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`
	`int main ()`
	`{`
	`int n,i,s=0;`
	`f>>n; // citim n`
	`for (i=1; i<=n; i++)`
	`s=s+i;`
	`g<<s; //afisam suma s`
	`}`

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

```
int main()
{
      int d,n;
     f>>n; // citim numarul
     if(n==1) cout<<1;
     else
     { cout<<1<<'  ' ;
          for(d=2;d<=n/2;d++)
               if(n%d==0)
               g<<d<<' '; //afisam divizorii
        g<<n;
     }
}
```

Număr prim

```
int main()
{
      int n,d,prim;
     f>>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)
          g<<"numar prim";
      else
          g<<"numarul nu este prim";
     }
}
```

Problema: Se citeste din fisierul numere.in un numar natural nenul n si apoi un sir de n numere naturale. Sa se afiseze in fisierul numere.out toate numerele impare din sir si numerele lor de ordine in cadrul sirului.

Exemplu

numere.in	numere.out
6 78 55 20 171 83 46	55 pozitia 2 171 pozitia 4 83 pozitia 5