

ȘIRURI (TABLOURI UNIDIMENSIONALE)

Problema 1

Enunț

Se citesc mai multe numere naturale, până la introducerea numărului 0 și se memorează într-un șir. Să se găsească toate numerele perfecte din șir. Un număr natural este perfect dacă el este egal cu suma tuturor divizorilor săi proprii (divizorii proprii se referă la toți divizorii numărului, cu excepția numărului însuși).

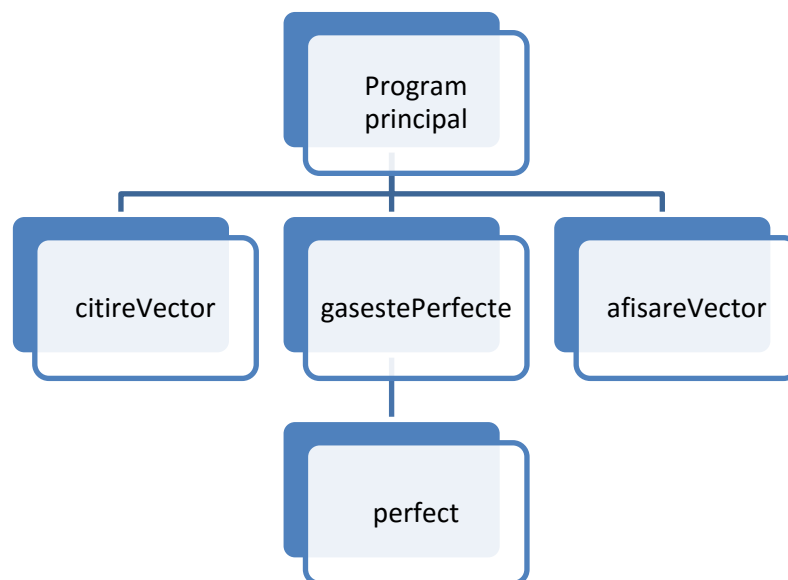
Exemplu:

- **6** este număr perfect. Divizorii proprii lui 6: 1, 2, 3. Suma divizorilor este egală cu numărul: $1 + 2 + 3 = 6$.
- **28** este număr perfect. $1 + 2 + 4 + 7 + 14 = 28$.

Se cere să se utilizeze subprograme, care să comunice între ele și cu programul principal prin parametri. Fiecare subprogram trebuie specificat.

Analiză

Identificarea subalgoritmilor



Specificarea subalgoritmilor

- Subalgoritmul **citireVector(a,n)**:

Descriere: Citește elementele unui vector până la întâlnirea numărului 0.

Date: -

Rezultate: a - vector cu elemente nenule de lungime n

$n \in \mathbb{N}$, $a = (a_i | i=1..n, a_i \in \mathbb{N}, a_i \neq 0)$.

- Functia **perfect(x)**:
Descriere: Verifică dacă numărul x este perfect.
Date: x – număr natural, $x \in \mathbb{N}$
Rezultate: returnează *true* dacă numărul x e perfect, *false* altfel.
- Subalgoritmul **gasestePerfecte(a,n,rez,m)**:
Descriere: Găsește numerele perfecte din șirul a de lungime n și le salvează în șirul rez , de lungime m .
Date: a, n – a vector de elemente nenule de lungime n , $n \in \mathbb{N}$.
 $a=(a_i, i=1..n, a_i \in \mathbb{N}^*)$
Rezultate: rez, m – rez e șirul numerelor perfecte din șirul a , m lungimea lui rez .
 $rez=(rez_i | i=1..m, rez_i \text{ numar perfect}, m \in \mathbb{N})$
- Subalgoritmul **afisareVector(rez, m)**:
Descriere: Afisează elementele din vectorul rez de lungime m .
Date: rez - vector de lungime m .
 $rez=(rez_i | i=1..m, rez_i \in \mathbb{N}, m \in \mathbb{N})$
Rezultate: se afișează elementele din vectorul rez .

Proiectare

Subalgoritmul **citireVector(a, n)** este:

```

n ← 0
@citeste x
cat-timp x <> 0 executa
    n ← n + 1
    a[n] ← x
    @citeste x
sf-cat-timp
Sf-subalgoritm

```

Functia **perfect(x)** este:

```

suma_div ← 1                                {initializam suma divizorilor cu 1}
i ← 2
cat-timp i <= x div 2 executa                {cautam urmatorii divizori proprii}
    daca x mod i = 0 atunci                  {daca i e un divizor, il adunam la suma divizorilor}
        suma_div ← suma_div+i
    sf-daca
    i ← i + 1
sf-cat-timp
daca suma_div = x atunci
    perfect ← true
altfel
    perfect ← false
sf-daca

daca x = 1 atunci
    perfect ← false
sf-daca
Sf-functie

```

Subalgoritmul **gasestePerfecte**(a,n, rez, m) este:

```

    m ← 0                                {initializam lungimea sirului rez}
    pentru i ← 1, n executa                {parcurgem sirul a}
        daca perfect(a[i]) atunci         {verificam daca elementul curent e perfect}
            m ← m + 1                       {in caz afirmativ, il adaugam}
            rez[m] ← a[i]
        sf-daca
    sf-pentru
Sf-subalgoritm

```

Subalgoritmul **afisareVector**(rez, m) este:

```

    daca m = 0 atunci
        @tipareste „Vectorul este vid”
    altfel
        pentru i ← 1, m executa
            @tipareste rez[i]
        sf-pentru
    sf-daca
sf-subalg

```

Algoritmul **NumerePerfecte** este:

```

    citireVector(a,n)
    gasestePerfecte(a,n,rez,m)
    afisareVector(rez,m)
Sf-Algoritm

```

Exemple

Date de intrare	Rezultate
2, 1, 28, 4, 6, 0	28, 6
1, 2, 3, 4, 5, 0	Nu exista numere perfecte in sir.
8128, 9, 28, 496, 10, 7, 15, 0	8128, 28, 496
100, 90, 1, 86, 165, 0	28

program numerePerfecte;

type vector= array[1..20] of integer;

```

{
    Citeste elementele unui vector pana la intalnirea numarului 0.
    Date: -
    Rezultate: a - vector cu elemente nenule de lungime n, n - numar natural.}
procedure citireVector(var a:vector; var n:integer);
var x:integer;
begin
    n:=0;
    writeln('Introduceti numerele');
    read(x);
    while(x>0) do
        begin
            n:=n+1;
            a[n]:=x;
            read(x);
        end;
end;

```

```

{ Afiseaza elementele din vectorul rez, de lungime n.
  Date: rez - vector cu elemente numere intregi, de lungime n, n - numar natural.
  Rezultate: se afiseaza elementele din vectorul rez.}
procedure afisareVector(a:vector;n:integer);
var i:integer;
begin
    if (n=0) then writeln ('Nu exista numere perfecte in sir.')
    else
        begin
            for i:=1 to n do
                write(a[i], ' ');
                writeln;
            end;
        end;
end;

{ Verifica daca numarul x este perfect.
  Date: x - numar natural.
  Rezultate: returneaza true daca numarul x este perfect, false altfel.}
function perfect(x:integer):boolean;
var sumaDiv,i:integer;
begin
    sumaDiv:=1;
    perfect:=false;
    for i:=2 to x div 2 do
        if (x mod i=0) then sumaDiv:=sumaDiv+i;
        if (sumaDiv=x) then perfect:=true;
        if (x = 1) then perfect := false;
    end;
end;

{ Gaseste numerele perfecte din sirul a de lungime n si le salveaza in sirul rez, de
  lungime m.
  Date: a - vector de elemente nenule de lungime n, n - numar natural.
  Rezultate: rez - sirul numerelor perfecte din sirul a, m - numar natural, lungimea lui
  rez.}
procedure gasestePerfecte(a:vector;n:integer;var rez:vector;var m:integer);
var i:integer;
begin
    m:=0;
    for i:=1 to n do
        if perfect(a[i])=true then
            begin
                m:=m+1;
                rez[m]:=a[i];
            end;
    end;
end;

{ Programul principal}
var a,b:vector;
    m,n:integer;

begin
    writeln(perfect(1));
    citireVector(a,n);
    gasestePerfecte(a,n,b,m);
    afisareVector(b,m);
    readln;
end.

```

Problema 2

Enunț

Se citesc n numere naturale (valoarea lui n este citită) și se memorează într-un șir. Să se insereze, după fiecare număr din șir următorul număr perfect.

Exemplu:

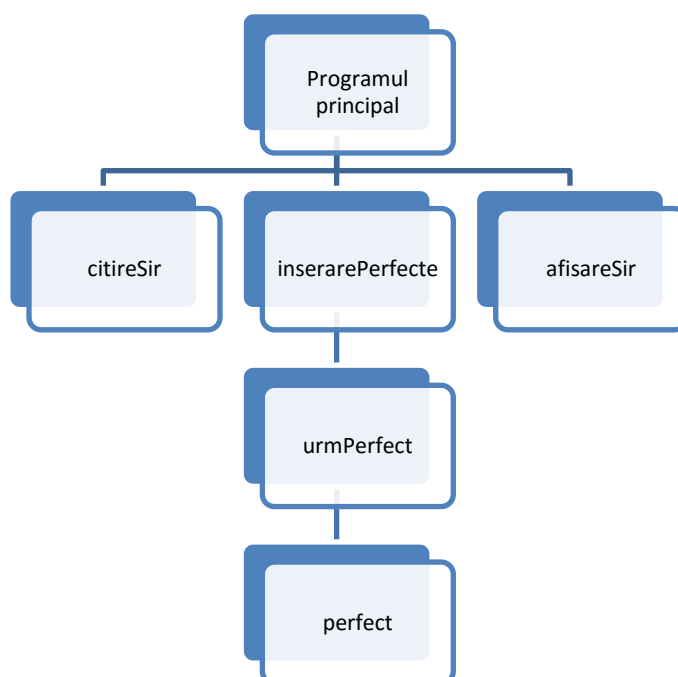
Dat fiind șirul: 100 18 30 5 5496.

Șirul rezultat va fi: 100 496 18 28 30 496 5 6 5496 8128.

Se cere să se utilizeze subprograme, care să comunice între ele și cu programul principal prin parametri. Fiecare subprogram trebuie specificat.

Analiză

Identificarea subalgoritmilor



Specificarea subalgoritmilor

- Subalgoritmul **citireSir**(a, n)
Descriere: Citește șirul de numere naturale a , de lungime n .
Date: -
Rezultate: a, n – a vector de numere naturale de lungime n .
 $a = (a_i \mid i=1..n, a_i \in \mathbb{N}), n \in \mathbb{N}$
- Subalgoritmul **inserarePerfecte**(a, n)
Descriere: Inserează în vectorul a de lungime n după fiecare element numărul perfect mai mare decât el.
Date: $a, n, a = (a_i \mid i=1..n, a_i \in \mathbb{N}), n \in \mathbb{N}$
Rezultate: a, n
 $a = (a_i \mid i=1..n, a_i \in \mathbb{N}), n \in \mathbb{N}$, a va avea inserat după fiecare element următorul număr perfect mai mare decât el

- Subalgoritmul **inserareElem**(a,n,pos,elem)

Descriere: Inserează elementul *elem* la poziția *pos* în vectorul *a* de lungime *n*.
Date: *a, n, pos, elem*
a - vector de lungime *n*, $n \in \mathbb{N}$,
 $pos \in \mathbb{N}, 0 \leq pos \leq n$, $elem \in \mathbb{N}$
Rezultate: *a, n, a=(a_i, i=1..n, a va contine elementul elem la pozitia pos), n \in \mathbb{N}*
- Subalgoritmul **afisareSir**(a,n)

Descriere: Afișează elementele din vectorul *a* de lungime *n*.
Date: *a, n* – *a* vector de lungime *n*, $n \in \mathbb{N}$. $a=(a_i | i=1..n, a_i \in \mathbb{N})$,
Rezultate: - se afișează elementele din vectorul *a*.
- Funcția **perfect**(x)

Descriere: Verifică dacă numărul *x* este perfect.
Date: *x* – număr natural, $x \in \mathbb{N}$
Rezultate: returnează *true* dacă numărul *x* e perfect, *false* altfel.
- Funcția **urmPerfect**(x)

Descriere: Găsește următorul număr perfect mai mare decât *x*.
Date: *x* – număr natural, $x \in \mathbb{N}$
Rezultate: returnează următorul număr perfect mai mare decât *x*.

Proiectare

Subalgoritmul **citireSir**(a,n) este:

```
@tipareste „Introduceti numarul de elemente”
@citeste n
@tipareste „Introduceti elementele”
pentru i ← 1, n executa
    @citeste a[i]
sf-pentru
sf-subalg
```

Funcția **perfect**(x) este:

```
suma_div ← 1 {initializam suma divizorilor cu 1}
i ← 2
cat-timp i <= x div 2 executa {cautam urmatorii divizori proprii}
    daca x mod i = 0 atunci {daca i e un divizor, il adunam la suma divizorilor}
        suma_div ← suma_div+i
    sf-daca
    i ← i + 1
sf-cat-timp
daca suma_div = x atunci
    perfect ← true
altfel
    perfect ← false
sf-daca

daca x = 1 atunci
    perfect ← false
sf-daca
```

Sf-functie

Functia **urmPerfect(x)** este:

```
y ← x + 1
cat-timp perfect(y) = false executa
  y ← y + 1
sf-cat-timp
urmPerfect ← y
```

sf-functie

Subalgoritmul **inserarePerfecte(a,n)** este:

```
i ← 1
cat-timp i <= n executa
  elem ← urmPerfect(a[i]) {cautam primul nr perfect mai mare decat elem. curent}
  inserareElem(a,n,i+1,elem) {inseram nr. perfect pe poz. urmatoare}
  i ← i + 2 {trecem la urmatorul numar din sir}
sf-cat-timp
```

sf-subalg

Subalgoritmul **inserarePerfecte(a,n)** este: {versiunea 1, fara translatarea sirului}

```
i ← n {pornim de la sfarsitul sirului}
cat-timp (i >= 1) executa {mutam elementele pe pozitia finala}
  a[2*i] ← urmPerfect(a[i])
  a[2*i-1] ← a[i]
  i ← i - 1
```

sf-cat-timp

n ← n * 2

{dublăm lungimea sirului}

Sf-subalg

Subalgoritmul **inserareElem(a,n,pos, elem)** este:

```
n ← n + 1 {incrementam dimensiunea vectorului}
i:=n {mutam elementele vectorului de la capat spre pozitia de inserare}
cat-timp (i > pos) executa
  a[i] ← a[i - 1]
  i ← i - 1
```

sf-cat-timp {i=pos}

a[i] ← elem {inseram elem la pozitia pos}

sf-subalg

Subalgoritmul **afisareSir(a,n)** este:

```
daca n = 0 atunci
  @tipareste „Vectorul este vid”
altfel
  pentru i ← 1, n executa
    @tipareste a[i]
```

sf-pentru

sf-daca

sf-subalg

Algoritmul **InserarePerfecte** este:

```
citireSir(a,n)
inserarePerfecte(a,n)
afisareSir(a,n)
```

Sf-Algorithm

Exemple

Date de intrare		Rezultate
n	sir	
5	100, 18, 30, 5, 5496	100, 496, 18, 28, 30, 496, 5, 6, 5496, 8128
4	10, 850, 1000, 2	10, 28, 850, 8128, 1000, 8128, 2, 6
6	-5, 1, 10, 10, 2, 2	-5, 6, 1, 6, 10, 28, 10, 28, 2, 6, 2, 6
1	5000	5000, 8128
5	1, 10, 1000, -1, 400	1, 6, 10, 28, 1000, 8128, -1, 6, 400, 496

```
program inserarePerfecte;
```

```
type vector=array[1..100] of integer;
```

```
{ Citeste elementele unui vector pana la intalnirea numarului 0.
```

```
  Date: -
```

```
  Rezultate:a - vector cu elemente nenule de lungime n, n - numar natural.}
```

```
procedure citireSir(var x:vector; var n:integer);
```

```
  var i:integer;
```

```
  begin
```

```
    write('Dati n: ');
```

```
    readln(n);
```

```
    write('Introduceti numerele: ');
```

```
    for i:=1 to n do
```

```
      read(x[i]);
```

```
    writeln;
```

```
  end;
```

```
{ Afiseaza elementele din vectorul rez, de lungime n.
```

```
  Date: rez - vector cu elemente numere intregi, de lungime n, n - numar natural.
```

```
  Rezultate: se afiseaza elementele din vectorul rez.}
```

```
procedure afisareSir(x:vector; n:integer);
```

```
  var i:integer;
```

```
  begin
```

```
    for i := 1 to n do
```

```
      write(x[i], ' ');
```

```
    writeln;
```

```
  end;
```

```
{ Verifica daca numarul x este perfect.
```

```
  Date: x - numar natural.
```

```
  Rezultate: returneaza true daca numarul x este perfect, false altfel.}
```

```
function perfect(x:integer):boolean;
```

```
  var sumaDiv,i:integer;
```

```
  begin
```

```
    sumaDiv:=1;
```

```
    perfect:=false;
```

```
    for i:=2 to x div 2 do
```

```
      if (x mod i=0) then sumaDiv:=sumaDiv+i;
```

```
    if (sumaDiv=x) then perfect:=true;
```

```
    if (x = 1) then perfect := false;
```

```
  end;
```



```

{   Insereaza elementul elem la pozitia pos in vectorul a de lungime n.
    Date: a - vector de elemente nenule de lungime n, n - numar natural.
        pos - numar natural, 0 <= pos < n.
        elem - numar intreg.
    Rezultate: noul vector a va contine elementul elem la pozitia pos.}
procedure inserareElem(var a:vector; var n:integer;pos,elem:integer);
    var i:integer;
    begin
        n:=n+1;
        for i:=n downto pos+1 do a[i]:=a[i-1];
        a[pos]:=elem;
    end;

{   Gaseste urmatorul numar perfect mai mare decat x.
    Date: x - numar natural.
    Rezultate: y - urmatorul numar perfect mai mare decat x.}
function urmPerfect(x:integer):integer;
    var y:integer;
    begin
        y:=x+1;
        while(perfect(y)=false) do y:=y+1;
        urmPerfect:=y;
    end;

{   Insereaza in vectorul a de lungime n dupa fiecare element numarul perfect mai mare decat el.
    Date: a - vector de elemente nenule de lungime n, n - numar natural.
    Rezultate: a - vectorul modificat, in care dupa fiecare numar a fost inserat primul numar
    perfect mai mare decat el.
        n - numar natural, lungimea vectorului modificat.}
procedure inserarePerfecte(var a:vector; var n:integer);
var i,elem:integer;
begin
    i:=1;
    while(i<=n)do begin
        elem:=urmPerfect(a[i]);
        inserareElem(a,n,i+1,elem);
        i:=i+2;
    end;
end;

{   Insereaza in vectorul a de lungime n dupa fiecare element numarul perfect mai mare decat el.
    Date: a - vector de elemente nenule de lungime n, n - numar natural.
    Rezultate: a - vectorul modificat, in care dupa fiecare numar a fost inserat primul numar
    perfect mai mare decat el.
        n - numar natural, lungimea vectorului modificat.}
procedure inserarePerfecte1(var a:vector; var n:integer);
var i:integer;
begin
    i:=n;
    while(i>=1) do begin
        a[2*i]:=urmPerfect(a[i]);
        a[2*i-1]:=a[i];
        i:=i-1;
    end;
    n:=n*2;
end;

// Programul principal
var a:vector;
    n:integer;
begin
    citireSir(a,n);
    inserarePerfecte1(a,n);
    afisareSir(a,n);
    readln;
end.

```

Problema 3

Enunț

Institutul Meteorologic Cluj are nevoie de o aplicație care să permită menținerea evidenței tuturor temperaturilor medii anuale (în grade Celsius), începând de la un an dat, până în anul precedent celui curent (până în 2015, în acest caz). Aplicația trebuie să permită:

- Introducerea unui an de început și apoi a temperaturilor medii anuale, până în anul precedent celui curent (2015). De exemplu, dacă anul de început este 1999, se vor reține 17 temperaturi medii (pentru 1999, pentru 2000, pentru 2001, ... pentru 2015).
- Afișarea celei mai lungi perioade și a valorilor temperaturilor asociate în care temperaturile au crescut în continuu.
- Afișarea temperaturilor maxime, din fiecare perioadă în care temperaturile au crescut și apoi au scăzut, precum și a temperaturii maxime din toată perioada indicată.

Exemplu:

Pentru an curent 2016, an de început 1999 și temperaturile medii anuale:

15.2, 12.1, 10.8, 11, 14.6, 15.8, 10, 11.3, 12.1, 12.7, 13.1, 13, 12.8, 12.6, 12, 11.8, 11.5

Cea mai lungă perioadă în care temperaturile au crescut în continuu și valorile temperaturilor:

2005 - 2009

10, 11.3, 12.1, 12.7, 13.1

Toate perioadele în care temperaturile au crescut, cu maximele lor (se cer doar maximele, dar sunt exemplificate și perioadele pentru o mai bună înțelegere):

10.8, 11, 14.6, 15.8 -> max: 15.8

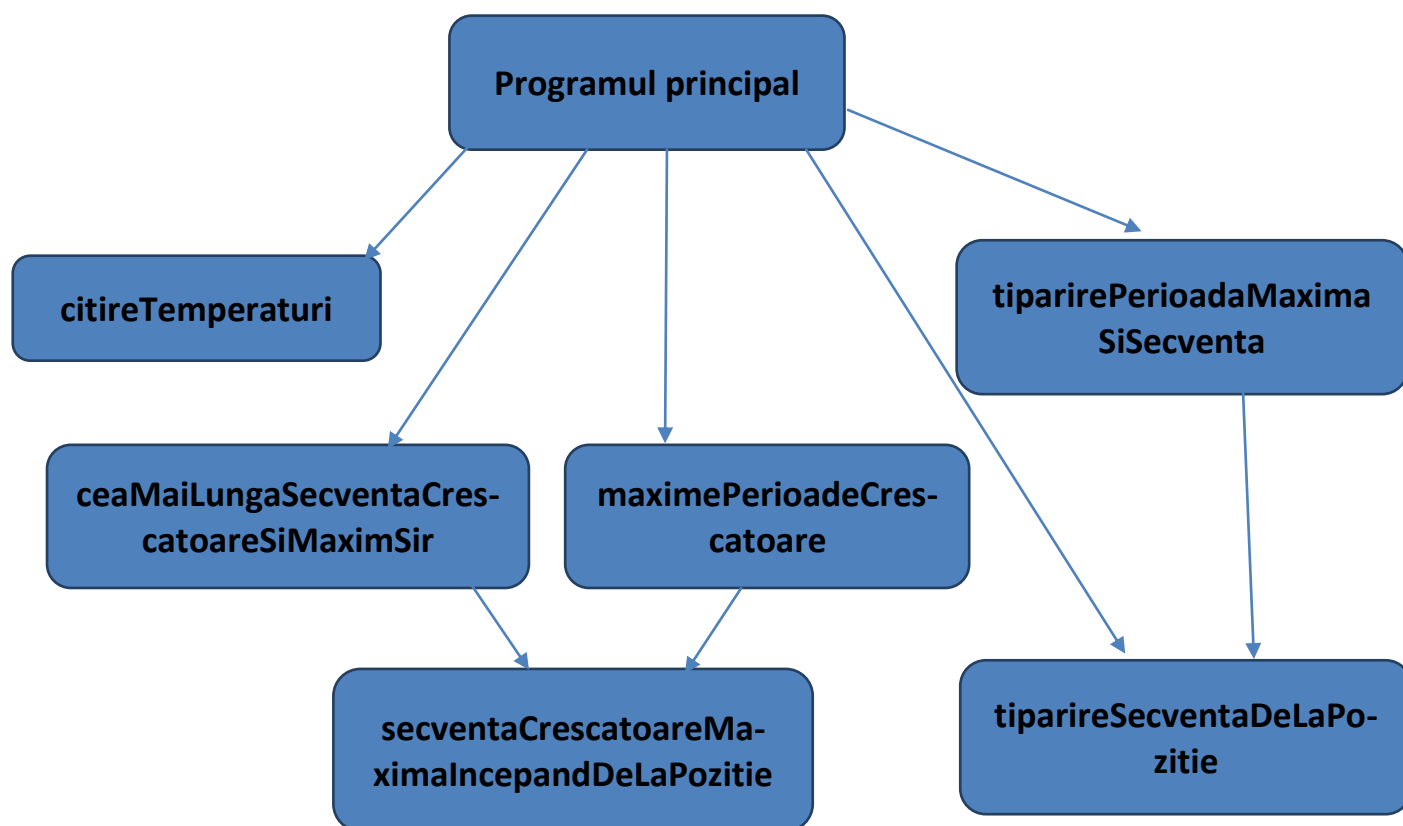
10, 11.3, 12.1, 12.7, 13.1 -> max: 13.1

Temperatura maximă din toată perioada indicată: 15.8

Se cere să se utilizeze subprograme, care să comunice între ele și cu programul principal prin parametri. Fiecare subprogram trebuie specificat.

Analiză

Identificarea subalgoritmilor



Specificarea subalgoritmilor

- Subalgoritmul **citireTemperaturi(sirTemperaturi, lungimeSir, anInceput)**:
Descriere: Citește anul de la care începe raportarea temperaturilor anuale, precum și șirul acestor temperaturi și calculează lungimea șirului, știindu-se anul curent.
Date: -
Rezultate: - *sirTemperaturi* – șir de numere reale, reprezentând fiecare câte o temperatură anuală
 $sirTemperaturi = (t_i \in \mathbf{R} | i=1..lungimeSir, lungimeSir \in \mathbf{N})$
- $lungimeSir \in \mathbf{N}$, lungimea șirului temperaturilor.
- $anInceput \in \mathbf{N}$, anul de la care începem înregistrarea temperaturilor.
- Subalgoritmul **secventaCrescatoareMaximalIncepandDeLaPozitie(sirTemperaturi, lungimeSir, pozitie, lungimeSecventa)**:
Descriere: Calculează secvența de numere crescătoare dintr-un șir, începând de la o poziție dată.
Date: - *sirTemperaturi* – șir de numere reale, reprezentând fiecare câte o temperatură anuală
 $sirTemperaturi = (t_i \in \mathbf{R} | i=1..lungimeSir, lungimeSir \in \mathbf{N})$
- $lungimeSir \in \mathbf{N}$, lungimea șirului temperaturilor.
- $pozitie \in \mathbf{N}$, poziția începând de la care se caută secvența crescătoare.
Rezultate: $lungimeSecventa \in \mathbf{N}$, lungimea celei mai lungi secvențe crescătoare începând de la poziția dată.

- Subalgoritmul **ceaMaiLungaSecventaCrescatoareSiMaximSir(sirTemperaturi, lungimeSir, lungimeSecventaMaxima, pozitieInceput, maxim):**

Descriere: Calculează cea mai lungă secvență de numere crescătoare dintr-un șir, precum și maximul din șir.

Date: - *sirTemperaturi* – șir de numere reale, reprezentând fiecare câte o temperatură anuală
sirTemperaturi = $(t_i \in \mathbf{R} | i=1..lungimeSir, lungimeSir \in \mathbf{N})$
 - *lungimeSir* $\in \mathbf{N}$, lungimea șirului temperaturilor.
 - *pozitie* $\in \mathbf{N}$, poziția începând de la care se caută secvența crescătoare.

Rezultate: - *pozitieInceput* $\in \mathbf{N}$, poziția din șir de la care începe cea mai lungă secvența de numere crescătoare.
 - *lungimeSecventa* $\in \mathbf{N}$, lungimea celei mai lungi secvențe de numere crescătoare din șirul dat.
 - *maxim* $\in \mathbf{R}$, numărul maxim din șir.
- Subalgoritmul **maximePerioadeCrescatoare(sirTemperaturi, lungimeSir, sirMaxime, lungimeSirMaxime):**

Descriere: Identifică maximele tuturor secvențelor de temperaturi crescătoare.

Date: - *sirTemperaturi* – șir de numere reale, reprezentând fiecare câte o temperatură anuală
sirTemperaturi = $(t_i \in \mathbf{R} | i=1..lungimeSir, lungimeSir \in \mathbf{N})$
 - *lungimeSir* $\in \mathbf{N}$, lungimea șirului temperaturilor.

Rezultate: - *sirMaxime* – șir de numere reale, conținând maximele tuturor secvențelor de temperaturi crescătoare
sirMaxime = $(t_i \in \mathbf{R} | i=1..lungimeSirMaxime, lungimeSirMaxime \in \mathbf{N})$
 - *lungimeSirMaxime* $\in \mathbf{N}$, lungimea șirului conținând elementele maxime ale secvențelor crescătoare.
- Subalgoritmul **tiparireSecventaDeLaPozitie (sir, lungimeSir, lungimeSecventa, pozitieInceput):**

Descriere: Tiparește secvența din șirul dat, care începe de la poziția dată, având lungimea dată.

Date: - *sir* – șir de numere reale, reprezentând fiecare câte o temperatură anuală
sir = $(t_i \in \mathbf{R} | i=1..lungimeSir, lungimeSir \in \mathbf{N})$
 - *lungimeSir* $\in \mathbf{N}$, lungimea șirului.
 - *lungimeSecventa* $\in \mathbf{N}$, lungimea secvenței de numere care trebuie tipărite.
 - *pozitieInceput* $\in \mathbf{N}$, poziția începând de la care trebuie tipărite numerele din șirul dat.

Rezultate: se afișează elementele subsecvenței din șirul dat, care începe de la poziția *pozitieInceput*, și are lungimea *lungimeSecventa*.
- Subalgoritmul **tiparirePerioadaMaximaSiSecventa (sirTemperaturi, lungimeSir, lungimeSecventa, pozitieInceput, anInceput):**

Descriere: Tiparește cea mai lungă perioadă în care temperaturile au crescut în continuu, precum și valorile temperaturilor asociate.

Date: - *sirTemperaturi* – șir de numere reale, reprezentând fiecare câte o temperatură anuală
sirTemperaturi = $(t_i \in \mathbf{R} | i=1..lungimeSir, lungimeSir \in \mathbf{N})$
 - *lungimeSir* $\in \mathbf{N}$, lungimea șirului.
 - *lungimeSecventa* $\in \mathbf{N}$, lungimea secvenței de numere care trebuie tipărite.
 - *pozitieInceput* $\in \mathbf{N}$, poziția începând de la care trebuie tipărite numerele din șirul dat.
 - *anInceput* $\in \mathbf{N}$, anul de la care sunt raportate temperaturile.

Rezultate: se afișează cea mai lungă perioadă în care temperaturile au crescut în continuu, precum și valorile temperaturilor asociate.

Proiectare

Subalgoritmul **citireTemperaturi**(sirTemperaturi, lungimeSir, anInceput) este:

```
@tipareste „Introduceti anul de inceput: ”
@citeste anInceput
@anCurent ← identifica anul curent
lungimeSir ← anCurent - anInceput
{citim sirul temperaturilor}
pentru i ← 1, lungimeSir executa
    @tipareste „Temperatura pentru anul ”, anInceput + i - 1, „: ”
    @citeste sirTemperaturi[i]
sf-pentru
sf-subalg
```

Subalgoritmul **secventaCrescatoareMaximaIncepandDeLaPozitie**(sirTemperaturi, lungimeSir, pozitie, lungimeSecventa) este:

```
lungimeSecventa ← 0
i ← pozitie + 1
cat-timp i < lungimeSir si sirTemperatur[i] > sirTemperaturi[i - 1] executa
    i ← i + 1
sf-cat-timp
lungimeSecventa ← i - pozitie
sf-functie
```

Subalgoritmul **ceaMaiLungaSecventaCrescatoareSiMaximSir**(sirTemperaturi, lungimeSir, lungimeSecventaMaxima, pozitieInceput, maxim) este:

```
i ← 1
cat-timp i <= lungimeSir executa
    lungimeSecventa ← 0
    secventaCrescatoareMaximaIncepandDeLaPozitie(sirTemperaturi, lungimeSir,
        i, lungimeSecventa)

    daca lungimeSecventa > lungimeSecventaMaxima atunci
        lungimeSecventaMaxima ← lungimeSecventa
        pozitieInceput ← i
sf-daca

{maximul celei mai lungi secvente crescatoare va fi valoarea ultimului
element din secventa}
maximSecventa ← sirTemperaturi[i + lungimeSecventa - 1]
daca maximSecventa > maxim atunci
    maxim ← maximSecventa
sf-daca
i ← i + lungimeSecventa
sf-subalgoritm
```

Subalgoritmul **maximePerioadeCrescatoare**(sirTemperaturi, lungimeSir, sirMaxime, lungimeSirMaxime) este:

```
i ← 1
lungimeSirMaxime ← 0
cat-timp i <= lungimeSir executa
    lungimeSecventa ← 0
    secventaCrescatoareMaximaIncepandDeLaPozitie(sirTemperaturi, lungimeSir,
        i, lungimeSecventa)

    {consideram doar acele perioade care contin cel putin 2 ani cu temperaturi
    crescatoare}
    daca lungimeSecventa > 1 atunci
        {maximul celei mai lungi secvente crescatoare va fi valoarea ultimului
        element din secventa}
        maximSecventa ← sirTemperaturi[i + lungimeSecventa - 1]
```

```

        lungimeSirMaxime ← lungimeSirMaxime + 1
        sirMaxime[lungimeSirMaxime] ← maximSecventa
    sf-daca
        i ← i + lungimeSecventa
    sf-cat-timp
sf-subalgoritm

```

Subalgoritmul **tiparireSecventaDeLaPozitie**(sir, lungimeSir, lungimeSecventa, pozitieInceput) este:

```

    i ← pozitieInceput
    cat-timp i < pozitieInceput + lungimeSecventa - 1
        @tipareste sir[i], “, “
        i ← i + 1
    sf-cat-timp
    @tipareste sir[i]
sf-subalgoritm

```

Subalgoritmul **tiparirePerioadaMaximaSiSecventa**(sirTemperaturi, lungimeSir, lungimeSecventa, pozitieInceput, anInceput) este:

```

    {tiparire perioada}
    anInceputPerioada ← anInceput + pozitieInceput - 1
    anSfarsitPerioada ← anInceputPerioada + lungimeSecventa - 1
    @tipareste “Cea mai lunga perioada in care temperaturile au crescut in continuu
este: “,anInceputPerioada, “ - “, anSfarsitPerioada

```

```

    {tiparim valorile temperaturilor}
    tiparireSecventaDeLaPozitie(sirTemperaturi, lungimeSir, lungimeSecventa,
                                pozitieInceput)
sf-subalgoritm

```

Algoritmul **InstitutMeteo** este:

```

    lungimeSir ← 0
    anInceput ← 0
    citireTemperaturi(sirTemperaturi, lungimeSir, anInceput)

    {identificam cea mai lunga secventa in care temperaturile cresc in continuu si o
afisam}
    lungimeSecventa ← 0
    maxim ← -100
    pozitieInceput ← 1
    ceaMaiLungaSecventaCrescatoareSiMaximSir(sirTemperaturi, lungimeSir,
lungimeSecventa, pozitieInceput, maxim)
    tiparirePerioadaMaximaSiSecventa(sirTemperaturi, lungimeSir, lungimeSecventa,
pozitieInceput, anInceput)

    {tiparire temperaturi maxime, din fiecare perioadă în care temperaturile au crescut
și apoi au scăzut}
    lungimeSirMaxime ← 0
    maximePerioadeCrescatoare(sirTemperaturi, lungimeSir, sirMaxime, lungimeSirMaxime)
    @tipareste “Temperaturile maxime, din fiecare perioada in care temperaturile au
crescut si apoi au scazut: “
    tiparireSecventaDeLaPozitie(sirMaxime, lungimeSirMaxime, lungimeSirMaxime, 1)

    {tiparire temperatura maxima}
    @tipareste “Temperatura anuala maxima din perioada indicata este: “, maxim
Sf-Algoritm

```

Exemple

Date de intrare		Rezultate
An inceput	Sir temperaturi	
1999	15.2, 12.1, 10.8, 11, 14.6, 15.8, 10, 11.3, 12.1, 12.7, 13.1, 13, 12.8, 12.6, 12, 11.8, 11.5	<p>Cea mai lunga perioada in care temperaturile au crescut in continuu este: 2005 – 2009. 10, 11.3, 12.1, 12.7, 13.1</p> <p>Temperaturile maxime, din fiecare perioada in care temperaturile au crescut si apoi au scazut: 15.8, 13.1.</p> <p>Temperatura anuala maxima din perioada indicata este: 15.8.</p>
2010	14, 15, 13, 14, 15, 16	<p>Cea mai lunga perioada in care temperaturile au crescut in continuu este: 2012 – 2015. 13, 14, 15, 16</p> <p>Temperaturile maxime, din fiecare perioada in care temperaturile au crescut si apoi au scazut: 15, 16.</p> <p>Temperatura anuala maxima din perioada indicata este: 16.</p>
2008	11, 12, 12.3, 14.8, 13.1, 14, 14.5, 14.2	<p>Cea mai lunga perioada in care temperaturile au crescut in continuu este: 2008 – 2011. 11, 12, 12.3, 14.8</p> <p>Temperaturile maxime, din fiecare perioada in care temperaturile au crescut si apoi au scazut: 14.8, 14.5.</p> <p>Temperatura anuala maxima din perioada indicata este: 14.8.</p>
2013	15, 15.5, 16	<p>Cea mai lunga perioada in care temperaturile au crescut in continuu este: 2013 – 2015. 15, 15.5, 16</p> <p>Temperaturile maxime, din fiecare perioada in care temperaturile au crescut si apoi au scazut: 16.</p> <p>Temperatura anuala maxima din perioada indicata este: 16.</p>

```

program InstitutMeteo;

uses sysutils;

type vector = array[1..100] of real;

{   Returneaza anul curent.
    Date: -
    Rezultate: an - integer, reprezinta anul curent.}
function anulCurent(): Word;
var an, luna, zi: Word;
begin
    DeCodeDate (Date, an, luna, zi);
    anulCurent := an;
end;

{   Citeste anul de la care incepe raportarea temperaturilor anuale, precum si sirul acelor
    temperaturi
    si calculeaza lungimea sirului, stiindu-se anul curent.
    Date: -
    Rezultate:sirTemperaturi - sir de elemente de tip double, reprezentand fiecare cate o
                temperatura anuala.
                lungimeSir - integer, lungimea sirului temperaturilor.}
procedure citireTemperaturi(var sirTemperaturi: vector; var lungimeSir: integer; var anInceput:
Word);
var i: integer;
begin
    write('Introduceti anul de inceput: ');
    read(anInceput);
    lungimeSir := anulCurent() - anInceput;

    // se citeste sirul temperaturilor
    for i := 1 to lungimeSir do
        begin
            write('Temperatura pentru anul ', anInceput + i - 1, ': ');
            read(sirTemperaturi[i]);
        end;
end;

{   Calculeaza secventa de numere crescatoare dintr-un sir, incepand de la o pozitie data.
    Date:      sirTemperaturi - sir de elemente de tip double, reprezentand fiecare cate o
                temperatura anuala.
                lungimeSir - integer, lungimea sirului temperaturilor.
                pozitie - integer, pozitia incepand de la care se cauta secventa crescatoare.
    Rezultate:lungimeSecventa - integer, lungimea celei mai lungi secvente crescatoare incepand de
                la pozitia data.}
procedure secventaCrescatoareMaximaIncepanDeLaPozitie(sirTemperaturi: vector; lungimeSir: integer;
pozitie: integer; var lungimeSecventa: integer);
var i: integer;
begin
    lungimeSecventa := 0;
    i := pozitie + 1;
    while (i <= lungimeSir) AND (sirTemperaturi[i] > sirTemperaturi[i - 1]) do
        i := i + 1;
    lungimeSecventa := i - pozitie;
end;

{   Calculeaza cea mai lunga secventa de numere crescatoare dintr-un sir, precum si maximul din
    sir.
    Date:      sirTemperaturi - sir de elemente de tip double, reprezentand fiecare cate o
                temperatura anuala.
                lungimeSir - integer, lungimea sirului temperaturilor.
    Rezultate:pozitieInceput - integer, pozitia din sir de la care incepe cea mai lunga secventa de
    numere creascatoare.
                lungimeSecventa - integer, lungimea celei mai lungi secvente de numere crescatoare
                din sirul dat.
                maxim - double, numarul maxim din sir.}
procedure ceaMaiLungaSecventaCrescatoareSiMaximSir(sirTemperaturi: vector; lungimeSir: integer; var
lungimeSecventaMaxima: integer; var pozitieInceput: integer; var maxim: real);

```



```

var i, lungimeSecventa: integer;
    maximSecventa: real;
begin
    i := 1;
    while (i <= lungimeSir) do
    begin
        lungimeSecventa := 0;
        secventaCrescatoareMaximaIncepandDeLaPozitie(sirTemperaturi, lungimeSir, i, lungimeSecventa);
        if (lungimeSecventa > lungimeSecventaMaxima) then
        begin
            lungimeSecventaMaxima := lungimeSecventa;
            pozitieInceput := i;
        end;
        // maximul celei mai lungi secvente crescatoare va fi valoarea ultimului element din secventa
        maximSecventa := sirTemperaturi[i + lungimeSecventa - 1];
        if (maximSecventa > maxim) then
            maxim := maximSecventa;
        i := i + lungimeSecventa;
    end;
end;

{   Identifica maximele tuturor secventelor de temperaturi crescatoare.
Date:    sirTemperaturi - sir de elemente de tip double, reprezentand fiecare cate o
         temperatura anuala.
         lungimeSir - integer, lungimea sirului temperaturilor.
Rezultate:sirMaxime - sir de elemente de tip double, continand maximele tuturor secventelor de
         temperaturi crescatoare.
         lungimeSirMaxime - integer, lungimea sirului continand elementele maxime ale
         secventelor crescatoare.}
procedure maximePerioadeCrescatoare(sirTemperaturi: vector; lungimeSir: integer; var sirMaxime:
vector; var lungimeSirMaxime: integer);
var i, lungimeSecventa: integer;
    maximSecventa: real;
begin
    lungimeSirMaxime := 0;
    i := 1;
    while (i <= lungimeSir) do
    begin
        lungimeSecventa := 0;
        secventaCrescatoareMaximaIncepandDeLaPozitie(sirTemperaturi, lungimeSir, i, lungimeSecventa);

        // consideram doar acele perioade care contin cel putin 2 ani cu temperaturi crescatoare
        if (lungimeSecventa > 1) then
        begin
            // maximul celei mai lungi secvente crescatoare va fi valoarea ultimului element din
            secventa
            maximSecventa := sirTemperaturi[i + lungimeSecventa - 1];
            lungimeSirMaxime := lungimeSirMaxime + 1;
            sirMaxime[lungimeSirMaxime] := maximSecventa;
        end;

        i := i + lungimeSecventa;
    end;
end;

{   Tipareste secventa din sirul dat, care incepe de la pozitia data, avand lungimea data.
Date:    sir - sir de elemente de tip double.
         lungimeSir - integer, lungimea sirului.
         lungimeSecventa - integer, lungimea secventei de numere.
         pozitieInceput: integer, pozitia de la care trebuie tiparite numerele din sirul dat.
Rezultate: se afiseaza elementele subsecventei din sirul dat, care incepe de la pozitia
         pozitieInceput, si are lungimea lungimeSecventa.}
procedure tiparireSecventaDeLaPozitie(sir: vector; lungimeSir: integer; lungimeSecventa: integer;
pozitieInceput: integer);
var i: integer;
begin
    i := pozitieInceput;
    while (i < pozitieInceput + lungimeSecventa - 1) do
    begin

```

```

        write(sir[i]:2:2, ', ');
        i := i + 1;
    end;
    writeln(sir[i]:2:2);
end;

{   Tipareste cea mai lunga perioada in care temperaturile au crescut in continuu, precum si
    valorile temperaturilor asociate.
    Date:      sirTemperaturi - sir de elemente de tip double, reprezentand fiecare cate o
                temperatura anuala.
                lungimeSir - integer, lungimea sirului temperaturilor.
                lungimeSecventa - integer, lungimea secventei de numere.
                pozitieInceput: integer, pozitia de la care trebuie tiparite numerele din sirul dat.
                anInceput - integer, anul de la care sunt raportate temperaturile.
    Resultate: se afiseaza cea mai lunga perioada in care temperaturile au crescut in continuu,
                precum si valorile temperaturilor asociate.}
procedure tiparirePerioadaMaximaSiSecventa(sirTemperaturi: vector; lungimeSir: integer;
lungimeSecventa: integer; pozitieInceput: integer; anInceput: Word);
var anInceputPerioada, anSfarsitPerioada: Word;
begin
    // tiparire perioada
    anInceputPerioada := anInceput + pozitieInceput - 1;
    anSfarsitPerioada := anInceputPerioada + lungimeSecventa - 1;
    writeln('Cea mai lunga perioada in care temperaturile au crescut in continuu este: ',
anInceputPerioada, ' - ', anSfarsitPerioada);

    // tiparire secventa valori temperaturi
    tiparireSecventaDeLaPozitie(sirTemperaturi, lungimeSir, lungimeSecventa, pozitieInceput);
end;

// Programul principal
var sirTemperaturi, sirMaxime: vector;
    lungimeSir, lungimeSecventa, pozitieInceput, lungimeSirMaxime: integer;
    anInceput: Word;
    maxim: real;

begin
    citireTemperaturi(sirTemperaturi, lungimeSir, anInceput);

    // identificam cea mai lunga secventa in care temperaturile cresc in continuu si o afisam
    lungimeSecventa := 0;
    maxim := -100;
    pozitieInceput := 1;
    ceaMaiLungaSecventaCrescatoareSiMaximSir(sirTemperaturi, lungimeSir, lungimeSecventa,
pozitieInceput, maxim);
    tiparirePerioadaMaximaSiSecventa(sirTemperaturi, lungimeSir, lungimeSecventa, pozitieInceput,
anInceput);

    // tiparire temperaturi maxime, din fiecare perioada in care temperaturile au crescut si apoi
    au scazut
    lungimeSirMaxime := 0;
    maximePerioadeCrescatoare(sirTemperaturi, lungimeSir, sirMaxime, lungimeSirMaxime);
    write('Temperaturile maxime, din fiecare perioada in care temperaturile au crescut si apoi au
scazut: ');
    tiparireSecventaDeLaPozitie(sirMaxime, lungimeSirMaxime, lungimeSirMaxime, 1);
    writeln();

    // tiparire temperatura maxima
    writeln('Temperatura anuala maxima din perioada indicata este: ', maxim:2:2, '.');
end.

```

Temă propusă

Considerând problema 3, să se afișeze:

- cea mai lungă perioadă în care temperaturile au scăzut în continuu și valorile temperaturilor asociate;
- temperaturile minime, din fiecare perioadă în care temperaturile au scăzut și apoi au crescut;
- temperatura minima din toată perioada indicată.