS, sPF:sirE; var nSF:integer);
var pos, ps,pf: integer;
```

29 februarie 2020 Conf. Dr. Andreea Vescan
Lect. Dr. Radu Găceanu

```
begin
   ps:=-1;pf:=-1;
   pos:=1;
   nSF:=0;
   while (pos<=(nF-2)) do
   begin
      cautaTriplet(sF,nF, pos, ps,pf);
     //  writeln('un triplet intre pozitiile ps=',ps, ' si pf=', pf);
      nSF:=nSF+1;
      sPS[nSF]:=ps;
      sPF[nSF]:=pf;
      //pos := pf+1;
      pos:=pos+1
   end;
end;
procedure eliminarePozitieData(var sF:sirE; var nF:integer; p:integer);
var i:integer;
begin
   for i:=p to nF-1 do
      sF[i]:=sF[i+1];
   nF := nF-1;
end;

procedure eliminareTriplet(var sF:sirE; var nF:integer; ps,pf:integer);
var i: integer;
begin
   for i:= 1 to (pf-ps +1) do
      eliminarePozitieData(sF,nF, ps);
end;
procedure jocEliminaTripletePanaLaSirVid_Versiune1(var sF:sirE;var nF:integer);
var existaTriplete:boolean;
   p,ps,pf:integer;
begin
   existaTriplete := false;
   p:=1;
   cautaTriplet(sF,nF,p,ps,pf);
   if (ps >-1) then
     existaTriplete := true;
   while (existaTriplete) and (p<=(nF-2))do
   begin
```

29 februarie 2020 Conf. Dr. Andreea Vescan
Lect. Dr. Radu Găceanu

```
        cautaTriplet(sF,nF,p,ps,pf);
        if (ps>-1) then
          begin
             existaTriplete:=true;
             eliminareTriplet(sF,nF,ps,pf);
             p:=ps;
          end
        else
            p:=p+1;
     if ((p>(nF-2)) and (nF>0)) then
       p:=1;
    end;
end;

var sirF:sirE;
    nF:integer;
    tipFiguraDeNumarat:integer;
    nrDefiguriNumarate:integer;
    p,ps,pf:integer;
begin
  citireFiguri(sirF,nF);
  afisareFiguri(sirF,nF);

  tipFiguraDeNumarat:=1;
  nrDefiguriNumarate :=  numarDeFiguriDeUnFel(sirF, nF,tipFiguraDeNumarat);
  writeln('Numar e 1=', nrDefiguriNumarate);

  tipFiguraDeNumarat:=1;
  nrDefiguriNumarate := numarDeFiguriDeUnFelRecursiv(sirF, nF,tipFiguraDeNumarat,nF);
  writeln('RECURSIV Numar e 1=', nrDefiguriNumarate);

  writeln('Figura predominanta este:');
  writeln(figuraPredominanta(sirF,nF));

 // p:=4;
  // cautaTriplet(sirF, nF, p,ps,pf);
//  cautaToateTripletele(sirF,nF);

  jocEliminaTripletePanaLaSirVid_Versiune1(sirF, nF);
  writeln('Sirul ramas dupa eliminare triplete:');
```

29 februarie 2020

Conf. Dr. Andreea Vescan
Lect. Dr. Radu Găceanu

```
 afisareFiguri(sirF,nF);

 readln;
end.
```

Varianta C++

```
#include <iostream>


void citireFiguri(int sF[100], int &nF){
std::cout<<"Dati numarul de figuri";
std::cin >> nF;
std::cout<<"numarul de figuri citite este="<<nF;
std::cout<<"dati figurile";
for (int i = 1; i <= nF; i++)
{
	std::cout<<"figura sF["<< i<<"]=";
	std::cin>>sF[i];
}
}
void afisareFiguri(int sF[100], int nF) {

	for (int i=1; i <= nF; i++) {
		std::cout <<"\n"<< "sF[" << i << "]=";
		std::cout << sF[i];
	}
}

int numarDeFiguriDeUnFel(int sF[100], int nF, int tipF) {
	int nFTip = 0;
	for (int i=1;i<=nF;i++)
		if (sF[i] == tipF)
			nFTip = nFTip + 1;
    return nFTip;
}

int numarDeFiguriDeUnFelRecursiv(int sF[100], int nF, int tipF, int i){

	if (i == 0)
		return 0;
	else
		if (sF[i] == tipF)
			return numarDeFiguriDeUnFelRecursiv(sF, nF, tipF, i - 1) + 1;
		else
			return numarDeFiguriDeUnFelRecursiv(sF, nF, tipF, i - 1);
}

int figuraPredominanta(int sF[100], int nF) {
```

29 februarie 2020

Conf. Dr. Andreea Vescan
Lect. Dr. Radu Găceanu

```
	int nC = numarDeFiguriDeUnFel(sF, nF, 1);
	int nP = numarDeFiguriDeUnFel(sF, nF, 2);
	int nR = numarDeFiguriDeUnFel(sF, nF, 3);
	int nPredominant = nC;
	int rezultat = 1;
	if (nPredominant < nP)
	{
		nPredominant = nP;
		rezultat = 2;
	}

	if (nPredominant < nR)
	{
		nPredominant = nR;
		rezultat = 3;
	}
	return rezultat;
}
void cautaTriplet(int sF[100], int nF, int poz, int &pS, int &pF) {
	bool gasitT;
	pS = -1;
	pF= -1; //(*cod de eroare cand nu gasesc triplet*)
	gasitT = false;
	while ((not gasitT) && (poz < nF)){
		while ((poz < nF) && (not (sF[poz] = 1)))
			poz = poz + 1;
		//(*daca am gasit pozitie cerc si mai am cel putin 2 pozitii de verificat*)
		if ((poz < nF) && ((poz + 2) <= nF)){
			if ((sF[poz + 1] == 2) && (sF[poz + 2] == 3)){
				pS = poz;
				pF = poz + 2;
				gasitT = true;
			}
			else
				poz = poz + 1;
		}
		else
			poz = poz + 1;
	}// (*while gasitT*)
}



void cautaToateTripletele(int sF[100], int nF, int sPS[100], int sPF[100], int &nSF){
	int ps = -1; int pf = -1;
	int pos = 1;
	nSF = 0;
	while (pos <=(nF - 2)) {
		cautaTriplet(sF, nF, pos, ps, pf);
		nSF = nSF + 1;
		sPS[nSF] = ps;
		sPF[nSF] = pf;
```

29 februarie 2020

Conf. Dr. Andreea Vescan
Lect. Dr. Radu Găceanu

```
			pos = pf+1;
		}

}
void eliminarePozitieData(int sF[100], int &nF,int p) {

		for (int i = p;i <= (nF - 1); i++)
			sF[i] = sF[i + 1];
		nF = nF - 1;
}

void eliminareTriplet(int sF[100], int &nF, int ps, int pf) {
		for (int i = 1; i <= (pf - ps + 1); i++)
			eliminarePozitieData(sF, nF, ps);
}

void jocEliminaTripletePanaLaSirVid_Versiune1(int sF[100], int &nF) {
		bool existaTriplete;

		existaTriplete = false;
		int p = 1;
		int ps, pf;
		cautaTriplet(sF, nF, p, ps, pf);
		if (ps > -1)
			existaTriplete = true;
		while ((existaTriplete) and (p <= (nF - 2))) {
			cautaTriplet(sF, nF, p, ps, pf);
			if (ps > -1) {
				existaTriplete = true;
				eliminareTriplet(sF, nF, ps, pf);
				p = ps;
			}
			else
				p = p + 1;
			if ((p > (nF - 2)) && (nF > 0))
				p = 1;
		}

}



int main(){

		int sirF[100];
		int nF, tipFiguraDeNumarat, nrDefiguriNumarate, p=-1;


		citireFiguri(sirF, nF);
		std::cout << "\n";
		afisareFiguri(sirF, nF);

		tipFiguraDeNumarat = 1;
```

29 februarie 2020

Conf. Dr. Andreea Vescan
Lect. Dr. Radu Găceanu

```
	nrDefiguriNumarate = numarDeFiguriDeUnFel(sirF, nF, tipFiguraDeNumarat);
	std::cout << "\n"<< "Numar e 1=" << nrDefiguriNumarate<< "\n";

	tipFiguraDeNumarat = 1;
	nrDefiguriNumarate = numarDeFiguriDeUnFelRecursiv(sirF, nF, tipFiguraDeNumarat,
nF);
	std::cout<<"\n"<<"RECURSIV Numar e 1="<<nrDefiguriNumarate<< "\n";

	std::cout << "\n"<<"Figura predominanta este:";
	int figPred = figuraPredominanta(sirF, nF);
	std::cout << figPred << "\n";


	 //p=1;
	// int pss=-1, pff=-1;
	// std::cout << "\nTriplet cautat incepand cu pozitia poz=:" << p << "\n";
	// cautaTriplet(sirF, nF, p,pss,pff);
	 //std::cout << "ps=:"<<pss<<" si pf="<<pff << "\n";

	 int sPS[100]; int sPF[100]; int nSF = -1;
	 cautaToateTripletele(sirF,nF,sPS,sPF,nSF);

	jocEliminaTripletePanaLaSirVid_Versiune1(sirF, nF);
	std::cout<<"\n"<<"Sirul ramas dupa eliminare triplete:";
	afisareFiguri(sirF, nF);



}
```

29 februarie 2020

Conf. Dr. Andreea Vescan
Lect. Dr. Radu Găceanu

Problema 2

Cadouri de la Iepurasul de Pasti

Enunț

Iepurasul de Pasti doreste sa distribuie cadouri copiilor.
Intr-un dosar are fise legate de activitatea fiecarui copil, informatii pe baza carora ajutoarele iepurasului au construit relatii de forma: copilul "x" a fost 'mai cuminte' decat copilul "y" in anul care a trecut.
Ajutati-l pe iepuras sa distribuie cadouri tuturor copiilor astfel incat orice copil 'mai cuminte' sa fie vizitat de iepuras inaintea copiilor despre care se stie ca au fost mai putin cuminti decat el.

Sa se scrie o functie care primeste ca parametru un tablou unidimensional de perechi <x, y> (cu seminficatia x 'mai cuminte' decat y; x, y de tip string) si returneaza un tablou unidimensional cu elemente de tip string (numele tuturor copiilor) astfel incat relatia 'mai cuminte' este respectata (orice copil 'mai cuminte' se afla in tablou inaintea oricarui copil mai putin cuminte decat el).

Fiind vorba despre foarte multi copii, s-ar dori o solutie in timp liniar (***).

(***) Observatii:
1. Cerinta nu se aplica pentru partea de codificare a datelor (din motive de timp etc - discutat la consultatii)
2. Se presupune ca numarul de relatii e cel mult liniar in raport cu numarul de copii.
---
ex:
<Ionel, Gigel>
<Maria, Ionel>
<Maria, Gigel>
<Ana, Gigel>
solutie1: [Maria, Ionel, Ana, Gigel]
solutie2: [Ana, Maria, Ionel, Gigel]
etc.

29 februarie 2020

Conf. Dr. Andreea Vescan
Lect. Dr. Radu Găceanu

```
#include <iostream>
#define DIM 10
using namespace std;
struct Per {
	string x;
	string y;
	explicit Per(string x = "", string y = "") : x(std::move(x)), y(std::move(y)) {}
};
struct TabPer {
	int n = 0;
	Per elem[DIM];
};
struct TabInt {
	int n = 0;
	int elem[DIM];
};
struct TabTabInt {
	int n = 0;
	TabInt elem[DIM];
};
struct TabStr {
	int n = 0;
	string elem[DIM];
};
TabPer citestePerechi() {
	string data[][DIM] = {{"ionel", "gigel"},
	                      {"maria", "ionel"},
	                      {"maria", "gigel"},
	                      {"ana",   "gigel"}};
	TabPer tp;
	tp.n = 4;
	for (int i = 0; i < tp.n; i++) {
		tp.elem[i] = Per(data[i][0], data[i][1]);
	}
	return tp;
}
bool gasit(const TabStr &str, const string &v) {
	for (const string &e: str.elem) {
		if (e == v) {
			return true;
		}
	}
	return false;
}

TabStr formeazaTablouNumeCopii(const TabPer &per) {
	TabStr res;
	res.n = 0;
	for (const Per &p:per.elem) {
```

29 februarie 2020

Conf. Dr. Andreea Vescan
Lect. Dr. Radu Găceanu

```
            if (!gasit(res, p.x)) {
                res.elem[res.n++] = p.x;
            }
            if (!gasit(res, p.y)) {
                res.elem[res.n++] = p.y;
            }
        }
        return res;
}
int getCod(const TabStr &numeCopii, const string &nume) {
        for (int i = 0; i < numeCopii.n; i++) {
            if (numeCopii.elem[i] == nume) {
                return i;
            }
        }
        return 0;
}
void adaugaRelatie(TabTabInt &relatii, int x, int y) {
        TabInt &copilX = relatii.elem[x];
        copilX.elem[copilX.n++] = y;
}
TabTabInt formeazaTablouRelatii(const TabPer &perechi, const TabStr &numeCopii) {
        TabTabInt res;
        res.n = numeCopii.n;
        for (int i = 0; i < perechi.n; i++) {
            Per p = perechi.elem[i];
            int x = getCod(numeCopii, p.x);
            int y = getCod(numeCopii, p.y);
            adaugaRelatie(res, x, y);
        }
        return res;
}
void initVizitat(const TabTabInt &relatii, bool vizitat[]) {
        for (int i = 0; i < relatii.n; i++) {
            vizitat[i] = false;
        }
}
void viziteazaCopil(TabTabInt relatii, int cod, bool vizitat[], TabInt &res) {
        vizitat[cod] = true;
        for (int i = 0; i < relatii.elem[cod].n; i++) {
            int c = relatii.elem[cod].elem[i];
            if (!vizitat[c]) {
                viziteazaCopil(relatii, c, vizitat, res);
            }
        }
        res.elem[res.n++] = cod;
}
```

29 februarie 2020

Conf. Dr. Andreea Vescan
Lect. Dr. Radu Găceanu

```
TabInt calculeazaOrdineCoduri(TabTabInt relatii) {
	TabInt res;
	res.n = 0;
	bool vizitat[relatii.n];
	initVizitat(relatii, vizitat);
	for (int i = 0; i < relatii.n; i++) {
		if (!vizitat[i]) {
			viziteazaCopil(relatii, i, vizitat, res);
		}
	}
	return res;
}
TabStr codes2Names(const TabInt &coduri, const TabStr &nume) {
	TabStr res;
	res.n = 0;
	for (int i = coduri.n - 1; i > -1; i--) {
		res.elem[res.n++] = nume.elem[coduri.elem[i]];
	}
	return res;
}
TabStr determinaOrdineCopii(const TabPer &perechi) {
	TabStr numeCopii = formeazaTablouNumeCopii(perechi);
	TabTabInt tablouRelatii = formeazaTablouRelatii(perechi, numeCopii);
	TabInt coduriCopii = calculeazaOrdineCoduri(tablouRelatii);
	TabStr res = codes2Names(coduriCopii, numeCopii);
	return res;
}
void afiseazaTablou(const TabStr &arr) {
	for (const string &s: arr.elem) {
		cout << s << " ";
	}
}
int main() {
	TabPer perechi = citestePerechi();
	TabStr res = determinaOrdineCopii(perechi);
	afiseazaTablou(res);
	std::cout << "bye" << std::endl;
	return 0;
}
```