Algoritmi fundamentali

Este vorba despre algoritmi de rezolvare a unor probleme des întâlnite în practică, care fie fac obiectul unor enunțuri, fie sunt utilizați ca etape, pași, în elaborarea unor răspunsuri. Aceşti algoritmi au fost concepuți spre a veni în ajutorul programatorilor, care îi folosesc ori de câte ori este necesar în probleme, fără a mai fi nevoie să-i elaboreze de fiecare dată. Aceştia se referă la separarea cifrelor unui număr (folosit de fiecare dată când în rezolvarea unei probleme este necesar accesul la cifrele unui număr), determinarea divizorilor proprii ai unui număr natural dat, testarea dacă un număr natural mai mare ca 1 este prim, determinarea celui mai mare divizor comun a două numere naturale date, descompunerea unui număr natural mai mare ca 1 în factori primi, determinarea maximului/minimului unui şir de numere citite, pe rând, de la dispozitivul de intrare.

În continuare, voi prezenta fiecare algoritm fundamental astfel: prezentarea algoritmului, descrierea lui în limbaj pseudocod şi în limbaj C++ şi rezolvarea câte unei aplicații în care intervine.

1. Separarea cifrelor unui număr

Se va folosi rezultatul din matematică conform căruia restul împărțirii la 10 al unui număr întreg pozitiv îl reprezintă ultima cifră a numărului (cea mai puțin semnificativă), iar câtul împărțirii la 10, numărul fără ultima cifră. Repetând această operație cât timp numărul mai are cifre de separat, obținem la fiecare pas o cifră a numărului, care poate fi prelucrată, de fiecare dată câtul obținut devenind deîmpărțit. În algoritm, marcarea încheierii separării cifrelor se face când numărul dat devine 0, deci nu mai sunt cifre de separat.

Exemplu: n=2415

	cât	rest
2415	:10=241	5
241	:10=24	1
24	:10=2	4
2	:10=0	2

Am obținut cifrele numărului în ordine inversă: 5, 1, 4, 2

Limbaj pseudocod

```
cât_timp n≠ 0 execută
        scrie ″ cifra este ″ ,n%10 {sau orice alta prelucrare}
    n=[n/10]
```

Limbaj C++

```
while(n!=0)
   {cout<<"cifra este "<<n%10;
    n=n/10;
    }
}
```

Exemple

1. Se citeşte de la tastatură un număr întreg a. Să se afişeze suma cifrelor numărului.

Limbaj pseudocod

```
intreg a,s
citeste a
s=0
 cât_timp a≠ 0 execută
     s=s+a%10
     a=[a/10]
scrie s
```

Limbaj C++

```
#include <iostream.h>
void main()
{int a,s=0;
cout<<"a=";cin>>a;
while(a!=0)
   {s=s+a%10;
    a=a/10;
    }
cout<<"s="<<s<<endl;
}
```

2. Se citeşte de la tastatură un număr întreg a. Să se precizeze câte cifre are numărul citit.

Limbaj pseudocod

```
intreg a,k
citeste a
k=0
 cât_timp a≠ 0 execută
     k=k+1 (număr cifrele)
     a=[a/10]
scrie k
```

Limbaj C++

```
#include <iostream.h>
void main()
{int a,k=0;
cout<<"a=";cin>>a;
while(a!=0)
   {k++;
    a=a/10;
    }
cout<<"numărul are "<<k<<″  cifre″ <<endl;
}
```

3. Se citeşte de la tastatură un număr întreg a. Să se afişeze oglinditul numărului.
Ex. a=3147; oglinditul este 7413.

Limbaj pseudocod

```
intreg a,ogl
citeste a
ogl=0
 cât_timp a≠ 0 execută
      ogl=ogl*10+a%10
      a=[a/10]
scrie ogl
```

Limbaj C++

```
#include <iostream.h>
void main()
{int a;
long ogl=0;
cout<<"a=";cin>>a;
while(a!=0)
    { ogl=ogl*10+a%10;
      a=a/10;
     }
cout<<"oglinditul numărului dat este "  <<ogl;
}
```

2. Determinarea divizorilor proprii ai unui număr natural dat

De exemplu, dacă n=100, divizorii proprii sunt: 2, 4, 5, 10, 20, 25, 50;
dacă n=45, divizorii proprii sunt: 3, 5, 9, 15;
dacă n=32, divizorii proprii sunt: 2, 4, 8, 16.
Putem continua cu exemplele, dar şi din acestea se poate observa că:
- cel mai mic divizor propriu posibil este 2
- cel mai mare divizor propriu posibil este jumătatea numărului [n/2]

Este suficient să testăm care din valorile cuprinse între 2 şi [n/2] împart exact numărul n dat şi astfel identificăm, pe rând, divizorii proprii ai numărului, care vor putea fi prelucrați conform cerințelor enunțului.

Limbaj pseudocod

```
 pentru i←2, [n/2] execută
      dacă n%i=0 atunci scrie i (sau orice altă
                                 prelucrare cerută)
```

Limbaj C++

```
for(i=2;i<=n/2;i++)
     if(n%i==0) cout<<i;
```

(am ales să fac operația de afişare a divizorilor, dar se va face operația necesară conform algoritmului din enunțul concret)

Exemplu

Se citeşte de la tastatură un număr natural x. Să se verifice dacă x este număr perfect. Un număr este perfect dacă este egal cu suma divizorilor săi, mai mici decât el. Ex. 6=1+2+3; 28=1+2+4+7+14

Limbaj pseudocod

```
întreg n,i,s
citeste n
s=1
 pentru i←2, [n/2] execută
      dacă n%i=0 atunci s=s+i

 dacă s=n atunci scrie "  număr perfect"
           altfel scrie "  numărul nu este perfect"
```

Limbaj C++

```
#include <iostream.h>
void main()
{int n,i,s=1;
cout<<"n=";cin>>n;
for(i=2;i<=n/2;i++)
    if(n%i==0) s=s+i;
if(s==n) cout<<"numarul dat este perfect";
else cout<<"numarul dat nu este perfect";
}
```

3. Testul de număr prim

Matematica ne spune că un număr este prim dacă are doar doi divizori, pe 1 şi numărul însuşi, deci când nu are divizori proprii. Sunt mai multe modalități de a verifica dacă un număr dat este prim sau nu. Noi o vom folosi pe cea conform căreia dacă numărul nu are divizori proprii atunci este prim, în caz contrar, dacă are cel puțin un divizor propriu, atunci numărul nu este prim.

Limbaj pseudocod

```
întreg n,i
logic prim
citeste n
prim←true
 pentru i←2, [√n] execută
        dacă n%i=0 atunci prim←false
dacă prim=true atunci scrie ″ nr prim″
                  altfel scrie ″ nr nu este prim″
```

Limbaj C++

```
#include<math.h>
#include <iostream.h>
void main()
{int n,i,prim;
cout<<"n=";cin>>n;
prim=1;
for(i=2;i<=sqrt(n);i++)
   if(n%i==0) prim=0;
if(prim==1) cout<<"numarul dat este prim";
else cout<<"numarul dat nu este prim";
}
```

Exemplu

Se citesc, pe rând, de la tastatură n numere naturale. Să se afişeze media aritmetică a valorilor prime citite.

Limbaj pseudocod

```
întreg n,i,j,s,k,x
logic prim
citeste n
s=0
k=0
  pentru i←1,n execută
        citeşte x
        prim←true
         pentru j←2, [√x] execută
              dacă x%j=0 atunci prim←false
        dacă prim=true atunci
                              s=s+x
                              k=k+1
dacă k≠0 atunci scrie ″ media aritmetică ″ ,s/k
                  altfel scrie ″ şirul dat nu conţine numere
prime″
```

Limbaj C++

```
#include <iostream.h>
void main()
{int n,i,j,prim,s=0,k=0,x;
cout<<"n=";cin>>n;
for(i=1;i<=n;i++)
      {cout<<"dati un numar";cin>>x;
       prim=1;
       for(j=2;j<=sqrt(n);j++)
              if(n%j==0) prim=0;
       if(prim==1)
                {s=s+x;
                 k=k+1;}
       }
if(k!=0) cout<<"media aritmetica este "<<(float)s/k;
else cout<<"sirul dat nu contine numere prime";
}
```

4. Determinarea celui mai mare divizor comun a două numere naturale date

Varianta 1 (prin împărţiri)

Să presupunem că avem două numere naturale a şi b, pentru care trebuie să aflăm cel mai mare divizor comun(cmmdc). Dacă numărul a se împarte exact la numărul b, atunci cmmdc(a,b)=b. Dacă cele două numere nu se împart exact unul la celălalt, atunci se procedează astfel: se împarte în mod repetat a la b, până când restul împărţiri devine 0. La fiecare pas se reţine restul, împărţitorul devine deimpartit, iar restul împărţitor. Cmmdc(a,b) este ultimul rest nenul obţinut. Şi aici avem două modalităţi de determinare, după cum folosim structura repetitivă cât_timp ... execută sau repetă ... cât_timp.

cât_timp ... execută

Limbaj pseudocod

```
r←a%b
  cât_timp r≠0 execută
        a←b
        b←r
        r←a%b
scrie b
```

Limbaj C++

```
r=a%b
while(r!=0)
     {a=b;
      b=r;
      r=a%b; }
cout<<"cmmdc="<<b;
```

repetă ... cât_timp

Limbaj pseudocod

```
repeta
      r←a%b
      a←b
      b←r
cat_timp r≠0
scrie a
```

Limbaj C++

```
do
{r=a%b;
 a=b;
 b=r;
}while(r!=0);
cout<<"cmmdc="<<a;
```

Varianta 2 (prin scăderi)

Algoritmul este următorul: cât timp cele două numere a şi b sunt diferite între ele, se scade din numărul mai mare numărul mai mic. În momentul în care cele două numere devin egale, cmmdc se află în oricare din cele două numere a sau b.

Limbaj pseudocod

```
cât_timp a≠b execută
      dacă a>b atunci a←a-b
               altfel  b←b-a
scrie a
```

Limbaj C++

```
while(a!=b)
      if(a>b) a=a-b;
      else b=b-a;
cout<<″ cmmdc=″<< a;
```

Exemple

1. Se dau două numere naturale a şi b. Să se verifice dacă cele două numere sunt prime între ele. Două numere sunt prime între ele dacă cmmdc al lor este 1.

Limbaj pseudocod

```
întreg a,b
citeşte a,b
cât_timp a≠b execută
      dacă a>b atunci a←a-b
               altfel  b←b-a
dacă a=1 atunci scrie ″ cele două nr sunt prime între ele″
               altfel scrie ″ cele două nr nu sunt prime între ele″
```

Limbaj C++

```
#include <iostream.h>
void main()
{int a,b;
cin>>a>>b;
while(a!=b)
      if(a>b) a=a-b;
      else b=b-a;
if(a==1) cout<<"cele doua nr sunt prime intre ele";
else cout<<"cele doua nr nu sunt prime intre ele";
}
```

2. Se dau două numere naturale a şi b. Să se determine cmmmc (cel mai mic multiplu comun) al celor două numere.

$$cmmmc\ (a,b) = \frac{a*b}{cmmdc\ (a,b)}$$

Limbaj pseudocod

```
intreg a,b,x
citeste a,b
x←a*b
cât_timp a≠b execută
      dacă a>b atunci a←a-b
               altfel  b←b-a
scrie ″ cmmmc=″ ,[x/a]
```

Limbaj C++

```
#include <iostream.h>
void main()
{int a,b,x;
cin>>a>>b;
x=a*b;
while(a!=b)
      if(a>b) a=a-b;
      else b=b-a;
cout<<"cmmmc="<<x/a;
}
```

5. Descompunerea în factori primi a unui număr natural dat, mai mare ca 1

Exemple

120	2
60	2
30	2
15	3
5	5
1	

45	3
15	3
5	5
1	

Algoritm:
se porneşte de la primul factor prim posibil, 2;
cât timp numărul dat este diferit de 1, se execută operaţiile:
cât timp numărul se împarte exact la un factor prim
se execută împărţirea, se prelucrează factorul şi câtul devine deîmpărţit
se trece apoi la următorul factor prim

Limbaj pseudocod

```
f←2
cât_timp n≠1 execută
    cât_timp n%f=0 execută
        scrie f   (sau orice alta prelucrare ceruta)
        n=[n/f]
    f←f+1
```

Limbaj C++

```
#include <iostream.h>
void main()
{int n,f;
cin>>n;
f=2;
while(n!=1)
    {while(n%f==0)
        {cout<<f;
         n=n/f;}
      f=f+1;
    }
}
```

Exemplu
Se citeşte de la tastatură un număr natural mai mare ca 1. Să se afişeze descompunerea numărului în factori primi sub forma f la puterea p, unde f este factorul prim şi p puterea la care apare în descompunere.
Ex. descompunerea numărului 120 va fi afişată astfel:
2 la puterea 3
3 la puterea 1
5 la puterea 1

Limbaj pseudocod

```
întreg n,f,p
citeşte n
f←2
cât_timp n≠1 execută
    p←0
    cât_timp n%f=0 execută
        p=p+1
        n=[n/f]
    dacă p≠0 atunci scrie f," la puterea " ,p
    f←f+1
```

Limbaj C++

```
#include <iostream.h>
void main()
{int n,f=2,p=1;
cin>>n;
while(n!=1)
    {p=0;
     while(n%f==0)
        {p=p+1;
         n=n/f;}
     if(p!=0) cout<<f<<" la puterea „<<p<<endl;
     f=f+1;
    }
}
```

6. Determinarea minimului/maximului unui şir de numere citite pe rând

Determinarea maximului

Se presupune că primul număr citit este maximul.Se citesc apoi, pe rând, numerele şi la fiecare pas se compară numărul citit cu maximul existent. Dacă numărul citi este mai mare decât maximul, se înlocuieşte maximul.

Limbaj pseudocod

```
intreg n,i,x,max
citeste n,x
max←x
pentru i←2,n executa
        citeste x
        daca x>max atunci max←x

scrie max
```

Limbaj C++

```
#include <iostream.h>
void main()
{int n,i,x,max;
cout<<"cate numere se dau?";cin>>n;
cout<<"dati un numar";cin>>x;
max=x;
for(i=2;i<=n;i++)
        {cout<<"dati un numar";cin>>x;
          if(x>max) max=x;}
cout<<"maximul numerelor citite este="<<max<<endl;
}
```

Determinarea minimului

Se presupune că primul număr citit este minimul.Se citesc apoi, pe rând, numerele şi la fiecare pas se compară numărul citit cu minimul existent. Dacă numărul citi este mai mic decât minimul, se înlocuieşte minimul.

Limbaj pseudocod

```
intreg n,i,x,min
citeste n,x
min←x
pentru i←2,n executa
        citeste x
        daca x<min atunci min←x

scrie min
```

Limbaj C++

```
#include <iostream.h>
void main()
{int n,i,x,min;
cout<<"cate numere se dau?";cin>>n;
cout<<"dati un numar";cin>>x;
min=x;
for(i=2;i<=n;i++)
        {cout<<"dati un numar";cin>>x;
          if(x<min min=x;}
cout<<"minimul numerelor citite este="<<min<<endl;
}
```

Determinarea în acelaşi timp a minimului şi maximului

Se presupune că primul număr citit este minimul şi în acelaşi timp maximul. .Se citesc apoi, pe rând, numerele şi la fiecare pas se compară numărul citit cu minimul şi maximul existente. Dacă numărul citi este mai mic decât minimul, se înlocuieşte minimul; dacă numărul citi este mai mare decât maximul, se înlocuieşte maximul.

Limbaj pseudocod

```
intreg n,i,x,max,min
citeste n,x
max←x
pentru i←2,n executa
        citeste x
        daca x>max atunci max←x

        daca x<min atunci min←x

scrie max,min
```

Limbaj C++

```
#include <iostream.h>
void main()
{int n,i,x,max,min;
cout<<"cate numere se dau?";cin>>n;
cout<<"dati un numar";cin>>x;
max=x; min=x;
for(i=2;i<=n;i++)
        {cout<<"dati un numar";cin>>x;
          if(x>max) max=x;
          if(x<min) min=x;}
cout<<"maximul numerelor citite este="<<max<<endl;
cout<<"minimul numerelor citite este="<<min<<endl;
}
```