Algoritmică şi programare – Laborator 3

Următorul algoritm calculează cel mai mare divizor comun şi cel mai mic multiplu comun a două numere naturale, nenule, a şi b, citite de la tastatură. Algoritmul are două variante: prin împărţiri repetate şi prin scăderi repetate.

Varianta 1. Se foloseşte algoritmul lui **Euclid**, care atribuie lui b restul împărţirii lui a la b, iar lui a vechea valoare a lui b. Algoritmul se termină când b=0.

Paşii algoritmului sunt:

1. Se împarte a la b şi se obţine restul r;
2. Se execută operaţiile de atribuire a<-b şi b<-r;
3. Dacă b este diferit de 0 atunci se revine la pasul 1, altfel c.m.m.d.c. este a.

```
#include<iostream>
using namespace std;
void main()
{
	int a, b, r, x, y;
	cout << "Programul calculeaza c.m.m.d.c. si c.m.m.m.c a doua numere
	date" << endl;
	cout << "Introduceti primul numar : ";
	cin >> a; x=a;
	cout << "Introduceti al doilea numar : ";
	cin >> b; y=b;
	while(b!=0) {
		r=a%b; // calculeaza restul impartirii lui a la b
		a=b;
		b=r;
	}
	cout << "c.m.m.d.c al numerelor "<< x << " si " << y << " este : " << a
	<< endl;
	cout << "c.m.m.m.c al numerelor "<< x << " si " << y << " este : " <<
	x*y/a << endl;
}
```

Varianta 2. Folosim algoritmul de scădere repetată a valorii celei mai mici din valoarea cea mai mare. Rezolvarea problemei se bazează pe condiţia a≠b.

Paşii algoritmului sunt:

1. Se scade din numărul mai mare numărul mai mic;
2. Dacă numerele sunt diferite se revine la pasul 1, altfel c.m.m.d.c. este a.

```
#include<iostream>
using namespace std;
void main()
{
	int a, b, x, y;
	cout << "Programul calculeaza c.m.m.d.c. si c.m.m.m.c a doua numere
	date" << endl;
	cout << "Introduceti primul numar : ";
```

```
	cin >> a; x=a;
	cout << "Introduceti al doilea numar : ";
	cin >> b; y=b;
	while (a!=b) {
		if(a>b)
			a=a-b;
		else
			b=b-a;
	    }
	cout << "c.m.m.d.c al numerelor "<< x << " si " << y << " este : " << a
	<< endl;
	cout << "c.m.m.m.c al numerelor "<< x << " si " << y << " este : " <<
	x*y/a << endl;
}
```

Observație. Pentru numerele naturale considerate a și b are loc următoarea relație:

$$a*b=c.m.m.d.c.(a,b)*c.m.m.m.c.(a,b),$$

formulă ce permite astfel determinarea și a celui mai mic multiplu comun a acestor două numere.

Algoritmi de interschimbare a valorilor a două variabile

Interschimbarea valorilor a două variabile de memorie x și y nu se face prin simpla atribuire a noii valori, deoarece secvența de atribuiri x<-y și y<-x ar duce la pierderea valorii lui x, iar secvența y<-x și x<-y ar duce la pierderea valorii lui y.

Pentru a realiza interschimbarea conținutului celor două variabile de memorie se poate folosi una din următoarele variante de algoritm:

Varianta 1. Interschimbarea valorilor a două variabile a și b prin folosirea unei variabile intermediare x. Pașii algoritmului sunt:

1. Se salvează valoarea primei variabile a în variabila x;
2. Se atribuie primei variabile valoarea celei de a doua variabile;
3. Se atribuie celei de a doua variabile b valoarea care a fost salvată în a treia variabilă x.

Exemplu. Se citesc două variabile de la tastatură. Să se interschimbe conținutul lor și să se afișeze apoi pe ecran.

```
#include<iostream>
using namespace std;
void main()
{
	int a, b, x;
	cout<<"Introduceti a : "; cin>>a;
	cout<<"Introduceti b : "; cin>>b;
	x=a; a=b; b=x ;
	cout<<"Valoarea lui a devine : "<<a<<endl;
	cout<<"Valoarea lui b devine : "<<b<<endl;
}
```

Varianta 2. Interschimbarea valorilor a două variabile a şi b fără folosirea unei variabile intermediare. Se folosesc identităţile matematice:

a=(a-b)+b şi b=((a-b)+b)-(a-b)

```
#include<iostream>
using namespace std;
void main()
{
	int a,b;
	cout<<"Introduceti a : "; cin>>a;
	cout<<"Introduceti b : "; cin>>b;
	a=a-b;
	b=a+b;
	a=b-a;
	cout<<"Valoarea lui a devine : "<<a<<endl;
	cout<<"Valoarea lui b devine : "<<b<<endl;
}
```

Problemă. La un concurs, comisia de notare este formată din n membri. Să se scrie algoritmul de calcul al mediei, ştiind că nota cea mai mică şi nota cea mai mare nu sunt luate în considerare la calcularea mediei.

Procedăm în felul următor. La o singură citire este determinat atât minimul cât şi maximul şi, de asemenea, toate notele sunt adunate la o sumă. La final din sumă sunt scăzute minimul şi maximul şi suma se împarte la n-2. Este necesar, de asemenea, şi un contor care să numere câte note au fost introduse.

```
#include<iostream>
using namespace std;
void main()
{
	double a, min, max, s=0, c=0, media;
	cout<<"Introduceti nota : "; cin>>a;
	min=a; max=a;
	while(a>0)
	{
		c=c+1; s=s+a;
		if (a>max) max=a;
		if (a<min) min=a;
			cout<<"Introduceti nota : "; cin>>a;
	}
	s=s-min; s=s-max;
	media=s/(c-2);
	cout<<"Media celor "<<c<<" note este "<<media<<endl;
}
```

Problemă. Să se citească un număr natural de la tastatură şi să se determine dacă acesta este număr prim sau nu.

Algoritmul de verificare dacă un număr natural n este prim constă în generarea tuturor numerelor naturale mai mari sau egale cu 2 şi mai mici sau egale cu sqrt(n) şi verificarea pentru fiecare număr generat dacă îl divide pe n. Dacă există cel puţin un astfel de număr, numărul n nu este prim. Pentru a şti dacă există cel puţin un număr care îl divide pe n, se va folosi o variabilă logică x, care va avea valoarea logică True dacă numărul este prim şi False dacă numărul nu este prim. Se presupune de la început că

numărul este prim (variabila x se iniţializează cu 1) şi pentru primul număr găsit în şirul de numere generate, care îl divide pe n, se va schimba valoarea variabilei x în False. Pentru generarea şirului de numere se foloseşte o variabilă contor care va fi iniţializată cu 2 şi care se va incrementa cu 1 până va avea valoarea [sqrt(n)].

Paşii algoritmului sunt:

1. Se iniţializează variabila x cu True;
2. Se generează primul număr din şirul de numere, prin operaţia i<-2;
3. Dacă n se divide cu i, atunci se schimbă valoarea variabilei x prin operaţia x<-F; altfel, se generează următoarea cifră din şirul de numere prin incrementarea contorului i++;
4. Dacă i<=sqrt(n) şi x=T se revine la pasul 3;
5. Dacă x=T se afişează mesajul "Numărul este prim" altfel se generează mesajul "Numarul nu este prim".

```
#include<iostream>
#include<math.h>
using namespace std;
void main()
{
	int i, n;
	bool x=1;
	cout<<"Introduceti numarul pe care doriti sa il verificati : "; cin>>n;
	for(i=2;i<sqrt(float(n));i++)
		if(n%i==0) x=0;
	if(x==1)
		cout<<"Numarul "<<n<<" este prim !"<<endl;
	else
		cout<<"Numarul "<<n<<" nu este prim !"<<endl;
}
```

Problemă. Să se elaboreze un algoritm care determină inversul unui număr natural citit de la tastatură şi să se verifice dacă el este palindrom sau nu (un număr este palindrom dacă el este egal cu inversul său).

Algoritmul determină inversul unui număr n prin extragerea pe rând a fiecărei cifre (începând cu cifra unităţilor) din numărul n şi compunerea unui nou număr cu aceste cifre.

Paşii algoritmului sunt:

1. Se citeşte numărul n;
2. Se iniţializează numărul inv cu valoarea 0;
3. Se extrage cifra cea mai semnificativă din numărul n şi se adună cifra la numărul inv*10, prin operaţia inv=inv*10+n%10;
4. Se elimină din numărul n cifra extrasă, cu operaţia n=n/10;
5. Dacă n este diferit de 0 se revine la pasul 3.

```
#include<iostream>
using namespace std;
void main()
{
	int n, nr, inv;
	cout<<"Introduceti numarul natural pe care vreti sa il inversati : ";
	cin>>n; nr=n;
	inv=0;
```

```
        while(n!=0) {
        inv=inv*10+n%10;
        n=n/10; }
        cout<<"Inversul numarului considerat este : "<<inv<<endl;
        if(nr==inv)
            cout<<"Numarul este palindrom !"<<endl;
        else
            cout<<"Numarul nu este palindrom !"<<endl;
}
```

Problemă. Să se elaboreze un algoritm care citeşte de la tastatură un număr natural şi îl descompune în factori primi.

Pentru a afişa numai divizorii primi ai unui număr n, din număr se elimină toţi divizorii i găsiţi la un moment dat, operaţia repetându-se până când sunt eliminaţi toţi divizorii din numărul n (n are valoarea 1).

Pasii algoritmului sunt:

1. Se iniţializează şirul de numere cu care se va împărţi n cu primul divizor posibil, adică i<-2;
2. Dacă i îl divide pe n, atunci se afişează i şi, atâta timp cât n se împarte la i, se execută împărţirea lui n la i pentru a elimina toate puterile lui i din n.
3. Se trece la următorul divizor posibil, prin incrementarea lui i;
4. Dacă n este diferit de 1 atunci se revine la pasul 2; altfel algoritmul se termină.

```
#include<iostream>
using namespace std;
void main()
{
      int n, i, k;
      cout<<"Introduceti nr. natural pe care vreti sa il descompuneti : ";
      cin>>n;
      i=2;
      while(n!=1) {
            if(n%i==0) {
                  k=0;
                  while(n%i==0) {
                      k++;
                      n=n/i;
                  }
                  cout<<i<<" la puterea "<<k<<endl;
            }
            i++;
      }
}
```

Problemă (schema lui Horner). Să se elaboreze doi algoritmi care să convertească un număr din baza 10 într-o baza b şi reciproc.

```
// Algoritm de conversie din baza 10 in baza b<10
#include<iostream>
using namespace std;
void main()
{
```

```
	int n10, nb, b, p;
	cout<<"Introduceti numarul in baza 10 ce trebuie transformat : ";
	cin>>n10;
	cout<<"Introduceti noua baza (<=9) : "; cin>>b;
	nb=0;
	p=1;
	while(n10!=0) {
		nb=nb+p*(n10%b);
		n10=n10/b;
		p=p*10;
	}
	cout<<"Numarul in baza "<<b<<" este "<<nb<<endl;
}

// Algoritm de conversie din baza b<10 in baza 10
#include<iostream>
using namespace std;
void main()
{
	int n, b, inv, inv2;
	cout<<"Introduceti numarul natural pe care vreti sa il convertiti : ";
	cin>>n;
	cout<<"Introduceti baza din care convertiti : "; cin>>b;
	// in secventa ce urmeaza determinam inversul numarului n
	inv=0;
	while(n!=0) {
		inv=inv*10+n%10;
	n=n/10; }
	// la o noua inversare trecem de la baza b la baza 10
	n=0;
	while(inv!=0) {
		n=n*b+inv%10;
		inv=inv/10; }
	cout<<"Numarul convertit in baza 10 este : "<<n<<endl;
}
```