1. Algoritmi in pseudocod

Sortare prin metoda bulelor

n, aux, numere naturale

a[101], sir de numere naturale

k, variabila de tip intrerupator

se citeste n si un sir de numere a[101] de n elemente

executa

{

k <- 1 ;

pentru i <- 0 , i < n – 1, i <- i + 1 executa

daca a[i] > a[i+1]

{

k <- 0

aux <- a[i]

a[i] <- a[i+1]

a[i+1] <- aux

}

Sfarsit daca

Sfarsit pentru

}

cat timp ≠ k

se afiseaza sirul a

1. Algoritmul lui Euclid (CMMDC)

n, m numere naturale

se citesc n si m

cat timp n ≠ m

daca n > m

n <- n – m;

sfarsit daca

altfel

m <- m – n;

sfarsit altfel

se afiseaza n

Descompunerea unui număr în factori primi

Citim un numar n. Parcurgem cu o structura repetitiva cu test initial (for) de la I = 2 pana la n inclusiv (cu cea de a doua conditie ca n sa fie diferit de 1) si parcugem din 1 in 1. Daca in urma impartirii lui I la n nu avem rest, atunci cu un while se imparte I la n si se numara de cate ori se face aceasta impartire intr-un p, dupa care (in acest daca) se afiseaza i-ul (factorul prim) si p-ul (puterea factorului prim).

Determinarea tuturor numerelor prime până la un n citi

Prima data scriem o functie prim, cu ajutorul careia putem determina daca un numar este prim sau nu, aceasta functie returneaza 0 daca numarul nu este prim si respectiv 1 daca numarul este prim.

In functia prim: daca numarul este mai mic decat 2 atunci se returneaza 0; altfel daca numarul este divizibil cu 2 si este diferit de 2 se returneaza 0; altfel se parcurge cu un for de la d = 3 pana la radical din numarul pe care vrem sa il testam daca este prim si parcurgerea este din 2 in 2; daca d se imparte la numarul nostru si nu avem rest, atunci se returneaza 0; in final, daca niciuna dintre conditii nu a fost indeplinita, atunci se returneaza 1, adica numarul nostru este prim.

In programul principal citim un numar n si parcurgem cu un for de la d = 3 pana la n inclusiv si din 2 in 2. Daca apelam functia prim de d si ne returneaza 1 inseamna ca d este un numar prim si se afiseaza.

numar\_cifre(n)

n, k numere naturale

{

k <- 0

cat timp n ≠ 0

{

K <- k + 1

n <- n / 10

}

returneaza k

}

Program principal

{

a, b, cifa, cifb, aux, i, j, nr, t, k = 0 numere naturale

s[20] = { 0 } sir

se citesc a, b

t <- numar\_cifre(a) + numar\_cifre(b)

aux <- b

cat timp a ≠ 0

{

cifa <- a % 10

a <- a / 10

b <- aux

k <- k + 1

i <- k, j <- k + 1

cat timp b ≠ 0

{

cifb <- b % 10

b <- b / 10

nr <- cifa \* cifb

s[i] <- s[i] + nr % 10, nr <- nr / 10

s[j] <- s[j] + nr % 10

i <- j

j <- j + 1

}

Sfarsit cat timp

}

Sfarsit cat timp

pentru i <- 1, i <= t, I <- I + 1

{

afiseaza s[i] " "

}

Sfarsit pentru

Afiseaza linie noua

z <- 0 numar natural

pentru i <- 1, i <= t, I <- I + 1

{

Daca s[i] >= 10

{

s[i + 1] <- (s[i] / 10) % 10 + s[i + 1]

daca s[i + 1] = 0

{

z <- z + 1

}

Sfarsit daca

alfel

{

z <- 0

}

Sfarsit altfel

s[i] <- s[i] % 10

}

}

Sfarsit pentru

daca s[i + 1] = 0

{

z <- z + 1

}

Sfarsit daca

alfel

{

z <- 0

}

Sfarsit altfel

t <- t - z

pentru i <- t, i >= 1, I <- I - 1

{

afiseaza s[i]

}

Sfarsit pentru

}

Pun algoritmul si in c++ daca vreti sa il testati:

#include<iostream>

using namespace std;

int numar\_cifre(int n)

{

int k = 0;

while (n)

{

k++;

n = n / 10;

}

return k;

}

int main()

{

int a, b, cifa, cifb, aux, i, j, nr, t, k = 0;

int s[20] = { 0 };

cin >> a >> b;

t = numar\_cifre(a) + numar\_cifre(b);

aux = b;

while (a)

{

cifa = a % 10;

a = a / 10;

b = aux;

k++;

i = k; j = k + 1;

while (b)

{

cifb = b % 10;

b = b / 10;

nr = cifa \* cifb;

s[i] = s[i] + nr % 10; nr = nr / 10;

s[j] = s[j] + nr % 10;

i = j;

j++;

}

}

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

{

cout << s[i] << " ";

}

cout << endl;

int z = 0;

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

{

if (s[i] >= 10)

{

s[i + 1] = (s[i] / 10) % 10 + s[i + 1];

if (s[i + 1] == 0)

{

z++;

}

else

{

z = 0;

}

s[i] = s[i] % 10;

}

}

if (s[i + 1] == 0)

{

z++;

}

else

{

z = 0;

}

t = t - z;

for (int i = t; i >= 1; i--)

{

cout << s[i];

}

}