```
{
Iteratia 01
 - Functionalitatea/Subfunctionalitatea F4;
 - F4 = verifica daca un numar este prim sau nu

Iteratia 02
 - Functionalitatea/Subfunctionalitatea F5;
 - F5 = cauta pozitia unui element intr-un sir de elemente (daca exista)


Iteratia 03
 - Functionalitatea/Subfunctionalitatea F2;
 - F2 = determinarea numerelor prime si a frecventelor lor


Iteratia 04
 - Functionalitatea/Subfunctionalitatea F1 si F3;
 - F1 = citirea sirului dat
 - F2 = afisarea sirului determinat

 }
uses crt;
const  MAX=100;
type sir=array[1..MAX] of integer;
     pereche=record
        numar,frecventa:integer;
     end;
     sirRez=array[1..MAX] of pereche;


{********************** Cod pentru Iteratie 01 ************************}
function EstePrim(nr:integer):boolean;
var d:integer;
    prim:boolean;
begin
    prim:=true;
    if (nr=0 ) or (nr=1) then prim:=false
    else
      begin
        d:=2;
        while ((d<=(nr div 2)) and (prim)) do
         if (nr mod d=0) then
           prim:=false
         else
           d:=d+1;
    end;
    EstePrim:=prim;
end;

{********************** Cod pentru Iteratie 02 ************************}
function CautaPozitie(e,d:integer;s:sirRez):integer;
var poz,p:integer;
begin
   poz:=-1;
   p:=1;
   While ((p<=d) and (poz=-1)) do
     If (e=s[p].numar) then
       poz:=p
     else
       p:=p+1;
   CautaPozitie:=poz;
end;

{********************** Cod pentru Iteratie 03 ************************}
procedure ProcesarePentruCerinte(ds:integer;vs:sir;var dpf:integer;var
vpf:sirRez);
var contor, poz:integer;
begin
  contor := 1;
  dpf:=0;
  While(contor<=ds) do
```

```
    begin
     If(EstePrim(vs[contor]) ) then
       begin
         poz:=CautaPozitie(vs[contor], dpf, vpf);
         If( poz= -1) then
           begin
             dpf := dpf+1;
             vpf[dpf].numar :=vs[contor];
             vpf[dpf].frecventa:=1;
           end
         else
             vpf[poz].frecventa:=vpf[poz].frecventa+1;
        end;
      contor:=contor+1;
    end;
end;

{********************** Cod pentru Iteratie 04 ************************}
procedure CitireSir(var ds:integer;var vs:sir);
var i:integer;
begin
    Write('Numarul de elemente:');
    Readln(ds);
    Writeln('Elementele');
    for i:=1 to ds do
      begin
        writeln('Elementul al-',i,'-lea=');
        read(vs[i]);
       end;
end;

procedure TiparireRezultate(dpf:integer; vpf:sirRez);
var i:integer;
begin
   for i:=1 to dpf do
           writeln('numar=',vpf[i].numar,' frecventa=',vpf[i].frecventa);
   writeln;
end;




{********************** Cod pentru testarea functionalitatii F4 - Iteratia 01
************************}

procedure assertEquals( message:string; a,b:boolean);
begin
      write(message, ' : ');
      if a=b then

         writeln('success')
      else

         writeln('FAILURE');
end;

procedure testareEstePrim;
begin
     assertEquals(' 0 nu e prim ', false, EstePrim(0));
     assertEquals(' 1 nu e prim ', false, EstePrim(1));
     assertEquals(' 2 este prim ', true, EstePrim(2));
     assertEquals(' 17 este prim ', true, EstePrim(17));
     assertEquals(' 7 este prim ', true, EstePrim(7));
     assertEquals(' 3 este prim ', true, EstePrim(3));
     assertEquals(' 15 nu este prim ', false, EstePrim(15));
end;

{********************** Cod pentru testarea functionalitatii F5 - Iteratia 02
************************}
procedure initializareSirNumere(s:string; var ds:integer; var vs:sirRez);
{
```

```
   Descriere: metoda care creeaza un vector de dim ds si valori vs din stringul s
   Date: s
   Precond: s string
   Resultare: ds,vs
   Postcond: ds - numar natural, reprezinta numarul de numere naturale separate
de spatiu din stringul s
             vs - valorile vectorului de numere naturale din stringul s
}
var i:integer;
    nr,cifra:integer;
    codDeEroare:integer ;
begin
      ds:=0; i:=1;
      while (i<=length(s)) do begin
        nr:=0;
        { se citeste cifra cu cifra din sirul de caractere=string
          si se formeaza numarul din cifre }
        while (s[i]<>' ') and (i<=length(s)) do
          begin
             Val( s[i], cifra , codDeEroare);
             {transforma caracterul s[i] in numarul cifra
               cu codul de eroare 0, daca s-a reusit conversia,
               cu codul de eroare pozitie cu neconcordanta}
             nr:=  nr*10+cifra;
             i:=i+1;
           end;
        ds:=ds+1;
        vs[ds].numar:=nr;
        i:=i+1;
      end;
end;
procedure initializareSirFrecvente(s:string; var ds:integer; var vs:sirRez);
{
   Descriere: metoda care creeaza un vector de dim ds si valori vs (frecvente)
din stringul s
   Date: s
   Precond: s string
   Resultare: ds,vs
   Postcond: ds - numar natural, reprezinta numarul de numere naturale separate
de spatiu din stringul s
             vs - valorile vectorului de numere naturale din stringul s
}
var i:integer;
    nr,cifra:integer;
    codDeEroare:integer ;
begin
      ds:=0; i:=1;
      while (i<=length(s)) do begin
        nr:=0;
        { se citeste cifra cu cifra din sirul de caractere=string
          si se formeaza numarul din cifre }
        while (s[i]<>' ') and (i<=length(s)) do
          begin
             Val( s[i], cifra , codDeEroare);
             {transforma caracterul s[i] in numarul cifra
               cu codul de eroare 0, daca s-a reusit conversia,
               cu codul de eroare pozitie cu neconcordanta}
             nr:=  nr*10+cifra;
             i:=i+1;
           end;
        ds:=ds+1;
        vs[ds].frecventa:=nr;
        i:=i+1;
      end;
end;

procedure assertEquals( message:string; a,b:integer);

begin
      write(message, ' : ');
      if a=b then
```

```
         writeln('success')
      else
         writeln('FAILURE');
end;

procedure testareCautaPozitie;
var unSir:sirRez;
    i,dUnSir:integer;
begin

    initializareSirNumere('5 2 3 1', dUnSir, unSir);
    initializareSirFrecvente('5 2 3 1', dUnSir, unSir);
    assertEquals('3 in sir', 3, CautaPozitie(3,dUnSir, unSir));
    assertEquals('1 in sir', 4, CautaPozitie(1,dUnSir, unSir));
    assertEquals('4 nu este in sir', -1, CautaPozitie(4,dUnSir, unSir));

    initializareSirNumere('34', dUnSir, unSir);
    initializareSirFrecvente('3', dUnSir, unSir);
    assertEquals('34 in sir', 1, CautaPozitie(34,dUnSir, unSir));
    assertEquals('5 nu este in sir', -1, CautaPozitie(5,dUnSir, unSir));

end;

{********************** Cod pentru testarea functionalitatii F2 - Iteratia
03************************}
procedure initializareSir(s:string; var ds:integer; var vs:sir);
{
   Descriere: metoda care creeaza un vector de dim ds si valori vs din stringul s
   Date: s
   Precond: s string
   Resultare: ds,vs
   Postcond: ds - numar natural, reprezinta numarul de numere naturale separate
de spatiu din stringul s
             vs - valorile vectorului de numere naturale din stringul s
}
var i:integer;
    nr,cifra:integer;
    codDeEroare:integer ;
begin
      ds:=0; i:=1;
      while (i<=length(s)) do begin
        nr:=0;
        { se citeste cifra cu cifra din sirul de caractere=string
          si se formeaza numarul din cifre }
        while (s[i]<>' ') and (i<=length(s)) do
          begin
             Val( s[i], cifra , codDeEroare);
             {transforma caracterul s[i] in numarul cifra
               cu codul de eroare 0, daca s-a reusit conversia,
               cu codul de eroare pozitie cu neconcordanta}
             nr:=  nr*10+cifra;
             i:=i+1;
           end;
        ds:=ds+1;
        vs[ds]:=nr;
        i:=i+1;
      end;
end;
procedure assertEquals( message:string; a,b,c,d,e,f:integer);

begin
      write(message, ' : ');
      if (a=d)  and (b=e) and(c=f)then
         writeln('success')
      else
         writeln('FAILURE');
end;

procedure CautaPozFrecventa(dAltSir:integer;altSir:sirRez;e:integer;var
p,f:integer);
{
```

```
 Descriere: determina pozitia p si frecventa f a numarului e in sirul altSir;
}
var i:integer;
begin
    i:=1;p:=-1;f:=-1;
    while(i<=dAltSir)and(p=-1) do
      begin
        if(altSir[i].numar=e) then
          begin
             p:=i;
             f:=altSir[i].frecventa;
          end
          else
           i:=i+1;
      end;
end;
procedure testareProcesarePentruCerinte;
var unSir:sir;
    e,p,f,dUnSir,dAltSir:integer;
    altSir:sirRez;
begin

    initializareSir('7 2 3 1 7 2 1 7', dUnSir, unSir);
    ProcesarePentruCerinte(dUnSir, unSir,dAltSir,altSir);

    e:=7;
    CautaPozFrecventa(dAltSir,altSir,e,p,f);
    assertEquals('7 in sir de 3 ori',7,1,3,e,p,f);

    e:=1;
    CautaPozFrecventa(dAltSir,altSir,e,p,f);
    assertEquals('1 in sir de 0 ori',1,-1,-1,e,p,f);

    e:=3;
    CautaPozFrecventa(dAltSir,altSir,e,p,f);
    assertEquals('3 in sir de 1 ori',3,3,1,e,p,f);


end;


procedure testareAlgoritmNumerePrimeSiFrecvente;
begin

      writeln ('Teste pentru EstePrim ');
      writeln('_____________________');
      testareEstePrim;

      writeln;
      writeln ('Teste pentru CautaPozitie ');
      writeln('_____________________');
      testareCautaPozitie;

      writeln;
      writeln ('Teste pentru ProcesarePentruCerinte ');
      writeln('_____________________');
      testareProcesarePentruCerinte;

  end;


{********************** Cod pentru executia algoritmului implementat - Iteratia
04************************}

procedure ExecutieAlgoritmNumerePrimeSiFrecvente;
var ds,dpf:integer;
    vs:sir;
    vpf:sirRez;
begin

    CitireSir(ds,vs);
```

```
    ProcesarePentruCerinte(ds,vs,dpf,vpf);
    TiparireRezultate(dpf,vpf);

end;


begin
   {testareAlgoritmNumerePrimeSiFrecvente;}
   ExecutieAlgoritmNumerePrimeSiFrecvente;
  readkey;
end.
```