

Probleme consultații 11 noiembrie 2017

Problema 1

Enunț

Fie i, j, k trei numere naturale, i și $k > 0$. Să scrie un subprogram care returnează restul împărțirii numărului (i^j) la k , deci (i^j) modulo k (iterativ și recursiv).

Antet subprogram (funcție) în **PASCAL**:

```
function Rest (i, j, k: integer) : integer;
```

Antet subprogram (funcție) în **C++**:

```
int Rest (int i, int j, int k)
```

Exemple:

(100^{100}) modulo 7 = 2

(125^{199}) modulo 999 = 800

(2^{10}) modulo 9 = 7

Analiză

- practic se înmulțesc resturile modulo k ; nu e nevoie de calculul expresiei i^j ;
- se calculează la fiecare pas de înmulțire restul produsului (modulo k);

Aceasta deoarece se știe că:

$$(a * b) \bmod c = (a \bmod c * b \bmod c) \bmod c$$

Atunci:

$$a^x \bmod c = (a \bmod c * a^{x-1} \bmod c) \bmod c$$

Specificarea funcției

Funcția **Rest(i, j, k)**:

Descriere: returnează restul împărțirii (i^j) la k .

Date: - i, j, k - numere naturale

Rezultate: R un număr natural: $R \in \{0, 1, \dots, k-1\}$

Implementare

Varianta iterativă Pascal

```
function Rest(i,j,k:integer):integer;           {functie ce determina restul}
var R,iModK:integer;
begin                                           {e suficient sa inmultim resturile}
    iModK:=i mod k;                            {expresia i Mod k e constanta in ciclu}
    R:=1;
    While (j>0) do
        begin
            R:=(R*(iModK)) mod k;              {atat lui i cat si produsului se aplica mod}
            j:=j-1;
        end;
    Rest:=R;
end;
```

Varianta recursivă Pascal

```
function Rest (i,j,k:integer):integer);    {recursiv, se autoapeleaza functia}
begin                                     {cat timp j>0}
    if (j>0) then Rest:= ((i mod k)*Rest(i,j-1,k)) % k
    else Rest:=1;                        {la final se inlocuieste cu 1, in}
end;                                       {sirul de produse.}
```

Varianta iterativă C++

```
int Rest(int i, int j, int k) {
    int iModK = i % k;           //expresia e constanta in ciclu
    int R = 1;
    while (j--)
        R = (R*iModK) % k;       //se aplica % lui i si produsului
    return R;
}
```

Varianta recursivă C ++

[illegible]

Exemple

Date de intrare	Rezultate
100, 100, 7	2
125, 199, 999	800
2, 10, 9	7
8, 0, 100	1
5, 151, 5	0

Problema 2

Enunț

Fie x, y două numere naturale. Să scrie un subprogram care returnează **true** dacă x și y sunt **asemenea** și **false** în sens contrar, fără a folosi tablouri. (Două numere naturale sunt **asemenea** dacă au aceleași cifre: 2131 e **asemenea** cu 32211 pentru că mulțimea cifrelor este aceeași: $\{1, 2, 3\}$).

Analiză

Problema are o rezolvare simplă dacă se folosește conceptul de vector de apariție pentru cifrele unui număr:

- se determină vectorii de apariție pentru x și y ;
- se compară apoi vectorii de apariție și se decide asemănarea.

Fără utilizarea vectorilor ar trebui să vedem dacă fiecare cifră a lui x este în y și invers.

Vom face 2 subprograme:

- Un subprogram care verifică dacă fiecare cifră a unui număr se află printre cifrele altui număr.
- Un subprogram care apelează apoi de 2 ori primul subprogram.

Antete subprograme în **PASCAL**:

```
function CifreAinB (A,B:longint):boolean;  
function Asemenea (X,Y:longint):boolean;
```

Antete subprograme în **C++**:

```
bool CifreAinB (long A,long B)  
bool Asemenea (long X,long Y)
```

Specificarea subalgoritmilor

Procedura **CifreAinB(A, B)**:

Descriere: verifică dacă fiecare cifră a lui A este în mulțimea cifrelor lui B.

Date: - $A, B > 0$ numere naturale

Rezultate: true, dacă mulțimea cifrelor lui A este inclusă în mulțimea cifrelor lui B;
false, în caz contrar

Procedura **Asemenea(X, Y)**:

Descriere: verifică dacă X și Y sunt asemenea.

Date: - $X, Y > 0$ numere naturale

Rezultate: true, dacă numerele date X și Y sunt asemenea (adică dacă CifreAinB(X,Y) este true **ȘI** CifreAinB(Y,X) este true);
false, în caz contrar

Implementare

Varianta Pascal

```
function CifreAinB(A,B:longint):boolean;
var CopieB :longint;
    UcifA :byte;
begin
    CopieB :=B;                                {se retine o clona a lui B in CopieB}
    CifreAinB:=true;                            {presupunem ca incluzinea exista}
    while (A>0) do
    begin
        UcifA:=A mod 10;                        {verifica fiecare cifra a lui A daca e B}
        B :=CopieB;                            {la fiecare cifra a lui A se reface B}
        while (B>0) and (UcifA<>(B mod 10)) do B:=B div 10;
        if (B=0)                               {daca B=0 => cifra curenta lui A nu e in B}
        then
            begin
                CifreAinB:=false;               {se pune numele functiei pe false si A pe 0}
                A :=0;                          {pentru a iesi din primul ciclu while}
            end
        else A:=A div 10;                       {daca B>0, cifra curenta a lui A e in B}
    end;                                        {se trece la următoarea cifra a lui A}
end;

function Asemenea(X,Y:longint):boolean;
begin
    if (CifreAinB(X,Y)) and (CifreAinB(Y,X))
    then Asemenea:=true
    else Asemenea:=false;
end;
```

Varianta C++ (varianta 1)

```
bool CifreAinB(long A, long B){
    long CopieB;
    short UcifA;
    CopieB=B;                                // retine o clona a lui B in CopieB
    while (A>0) {
        UcifA = A % 10;                      // verifica fiecare cifra a lui A daca e in B
        B = CopieB;                          // la fiecare cifra a lui A se reface B
        while (B>0 && UcifA != (B %10))
            B = B/10;
        if (B==0) return false;              // daca B=0 => cifra curenta lui A nu e in B
        A = A /10;                           // daca B>0, cifra curenta a lui A e in B
    }
```

```

    } // se trece la următoarea cifra a lui A
    return true; // la iesire din while fiecare cifra a lui A
} // este in B
bool Asemenea(long X, long Y){
    return CifreAinB(X,Y) && CifreAinB(Y,X);
}

```

Varianta C++ (varianta 2) O varianta factorizata

```

bool CifraInB(short Cifra, long B){ // verifica daca o cifra e in B
    while (B>0 && (Cifra!=(B%10)))
        B=B/10;
    if(B==0) return false; // daca B=0 => cifra nu e in B
    return true; // cifra e in B
}
bool CifreAinB(long A, long B){ // verifica fiecare cifra a lui A daca e in B
    while (A>0) {
        if (!CifraInB(A%10,B)) return false; // cifra curenta lui A nu e in B
        A=A /10;
    }
    return true; // fiecare cifra a lui A este in B
}
bool Asemenea(long X, long Y){
    if (CifreAinB(X,Y) && CifreAinB(Y,X)) return true;
    return false;
}

```

Exemple

Date de intrare	Rezultate
1222331, 123	true
122235, 123	false
5656565, 56	true
5656565, 5	false
1, 8	false

Problema 3

Enunț

Să se scrie un subprogram care să afișeze perechile de numere naturale „prietene” dintr-un interval dat $[a,b]$, a, b – numere naturale, $a < b$. Două numere naturale x și y sunt „prietene” dacă suma divizorilor lui x este egală cu suma divizorilor lui y . Pentru un număr, se vor considera toți divizorii, inclusiv 1 și numărul însuși.

Analiză

Vom scrie un subprogram care verifică dacă x și y din $[a, b]$ au aceeași sumă s a divizorilor. Avem nevoie și de un subprogram care determină suma divizorilor. Vom realiza acest subprogram cu cea mai bună complexitate, adică $O(\sqrt{n})$.

Specificarea subalgoritmilor

- Subalgoritmul **SumaDiv(n)**:
Descriere: Calculează suma divizorilor numărului dat.
Date: n – număr natural.
Rezultate: s – număr natural, reprezentând suma divizorilor numărului dat.
- Subalgoritmul **AfisPrietene(a, b)**:
Descriere: Afișează toate numerele prietene din intervalul $[a, b]$.
Date: a, b – numere naturale, $a < b$
Rezultate: Se afișează toate numerele prietene din intervalul dat.

Antete subprograme în **PASCAL**:

```
function SumaDiv (n:longint):longint;  
procedure AfisPrietene (a,b:longint);
```

Antete subprograme în **C**:

```
long SumaDiv (long n);  
void AfisPrietene (long a, long b);
```

Implementare

Varianta C++

```
long sumDiv(long n)  
{long S = n + 1; //initializarea sumei S cu n+1  
  long d = 2;  
  for (d = 2; d*d<n; d++) //se verifica pentru d=2,3,...,rad(n)-1  
    if (n%d == 0) S += d + n / d; //se aduna factorii complementari  
  if (d*d == n) S += d; //daca n este patrat perfect se mai aduna  
  return S; //d, atentie la iesire din for d>=rad(n)  
}
```

O variantă recursivă a acestui subprogram:

```
long SumDivRec(long n, long d)      //e nevoie si de d, la apel d=2
{ if (d*d>n) return 1 + n;          //la iesire se inlocuieste apelul cu n+1
  //daca n este patrat perfect
  if (d*d == n) return d + SumDivRec(n, d + 1);
  //se verifica pentru d=2..rad(n)
  //se aduna factorii complementari
  if (n%d == 0) return d + (n / d) + SumDivRec(n, d + 1);
  return SumDivRec(n, d + 1);      //daca d nu e divizor
}
```

Apel: SumDiv(n, 2)

```
long sumDiv(long n)
{
    return SumDivRec(n, 2);
}
```

```
void AfisPrietene(long a, long b)
{   for(long x=a; x<b; x++)
    for(long y=a+1; y<=b; y++)
        if(SumaDiv(x)==SumDiv(y)) cout<<x<<" si "<<y<<" prietene \n";
}
```

Varianta Pascal

```
function SumDiv(n: longint): longint;
var s, d: longint;
begin
    s:= n + 1;
    d := 2;
    while (d*d < n) do
    begin
        if (n mod d = 0) then
        begin
            if (n / d = d) then s := s + d
            else s := s + d + n div d;
        end;
        d := d + 1;
    end;
    if (d*d = n) then s := s + d;
    sumDiv := s;
end;

procedure AfisPrietene(a, b: longint);
var i, j: longint;
begin
    for i := a to b do
        for j := i + 1 to b do
            if (SumDiv(i) = SumDiv(j)) then writeln(i, ' ', j);
end;
```


Exemple

Date de intrare	Rezultate
1, 20	6, 11 10, 17 14, 15
15, 25	15, 23 16, 25
1, 10	-
100, 140	102, 110 104, 116 114, 135 115, 119 132, 140
1023, 1050	1030, 1035

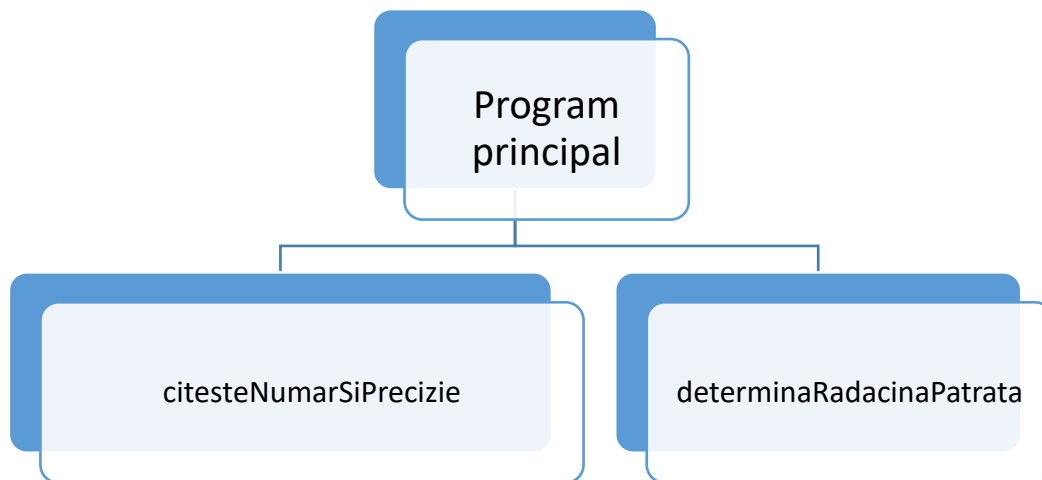
Problema 4

Enunț

Dat fiind un număr real x , să se determine rădăcina pătrată r a acestuia, cu precizia *epsilon*.

Analiză

Se va folosi metoda înjumătățirii intervalului: pornind de la un interval inițial pentru rădăcina pătrată, se selectează iterativ subintervalul în care se află rădăcina, până când se ajunge la o lungime maximă a intervalului mai mică decât precizia cerută. Precizia epsilon poate fi $10^{-k} = 0.0\dots1$ cu $(k-1)$ de zero (adică cu k zecimale exacte). Exemplu epsilon: $\epsilon = 10^{-3} = 0.001$.



Specificarea subalgoritmilor

- Subalgoritmul **citesteNumarSiPrecizie(x, eps)**:
 - Descriere*: Citește două numere reale.
 - Date*: -
 - Rezultate*: x , eps – numere reale, $x > 0$.
- Subalgoritmul **determinaRadacinaPatrata(n, eps)**:
 - Descriere*: Determina rădăcina pătrată a unui număr dat, folosind o Precizie dată.
 - Date*: x , eps – numere reale, $x > 0$
 - Rezultate*: rădăcina pătrată a lui x , cu Precizia eps .

Implementare

Varianta Pascal

```
procedure citesteNumarSiPrecizie(var x: real; var eps: real);
begin
    writeln('Introduceti numarul real si apoi Precizia: ');
    readln(x, eps);
end;

function determinaRadacinaPatrataIterativ(x, eps: real): real;
var dreapta, stanga, mij: real;
begin
    dreapta := x;
    stanga := 0;
    mij := (dreapta + stanga) / 2;

    while (abs(dreapta - stanga) > eps) do
    begin
        if (mij * mij > x) then
            dreapta := mij
        else
            stanga := mij;

        mij := (dreapta + stanga) / 2;
    end;

    determinaRadacinaPatrataIterativ := mij;
end;

function determinaRadacinaPatrataRec(x, eps, stanga, dreapta: real): real;
var mij: real;
begin
    mij := (dreapta + stanga) / 2;
    if (abs(dreapta - stanga) < eps) then
        determinaRadacinaPatrataRec := mij
    else
        if (mij * mij > x) then
            determinaRadacinaPatrataRec := determinaRadacinaPatrataRec(x, eps,
                stanga, mij)
        else
            determinaRadacinaPatrataRec := determinaRadacinaPatrataRec(x, eps, mij,
                dreapta);
    end;
end;

function determinaRadacinaPatrataRecurziv(x, eps: real): real;
begin
    determinaRadacinaPatrataRecurziv := determinaRadacinaPatrataRec(x, eps, 0,
        x);
end;
```

```

var x, eps: real;
begin
    citesteNumarSiPrecizie(x, eps);
    writeln('Iterativ: ');
    writeln('Radacina patrata cu Precizia data este: ',
determinaRadacinaPatrataIterativ(x, eps));
    writeln('Recurziv: ');
    writeln('Radacina patrata cu Precizia data este: ',
determinaRadacinaPatrataRecurziv(x, eps));
end.

```

Varianta C++

```

/*
Descriere: Citeste doua numere reale, reprezentand numarul caruia trebuie sa ii fie
calculata radacina patrata si Precizia (eroarea).
Date: -
Rezultate: se citesc cele doua numere.
*/
void citesteNumarSiPrecizie(double& x, double& eps)
{
    cout << "Introduceti numarul real si apoi precizia: ";
    cin >> x >> eps;
}

/*
Descriere: Determina radacina patrata a unui numar dat, folosind o Precizie data.
Date: x, eps - numere reale.
Rezultate: radacina patrata a lui x, cu Precizia eps.
*/
double determinaRadacinaPatrataIterativ(double x, double eps)
{
    double dreapta = x;
    double stanga = 0;
    double mij = (dreapta + stanga) / 2;
    while (abs(dreapta - stanga) > eps)
    {
        if (mij * mij > x) dreapta = mij;
        else stanga = mij;
        mij = (dreapta + stanga) / 2;
    }
    return mij;
}
double determinaRadacinaPatrataRec(double x, double eps, double stanga, double dreapta)
{
    double mij = (dreapta + stanga) / 2;

    if (abs(dreapta - stanga) < eps)
        return mij;

    if (mij * mij > x)
        return determinaRadacinaPatrataRec(x, eps, stanga, mij);

    return determinaRadacinaPatrataRec(x, eps, mij, dreapta);
}

```

```

/*
    Descriere: Determina radacina patrata a unui numar dat, folosind o Precizie data.
    Date: x, eps - numere reale.
    Rezultate: radacina patrata a lui x, cu Precizia eps.
*/
double determinaRadacinaPatrataRecurziv(double x, double eps)
{
    return determinaRadacinaPatrataRec(x, eps, 0, x);
}

int main()
{
    double x = 0, eps = 0;
    citesteNumarSiPrecizie(x, eps);
    cout << "Iterativ: ";
    cout << "Radacina patrata a numarului " << x << ", cu Precizia " << eps << "
este: " << determinaRadacinaPatrataIterativ(x, eps) << endl;
    cout << endl;
    cout << "Iterativ: ";
    cout << "Radacina patrata a numarului " << x << ", cu Precizia " << eps << "
este: " << determinaRadacinaPatrataRecurziv(x, eps) << endl;
    return 0;
}

```

Exemple

Date de intrare	Rezultate
5, 0.01	2.23145
5, 0.0001	2.23606
100, 0.0000001	10
100, 0.1	10.0098
2, 0.01	1.41797

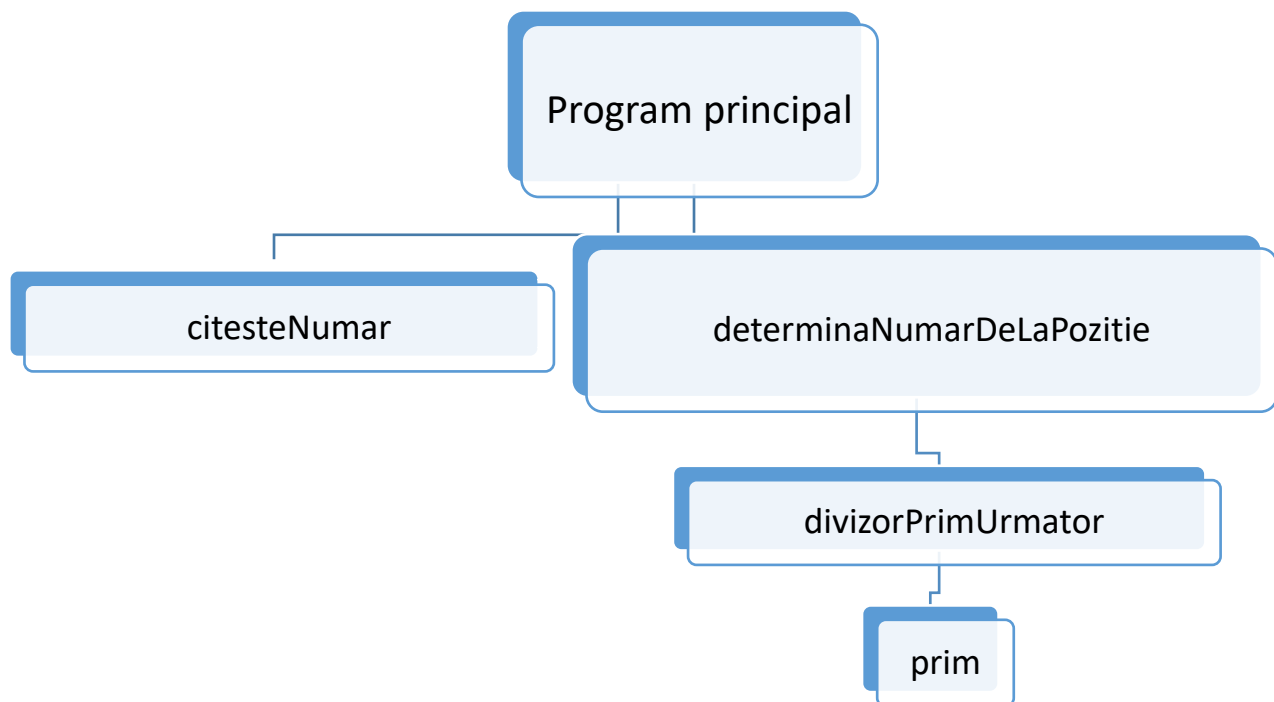
Problema 5

Enunț

Fie șirul $X = (1, 2, 3, 2, 2, 5, 2, 2, 3, 3, 3, 7, 2, 2, 3, 3, 3, 2, 2, 5, 5, 5, 5, 5, \dots)$, obținut din secvența numerelor naturale, în care numerele compuse sunt înlocuite cu divizorii lor primi, fiecare divizor d fiind scris de d ori. Scrieți un algoritm care determină valoarea elementului de pe poziția k , fără a păstra șirul în memorie (k – număr natural). Indexarea șirului începe de la 0.

Analiză

Identificarea subalgoritmilor



Specificarea subalgoritmilor

- Subalgoritmul **citesteNumar()**:
Descriere: Citește și returnează un număr întreg.
Date: -
Rezultate: n – număr întreg, n – este numărul citit.
- Funcția **prim(n)**:
Descriere: Verifică dacă un număr dat este număr prim.
Date: n – număr natural
Rezultate: *True* – dacă numărul dat este prim
False – altfel

- Funcția **divizorPrimUrmator(n, div)**:

Descriere: Găsește, pentru numărul n , primul divizor prim după div și mai mic decât n . Dacă un astfel de divizor nu există, se returnează -1.

Date: n, div – numere naturale.

Rezultate: i – primul divizor prim al lui n , mai mare decât div , dacă un astfel de divizor există.

-1 – dacă un astfel de divizor nu există.

- Funcția **determinaNumarDeLaPozitie (poz)**:

Descriere: Determină numărul din șirul X (format după cum se specifică în enunțul problemei), de la poziția poz .

Date: poz – număr natural.

Rezultate: rez – numărul de pe poziția poz , din șirul X (format după cum se specifică în enunțul problemei).

Implementare

Varianta Pascal

```
function citesteNumar: integer;
var n: integer;
begin
    writeln('Introduceti numarul: ');
    readln(n);
    citesteNumar := n;
end;

function prim(n: integer): boolean;
var d: integer; ok: boolean;
begin
    ok := true;
    if (n < 2) then ok := false
    else
        begin
            if ((n > 2) and (n mod 2 = 0)) then
                ok := false
            else
                begin
                    d := 3;
                    while (d * d <= n) do
                        begin
                            if (n mod d = 0) then ok := false;
                            d := d + 2;
                        end;
                    end;
                end;
            end;
        end;
    prim := ok;
end;

function divizorPrimUrmator(n, d: integer): integer;
```

```

var i, rez: integer;
begin
    if (prim(n)) then rez := -1
    else
        begin
            i := d + 1;
            while ((not prim(i)) or (n mod i <> 0)) and (i <> n) do
                i := i + 1;
            if (i = n) then    rez := -1
                else    rez := i;
        end;
        divizorPrimUrmator := rez;
    end;
function determinaNumarDeLaPozitie(poz: integer): integer;
var rezultat, numarCurentInSecventa, pozitieCurenta, divUrm: integer;
begin
    if (poz < 3) then rezultat := poz + 1
    else
        begin
            rezultat := 0;
            numarCurentInSecventa := 3;
            pozitieCurenta := 2;
            while (poz > pozitieCurenta) do
                begin
                    numarCurentInSecventa := numarCurentInSecventa + 1;
                    if (prim(numarCurentInSecventa)) then
                        {daca numarul este prim, pozitia va creste cu 1}
                        begin
                            pozitieCurenta := pozitieCurenta + 1;
                            rezultat := numarCurentInSecventa;
                        end
                    else
                        begin
                            divUrm := divizorPrimUrmator(numarCurentInSecventa, 1);
                            {se iau pe rand divizorii primi ai numarului}
                            while ((poz > pozitieCurenta) and (divUrm <> -1)) do
                                begin
                                    pozitieCurenta := pozitieCurenta + divUrm;
                                    {valoarea divUrm va aparea de divUrm ori in sir,}
                                    {deci se pot sari divUrm pozitii}
                                    rezultat := divUrm;
                                    {toate aceste pozitii vor contine valoarea divUrm}
                                    divUrm := divizorPrimUrmator(numarCurentInSecventa, divUrm);
                                end;
                            end;
                        end;
                    end;
                end;
            rezultat := rezultat;
        end;
    end;
    determinaNumarDeLaPozitie := rezultat;
end;
var n: integer;

```



```

begin
    n := citesteNumar();
    writeln('Numarul din sirul X de la pozitia data este: ',
determinaNumarDeLaPozitie(n));
end.

```

Varianta C++

```

#include <iostream>

using namespace std;

/*
Descriere: Citeste si returneaza un numar natural.
Date: -
Rezultate: se returneaza numarul citit.
*/
int citesteNumar()
{
    int n = 0;
    cout << "Introduceti un numar natural: ";
    cin >> n;
    return n;
}

/*
Descriere: Verifica daca numarul dat este prim.
Date: n - numar natural
Rezultate: True - daca numarul dat este prim
           False - altfel
*/
bool prim(int n)
{
    if (n < 2)        return false;           // numerele mai mici ca 2 nu sunt prime
    if (n == 2)       return true;            // 2 este prim
    if (n % 2 == 0)   return false;           // numerele pare nu sunt prime
    for(int div = 3; div * div <= n; div+=2)   // verificare pentru n impar
        if (n % div == 0) return false;       // iesire n nu e prim
    return true;      // la finalul lui for n e prim
}

/*
Descriere: Gaseste, pentru numarul n, primul divizor prim dupa div. Daca un astfel de
           divizor nu exista, se returneaza -1.
Date:      n, div - numere naturale.
Rezultate: i - primul divizor prim al lui n, mai mare decât div, daca un astfel de
           divizor exista.
           -1 - daca un astfel de divizor nu exista.
*/
int divizorPrimUrmator(int n, int div)
{
    if (prim(n))      return -1;
    int i = div + 1;
    while ((!prim(i) || (n % i != 0)) && i != n)
        i++;
    if (i == n)        return -1;
    return i;
}

/*

```

Descriere: Determina numarul din sirul X (format dupa cum se specifica in enunțul problemei), de la pozitia poz.
Date : poz - numar natural.
Rezultate : rez - numarul de pe pozitia poz, din sirul X (format dupa cum se specifica in enunțul problemei).

*/

```
int determinaNumarDeLaPozitie(int poz)
{
    if (poz < 3) return poz + 1;
    int rezultat = 0;
    int numarCurentInSecventa = 3;
    int pozitieCurenta = 2;
    while (poz > pozitieCurenta)
    {
        numarCurentInSecventa++;
        if (prim(numarCurentInSecventa))                // daca numarul este prim, pozitia
                                                         // va creste cu 1
        { pozitieCurenta += 1;
          rezultat = numarCurentInSecventa;
        }
        else
        {
            int divUrm = divizorPrimUrmator(numarCurentInSecventa, 1);
                                                         // se iau pe rand divizorii primi ai numarului
            while (poz > pozitieCurenta && divUrm != -1)
            { pozitieCurenta += divUrm;
              // valoarea divUrm va aparea de divUrm ori in
              // sir, deci se pot sari divUrm pozitii
              rezultat = divUrm;
              // toate aceste pozitii vor contine valoarea
              // divUrm
              divUrm = divizorPrimUrmator(numarCurentInSecventa, divUrm);
            }
        }
    }

    return rezultat;
}

int main()
{ /*for (int i = 0; i < 100; i++)
   cout << i << " " << determinaNumarDeLaPozitie(i) << endl;*/

    int poz = citesteNumar();
    cout << "Numarul din sirul X de pe pozitia " << poz << " este: "
          << determinaNumarDeLaPozitie(poz) << ".";
    system("pause");
    return 0;
}
```

Exemple

Date de intrare	Rezultate
1	2
8	3
11	7
20	5
30	13
34	7
35	7
77	11

Probleme tip grilă

1. Știind că variabilele a și i sunt întregi, stabiliți ce reprezintă valorile afișate de algoritmul de mai jos. S-au folosit notațiile $x\%y$ pentru restul împărțirii numărului întreg x la numărul întreg y , și $[x]$ pentru partea întreagă a numărului real x .

```
a ← 10
pentru i ← 1,6 execută
    scrie [a/7]
    a ← a % 7 * 10
sfârșit pentru
```

- a. primele 6 zecimale ale lui $1/7$
- b. primele 7 zecimale ale lui $1/6$
- c. primele 6 cifre ale lui $10/7$ (CORECT)
- d. primele 7 zecimale ale lui $10/6$

2. Ce va afișa algoritmul pseudocod de mai jos pentru două numere naturale nenule a și b ? S-a notat cu $x\%y$ restul împărțirii numerelor întregi x și y .

```
citește a,b
c ← 1
cât-timp a * c % b ≠ 0 execută
    c ← c + 1
sfârșit-cât-timp
scrie a*c
```

- a. a^b
- b. cel mai mic multiplu comun (CORECT)
- c. cel mai mare divizor comun
- d. $a*b$

3. Se consideră funcția de mai jos, care primește ca parametru un număr natural.

```
int f(int a)
{
    int x = a % 10;

    if (x == a)
        if (x % 2 == 0)
            return a;
        else
            return 0;

    if (x % 2 == 0)
        return 10 * f((int)(a/10)) + x;
    return f((int)(a/10));
}
```

- I. De câte ori se apelează funcția pentru $a = 253401976$?
- a. De 3 ori
 - b. De 9 ori (CORECT)
 - c. De 6 ori
 - d. De 7 ori
- II. Care este rezultatul funcției pentru $a = 253401976$?
- a. 206
 - b. 53019
 - c. 2406 (CORECT)
 - d. 0