
 Universitatea "Politehnica" din București  
 Facultatea de Electronică, Telecomunicații și  
 Tehnologia Informației

# Programarea Calculatoarelor (limbajul C)

## Curs 5 – Instrucțiunile Condiționale și Structurile Repetitive

Prof. Bogdan IONESCU

### Instrucțiunile condiționale

> Limbajul de programare C propune următoarele instrucțiuni și operatori condiționali:

- structura **if else** (dacă ... atunci ...),
- operatorul condițional **?:** (echivalent if else),
- structura de selecție **switch case** (if generalizat)

Curs Programarea Calculatoarelor, Prof. Bogdan IONESCU 3/59

### Cuprins

- 5.1. Instrucțiunile condiționale
- 5.2. Structurile repetitive
- 5.3. Instrucțiunile break, continue și exit
- 5.4. Probleme recapitative

Curs Programarea Calculatoarelor, Prof. Bogdan IONESCU 1/59

### Instrucțiunile condiționale (continuare)

- Structura if else

> Permite executarea unor instrucțiuni în funcție de valoarea de adevăr a unei expresii.

*Sintaxă:*

```
if (<expresie>
{
<secvență instrucțiuni>;
}
```

*său:*

```
if (<expresie>
{
<secvență instr. 1>;
}
else
{
<secvență instr. 2>;
}
```

Curs Programarea Calculatoarelor, Prof. Bogdan IONESCU 4/59

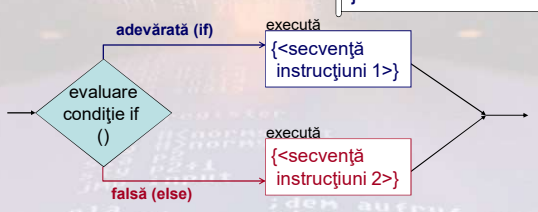
### 5.1. Instrucțiunile condiționale

Curs Programarea Calculatoarelor, Prof. Bogdan IONESCU 2/59

### Instrucțiunile condiționale (continuare)

- Structura if else (continuare)

*Mod de execuție:*



```
if (<expresie>
{
<secvență instrucțiuni 1>;
}
else
{
<secvență instrucțiuni 2>;
}
```

Curs Programarea Calculatoarelor, Prof. Bogdan IONESCU 5/59

### Instrucțiunile condiționale (continuare)

- Structura if else (continuare)

*Exemplu simplu:*

```
if (x==100)
{
    printf("x are valoarea 100");
}
else
{
    printf("x este diferit de 100");
}
```

dacă x are valoarea 100 atunci se afișează primul text.

în caz contrar, se afișează al doilea text.

Curs Programarea Calculatoarelor, Prof. Bogdan IONESCU 6/59

### Instrucțiunile condiționale (continuare)

- Structura if else (continuare)

*Exemplu 4:*

```
if (x>0)
{
    if (y==5)
    {
        if (z!=14)
        {
            printf("3 conditii adevarate");
        }
        else
        {
            printf("2 conditii adevarate");
        }
    }
    else
    {
        printf("din ce ce if face parte?");
    }
}
```

aceste instrucțiuni fac parte din if (x>0)

if (y==5) nu are else.

acest else este al primului if.

Curs Programarea Calculatoarelor, Prof. Bogdan IONESCU 9/59

### Instrucțiunile condiționale (continuare)

- Structura if else (continuare)

*Exemplu 2:*

```
if (x>0)
{
    printf("x este pozitiv");
}
else if (x<0)
{
    printf("x este negativ");
}
else
{
    printf("x este 0");
}
```

dacă x > 0 atunci se afișează primul text.

în caz contrar, se verifică dacă x < 0.

dacă x < 0 se afișează textul al doilea.

dacă x nu este nici > 0 și nici < 0, atunci se afișează textul al treilea.

Curs Programarea Calculatoarelor, Prof. Bogdan IONESCU 7/59

### Instrucțiunile condiționale (continuare)

- Structura if else (continuare)

*Observații:*

> În cazul în care după if sau else este executată o singură instrucțiune, nu este obligatorie folosirea acoladelor { }.

```
if (a<10)
    a=a+1;
    printf("aceasta linie nu face parte din if");
```

> Convenția de evaluare a condițiilor de tip if (variabilă):

if (a)	(a) este <b>adevărată</b> dacă a != 0
else	(a) este <b>falsă</b> dacă a = 0

Curs Programarea Calculatoarelor, Prof. Bogdan IONESCU 10/59

### Instrucțiunile condiționale (continuare)

- Structura if else (continuare)

*Exemplu 3:*

```
1 if (x>0)
2   if (y==5)
3     if (z!=14)
        printf("3 conditii adevarate");
    else
        printf("2 conditii adevarate");
    else
        printf("1 conditie adevarata");
else
    printf("nici o conditie adevarata");
```

if (x>0) are ca secvență de instrucțiuni **if-ul 2**

if (y==5) are ca secvență de instrucțiuni **if-ul 3**

Curs Programarea Calculatoarelor, Prof. Bogdan IONESCU 8/59

### Instrucțiunile condiționale (continuare)

- Operatorul condițional ?:

> Operatorul condițional reprezintă o scriere prescurtată a instrucțiunii if-else:

*Sintaxă:*

```
<(condiție)> ? <expresie_1> : <expresie_2>
```

*Mod de execuție:*

- ↳ dacă (condiție) este **adevărată** → returnează expresie\_1
- ↳ dacă (condiție) este **falsă** → returnează expresie\_2

*Exemplu simplu:*

```
x=(3>1) ? 1 : 0;    x= 1    x=(-6) ? 1 : 0;    x= 1
```

Curs Programarea Calculatoarelor, Prof. Bogdan IONESCU 11/59

### Instrucțiunile condiționale (continuare)

- Operatorul condițional ?: (continuare)

**P** Enunț: să se calculeze maximul a două numere întregi x și y cu ajutorul operatorului condițional.

```
int main()
{
    int max, x, y;
    ...
    max=(x>=y) ? x : y;
    ...
}
```

cu if...

```
int main()
{
    int max, x, y;
    ...
    if (x>=y)
        max=x;
    else
        max=y;
    ...
}
```

Curs Programarea Calculatoarelor, Prof. Bogdan IONESCU 12/59

### Instrucțiunile condiționale (continuare)

- Structura switch (continuare)

```
switch (expresie)
{
    case const1:
        <secvență instr.1>;
        break;
    ...
    case constN:
        <secvență instr.N>;
        break;
    default: <secvență instr. implicită>;
}
```

Mod de execuție:

```

    graph TD
      A[evaluare expresie] -- "expresie == const1" --> B[secvență instr.1]
      A -- "expresie == constN" --> C[secvență instr.N]
      A -- "default ?" --> D[secvență instr. implicită]
      B --> B1[break]
      C --> C1[break]
      D --> D1[break]
  
```

Curs Programarea Calculatoarelor, Prof. Bogdan IONESCU 15/59

### Instrucțiunile condiționale (continuare)

- Operatorul condițional ?: (continuare)

**P** Enunț: să se calculeze minimul a trei numere întregi x, y și z cu ajutorul operatorului condițional (1 linie).

```
int main()
{
    int min,x,y,z;
    ...
    min=((x<=y)&&(x<=z)) ? x : (((y<=x)&&(y<=z)) ? y : z);
    ...
}
```

dacă x este minim returnăm x altfel →

dacă y este minim returnăm y altfel →

altfel înseamnă că z este minim

Curs Programarea Calculatoarelor, Prof. Bogdan IONESCU 13/59

### Instrucțiunile condiționale (continuare)

- Structura switch (continuare)

> Structura de selecție **switch case** poate fi văzută ca o structură if-else generalizată,

> În funcție de valoarea unei expresii, permite alegerea dintr-o multitudine de variante într-un singur nivel condițional, ceea ce este mult mai convenabil decât scrierea unor instrucțiuni if multiple,

> **Limitare:** valoarea expresiei trebuie să fie neapărat întreagă, sau să poată fi convertită la o valoare întreagă,

> **Limitare:** constantele (const1, ..., constN) nu pot să conțină expresii sau variabile din program,

Curs Programarea Calculatoarelor, Prof. Bogdan IONESCU 16/59

### Instrucțiunile condiționale (continuare)

- Structura switch

> Permite integrarea mai multor decizii într-o singură structură.

Sintaxă:

```
switch (expresie)
{
    case const1:
        <secvență instr.1>;
        break;
    ...
    case constN:
        <secvență instr.N>;
        break;
    default: <secvență instr. implicită>;
}
```

Curs Programarea Calculatoarelor, Prof. Bogdan IONESCU 14/59

### Instrucțiunile condiționale (continuare)

- Structura switch (continuare)

Exemplu simplu:

```
scanf("%d",&x);
switch (x)
{
    case 1:
        printf("x are valoarea 1");
        break;
    case 2:
        printf("x are valoarea 2");
        break;
    default:
        printf("nu stiu valoare x");
}
```

dacă x are valoarea 1 se afișează ???

dacă x are valoarea 2 se afișează ???

dacă x are valoarea -3 se afișează ???

Curs Programarea Calculatoarelor, Prof. Bogdan IONESCU 17/59

### Instrucțiunile condiționale (continuare)

- Structura switch (continuare)

*Exemplu:*

```
scanf("%c",&litera);
switch (litera)
{
case 'a':
case 'A':
printf("s-a introdus litera a");
break;
case ' ':
nr_spatii++;
break;
}
```

ce se întâmplă dacă omitem corpul unui case ? se execută până la prima instrucțiune break.

**Observații:**

- break este opțional, poate constitui un avantaj,
- default este opțional, dar recomandat.

Curs Programarea Calculatoarelor, Prof. Bogdan IONESCU 18/59

### Structurile repetitive

> Sunt motivate de necesitatea de *execuție repetitivă* (în buclă) a anumitor instrucțiuni sau părți din program. Această execuție este reglementată de utilizator.

> Constituie baza unui limbaj de programare. Aproape toate problemele de calcul sunt rezolvate folosind astfel de structuri.

> Limbajul de programare **C** propune următoarele structuri repetitive (cicluri):

- ciclul cu test final **do while**,
- ciclul cu test inițial **while**,
- ciclul cu contor **for**.

Curs Programarea Calculatoarelor, Prof. Bogdan IONESCU 21/59

### Instrucțiunile condiționale (continuare)

- Structura switch (continuare)

**P** *Enunț:* se citește de la tastatură o valoare întreagă, folosind structura switch să se realizeze un program care spune dacă numărul este divizibil cu 3 sau nu.

```
scanf("%d",&a);
switch (a%3==0)
{
case 1:
printf("este divizibil cu 3");
break;
case 0:
printf("nu este divizibil cu 3");
break;
}
```

a%3==0  
→ valoare 1 (adevărat) dacă se împarte exact,  
→ valoare 0 (fals) dacă nu se împarte exact

Curs Programarea Calculatoarelor, Prof. Bogdan IONESCU 19/59

### Structurile repetitive (continuare)

- Structura do while

> execută cât timp:

*Sintaxă:*

```
do
{
<secvență instrucțiuni>;
} while (<condiție>;);
```

corpul structurii este specificat între "{" și "}"

condiția de execuție este specificată între "(" și ")", și se încheie cu ";"

> Permite **executarea repetitivă** a mai multor instrucțiuni **cât timp** o anumită condiție este îndeplinită (adevărată).

Curs Programarea Calculatoarelor, Prof. Bogdan IONESCU 22/59

## 5.2. Structurile repetitive

Curs Programarea Calculatoarelor, Prof. Bogdan IONESCU 20/59

### Structurile repetitive (continuare)

- Structura do while (continuare)

*Mod de execuție:*

```
do
{
secvență
instrucțiuni
}
while
{
evaluare
condiție ()
}
```

adevărată

falsă

*Exemplu:*

```
do
{
scanf("%d",&numar);
} while (numar<10);
```

*Execuție:*

3 (enter) (3<10 da)  
5 (enter) (5<10 da)  
11 (enter) (11<10 nu) **stop**

Curs Programarea Calculatoarelor, Prof. Bogdan IONESCU 23/59

### Structurile repetitive (continuare)

- Structura do while (continuare)

**Exemplu:**

```
int i=0;
do
{
    i++;
} while (i<10);
```

de câte ori se execută această buclă **10 ori**

**Exemplu:**

```
int x;
do
{
    scanf("%d",&x);
    if (x=0) printf("Atentie numarul este 0");
} while (x<10);
```

de câte ori se execută această buclă **de multe multe ori**

**Atenție:** nu modificați involuntar condiția de oprire !

Curs Programarea Calculatoarelor, Prof. Bogdan IONESCU 24/59

### Structurile repetitive (continuare)

- Structura do while (continuare)

**P** **Enunț:** folosind structura do while să se afișeze pe ecran toate numerele întregi cuprinse între două numere întregi pozitive **b** și **a** introduse de la tastatură ( $b > a$ ).

**Structură program:**

**Variabile de intrare:**

- citire a și b,
- definire contor i,

**Variabile de ieșire:**

- afișare repetitivă a contorului pentru valori între b și a,

```
unsigned int a,b;
int i;
```

Curs Programarea Calculatoarelor, Prof. Bogdan IONESCU 27/59

### Structurile repetitive (continuare)

- Structura do while (continuare)

**P** **Enunț:** folosind structura do while să se realizeze un program care permite ghicirea unui anumit număr ascuns între 1 și 999. Dacă utilizatorul introduce valoarea 0 stop.

**Constante program:**

```
int numar_ascuns=123;
```

**Variabile de intrare:**

```
int numar_introdus;
```

**Variabile de ieșire: nu**

am omis: să definim un indicator, 1 găsit, 0 abandon

**Structură program:**

- informare utilizator condiții
- citire repetitivă
- verificare număr introdus
- da, modificare indicator
- nu, furnizare indicații (<, >)
- condiție de încheiere
- verificare cauză încheiere

Curs Programarea Calculatoarelor, Prof. Bogdan IONESCU 25/59

- Structura do while (continuare)

**P** **Enunț:** folosind structura do while să se afișeze pe ecran toate numerele întregi cuprinse între două numere întregi pozitive **b** și **a** introduse de la tastatură ( $b > a$ ).

```
int a, b, i;
printf("a="); scanf("%d",&a);
printf("b="); scanf("%d",&b);
i=b;
do
{
    printf("%d ",i--);
} while (i>=a);
```

Curs Programarea Calculatoarelor, Prof. Bogdan IONESCU 28/59

- Structura do while (continuare)

```
int numar_ascuns=123, numar_introdus, flag=0;
printf("Ghicire numar intre 1 si 999\n");
do
{
    printf("Introduceti un intreg"); scanf("%d", &numar_introdus);
    if (numar_introdus==numar_ascuns)
    {
        printf("Numar ghicit, %d este numarul cautat", numar_introdus);
        flag=1;
    }
    else if (numar_introdus>numar_ascuns)
        printf("Prea mare\n");
    else printf("Prea mic\n");
} while (!flag && numar_introdus!=0);
if (!flag)
    printf("Ati abandonat cautarea");
```

Curs Programarea Calculatoarelor, Prof. Bogdan IONESCU 26/59

### Structurile repetitive (continuare)

- Structura while

> cât timp:

**Sintaxă:**

```
while (<condiție>
{
    <secvență instrucțiuni>
}
```

condiția de execuție este specificată între "(" și ")",

corpul structurii este specificat între "{" și "}"

> **Cât timp** o anumită condiție este îndeplinită (adevărată) se execută o serie de instrucțiuni.

Curs Programarea Calculatoarelor, Prof. Bogdan IONESCU 29/59

### Structurile repetitive (continuare)

- Structura while (continuare)

Mod de execuție:

```
while (<condiție>)
{
<secvență instrucțiuni>;
}
```

Exemplu:

```
char c=' ';
while (c!='x')
{
scanf("%c",&c);
}
```

Execuție:

a (enter) ('a'!='x' da)  
y (enter) ('y'!='x' da)  
x (enter) ('x'!='x' nu) **stop**

Curs Programarea Calculatoarelor, Prof. Bogdan IONESCU 30/59

### Structurile repetitive (continuare)

- Structura while (continuare)

Exemplu:

```
int x=30;
while (x-=5);
printf("%d",x);
```

Ce valori sunt afișate pe ecran? **0**

Greșeală frecventă: ";" ca la do while, rezultat: bucla se termină la ;

Observație: "{" și "}" nu sunt obligatorii pentru o singură instrucțiune.

Exemplu corect:

```
int x=30;
while (x-=5)
printf("%d",x);
```

Ce valori sunt afișate pe ecran? **25 20 15 10 5**

Curs Programarea Calculatoarelor, Prof. Bogdan IONESCU 33/59

### Structurile repetitive (continuare)

- Structura while (continuare)

Exemplu:

```
int x=9;
while (x--)
{
printf("Un mesaj insistent");
}
```

- se evaluează condiție while (x),  
- se decrementează valoare x  
de câte ori se afișează acest mesaj **9 ori**

Exemplu:

```
int x=9;
while (--x)
{
printf("Un mesaj mai puțin insistent?");
}
```

dar acesta **8 ori**

Curs Programarea Calculatoarelor, Prof. Bogdan IONESCU 31/59

### Structurile repetitive (continuare)

- Structura while (continuare)

**P** Enunț: să se realizeze un program care permite citirea a două numere întregi a și b. Folosind structura while, calculați media aritmetică a tuturor numerelor întregi dintre a și 0 ce sunt divizibile cu b (verificați că a>10).

Structură program:

- citesc repetitiv pe a până când a>10,
- parcurs numerele de la a la 0 și le însumez pe cele modulo b
- număr câte sunt și la sfârșit calculez media.

Variabile de intrare/lucru:

```
int a,b;
int numar_valori;
int contor;
```

Variabile de ieșire:

```
float medie_aritmetica;
```

Curs Programarea Calculatoarelor, Prof. Bogdan IONESCU 34/59

### Structurile repetitive (continuare)

- Structura while (continuare)

Exemplu:

```
char ch;
int s=0;
while ((ch=getche())!='x')
{
s=s+ch;
}
```

Ce face funcția getch() ?  
funcția getch() citește un caracter de la tastatură (e – echo, afișare caracter pe ecran)

Cum se execută ?  
- mai întâi ch=getche(),  
- se evaluează ch!='x',  
- dacă da, se execută instrucțiuni.

Curs Programarea Calculatoarelor, Prof. Bogdan IONESCU 32/59

- Structura while (continuare)

```
int a=0, b, contor, numar_valori=0;
float medie_aritmetica=0.0;
printf("b="); scanf("%d", &b);
while (a<=10)
{
printf("a="); scanf("%d", &a);
}
contor=a; // încep parcurgerea numerelor de la a la 0
while (contor>0)
{
if ((contor%b)==0)
{
medie_aritmetica+=contor; // adaug la media aritmetică valorile numar_valori++; // număr câte valori am adăugat
}
contor--; // mă îndrept spre 0 cu pas 1
}
printf("Media este:%f", (float)medie_aritmetica/numar_valori);
```

Curs Programarea Calculatoarelor, Prof. Bogdan IONESCU 35/59

### Structurile repetitive (continuare)

- Structura for

> pentru:

**Sintaxă:**

```
for (<init.contor> ; <cond.exec.> ; <modif.contor>)
{
    <secvență instrucțiuni>;
}
```

> Permite **executarea** iterativă a unei secvențe de instrucțiuni. Numărul de execuții este contabilizat de un contor. Acesta poate fi folosit în calcule pentru a identifica iterația curentă.

Curs Programarea Calculatoarelor, Prof. Bogdan IONESCU 36/59

### Structurile repetitive (continuare)

- Structura for (continuare)

> După modul de execuție, structura **for** este similară unei structuri **while** (cât timp execută) deoarece condiția de oprire este verificată înainte de execuție.

**Exemplu simplu:**

```
int x;
for (x=2 ; x>0 ; x--)
```

```
{
    printf("x are valoarea:%d\n",x);
}
```

**Execuție:**

```
x=2
2>0 (da)
>x are valoarea 2
x=2-1
2-1>0 (da)
>x are valoarea 1
x=2-1-1
2-1-1>0 (nu) stop
```

Curs Programarea Calculatoarelor, Prof. Bogdan IONESCU 39/59

### Structurile repetitive (continuare)

- Structura for (continuare)

```
for (<init.contor> ; <cond.exec.> ; <modif.contor>)
{
    <secvență instrucțiuni>;
}
```

**Semnificație parametri:**

<init.contor> : conține o expresie în care se inițializează contorul ce va monitoriza numărul de execuții al buclei for.

<cond.contor> : conține o condiție prin care se specifică indirect de câte ori se va executa corpul instrucțiunii for și anume: **cât timp** <cond.exec.> este adevărată.

<modif.contor> : specifică modul în care va fi schimbată valoarea contorului la fiecare iterație.

Curs Programarea Calculatoarelor, Prof. Bogdan IONESCU 37/59

### Structurile repetitive (continuare)

- Structura for (continuare)

> Specificarea expresiilor care definesc bucla **for** este *foarte flexibilă* după cum se poate vedea din exemplele următoare:

**Exemplu:**

```
for (x=0 ; ((x>3)&&(x<9)) ; x++)
```

```
printf("x are valoarea:%d\n",x);
```

condiția de oprire poate fi compusă.  
care este efectul acestui for **doar x=0**

**Exemplu:**

```
for (x=0,y=4 ; ((x==0)&&(x<3)) ; x++, y+=2)
```

```
printf("x=%d, y=%d\n",x,y);
```

la specificarea expresiilor se poate folosi operatorul ","

**Execuție:**

```
>x=0, y=4
>x=1, y=6
>x=2, y=8
stop
```

Curs Programarea Calculatoarelor, Prof. Bogdan IONESCU 40/59

### Structurile repetitive (continuare)

- Structura for (continuare)

> mod de execuție:

```
for (<init.contor> ; <cond.exec.> ; <modif.contor>)
{
    <secvență instrucțiuni>;
}
```

Curs Programarea Calculatoarelor, Prof. Bogdan IONESCU 38/59

### Structurile repetitive (continuare)

- Structura for (continuare)

**Exemplu:**

```
for (x=0,y=4,z=4000 ; z ; z/=10)
```

```
printf("%d,%d,%d\n",x, y, z);
```

care este efectul acestui for ?

**Execuție:**

```
>x=0, y=4, z=4000
>x=0, y=4, z=400
>x=0, y=4, z=40
>x=0, y=4, z=4
stop
```

**Exemplu:**

```
for (x=0 ; x<3 ; x++)
```

```
{
    for (y=0 ; y<4 ; y++)
        printf("a ");
    printf("\n");
}
```

**Execuție:**

```
>a a a a
>a a a a
>a a a a
```

Curs Programarea Calculatoarelor, Prof. Bogdan IONESCU 41/59

**Structurile repetitive (continuare)**

- Structura for (continuare)

> Câteva observații:

- structura **for** este folosită de regulă când se poate estima numărul de iterații,
- de asemenea, structura **for** se folosește de regulă atunci când în calcul este necesară cunoașterea interației curente (valoarea contorului),

> Cazuri particulare:

<code>for (; i&lt;10; i++)</code>	s-a omis inițializarea contorului, ce se întâmplă în acest caz ? <i>i pornește de la val. din program</i>
<code>for (; ;)</code>	sau toate expresiile, ce se întâmplă în acest caz ? <i>se execută la infinit secvența de instr.</i>

Curs Programarea Calculatoarelor, Prof. Bogdan IONESCU 42/59

**Structurile repetitive (continuare)**

- Structura for (continuare)

**P** *Enunț:* să se calculeze folosind structura for varianța numerelor întregi cuprinse între numerele **a** și **b** ( $b > a$ ).

$$\sigma^2 = \frac{1}{b-a+1} \sum_{i=a}^b (i - \mu_i)^2$$

*medie valori i*

<b>Variabile de intrare/lucru:</b>	<b>Structură program:</b>
<code>int a,b;</code>	- citesc <b>a</b> și <b>b</b> ,
<code>int i;</code>	- parcurg numerele de la <b>a</b> la <b>b</b> și calculez media,
<code>float medie;</code>	- parcurg numerele de la <b>a</b> la <b>b</b> și calculez varianta,
<b>Variabile de ieșire:</b>	
<code>float varianta;</code>	

Curs Programarea Calculatoarelor, Prof. Bogdan IONESCU 45/59

**Structurile repetitive (continuare)**

- Structura for (continuare)

**P** *Enunț:* se introduc de la tastatură trei numere întregi: **a**, **b**, **m** ( $b > a$ ). Să se realizeze un program care calculează produsul numerelor întregi divizibile cu **m** din intervalul [a;b] folosind structura for.

<b>Variabile de intrare/lucru:</b>	<b>Structură program:</b>
<code>int a,b,m;</code>	- citesc <b>a</b> , <b>b</b> și <b>m</b> ,
<code>int i;</code>	- parcurg numerele de la <b>a</b> la <b>b</b> și le înmulțesc pe cele modulo <b>m</b> ,
<b>Variabile de ieșire:</b>	
<code>int produs;</code>	

Curs Programarea Calculatoarelor, Prof. Bogdan IONESCU 43/59

**Structurile repetitive (continuare)**

- Structura for (continuare)

```
#include<math.h>
int a, b, i, dim;
float medie=0.0, varianta=0.0;
// citire date
printf("Introduceti a si b: "); scanf("%d %d", &a, &b);
dim=b-a+1;
// calcul medie
for (i=a; i<=b; i++)
    medie+=i;
medie=medie/dim;
// calcul varianta
for (i=a; i<=b; i++)
    varianta+=pow(i-medie,2); //ridicare la puterea 2
varianta=varianta/dim;
printf("Varianta este: %f",varianta);
```

Curs Programarea Calculatoarelor, Prof. Bogdan IONESCU 46/59

**Structurile repetitive (continuare)**

- Structura for (continuare)

```
int a=1, b=0, m, i;
int produs=1;
while (a>b)
{
    printf("Introduceti a si b: ");
    scanf("%d %d", &a, &b);
}
printf("Introduceti divizorul: ");
scanf("%d", &m);
for (i=a; i<=b; i++) // parcurgerea numerelor de la a la b
    if ((i%m)==0)
        produs=produs*i;
printf("Produsul este:%d", produs);
```

Curs Programarea Calculatoarelor, Prof. Bogdan IONESCU 44/59

### 5.3. Instrucțiunile break, continue și exit

Curs Programarea Calculatoarelor, Prof. Bogdan IONESCU 47/59



### Instrucțiunile break, continue și exit

> Introducerea structurilor repetitive, datorită execuției automate în buclă, a instrucțiunilor, a ridicat problema controlării în totalitate de către programator a execuției acestora.

→ o condiție de oprire scrisă eronat conduce la *repetarea infinită* a instrucțiunilor din buclă.

> Pentru a controla execuția, limbajul C pune la dispoziția programatorului trei comenzi speciale, și anume:

- instrucțiunea **break**
- instrucțiunea **continue**
- funcția **exit**

### Instrucțiunea continue

• **continue**: determină programul să abandoneze executarea instrucțiunilor rămase din buclă pentru iterația curentă, ca și cum s-ar fi ajuns la evaluarea condiției de repetare,

→ efectul constă în trecerea forțată la începutul iterației următoare.

**Sintaxă:**

**continue;** - se apelează ca o procedură, nu are parametri.

**Exemplu:**

```
for ( i=0; i<10; i++ )
{
    if (i==5) continue;
    printf("%d",i);
}
```

ce valori sunt afișate pe ecran  
0 1 2 3 4 6 7 8 9

### Instrucțiunea break

• **break**: înseamnă întrerupere, determină întreruperea unui ciclu (do while, while, for) sau a selecției (switch), chiar dacă condiția de terminare nu este îndeplinită.

**Sintaxă:**

**break;** - se apelează ca o procedură, nu are parametri.

**Exemplu:**

```
for ( ;; )
{
    scanf("%d", &x);
    if (x>10) break;
}
```

acest for se execută la infinit.

ciclu se va încheia atunci când valoarea introdusă este >10 datorită lui **break**.

### Instrucțiunea continue (continuare)

**Exemplu:**

```
int a, b, i;
printf("Introduceti doua numere");
scanf("%d %d",&a, &b);

i=a;
do
{
    if (i%2!=0)
        continue;
    printf("%d este divizibil cu 2\n",i);
} while (i++<b);
printf("i=%d",i);
```

ce face acest program ???

cum se execută ???

**Execuție:**

```
>Introduceti ...
>3 5
>i=3
> continue (3<5, i=3+1)
> 4 este divizibil cu 2
> (4<5 da, i=4+1)
> continue (5<5 nu, i++)
stop while
> i=6
```

### Instrucțiunea break (continuare)

**Exemplu:**

```
for (i=0; i<4; i++)
for (j=0; j<10; j++)
{
    if (j==3)
        break;
    printf("De cate ori apare mesajul");
}
```

de câte ori apare mesajul pe ecran dacă nu ar fi existat condiția if **4 x 10 = 40 de ori**

dar cu condiția if ???  
**4 x 3 = 12 de ori**

> **Observație:** instrucțiunea break nu permite ieșirea din toate structurile repetitive din program, ci doar din ultima structură repetitivă ce o conține.

### Instrucțiunea exit

• **exit**: este o funcție din biblioteca stdlib.h ce determină **încheierea execuției** programului curent (funcție, procedură) și returnarea unui cod de eroare.

**Prototipul funcției:**

```
void exit(int codEroare);
```

**Exemplu:**

```
int main()
{
    int i;
    while (1)
    { printf("Introduceti o valoare care sa-mi placa"); scanf("%d",&i);
      if (i==13)
          exit(1); } // raspuns gresit
}
```

programul citește de la tastatură valori până la introducerea valorii 13, când execuția se încheie cu codul 1

## 5.4. Probleme recapitulative

### Problema 2

**P** *Enunț:* se cunosc valorile funcției  $x^2$  pentru  $x$  în intervalul  $[-a;a]$ , unde  $a$  este o valoare naturală introdusă de la tastatură. Să se parcurgă aceste valori cu un pas de analiză  $p$  (introdus de la tastatură). Pentru fiecare valoare parcursă să se calculeze media valorilor din fereastra  $w$  (introdus de la tastatură).

valorile lui  $x$  sunt:

[ -a   -a+p   -a+2\*p   ...   -a+i\*p   ...   a-p ]

valorile lui  $x^2$  sunt:

[ (-a)<sup>2</sup>   (-a+p)<sup>2</sup>   (-a+2\*p)<sup>2</sup>   ...   (-a+i\*p)<sup>2</sup>   ...   (a-p)<sup>2</sup> ]

↓  
fereastra  $w = w$  valori inclusiv valoarea curentă

### Problema 1

**P** *Enunț:* scrieți un program care citește un număr natural  $n$  și care afișează valoarea expresiei:

$$s = 0! + 1! + 2! + \dots + i! + \dots + n!$$

*indicație:* se folosește structura for.

*Variabile de intrare/lucru:*      *Structură program:*

<b>int</b> n;	- citește $n$ și verifică că $n > 0$
<b>int</b> i, j;	- parcurg numerele de la 0 la $n$ ,
<b>int</b> tmp;	- calculez factorialul și adaug
	valoarea la suma $s$ ,

*Variabile de ieșire:*

**int** s;

### Problema 2 - rezolvare

```
int a, w, nr;
float p, i, contor, valoare, medie;
printf("a="); scanf("%d", &a);
printf("pas="); scanf("%f", &p);
printf("fereastra w="); scanf("%d", &w);
for (contor=-a; contor<=a; contor+=p)
{ printf("%.2f ", contor*contor); //afisare valoare x*x
  medie=0.0; nr=0; //initializare medie si numar valori
  for (i=contor; i<=a; i+=p) //parcure fereastra de w valori
  { if (i<=(contor+w*p)) //daca sunt in fereastra calculez medie
    { medie+=i*i;
      nr++; // numar valorile
    }
    else break;
  }
  medie=medie/nr;
  printf("media este: %.2fn", medie); }
```

### Problema 1 - rezolvare

```
int i, j, n; double tmp, s=1;
// citire date
do
{ printf("Introduceti n: "); scanf("%d", &n);
} while (n<0);
// calcul suma factoriali
for (i=1; i<=n; i++)
{ tmp=1;
  for (j=1; j<=i; j++)
    tmp=tmp*j;
  s=s+tmp;
}
printf("Suma s este: %e", s);
```

## Sfârșitul Cursului 5