Pregătire pentru examenul de Bacalaureat şi examenul de admitere la Facultatea de Matematică şi Informatică

Probleme propuse pentru sesiunea 1

16 decembrie 2017

INFORMATICA

Tema 1: Algoritmi de prelucrare a datelor simple:

- Prelucrări asupra cifrelor (extragerea şi analiza cifrelor unui număr natural)
- Proprietăţi de divizibilitate (verificare divizibilitate, determinarea celui mai mare divizor comun, verificare şi generare numere prime, descompunere în factori primi)
- Calcule de sume şi produse
- Generare şiruri date prin relaţii de recurenţă

1. Fie *n* un număr natural nenul. Descrieţi algoritmi pentru:
   a. Determinarea sumei tuturor cifrelor lui *n*. De exemplu, pentru *n=26326* se obţine valoarea *19*.

Se determină suma cifrelor unui număr prin extragerea pe rând a ultimei cifre a numărului prin n mod 10 şi adunarea acesteia într-o variabilă, apoi se va elimina din număr cifra extrasă prin n div 10; procesul se va repeta până la epuizarea tuturor cifrelor.

```
întreg n, s=0;
cât timp n<>0 execută
        s←s+ n mod 10;
        n←n div 10;
sfârşit_cât_timp;
```

```
int suma_cifre(int n)
{
   int s=0;
   while (n)
   {
      s+=n%10;
      n/=10;
   }
   return s;
}
```

   b. Determinarea valorii obţinute prin inversarea cifrelor numărului n. De exemplu, pentru valoarea *26326* se obţine valoarea *62362*.

Se extrag pe rând cifrele din număr (începând cu cifra unităților) și se construiește numărul inv=inv*10+cifra

```
întreg n, inv;
cât timp n<>0 execută
```


Blvd. V. Pârvan 4, 300223, Timişoara, România
Tel: +40-(0)256-592.155
Email: mateinfo@e-uvt.ro
www.math.uvt.ro

```
    inv←inv*10 + n mod 10;
    n←n div 10;
sfârșit_cât_timp;
```

```
long nr_inversat(int n)
{
  long inv;
  inv=0;
  while(n)
  {
    inv=inv*10+n%10;
    n/=10;
  }
  return inv;
}
```

c. Determinarea tuturor cifrelor binare ale lui *n*.

Se determină fiecare cifră a numărului și se testează dacă este egală cu 0 sau 1, caz în care variabila auxiliară test va lua valoarea 1. Daca valoarea lui test rîmâne 0 (neschimbată de la inițializarea făcută) se va afișa mesajul "nr nu are cifre binare".

```
întreg n, test=0;
cât timp n<>0 execută
        dacă (n mod 10 =0 sau n mod 10 =1)
           atunci   scrie n mod 10;
                    test ←1
        sfîrșit_dacă
        n←n div 10;
sfârșit_cât_timp;
```

```
void cifre_binare(int x)
{
  int n=x,test=0; // test=0 ne va indica ca n nu are cifre de 0 sau 1
  while (n)
  {
    if(n%10==0 or n%10==1)
      {
        cout<<n%10<<endl;  //se tipareste fiecare cifra binara
        test=1;
      }
    n/=10;
  }
  if (!test) cout<<"nr nu are cifre binare"<<endl;
}
```


Blvd. V. Pârvan 4, 300223, Timişoara, România
Tel: +40-(0)256-592.155
Email: mateinfo@e-uvt.ro
www.math.uvt.ro

d. Determinarea tuturor divizorilor proprii ai lui *n*.

Se verifică dacă n este divizibil cu d și în caz afirmativ se afișează d. Se repetă acțiunea de la d=2 până la d= n div 2.

```
întreg n, d;
pentru d←2, n div 2, pas 1  execută
      dacă (n mod d =0)
                  atunci   scrie d;
      sfârșit_dacă
sfârșit_pentru;
```

```
void divizori_proprii(int n)
{
   int d;
   for(d=2;d<=n/2;d++)
   //d este initalizat cu 2 (primul divizor propriu posibil)
   //instructiunea merge pana la n/2 deoarece nu pot exista divizori proprii intre n/2 si n
   if(n%d==0) //conditia ca d sa fie divizor propriu al lui d
         cout<<d<<" "; // se afiseaza pe rand divizorii proprii ai lui n
}
```

e. Afișarea mesajului „da” dacă cea mai semnificativă cifră a unui număr n este strict mai mare decât cifra unităţilor sau mesajul „nu” în caz contrar. De exemplu, pentru *n=5832* se va afișa „da” pentru că *5>2*, iar pentru *n=4539* se va afișa „nu”.

Se calculează ultima cifră (n mod 10), apoi se calculează prima cifră.
Se compară cifrele obținute: dacă primă cifră este strict mai mare decât ultima cifră se va afișa DA, in caz contrar NU.

```
întreg n, c1,cu;
cu←n mod 10;
cât timp n<>0 execută;
            c1←n mod 10;
            n←n div 10;
sfârşit_cât_timp;
dacă c1>cu
            atunci scrie “DA”
            altfel scrie “NU”
sfârșit_dacă
```

```
 void test_max(int n)
{
  int c1; // c1 variabila ce retine prima cifra
  int cu=n%10; // cu variabila ce retine ultima cifra
   cout<<"ultima cifra"<<cu<<endl;
   while (n)
   {
      c1=n%10;
      n/=10;
```


Blvd. V. Pârvan 4, 300223, Timişoara, România
Tel: +40-(0)256-592.155
Email: mateinfo@e-uvt.ro
www.math.uvt.ro

```
    }
    cout<<"prima cifra"<<c1<<endl;
    if (c1>cu) cout<<"DA"<<endl;
        else cout<<"NU"<<endl;
}
```

2. *Cifra destinului*. Cifra destinului este cifra obţinută prin adunarea cifrelor ce intervin în data naşterii; adunarea cifrelor rezultatului obţinut se repetă pâna se ajunge la o singură cifră. De exemplu, pentru 01.02.1999 se obţine: 1+2+1+9+9+9=31 → 3+1=4. Descrieţi un algoritm care calculează cifra destinului pornind de la data naşterii specificată prin cele trei valori (zi, luna, an).

   Se determină suma cifrelor unui număr prin extragerea pe rând a ultimei cifre a numărului prin n mod 10 şi adunarea acesteia într-o variabilă, apoi se va elimina din număr cifra extrasă prin n div 10; procesul se va repeta până la epuizarea tuturor cifrelor. Acest algoritm se va aplica pentru cele trei valori (zi, luna, an), se va face suma și la sfîrșit algoritmul se va aplica și pentru valoarea sumei.

```
întreg n, s=0;
cât timp n<>0 execută
        s←s+ n mod 10;
        n←n div 10;
sfârșit_cât_timp;


întreg zi, luna, an, suma;
citește zi, luna, an;
suma ← SumaCifre(zi)+SumaCifre(luna)+SumaCifre(an);
cât timp suma>=10 execută
     suma ← SumaCifre(suma)ș
sfârșit_cât_timp;
tipărește suma;


#include <iostream>
using namespace std;
unsigned int SumaCifre(unsigned long int a)
{
   unsigned int s=0;
   while(a!=0)
   {
      s=s+a%10;
      a=a/10;
   }
   return s;
}
int main()
{
```


Blvd. V. Pârvan 4, 300223, Timişoara, România
Tel: +40-(0)256-592.155
Email: mateinfo@e-uvt.ro
www.math.uvt.ro

```
    unsigned long int zi, luna,an;
    unsigned int suma;
    cout<<"ziua nasterii ";
    cin>>zi;
    cout<<"luna nasterii ";
    cin>>luna;
    cout<<"anul nasterii ";
    cin>>an;
    cout<<endl;
    suma=SumaCifre(zi)+SumaCifre(luna)+SumaCifre(an);
    while(suma>=10)
       suma=SumaCifre(suma);
    cout<<"Cifra destinului este : "<<suma;
    cout<<endl;
    return 0;
}
```

3. *Numere asemenea*. Două numere naturale sunt asemenea dacă scrierile celor două numere în baza *10* au aceleași cifre. De exemplu, numerele *23326* și *623* sunt asemenea, deoarece mulțimea cifrelor este aceiași (*{2,3,6}*). Descrieți un algoritm care preia o pereche de numere naturale și determină dacă sunt sau nu asemenea.

Algoritmul va testa dacă fiecare cifră din numărul n1 se găsește printre cifrele numărului n2. În caz în care s-a găsit o cifră în n1 care nu se află printre cifrele numărului n2, atunci numerele nu sunt asemenea.

```
întreg n1,n2,auxa,aux2,test;
citește n1,n2;
 aux1← n1, aux2← n2;
 cât timp aux1<>0 execută
        test← 0;
        cât timp aux2<>0 execută
                dacă aux1 mod 10 = aux2 mod 10
                      atunci
                           test← 1;
                           break;
                sfârșit_dacă;
                aux2←aux2 div 10;
      sfârșit_cât_timp;
      dacă test = 0
                atunci break;
                altfel
                        aux2 ← n2;
                        aux1← aux1 div 10;
     sfârșit_dacă;
  sfârșit_cât_timp;
  dacă test=1
```


Blvd. V. Pârvan 4, 300223, Timişoara, România
Tel: +40-(0)256-592.155
Email: mateinfo@e-uvt.ro
www.math.uvt.ro

atunci tipărește “numere asemenea”
altfel tipărește “nu sunt numere asemenea”
sfârșit_dacă

```
#include <iostream>
using namespace std;
int main()
{
    int n1,n2,aux1,aux2,c1,c2, test;
    cout<<"n1=";
    cin>>n1;cout<<endl;
    cout<<"n2=";
    cin>>n2;cout<<endl;
    aux1=n1;aux2=n2;
    while (aux1)
    {
      c1=aux1%10;
      test=0;
      while (aux2)
      {
      c2=aux2%10;
      if (c1==c2)
         {test=1;
          break;
         }
      aux2/=10;
      }
      if(!test) break;
      aux2=n2;
      aux1/=10;
    }
    if(test) cout<<"Numerele sunt asemenea "<<n1<<"  "<<n2<<endl;
      else cout<<"Numerele Nu sunt asemenea "<<n1<<"  "<<n2<<endl;
     return 0;
}
```

4. *Conversie între baze de numeraţie.* Se consideră că numărul natural nenul *n* este reprezentat în baza de numeraţie *b1* (*2≤ b1 ≤ 10*) şi se doreşte construirea numărului natural *m* care reprezintă aceeaşi valoare însă în baza *b2* (*2≤ b2 ≤ 10*). Atât numărul iniţial cât şi cel convertit sunt specificate prin variabile întregi (nu este posibilă utilizarea unui tablou pentru stocarea cifrelor). De exemplu pentru *n=2210* şi *b1=3*, *b2=5* se obţine *m=300*.

Pentru a realiza conversia unui număr din baza b1 in baza b2 vom proceda astfel:
- transformăm numărul din baza b1 in baza 10; apoi
- transformăm numărul obținut anterior din baza 10 in baza b2.

Pentru transformarea numărului din baza b1 în baza 10 se determină fiecare cifră a numărului n și se generează numărul prin

$n=x_{k-1}x_{k-2}\ldots x_1x_0 \Rightarrow nr= x_{k-1}*b^{k-1}+x_{k-2}*b^{k-2}+\ldots x_1*b^1+x_0*b^0$

Tranformarea numărului nr obținut astfel din baza 10 în baza b2 se va face tot prin determinarea fiecărei cifre a numărului și generarea numărului m prin

Blvd. V. Pârvan 4, 300223, Timişoara, România
Tel: +40-(0)256-592.155
Email: mateinfo@e-uvt.ro
www.math.uvt.ro

$nr=c_{k-1}c_{k-2}\ldots c_1c_0 \Rightarrow m= c_{k-1}*b^{k-1}+c_{k-2}*b^{k-2}+\ldots c_1*b^1+c_0*b^0$

```
citește n,b1,b2;
// transformăm numărul n din baza b1 în baza 10
p←1;
m←0;
cât_timp n<>0 execută
      r←n mod 10;
      m←m+r*p;
      p←p*b1;
      n←n div 10;
sfârșit_cât_timp;

//conversia lui m din baza 10 in b2
 p←1;
 n←0;
 cât_timp m<>0 execută
      r← m mod b2;
      n← n+r*p;
      p← p*10;
      m←m div b2;
 sfârșit_cât_timp;
```

```
#include <iostream>
using namespace std;
int main()
{
   long n,m,r,p,x,b1,b2;
   cout<<"Numarul de transformat n=";
   cin>>n;
   do
   {
   cout<<"Baza b1 in care este scris numarul =";
   cin>>b1;
    }while(b1<2 || b1>10);
   do
   {
   cout<<"Baza b2 in care se  face conversia =";
   cin>>b2;
   }while(b2<2 || b2>10);
   x=n;
   //conversia lui n din b1 in baza 10
   p=1;
   m=0;
   while(n!=0)
     {r=n%10;
      m=m+r*p;
      p=p*b1;
      n=n/10;
     }
    cout<<"numarul in baza 10 este "<<m<<endl;
    //conversia lui m din baza 10 in b2
```


Blvd. V. Pârvan 4, 300223, Timişoara, România
Tel: +40-(0)256-592.155
Email: mateinfo@e-uvt.ro
www.math.uvt.ro

```
    p=1;
    n=0;
    while(m!=0)
     {r=m%b2;
      n=n+r*p;
      p=p*10;
      m=m/b2;
     }
    cout<<"numarul in baza "<<b2<<" este "<<n;
    cout<<endl;
    return 0;
 }
```

5. *Reguli de codificare.* Se consideră un număr natural *n* constituit din $k \geq 2$ cifre ($n=c_k c_{k-1} \ldots c_1$) şi se pune problema codificării lui *n* prin schimbarea poziţiei unor cifre fără a stoca cifrele lui n într-un tablou. Descrieţi algoritmi pentru următoarele două variante de codificare:

   a. *Permutare circulară a cifrelor.* Fiind dată o valoare *p* ($1 \leq p \leq k-1$), se să construiască numărul $m=c_p c_{p-1} .. c_1 \, c_k c_{k-1} \ldots \, c_{p+1}$. De exemplu pentru $n=45612$ şi $p=2$ se obţine $m=12456$.
   b. *Interschimbare a două cifre.* Fiind date două valori *p1* şi *p2* ($1 \leq p2 \leq p1 \leq k-1$) să se construiască numărul obţinut prin interschimbarea cifrelor de pe poziţiile indicate de *p1* şi *p2* (din $n=c_k c_{k-1} \ldots c_{p1} \ldots c_{p2} .. c_1$ se obţine $m= c_k c_{k-1} \ldots c_{p2} \ldots c_{p1} .. c_1$), toate celelalte cifre rămânând pe poziţia lor iniţială. De exemplu pentru $n=45612$, $p1=4$, $p2=2$ se obţine $m=41652$.

```
/*
Permutare circulara a cifrelor.
*/

// se determina din cate cifre este format numarul
citește n,p;
aux ← n;
k ← 0;
cât timp aux<>0 execută
      aux ← aux div 10;
      k ← k+1;
sfârșit_cât_timp;
// se determina ultimele p cifre ale numarului
aux ← n;
pentru i←1,p, pas 1execută
      d ← d*10;
sfârșit_pentru;
rest ← aux mod d;
aux ← aux div d;
// se determina numarul final
d←1;
pentru i←1,k-p, pas 1execută
      d ← d*10;
sfârșit_pentru;
m ← rest*d+aux;
```


Blvd. V. Pârvan 4, 300223, Timişoara, România
Tel: +40-(0)256-592.155
Email: mateinfo@e-uvt.ro
www.math.uvt.ro

```
// Interschimbarea a  doua cifre
citește p1,p2;
// se determina cifrele de pe pozitiile p1 si p2
d← 0;
cât timp aux<>0 execută
       rest← aux mod 10;
       aux ←aux div 10;
       d ←d+1;
      dacă d = p2
           atunci c2 ←rest;
       sfârșit_dacă;
      dacă d = p1
           atunci c1 ←rest;
       sfârșit_dacă;
sfârșit_cât_timp;
// se parcurge numarul cifra cu cifra de la sfarsit spre inceput
// pana la intalnirea cifrei c1
// apoi, se muta cifra c2 pe pozitia lui c1
//si cifra c1 pe pozitia cifrei c2
// si se reface numarul
aux ← n;
rest ← aux mod 10;
nr ← 0; d ←1;
cât timp aux<>0 execută
      dacă rest = c1
           atunci rest ←c2;
           altfel dacă rest = c2
                      atunci rest ←c1;
                  sfârșit_dacă;
       sfârșit_dacă;
       nr ← rest*d+nr;
       aux ← aux div 10;
       d ← d*10;
       rest ← aux mod 10;
  sfârșit_cât_timp;
  tipărește nr;


#include <iostream>
using namespace std;
int main()
{
int n,p,k,i,aux,m,rest,d=1;
cout<<"numarul=";
cin >> n;
cout<<"p=";
cin>>p;
/*
Permutare circulara a cifrelor.
*/
aux=n;
k=0;
```


Blvd. V. Pârvan 4, 300223, Timişoara, România
Tel: +40-(0)256-592.155
Email: mateinfo@e-uvt.ro
www.math.uvt.ro

```
while ( aux )
{
k++;
aux /=10;
}
aux=n;
cout<<"nr cifre="<<k<<endl;
for(i=1;i<=p;i++)
   d*=10;
rest=aux%d;
aux/=d;
d=1;
for(i=1;i<=k-p;i++)
   d*=10;
m=rest*d+aux;
cout<<"nr obtinut="<<m;cout<<endl;
cout << '\n';
// Interschimbare a  doua cifre
int c1,c2;
int p1,p2,nr;
aux=n;
do
{
   cout<<"p1=";cin>>p1;
   cout<<"p2=";cin>>p2;
}while(p2 >p1 || p1>=k || p2>=k);
d=0;
while (aux)
   {
      rest=aux%10;
      aux/=10;
      d++;
      if(d==p2) c2=rest;
      if(d==p1) c1=rest;
   }
cout<<"cifra de pe pozitia "<<p1<<" este ="<<c1<<endl;
cout<<"cifra de pe pozitia "<<p2<<" este ="<<c2<<endl;
aux=n;
rest=aux%10;
nr=0;d=1;
while(aux)
   {
      if(rest==c1)rest=c2;
         else if(rest==c2)
                rest=c1;
      nr=rest*d+nr;
      aux/=10;
      d*=10;
      rest=aux%10;
   }
 cout<<"numarul format este "<<nr<<endl;
   return 0;
}
```


Blvd. V. Pârvan 4, 300223, Timişoara, România
Tel: +40-(0)256-592.155
Email: mateinfo@e-uvt.ro
www.math.uvt.ro

6. Pentru un număr natural *n*, să se scrie un algoritm care determină numărul maxim de divizori pe care îi are un număr din mulţimea *{2,3,…,n}*. De exemplu, dacă *n=15*, numărul maxim de divizori este 5 (numărul 12 are 5 divizori: 2,3,4,6,12).

   Algoritmul determină pentru fiecare număr cîți divizori are si se compară cu numărul divizorilor numărului precedent.

```
citește n;
pentru i←2,n, pas 1execută
      nr_div ← 0;
      d ← 2;
      cât timp d<=i execută
           dacă i mod d =0
                 atunci nr_div ← nr_div+1;
           sfârșit_dacă;
           d ← d+1;
      sfârșit_cât_timp;
      dacă nr_div>nr_max
           atunci
                 nr_max ← nr_div;
                 nr ← i;
      sfârșit_dacă;
 sfârșit_cât_timp;
tipărește nr_max;
```

```
#include <iostream>
using namespace std;
int main()
{
   int n,i,d,nr,nr_div,nr_max=0;
    cout<<"nr=";
    cin>>n;cout<<endl;
    for(i=2;i<=n;i++)
      nr_div=0;
      d=2;
      while(d<=i )
         {
         if(i%d==0)
           nr_div++;
         d++;
         }
      if (nr_div>nr_max)
         {   nr_max=nr_div;
            nr=i;
         }
    }
   cout<<nr<<" are cel mai mare numar de divizori ="<<nr_max<<endl;
   return 0;
}
```


Blvd. V. Pârvan 4, 300223, Timişoara, România
Tel: +40-(0)256-592.155
Email: mateinfo@e-uvt.ro
www.math.uvt.ro

7. Descrieţi un algoritm care pentru două numere naturale nenule a şi b determină numitorul şi numărătorul fracţiei ireductibile egale cu a/b.

Algoritmul impune determinarea cmmdc(numator, numitor) folosind algoritmul lui Euclid.
Apoi se imparte numaratorul si numitorul la cmmdc determinat

```
cmmdc (a,b)
cât timp b<>0 execută
        rest ← a mod b;
        a ← b;
        b ← rest;
sfârșit_cât_timp;
a ← cmmdc

dacă cmmdc(n1,n2) =1
            atunci tipărește “fracție ireductibilă”;
            altfel a ← n1 div cmmdc(n1,n2)
                   b ← n2 div cmmdc(n1,n2)
                   tipărește a/b
sfârșit_dacă;
```

```
#include <iostream>
using namespace std;
// se obtine fractie ireductibila daca
// atat numaratorul, cat si numitorul se impart la cmmdc(numarator,numitor)

int cmmdc(int a, int b)
{
   int rest;
   while (b) {
      rest = a % b;
      a = b;
      b = rest;
   }
   return a;
}
int main()
{
   int n1,n2,a,b;
    cout<<"numaratorul=";
    cin>>n1;cout<<endl;
    cout<<"numitorul=";
    cin>>n2;cout<<endl;
    if (cmmdc(n1,n2)==1)
       {cout<<"Fractia este ireductibila "<<n1<<"/"<<n2;
        cout<<endl;
       }
       else
       {
```


Blvd. V. Pârvan 4, 300223, Timişoara, România
Tel: +40-(0)256-592.155
Email: mateinfo@e-uvt.ro
www.math.uvt.ro

```
            a=n1/cmmdc(n1,n2);
            b=n2/cmmdc(n1,n2);
            cout<<"Fractia ireductibila "<<a<<"/"<<b;cout<<endl;
        }
    return 0;
}
```

8. Să se descrie un algoritm care verifică dacă un număr *n* este prim sau nu folosind teorema lui Wilson. (Condiția necesară și suficientă ca un număr natural *p*, *p>1*, să fie prim este ca *(p-1)!+1* să fie divizibil cu *p*.)

   Algoritmul calculează (n-1)! si apoi se testeaza dacă (n-1)! +1 este divizibil cu n, daca restul este 0 rezulta ca n este prim, in caz contrar n nu este prim

```
citește n;
pentru i←1,n-1, pas 1execută
      fact ← fact*i;
sfârșit_pentru;
dacă fact+1 mod n =0
                atunci tipărește “numărul este prim”;
                altfel tipărește “numărul nu este prim”;
sfârșit_dacă;
```

```
#include <iostream>
using namespace std;
int main()
{
   int n,i,c,fact;
   cout<<"nr=";
   cin>>n;cout<<endl;
   for(i=1,fact=1;i<n;i++)
      fact*=i;
   if((fact+1)%n==0)
      cout<<"Numarul este prim";
   else
      cout<<"Numarul NU este prim";
   cout<<endl;
   return 0;
}
```

9. Să se descrie un algoritm care afișează toate numerele prime dintr-un interval dat *[a,b]*, *a*, *b* numere naturale, *a<b*.

   Algoritmul constă în a testa dacă fiecare număr din intervalul *[a,b]* este prim. Un număr este prim dacă este divizibil doar cu 1 și el însuși.

Blvd. V. Pârvan 4, 300223, Timişoara, România
Tel: +40-(0)256-592.155
Email: mateinfo@e-uvt.ro
www.math.uvt.ro

```
citește a,d;
pentru i←a,b, pas 1execută
      c ← 0;
      pentru j←2,i-1, pas 1execută
            dacă i mod j =0
                        atunci  c ← c+1;
               sfârşit_dacă;
         sfâșit_pentru;
         dacă c=0
               atunci tipărește i
          sfârșit_dacă;
sfârşit_pentru;
```

```
#include <iostream>
using namespace std;
int main()
{
   int n,i,a,b,c,j,nr=0;
   do
   {
      cout<<"a=";
      cin>>a;cout<<endl;
      cout<<"b=";
      cin>>b;cout<<endl;
   }while(a>=b);
   for(i=a;i<=b;i++)
   {
      c=0;
      for (int j=2; j<i; j++)
      {
         if (i%j==0)
         c++;
      }
      if(!c)
      {
         cout<<i<<"  ";
         nr++;
      }
   }
   cout<<endl;
   cout<<"sunt "<<nr<<" numere prime in intervalul ["<<a<<","<<b<<"]"<<endl;
   return 0;
}
```

10. Se consideră un natural n>2. Descrieți un algoritm care să afișeze toate numerele *x* mai mici decât n, care au proprietatea că *x-1* și *x+1* sunt numere prime. Exemplu: pentru n=43, se vor afișa numerele: 4 6 12 18 30 42.

Blvd. V. Pârvan 4, 300223, Timişoara, România
Tel: +40-(0)256-592.155
Email: mateinfo@e-uvt.ro
www.math.uvt.ro

```
citește n;
pentru i←2,n-1, pas 1execută
      dacă i-1 este prim && i+1 este prim
                atunci tipărește i;
      sfârșit_dacă;
sfârșit_pentru;
```

```
#include <iostream>
using namespace std;
int prim(int x)
{
   int k,c=0;
   for (int k=2; k<x; k++)
      {
         if (x%k==0)
           c++;
      }
      if(!c)
         return 1;
         else return 0;
}
int main()
{
   int n,i,a,b,c,j,nr=0;
   cout<<"n=";
   cin>>n;cout<<endl;
   for(i=2;i<n;i++)
   {
      if (prim(i-1)&& prim(i+1))
         cout<<i<<"  ";
         }
   cout<<endl;
   return 0;
}
```

11. Scrieți un algoritm care afișează descompunerea unui număr natural *n* în factori primi. De exemplu pentru *n=18*, se va afișa *2^1x3^2*.

```
citește n;
întreg p,d=2;
cât timp n>1 execută
          p ← 0;
          cât timp n mod d =0 execută
                p ← p +1;
                n ← n div d;
          sfârșit_cât_timp;
          dacă p<>0
                atunci tipărește d la puterea p
          sfârșit_dacă;
          d ← d +1;
sfârșit_cât_timp;
```


Blvd. V. Pârvan 4, 300223, Timişoara, România
Tel: +40-(0)256-592.155
Email: mateinfo@e-uvt.ro
www.math.uvt.ro

```
#include <iostream>
using namespace std;
int main()
{
   int n,d=2,p;
   cout<<"numarul=";cin>>n;
   while(n>1)
   {
   p=0;
   while(n%d==0)
   {
      p=p+1;
      n=n/d;
   }
   if(p) cout<<d<<" la puterea "<<p<<endl;
   d=d+1;
   }
   return 0;
}
```

12. Să se scrie un algoritm care pentru numărul natural *n* dat, determină sumele:

    a. S=1*2+2*3+3*4...+n*(n+1)
    b. S=1*n+2*(n-1)+3*(n-2)+...+n*1

```
citește n;
întreg i, s1=0, s2=0;
pentru i ← 1,n, pas 1 execută
          s1← s1+i*(i+1);
          s2 ← s2+i*(n-i+1);
sfârșit_pentru;
tipărește s1, s2;
```

```
#include <iostream>

using namespace std;
//calcul sume
int main()
{
   int s1=0,s2=0,i,n;
   cout<<"numarul=";cin>>n;
   for(i=1;i<=n;i++)
   {
     s1+=i*(i+1);
     s2+=i*(n-i+1);
   }
   cout<<"suma 1 = "<<s1<<endl;
```


Blvd. V. Pârvan 4, 300223, Timişoara, România
Tel: +40-(0)256-592.155
Email: mateinfo@e-uvt.ro
www.math.uvt.ro

```
    cout<<"suma 2 = "<<s2<<endl;
    return 0;
}
```

13. Estimaţi cu precizia ε>0 limitele şirurilor următoare (pentru punctele a. şi b. se consideră că *x* este o valoare dată din intervalul (0,1)):
    a. $s_n = \sum_{i=0}^{n} \frac{(-1)^i x^{2i+1}}{(2i+1)!}$
    b. $s_n = \sum_{i=0}^{n} \frac{(-1)^i x^{2i}}{(2i)!}$
    c. $s_n = \sum_{i=1}^{n} \frac{1}{(2i+1)^2}$

    Indicaţie. Calculul se opreşte atunci când diferenţa dintre doi termeni consecutivi ai şirului este mai mică decât constanta ε ($|s_n - s_{n-1}| < \varepsilon, \varepsilon = 0.001$).

```
întreg i;
real e=0.001, suma=0, sn1,sn2, x=0.2;
i ← 0;
sn1 ←putere(x,2*i+1)/fact(2*i+1);
execută
    suma ← suma+sn1;
    sn2 ←sn1;
    i ← i+1;
   dacă i mod 2 =0
        atunci sn1 ← putere(x,2*i+1)/fact(2*i+1);
        altfel sn1 ← - putere(x,2*i+1)/fact(2*i+1);
   sfârșit_dacă;
cât_timp (fabs(sn1-sn2)>e)
tipărește suma;
```

```
întreg i;
real e=0.001, suma=0, sn1,sn2;
i ← 1;
sn1 ←1/ ((2*i+1)*(2*i+1));
execută
    suma ← suma+sn1;
    sn2 ←sn1;
    i ← i+1;
   sn1 ←1/ ((2*i+1)*(2*i+1));
cât_timp (fabs(sn1-sn2)>e)
tipărește suma;
```


Blvd. V. Pârvan 4, 300223, Timişoara, România
Tel: +40-(0)256-592.155
Email: mateinfo@e-uvt.ro
www.math.uvt.ro

```
#include <iostream>
#include <math.h>
using namespace std;
int main()
{
   int i;
   double e=0.001, suma=0,sn1,sn2;
   i=1;
   sn1= 1./((2*i+1)*(2*i+1));
   do
   {
     suma+=sn1;
     sn2=sn1;
     i++;
     sn1= 1./((2*i+1)*(2*i+1));
   }while(fabs(sn1-sn2)>e);
   cout<<"Suma = "<<suma<<endl;
   return 0;
}
```

14. Să se scrie un algoritm care determină dacă un număr natural dat *m* face parte din șirul lui Fibonacci sau nu. Șirul lui Fibonacci este definit prin relațiile: *f(0)=1*, *f(1)=1*, *f(n)=f(n-1)+f(n-2)*, pentru *n>=2*. Nu se vor folosi tablouri.

```
citește n;
întreg i,x,y,f;
x ← 1; y← 1;
dacă n=1
      atunci tipărește " numarul este in sirul lui Fibonacci";
sfârșit_dacă;
dacă n<>1
      atunci
         execută
               f← x+y;
               x← y;
               y← f;
         cât_timp (f<n);
         dacă f=n
               atunci tipărește "numarul este in sirul lui Fibonacci";
               altfel tipărește "numarul nu este in sirul lui Fibonacci";
         sfîrșit_dacă;
   sfârșit_dacă;
```


Blvd. V. Pârvan 4, 300223, Timişoara, România
Tel: +40-(0)256-592.155
Email: mateinfo@e-uvt.ro
www.math.uvt.ro

```
#include <iostream>
using namespace std;
int main()
{
   int n,i,x,y,f;
   cout<<"numarul=";cin>>n;
   x=1,y=1;
   if(n==1) cout<<"Numarul este in sirul lui Fibonacci "<<n<<endl;
   if(n!=1)
   {
      do
      {
         f=x+y;
         x=y;
         y=f;
      }while(f<n);
      if (f==n) cout<<"Numarul "<<n<<" apartine sirului ";
         else cout<<"Numarul "<<n<<" nu apartine sirului ";
   }
   cout<<endl;
   return 0;
}
```


Blvd. V. Pârvan 4, 300223, Timişoara, România
Tel: +40-(0)256-592.155
Email: mateinfo@e-uvt.ro
www.math.uvt.ro