Structuri de bază(liniară, alternativă şi repetitivă)

Programarea structurată este o manieră de concepere a programelor, potrivit unor reguli bine definite şi independent de limbajul de programare. Scopul programării structurate este elaborarea unor programe uşor de scris, de înţeles şi de modificat.

Programarea structurată are la bază teorema de structură care afirmă că orice algoritm cu o singură intrare şi o singură ieşire poate fi reprezentat ca o combinaţie de trei tipuri de structuri de control – secvenţa, decizia şi ciclul cu test iniţial. Se mai admite folosirea a încă trei tipuri de structuri de control – selecţia, ciclul cu test final şi ciclul cu contor.

Reprezentarea algoritmilor în pseudocod

1. Declararea variabilelor

variabilă tip;

2. Atribuirea variabilă←expresie;

3. Operaţia de citire(intrare) citeşte variablia1, variabila2, ... , variabila$_n$;

4. Operaţia de scriere(ieşire) scrie expresie1, expresie2, ... , expresie$_n$;

5. Structura liniară(secvenţa)

Este o succesiune de operaţii ce realizează o prelucrare(transformare) a datelor. Operaţiile sunt executate una după alta, în ordinea scrierii.

6. Structura alternativă(decizia)

Permite alegerea unei operaţii/secvenţe de operaţii din două alternative posibile.

a. Cu două ramuri

```
dacă C atunci A;
          altfel   B;
```

Am notat cu C condiţia care este o propoziţie logică ce poate avea doar una din valorile 1 sau 0 (adevărat sau fals). Dacă rezultatul propoziţiei logice C este 1(adevărat) atunci se execută secvenţa de operaţii A, altfel se execută secvenţa de operaţii B.

b. Cu o ramură dacă C atunci A;

```
Dacă condiţia C este adevărată,
     atunci se execută secvenţa A.
```

7. STRUCTURA REPETITIVA

Exista trei tipuri de structuri repetitive:

1) Structura cu numar cunoscut de repetitii (FOR)

2) Structura cu numar necunoscut de repetitii si cu test initial (WHILE)

3) Structura cu numar necunoscut de repetitii si cu test final (DO-WHILE)

Atunci când anumite operații trebuie repetate de un număr de ori cunoscut (de obicei de un număr mare de ori, care nu permite scrierea repetată a operațiilor în algoritm), se utilizează structura repetitivă pentru.

```
pentru (contor= val_initiala; contor <= val_finala ;contor = contor + pas)
     Instructiuni
Sfarsit pentru
```

Instructiunea repetitiva for in limbajul C++

```
for(contor=expresie_initiala; contor<=expresie_finala; contor=contor+pas)
     instructiuni
```

Executia instructiunii for are etapele urmatoare:

Etapa 1. Se inițializează contorul (variabila care numără pașii)

Etapa 2. Se verifică daca valoarea contorului este mai mica sau egala cu valoarea finala

- Dacă DA, se execută instrucțiuni, apoi contorul creste cu valoarea pasuluișsi revenim la Etapa 2
- Dacă NU este îndeplinită, se oprește execuția instructiunii for

Obs: daca sunt două sau mai multe instrucţiuni în for se vor grupa între acolade {...}

STRUCTURA REPETITIVA CU NUMAR CUNOSCUT DE PASI - FOR

Exemplu :Calculul sumei primelor n numere naturale

Algoritm suma

n,i,s intregi;
citeste n;
s=0;
pentru i=1,n executa
s=s+i;
scrie s;
sfarsit algoritm
	// suma primelor n numere naturale
	#include <iostream>
	using namespace std;
	int main ()
	{
	int n,i,s=0;
	cin>>n; // citim n
	for (i=1; i<=n; i++)
	s=s+i;
	cout << endl<<s;
	}

Divizorii unui numar natural Sa se afiseze toti divizorii unui numar natural nenul n.

```
int n,d;
cin>>n; // citim numarul
if(n==1) cout<<1;
else
```

```
{ cout<<1<<’  ’ ;
 for(d=2;d<=n/2;d++)
 if(n%d==0)
 cout<<d<<’ ’;
cout<<n;
}
```

Verificarea daca un numar este prim

```
 int n,d,prim;
cin>>n; // citim numarul
if(n<2) prim=0;
else
 {  prim=1; //presupunem ca numarul este prim
 for(d=2;d<=n/2;d++)
 if(n%d==0)     { prim=0; break;} }
if (prim==1)
cout<<”numar prim”;
 else
cout<<”numarul nu este prim”; }
```

Citirea si prelucrarea pe rând a n numere naturale

```
int n, i, x;
cin >> n;
for (i =1 ; i <= n; i++ )
{ cin >> x;
  //prelucrarea numarului x
 }
```

STRUCTURA REPETITIVA CU TEST INITIAL WHILE –Aplicatii

rezolvate

Exemplu : Calculul celui mai mare divizor comun a doua numere naturale date

Cmmdc(a,b)
a, b, rest intregi;
citeste a, b; rest=a %b
cat timp rest !=0 executa
  a=b
  b=rest
  rest=a % b;
scrie b;
sfarsit algoritm
	// cmmdc(a,b)
	int main ()
	{
	int a,b,rest;
	cin>>a>>b; // aici citim a si b
	rest=a%b; while (rest!=0) { a=b; b=rest;
	rest=a%b;
	}
	cout << endl<<b;
	}

Probleme propuse

1) Dandu-se doua numere naturale cu maxim patru cifre in variabilele a si b, se cere sa se afiseze cate numere impare sunt mai mici sau egale decat b si mai mari sau egale decat a.

2) Sa se determine cate numere din intervalul [a,b] sunt divizibile cu un numar dat k (a,b,k se vor citi).

3) Determinati numarul numerelor naturale mai mici sau egale decat n care sunt divizibile cu 2 sau 3.

4) Se citesc de la tastatura 2 numere naturale a si b (a<b,a>1,b<1000). Sa se afiseze toate numerele din intervalul [a,b], care au un numar par de divizori proprii. Ex. Pentru a=10,b=25, se vor afisa numerele 10,12,14,15,18,20,21,22,24 (Numarul 10 are 2 divizori proprii: 2 si 5; 2 este numar par)

5) Să se afişeze toate numerele până la 100 care au patru divizori.