```
{
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)
}
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 testarea functionalitatii F4 - Iteratie 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  Iteratie 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;
        i:=i+1;
        ds:=ds+1;
        vs[ds].numar:=nr;
      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;
        i:=i+1;
        ds:=ds+1;
        vs[ds].frecventa:=nr;
      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;
    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 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 in sir', -1, CautaPozitie(5,dUnSir, unSir));

end;


procedure testareAlgoritmNumerePrimeSiFrecvente;
begin

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

      writeln;
      writeln ('Teste pentru CautaPozitie ');
      writeln('_____________________________');
      testareCautaPozitie;
  end;

begin
   testareAlgoritmNumerePrimeSiFrecvente;
  readkey;
end.
```