Introducere în Programarea calculatoarelor
Laborator 4

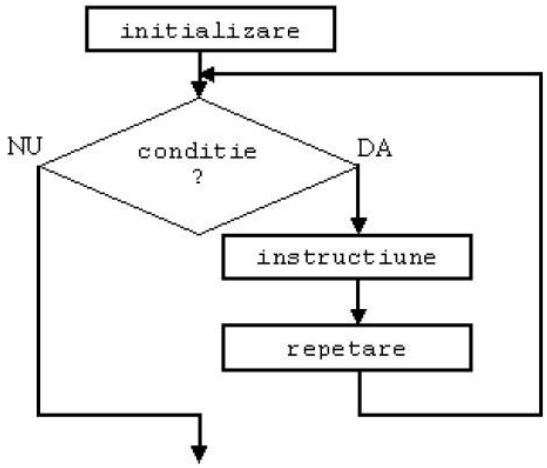
Instrucțiunile repetitive

Instrucțiunea *for*

Instrucțiunea *for* permite repetarea unei secvențe de instrucțiuni atâta timp cât o condiție este îndeplinita. În plus, oferă posibilitatea execuției unei expresii de inițializare (înainte de a începe bucla) şi a unei alte expresii la sfârșitul fiecărei iterații. Sintaxa este următoarea:

```
for (inițializare; condiție; repetare)
instrucțiune;
```

Schema logica aferenta acestei bucle *for* este următoarea:



Schemă logică - instrucțiunea for

Se observă că bucla *for* reprezintă un *ciclu cu test inițial*, daca nu este îndeplinita condiția, există posibilitatea ca instrucțiunea din corpul buclei sa nu se execute niciodată.
Dacă se dorește repetarea mai multor instrucțiuni în corpul buclei, acestea trebuie încadrate între acolade:

```
for (inițializare; condiție; repetare)
{
  instrucțiune_1;
  instrucțiune_2;
instrucțiune_3;
}
```

Introducere în Programarea calculatoarelor
Laborator 4

Exemplu: Sa se scrie un program în C care sa afișeze valorile funcției sinus între doua limite specificate de utilizator, cu un pas specificat de utilizator:

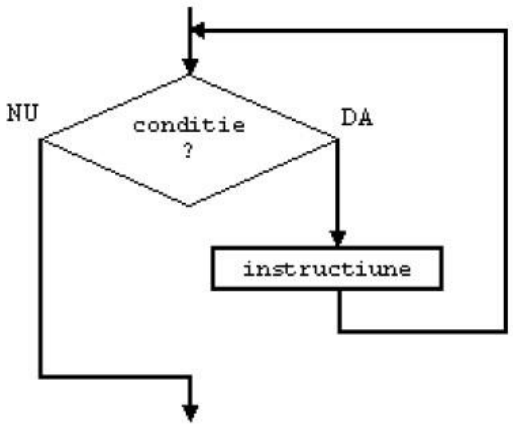
```
#include <stdio.h>
#include <math.h> int
main (void)
{ float start, stop, pas, x, y;
   printf ("Valoarea de inceput: ");
   scanf ("%f", &start);
   printf ("Valoarea de sfarsit: ");
   scanf ("%f", &stop);
   printf ("Pasul: ");
   scanf ("%f", &pas);
  for (x = start; x <= stop; x = x + pas) {
        y = sin (x);
       printf ("sin(%f)=%f \n", x, y);
}
return 0;
}
```

Instrucțiunea *while*

Instrucțiunea *while* oferă posibilitatea de a repeta o anumita secvență de program atâta timp cât o anumita condiție este îndeplinita. Sintaxa este următoarea:

```
while (conditie)
instructiune;
```

Instrucțiunea while permite construirea de *cicluri cu test inițial*, deci dacă nu este îndeplinita condiția, instrucțiunea nu se va executa niciodată. Schema logica aferenta este următoarea:



Introducere în Programarea calculatoarelor
Laborator 4

Dacă se dorește repetarea mai multor instrucțiuni în corpul buclei, acestea trebuie grupate între acolade:

```
while (condiție)
{
instrucțiune_1;
instrucțiune_2;
instrucțiune_3;
}
```

Exemplu: Sa se scrie un program care sa simuleze inițializarea si actualizarea stocului unui magazin:

```
#include <stdio.h>
 int main (void)
{ int stoc;
  int cant;
  printf ("Stocul initial=");
 scanf ("%d", &stoc);
 while (stoc > 0)
          {
            printf ("Ce cantitate doriti sa cumparati ?");
           scanf ("%d", &cant);
           stoc = stoc – cant;
          }
          printf ("Stocul a fost epuizat ! \n");
return 0; }
```

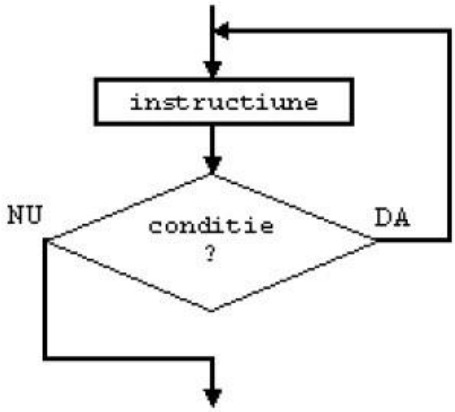
Instrucțiunea *do…while*

Această instrucţiune permite repetarea unei secvențe de instrucțiuni atâta timp cât o condiție este îndeplinita. Permite implementarea unui *ciclu cu test final*, deci instrucțiunile din interiorul buclei se executa sigur cel puțin o data.

Sintaxa acestei instrucțiuni este următoarea:

```
do instrucțiune;
while (condiție);
```

Schema logica aferenta este:



Schemă logică - instrucțiunea do-while

Dacă se dorește repetarea mai multor instrucțiuni în corpul buclei, acestea trebuie încadrate între acolade:

```
do
{
instructiune_1;
instructiune_2;
instructiune_3;
}while (conditie);
```

Exemplu: Sa se scrie un program C care sa citească de la tastatura nota unui student. Programul trebuie sa permită numai introducerea unei note corecte (de la 1 la 10).

```
#include
<stdio.h>
int main
(void)
{
        int nota;
        do
        {
        printf ("nota="); scanf
        ("%d", &nota);
        }while ((nota < 1) ||
        (nota > 10));
```

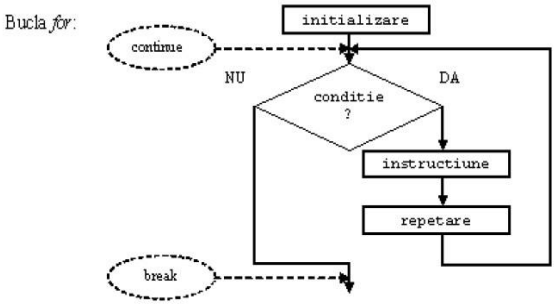
```
        printf ("Nota introdusa
        este %d \n", nota);
return 0; }
```

Instrucțiunile *break* și *continue*

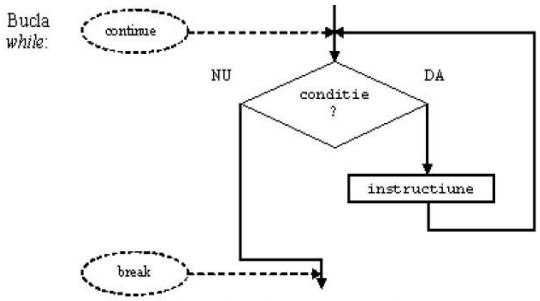
În limbajul C, în cadrul oricăreia dintre cele trei instrucțiuni repetitive prezentate anterior exista posibilitatea de a întrerupe forțat execuția iterației curente sau de a întrerupe forțat execuția întregii bucle.

Pentru a întrerupe iterația curenta si a forța re-evaluarea condiției buclei, se utilizează instrucțiunea *continue* în corpul buclei.

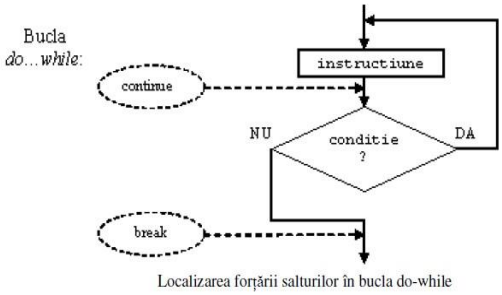
Pentru a întrerupe execuția întregii bucle, se folosește instrucțiunea *break* în corpul buclei. Locațiile unde se forțează salturile sunt indicate în figurile următoare:



Localizarea forțării salturilor în bucla for



Localizarea forțării salturilor în bucla while



Localizarea forțării salturilor în bucla do-while

Introducere în Programarea calculatoarelor
Laborator 4

Exemplu: Următorul program citește N numere de la tastatura si număra câte sunt pare si câte sunt impare:

```
#include <stdio.h>
int main (void)
{
        int N, x, i, pare;
        printf ("N=");
        scanf ("%d", &N);
        pare = 0;
        for (i = 0; i < N; i++) {
         scanf ("%d", &x);
        if ((x % 2) != 0)
                continue; /* salt la urmatoarea iteratie */
        pare++;
        }
        printf ("pare: %d impare: %d \n", pare, N - pare);
        return 0; }
```

Exemplu: Următorul program citește numere de la tastatura până când s-a introdus numărul 0:

```
#include <stdio.h>
void main (void)
{ int x;
        while (1) /* bucla infinita, se poate intrerupe doar cu break*/
        {
        scanf ("%d", &x);
        printf ("ati tastat: %d", x);
         if (x == 0)
            break;
        }
return 0;
        }
```

Introducere în Programarea calculatoarelor
Laborator 4

Probleme propuse:

1. Să se scrie un program în C care citește de la tastatură un număr și afișează toți divizorii acestuia.

2. Să se scrie un program în C care citește de la tastatură un număr întreg și afișează daca acesta este număr prim sau nu.

3. Să se scrie un program în C care citește de la tastatură un șir de numere încheiat cu numărul 0 și afișează suma numerelor introduse.

4. Să se scrie un program în C care citește de la tastatură un șir de numere încheiat cu numărul 0 și afișează media aritmetică a numerelor introduse.

5. Să se scrie un program în C care citește de la tastatură un șir de numere încheiat cu numărul 0 și afișează maximul dintre numerele introduse.

6. Se citește un număr n. Să se calculeze produsul cifrelor pare din număr.

7. Se citesc n numere. Să se calculeze pentru fiecare număr produsul cifrelor prime.

8. Se citesc n numere. Să se determine suma numerelor divizibile cu 5 și produsul celor prime.

9. Să se afișeze toate numerele prime de la 1 la n.

10. Se citește de la tastatură un șir de n numere. Să se afișeze câte din acestea sunt mai mici sau egale cu un număr dat k și să se facă suma lor.

11. Să se afișeze pe ecran următoarele secvențe de numere pentru un număr n citit de la tastatură.

Exemplu: pentru n=4 se va afișa:

```
1 2 3 4
1 2 3
1 2
1

1
1 2
1 2 3
1 2 3 4

4 3 2 1
4 3 2
4 3
4
```

4
4 3
4 3 2
4 3 2 1

12. Se citesc de la tastatură n numere. Să se calculeze pentru fiecare număr produsul cifrelor prime și să se afișeze toate produsele care sunt mai mici decât un număr nr citit anterior.

13. Se citesc numere de tip întreg până se vor introduce 2 numere consecutive. Să se afișeze câte din aceste numere sunt pătrate perfecte.

14. Se dă de la tastatură un număr cu cel mult 9 cifre. Să se afișeze cifrele numărului împreună cu frecvența lor de apariție.

15. Se citesc numere până la întâlnirea numărului 0. Să se afișeze numerele atunci când al doile număr este egal cu suma cifrelor pare din primul număr.