Algoritmi fundamentali – numerea naturale C++

ALGORITMI FUNDAMENTALI

Aceşti algoritmi au fost concepuţi 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 în factori primi, determinarea maximului/minimului unui şir de numere citite, pe rând, de la dispozitivul de intrare.

1. Separarea cifrelor unui număr

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

Exemplu: n=2954

operaţia	cât	rest
2954:10	295	4
295:10	29	5
29:10	2	9
2:10	0	2

Limbajul C++

```
#include<iostream>
using namespace std;
int main()
{
    int n;
    cout<<"n=";cin>>n;
    while(n)
    {
        cout<<n%10<<" ";
        n=n/10;
    }
}
```

Am obţinut cifrele numărului în ordine inversă: 4, 5, 9, 2.

2. Determinarea divizorilor proprii ai unui număr natural dat

De exemplu, dacă n=50, divizorii proprii sunt: 2, 5, 10, 25;
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 prelucraţi conform cerinţelor enunţului.
Exemple cu cele trei structuri repetitive:

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

```
#include<iostream>
using namespace std;
int main()
{
    int n,d;
    cout<<"n=";cin>>n;
    d=2;
    do
    {
        if(n%d==0)
            cout<<d<<" ";
        d++;
    }
    while(d<=n/2);
}
```

```
#include<iostream>
using namespace std;
int main()
{
    int n,d;
    cout<<"n=";cin>>n;
    for(d=2;d<=n/2;d++)
        if(n%d==0)
            cout<<d<<" ";
}
```

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 modalităţi 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 puţin un divizor propriu, atunci numărul nu este prim.

```
#include<iostream>
using namespace std;
int main()
{
    int n,d,ok=1;
    cout<<"n=";cin>>n;
    for(d=2;d<=n/2 && ok==1;d++)
        if(n%d==0)
            ok=0;
        if(ok!=0)
            cout<<"Numarul este prim.";
        else
            cout<<"Numarul nu este prim.";
}
```

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

Algoritmul lui Euclid

Să presupunem că avem două numere naturale a şi b, pentru care trebuie să aflăm cel mai mare divizor comun(cmmdc).Se reţine în variabila r restul împărţirii lui a la b.Variabila a ia valoarea variabilei b iar b ia valoarea restului obţinut în urma împărţirii lui a la b. Aceste operaţii se execută cât timp b este diferit de 0. Cel mai mare divizor comun va fi variabila a.

```
#include<iostream>
using namespace std;
int main()
{
    int a,b,r;
    cout<<"a=";cin>>a;
    cout<<"b=";cin>>b;
    while(b)
    {
        r=a%b;
        a=b;
        b=r;
    }
    cout<<"cmmdc este "<<a;
}
```

Algoritmul scăderilor repetate

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.

```
#include<iostream>
using namespace std;
int main()
{
    int a,b;
    cout<<"a=";cin>>a;
    cout<<"b=";cin>>b;
    while(a!=b)
    {
        if(a>b)
            a=a-b;
        else
            b=b-a;
    }
    cout<<"cmmdc este "<<a;
}
```

5. Descompunerea în factori primi a unui număr natural

Exemple

120	2
60	2
30	2
15	3
5	5
1	

45	3
15	3
5	5
1	

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

Algoritm:

se porneşte de la primul factor prim posibil, 2;

cât timp numărul dat este diferit de 1, se execută operaţiile:

dacă factorul îl divide pe n îl afişăm

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

6. Determinarea valorii minime/maxime dintr-un şir de numere

Determinarea valorii maxime

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 citit este mai mare decât maximul, se înlocuieşte maximul.

Determinarea valorii minime

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 citit este mai mic decât minimul, se înlocuieşte minimul.

```
#include<iostream>
using namespace std;
int main()
{
    int n,a,max,i;
    cout<<"n=";cin>>n;
    cout<<"a=";cin>>a;
    max=a;
    for(i=2;i<=n;i++)
    {
        cout<<"a=";cin>>a;
        if(a>max)
            max=a;
    }
    cout<<max;
}
```

```
#include<iostream>
using namespace std;
int main()
{
    int n,a,min,i;
    cout<<"n=";cin>>n;
    cout<<"a=";cin>>a;
    min=a;
    for(i=2;i<=n;i++)
    {
        cout<<"a=";cin>>a;
        if(a<min)
            min=a;
    }
    cout<<min;
}
```

http://www.cnchogastecuci.ro/portofoliu/solutii/Notiuni_teoretice.pdf