Algoritmi Elementari

Determinarea minimului

Să rezolvăm următoarea problemă: Se dau n numere întregi. Calculaţi cel mai mic dintre cele n numere date.

Numărul de valori se cunoaște de la început. Algoritmul de rezolvare va fi:

* inițializăm variabila min corespunzător
* citim cele n numere
  + fiecare număr se compară cu min și dacă este mai mic, se actualizează min
* inițializăm min cu prima dintre cele n valori; celelalalte n-1 valori se vor compara cu min

În secvența care urmează, n este numărul de valori citite, în x se citesc pe rând valorile, iar în min vom determina valoarea minimă:

cin >> n >> x;

min = x;

for(int i =1 ; i < n ; i ++)

{

cin >> x;

if(x < min)

min = x;

}

Determinarea maximului

Să rezolvăm următoarea problemă: Se citesc numere întregi până la apariția lui 0, care nu se ia în considerare. Calculaţi maximul lor. Numărul de valori nu se cunoaște de la început. Algoritmul de rezolvare va fi:

* inițializăm variabila max cu prima valoare
* citim un număr în x
* cât timp x este nenul
  + x se compară cu max și dacă este mai mare, se actualizează max
  + citim altă valoare pentru x
* În secvența care urmează, n este numărul de valori citite, în x se citesc pe rând valorile, iar în max vom determina valoarea minimă:

cin >> x;

max = x;

cin >> x;

while(x != 0)

{

if(x > max)

max = x;

cin >> x;

}

Divizibilitate

* Spunem că numărul n divide numărul m dacă există un număr p astfel încât n \* p = m. De regulă, este echivalent cu faptul că restul împărțirii lui m la n este 0, în C/C++ m % n este 0.
* Dacă n divide pe m, spunem că n este divizor al lui m, iar m este multiplu al lui n.

O primă metodă de determinare a divizorilor constă în a observa că toți divizorii lui n sunt între 1 și n, inclusiv. Putem parcurge numerele din acest interval și verifica dacă sunt într-adevăr divizori ai lui n, caz în care sunt luați în considerare. Următorul program afișează divizorii lui n în acest fel.

#include <iostream>

using namespace std;

int main()

{

int n;

cin >> n;

for(int d =1 ; d <= n ; d ++ )

if(n % d == 0)

cout << d << " ";

return 0;

}

De exemplu, pentru n = 24 se va afișa:

1 2 3 4 6 8 12 24

1. Numerele care nu sunt pătrate perfecte au număr par de divizori.
2. Singurele numere cu număr impar de divizori sunt pătratele perfecte.
3. Cel mai mic divizor propriu al unui număr natural (diferit de 1 și de numărul însuși) este număr prim.

# Cel mai mare divizor comun (CMMDC) a două numere în C++

Se dau două numere naturale a și b și se cere să aflăm CMMDC-ul celor două numere (**c**el **m**ai **m**are **d**ivizor **c**omun).

**Exemplu:** pentru a = 14, b = 6, cmmdc(a, b) = 2.

În acest articol vorbim despre varianta normală (pentru clasa a IX-a). Pentru varianta recursivă (clasa a X-a), puteți intra [pe această lecție](https://infoas.ro/lectie/21/cmmdc-recursiv-a-doua-numere-naturale-in-c).

## Determinarea celui mai mare divizor comun în C++

Există câteva metode de a determina cel mai mare divizor comun a două numere naturale nenule a și b. Vom prezenta două variante, ambele bazate pe același algoritm, **algoritmul lui Euclid**.

### **Metoda 1. Algoritmul lui Euclid cu scăderi**

Algoritmul lui Euclid cu scăderi este o metodă mai puțin eficientă, însă mai ușor de înțeles. Acest algoritm se bazează pe principiul următor: dacă d | a și d | b, atunci d | a - b (unde a ≥ b) — dacă d divide ambele numere, atunci divide și diferența lor.

Iată algoritmul propriu zis:

* cât timp numerele sunt diferite, se scade numărul mai mic din cel mai mare;
* când numerele devin egale, atunci CMMDC-ul lor este chiar numărul lor.

Iată implementarea algoritmului în C++:

#include <iostream>

using namespace std;

int main()

{

int a, b;

cin >> a >> b;

while(a != b)

{

if(a > b) a = a - b;

else b = b - a;

}

cout << "cmmdc-ul este " << a;

return 0;

}

### **Metoda 2. Algoritmul lui Euclid clasic (cu împărțiri)**

Algoritmul lui Euclid cu împărțiri este mai dificil de înțeles, însă este mai eficient (acesta este algoritmul recomandat pentru rezolvarea problemelor).

Pe scurt, observăm că în codul anterior facem scăderi repetate. Aceste scăderi se pot înlocui cu împărțiri, astfel că algoritmul este următorul:

* cât timp b este diferit de 0, a primește b, iar b primește restul împărțirii lui a la b;
* răspunsul în final se află în a.

#include <iostream>

using namespace std;

int main()

{

int a, b;

cin >> a >> b;

while(b != 0)

{

int r = a % b;

a = b;

b = r;

}

cout << "cmmdc-ul este " << a;

return 0;

}

# CMMMC a două numere în C++ (cel mai mic multiplu comun)

Dându-se două numere naturale a și b, vrem să aflăm CMMMC-ul (**c**el **m**ai **m**ic **m**ultiplu **c**omun) al celor două numere.

**Exemplu:** Pentru a = 3, b = 7, cmmmc(a, b) = 21 — 21 este cel mai mic număr care este simultan multiplu de 3 și de 7.

## Explicatie:

Pentru două numere naturale, avem egalitatea cmmdc(a, b) \* cmmmc(a, b) = a \* b. De aici, putem să extragem faptul că cmmmc(a, b) = a \* b / cmmdc(a, b).

## Codul sursă în C++

#include <iostream>

using namespace std;

int main()

{

int a, b;

cin >> a >> b;

int cmmdcNr;

int copie a = a, copie b = b;

while(copie b)

{

int aux = copie a % copie b;

copie a = copie b;

copie b = aux;

}

cmmdcNr = copiea;

int cmmmcNr = a \* b / cmmdcNr;

cout << "cmmmc(" << a << ", " << b << ") = " << cmmmcNr;

return 0;

}