Structura repetitivă cu test final

- Atunci când se dorește repetarea unei acțiuni de un anumit număr de ori este necesară utilizarea unei structuri repetitive. În funcție de numărul repetărilor acestea se clasifică în:
- Structuri repetitive cu număr nedeterminat de pași
  - Cu test inițial
  - Cu test final
- Structuri repetitive cu număr fix de pași

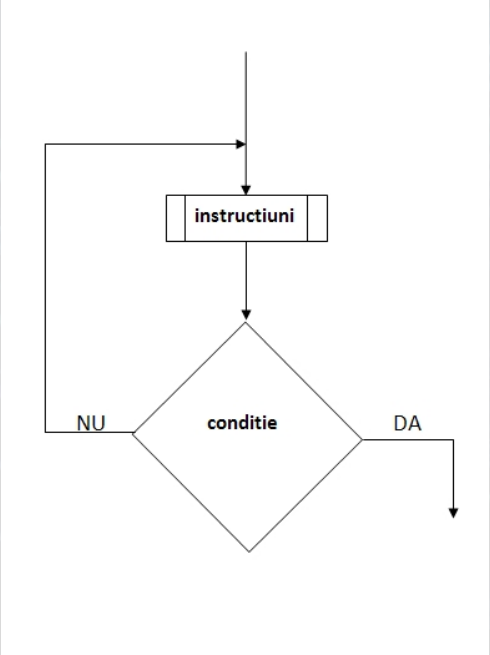
Structura repetitivă cu test final

- Numele acestei structuri repetitive provine de faptul că după executarea unei instrucțiuni se verifică îndeplinirea unei expresii logice. În limbajul C++ este cunoscută sub denumirea de structura **do-while**.
- Reprezentarea acestei structuri este:

pseudocod

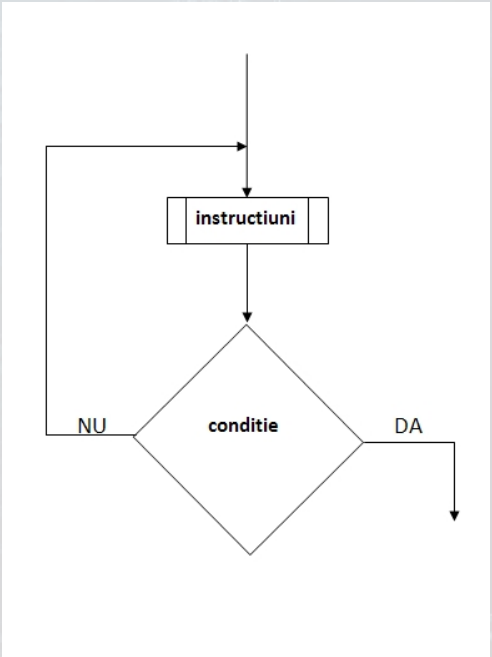
```
execută
    instrucțiune
cât timp (expresie logică)
```

schema logică



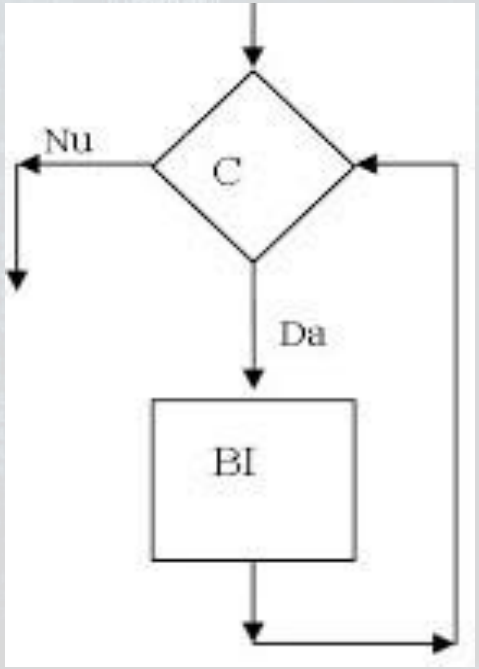
Mod de execuție:

- Pas 1 :se executa instructiuni
- Pas 2: Se determină valoarea de adevăr a expresiei logice (C);
- Pas 3: daca conditia este **falsă**, atunci se revine la pasul 1 (se executa instructiuni)
- Pas 3: Dacă expresia logică este **adevărată**, atunci se părăsește structura repetitivă.



Observații

- structura repetitiva cu test final este o structura repetitiva cu numar necunoscut de pasi
- instructiunile se executa pana cand conditia este adevarata
- instructiunile se executa cel putin odata
- conditia trebuie sa devina adevarata pentru ca structura repetitiva sa se termine



- Instrucțiunea **do while** este instrucțiunea C++ folosită pentru a implementa în limbaj de programare o structură repetitivă cu număr necunoscut de pași și test final.
- Sintaxa instrucțiunii este:

```
Do
    Instrucțiune;
while (Expresie);
```

```
do
{
Instrucțiune1;
Instrucțiune2;
..............
} while (Expresie);
```

Mod de execuție:

- Se execută `Instrucțiune`;
- Se evaluează `Expresie`
- Dacă valoarea `Expresie` este **fals**
  - Se execută `Instrucțiune`;
- Dacă `Expresie` este **adevărat**, se trece la instrucțiunea de după **while**.

Observații

- `Instrucțiune;` se execută cât timp `Expresie` este nulă – condiție falsă.
- `Instrucțiune;` se execută cel puțin o dată.
- `Instrucțiune;` poate fi orice fel de instrucțiune, dar una singură. Dacă sunt necesare mai multe instrucțiuni, se va folosi instrucțiunea compusă.
- Este necesar ca cel puțin o variabilă care apare în `Expresie` să-și modifice valoarea în `Instrucțiune;`. Altfel se obține o buclă infinită.

```
cin>>n;
inițializare variabile de manevră (sume, contoare cu valoarea 0, produse cu valoarea 1
prelucrare caz n=0
Do
{ ……;//prelucrează ultima cifră (n%10)
n=n/10;//  ”taie” ultima cifră din numărul n
}
while(n);
```

Atenție!

- La sfârșitul acestui algoritm valoarea lui **n** este **0**
- Un caz particular îl reprezintă situația în care inițial valoarea lui **n** este **0**, el trebuie prelucrat separat înainte de instrucțiunea **while**
- Numărul de operații efectuate de algoritm este egal cu numărul cifrelor lui **n**

1. Produsul cifrelor impare ale unui număr **n**, întreg.

```
#include <iostream>
using namespace std;
int main()
{
    int n,p=1,c;
    cin>>n;
    if(n==0)
        cout<<"introduceti un numar diferit de 0";
    else
    {
        do
        {
            c=n%10;
            if(c%2==1)
                p=p*c;
            n=n/10;
        }
        while(n);
        cout<<"produsul cifrelor impare este : "<<p;
    }
    return 0;
}
```

2. Suma cifrelor unui număr întreg **n**

```
#include <iostream>
using namespace std;
int main()
{
    int n,s=0,c;
    cin>>n;
    if(n==0)
        cout<<"introduceti un numar diferit de 0";
    else
    {
        do
        {
            c=n%10;
            s=s+c;
            n=n/10;
        }
        while(n);
        cout<<"Suma cifrelor numarului este : "<<s;
    }
    return 0;
}
```