UNIVERSITATEA TEHNICĂ A MOLDOVEI

Programarea calculatoarelor

Instrucțiuni în limbajul C

Profesoară: Maria Guțu

Instrucțiuni decizionale în C

- **Instrucțiunea IF;**
- **Instrucțiunea IF-ELSE;**
- **Instrucțiunea IF imbricată;**
- **Instrucțiunea Switch;**
- Operatorul condiționat.

Instrucțiunea decizională IF

Sintaxa:

if(expression)

program statement

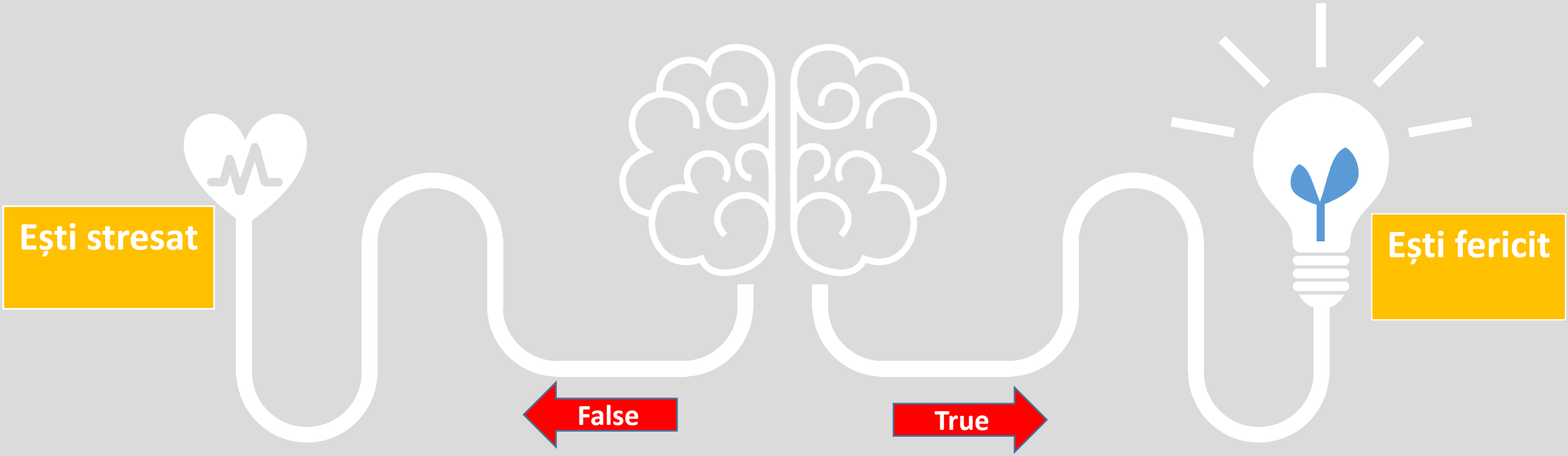
Semantica:

If (plouă)

iau umbrela

Instrucțiunea IF

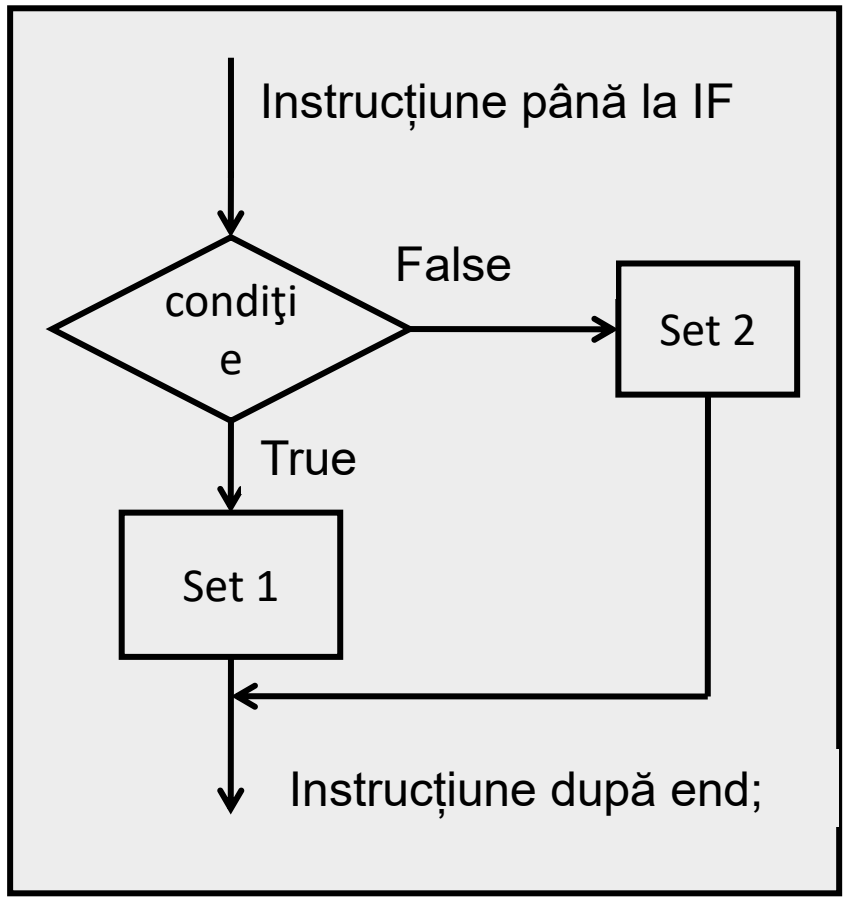
Dacă ai învățat și ești pregătit de lecție



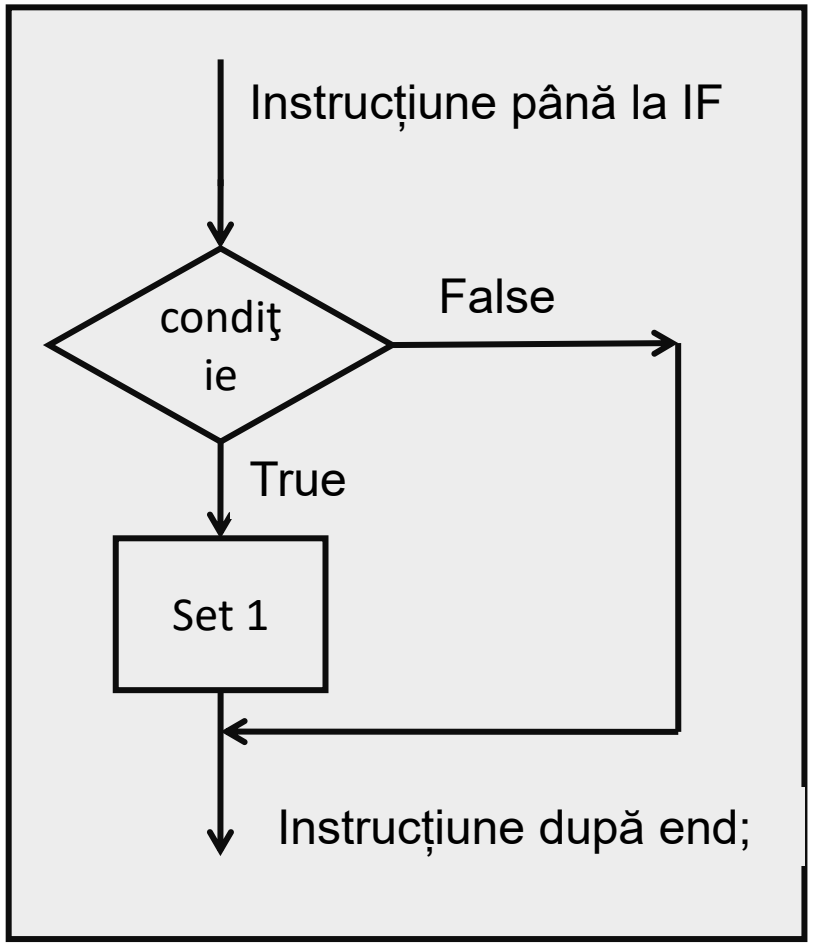
Condiții

- Less than: `a < b`
- Less than or equal to: `a <= b`
- Greater than: `a > b`
- Greater than or equal to: `a >= b`
- Equal to `a == b`
- Not Equal to: `a != b`

Instrucțiunea IF

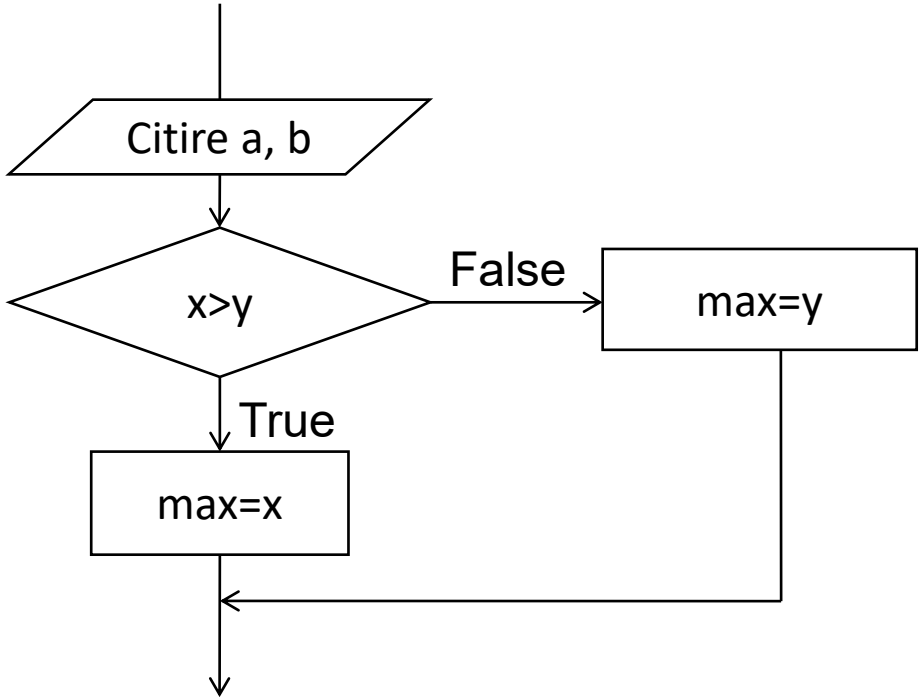


Dacă rezultatul evaluării *Condiției* este *true*, atunci se execută Set1 iar, dacă *Condiția* are valoare *false*, atunci se execută Set2, dacă există, în caz contrar se trece la următoarea secvență de cod din program.



Exemplu 1

Se consideră două numere întregi distincte. Să se scrie un program care determină numărul mai mare.

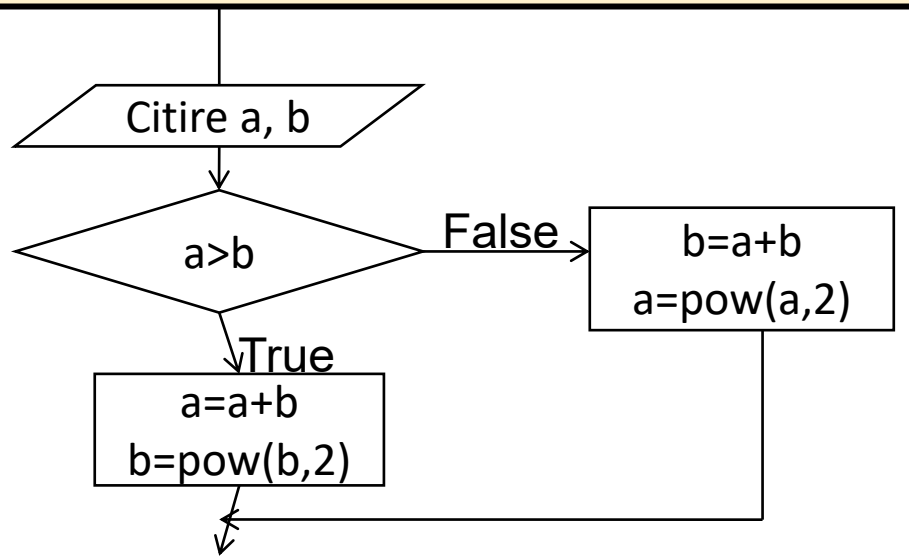



```
#include <stdio.h>
int main()
{
    int x, y, max;
    scanf("%d%d",&x,&y);
    if (x>y) max = x;
        else max = y;
    printf("El.max: %d",max);
    return 0;
}
```

Exemplu 2

```
#include <stdio.h>
#include <math.h>
int main(){
    int a, b;
    scanf("%d%d",&a,&b);
    if (a>b) { a = a+b;
               b = pow(b,2);
        } else { b = a+b;
                 a = pow(a,2);
    }
    printf("a=%d; b=%d",a, b);
    return 0;
}
```

Se consideră două numere întregi distincte. Să se scrie un program care înlocuiește numărul mai mare cu suma numerelor date, iar numărul mai mic – cu pătratul său.



Exemplu 2

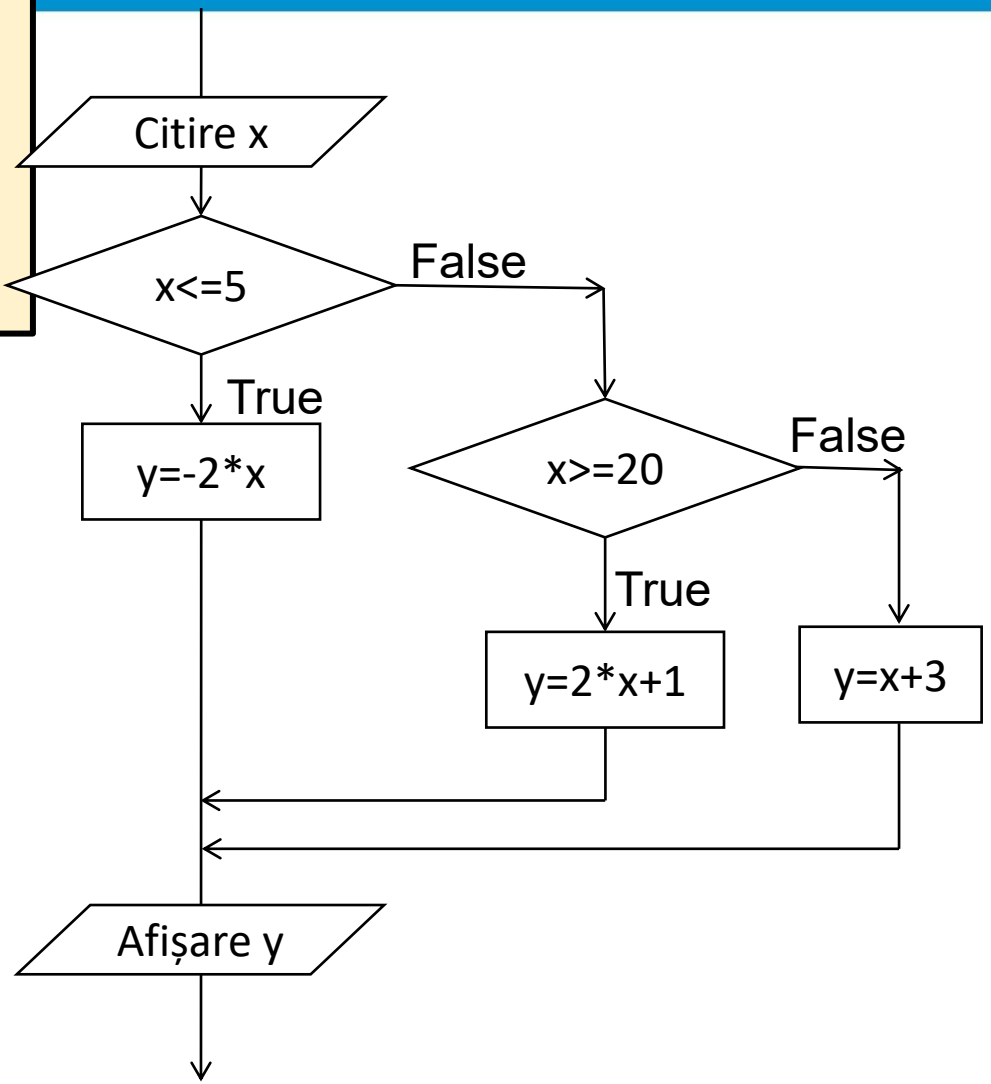

```
#include <stdio.h>
int main(){
    int time = 22;
      if (time < 10) {
            printf("Good morning.");
            } else if (time < 20) {
                        printf("Good day.");
                        } else {
                                printf("Good evening.");
                                }
return 0;
}
```

Exemplu 3

Să se calculeze valoarea funcției *y* definită pe mulțimea numerelor reale, pentru un număr *x* citit de la tastieră.

$$y = \begin{cases} -2x, & x \leq 5 \\ x + 3, & 5 < x < 20 \\ 2x + 1, & x \geq 20 \end{cases}$$

```
#include <stdio.h>
int main(){
    int x, y;
    scanf("%d",&x);
    if (x<=5) y = -2*x;
        else if (x>=20) y = 2*x+1;
            else y = x+3;
    printf("y=%d",y);
return 0;}
```



Exemplu 4

Determinarea dacă un an este bisect sau nu

```
#include <stdio.h>
int main()
{
int an;
scanf("%d", &an);
if(an % 400 == 0 || (an % 4 == 0 && an % 100 != 0))
    printf(" este un an bisect\n");
  else
    printf(" nu este un an bisect\n");
    return 0;
}
```

Instrucțiunea decizională IF

Instrucțiunea **IF** este utilizată pentru a stabili execuția unei instrucțiuni de program (sau declarații dacă sunt incluse în paranteze) pe baza unor condiții specificate.

Instrucțiunea **if / else if /else** se folosește pentru a implementa structuri decizionale cu mai mult de două alternative și se numesc **instrucțiuni imbricate**.

Instrucțiunea decizională Switch

Forma generală:

```
switch (opțiune) {
        case c1: instructiuni 1; break;
        case c2: instructiuni 2; break;
        .....
        case cn: instructiuni n; break;
        default: instructiuni; break;
}
```

Expresia **opțiune** aflată între paranteze este comparată succesiv cu valorile **c1, c2..., cn**, care *trebuie să fie ori constante simple ori expresii constante*.

Dacă se găsește un caz a cărui valoare este egală cu valoarea din o**pțiune**, urmează instrucțiunile de program care execută cazul.

Nu uitați să includeți declarația de pauză break la sfârșitul fiecărui caz. Dacă uitați să faceți acest lucru pentru un caz particular execuția programului va continua cu următorul caz, ori de câte ori acest caz va fi executat.

Instrucțiunea decizională Switch

Forma generală:

```
switch (opțiune) {
        case c1: instructiuni 1; break;
        case c2: instructiuni 2; break;
        ……
        case cn: instructiuni n; break;
        default: instructiuni; break;
}
```

Cazul opțional special numit **default** este executat *dacă valoarea expresiei nu se potrivește cu niciuna din valorile cazului*.

De îndată ce este întâlnită instrucțiunea **break;** într-un ciclu, iterațiile ciclului se opresc în punctul unde se întâlnește ***break;*** și controlul revine din ciclu imediat la prima declarație de după ciclu.

Exemplu 1

```
// Program to create a simple calculator
#include <stdio.h>

int main() {
   char operation;
   double n1, n2;

   printf("Enter an operator (+, -, *, /): ");
   scanf("%c", &operation);
   printf("Enter two operands: ");
   scanf("%lf %lf",&n1, &n2);
```

```
switch(operation) {
      case '+':
         printf("%.1lf + %.1lf = %.1lf",n1, n2, n1+n2);  break;
      case '-':
         printf("%.1lf - %.1lf = %.1lf",n1, n2, n1-n2); break;
      case '*':
         printf("%.1lf * %.1lf = %.1lf",n1, n2, n1*n2); break;
      case '/':
         printf("%.1lf / %.1lf = %.1lf",n1, n2, n1/n2); break;
      // operator doesn't match any case constant +, -, *, /
      default:
         printf("Error! operator is not correct");
   }
   return 0;
}
```

Exemplu 2

```
#include <stdio.h>
int main() {
   int num;
   printf("Input a one-digit number\n");     scanf("%d",&num);
                    switch (num) {
                    case 0: case 2: case 4: case 6: case 8:
                              printf("%d - an even number\n", num); break;
                    case 1: case 3: case 5: case 7: case 9:
                              printf("%d - an odd number\n", num); break;
                    default:
                              printf("%d not a one-digit number \n", num); break;
                    }
return 0; }
```

Exemplu 3

```
#include <stdio.h>
int main(){
    char ch;
    printf("Input a character\n");
    scanf("%c",&ch);
```

```
switch (ch) {
        case 'A' ... 'Z':
                printf("%c in range A to Z\n", ch);
                printf("%d in range %d to %d\n", ch, 'A', 'Z');
                break;
        case 'a' ... 'z':
                printf("%c in range 'a' to 'z'\n", ch);
                printf("%d in range %d to %d\n", ch, 'a', 'z');
                break;
        default:
                printf("%d not in range\n", ch);
                break;
        }
return 0; }
```

Exemplu 4

Un exemplu de utilizare a instrucțiunii SWITCH atunci când avem un interval prestabilit de valori.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main() {
    srand(time(0));
    int num = rand() % 100 + 1;
```

Reține:
Corect - **case 1 ... 5:**
// cele trei puncte sunt precedate și succedate de câte un spațiu
Greșit - **case 1...5:**

```
switch (num) {
            case 1 ... 10:
                printf("%d in range 1 to 10\n", num);
                break;
            case 19 ... 60:
                printf("%d in range 19 to 60\n", num);
                break;
            default:
                printf("%d not in range\n", num);
                break;
            }
    return 0;
}
```

Instrucțiuni Repetitive:

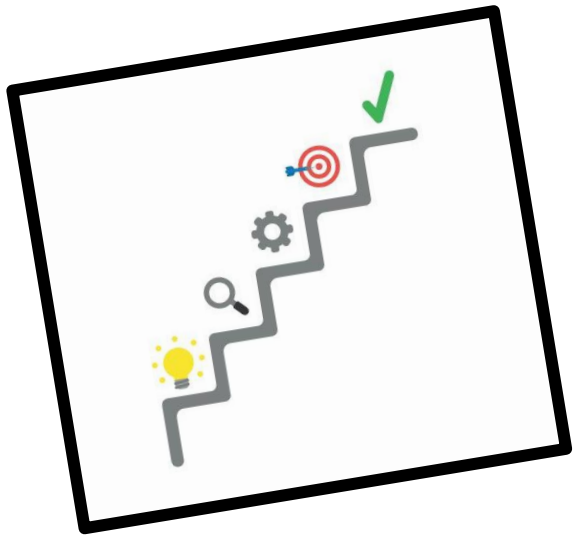
Instrucțiunea FOR

Instrucțiunea While

Instrucțiunea Do...While

Sintaxa și semanitica instrucțiunii FOR

For (**initializare**; **verificare**; **reinitializare**) { instructiune }



unde: **Initializare** – se atribuie o valoare de început;
Verificare – se verifică satisfacerea condiției;
Reinitializare – se incrementează sau decrementează valoarea variabilei.

Ce se va afișa în consolă?

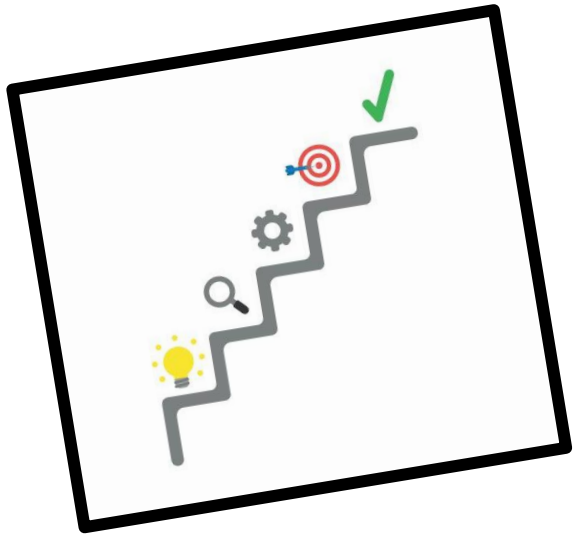
```
for (int i=1; i<=3; ++i)
    printf("%2d", i);
```

```
1 2 3
```

Cuvânt-cheie | Inițializare | Conditie | Reinițializare | Instrucțiune

Sintaxa și semanitica instrucțiunii FOR

For (**initializare**; **verificare**; **reinitializare**) { instructiune }



unde: **Initializare** – se atribuie o valoare de început;
Verificare – se verifică satisfacerea condiției;
Reinitializare – se incrementează sau decrementează valoarea variabilei.

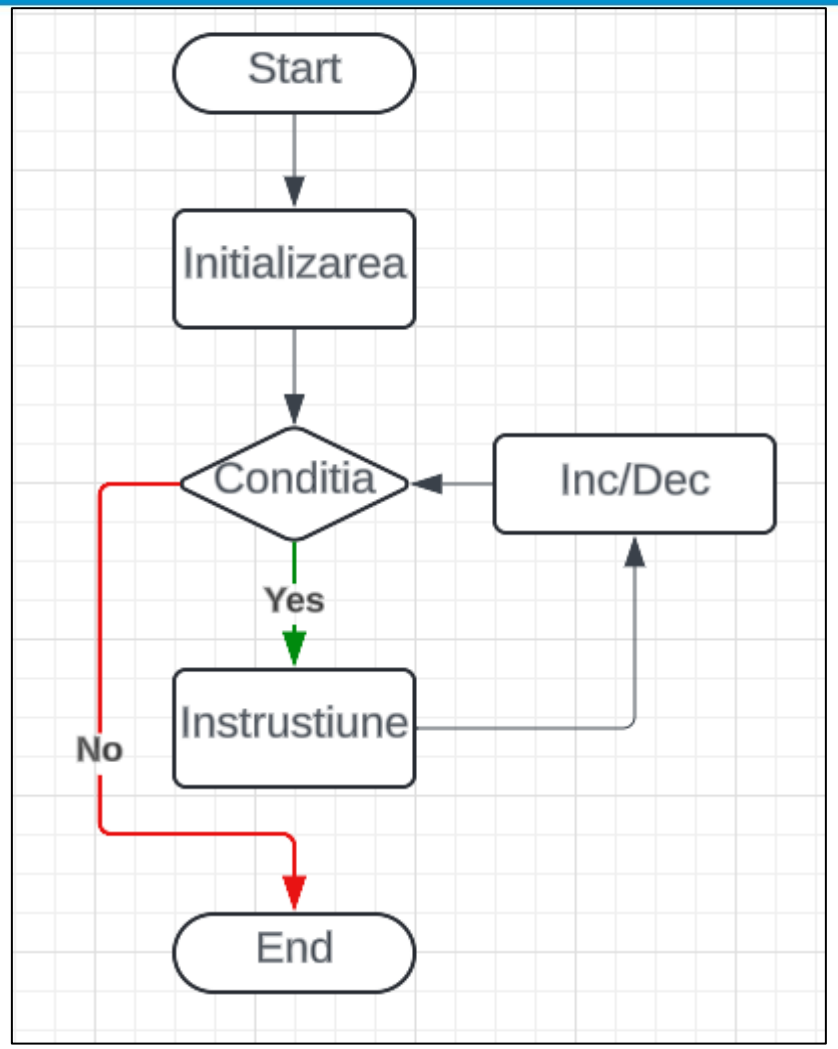
Ce se va afișa în consolă?

```
for (int i=5; i>=2; --i)
    printf("%2d", i);
```

```
5 4 3 2
```

Cuvânt-cheie | Inițializare | Conditie | Reinițializare | Instrucțiune

Exemplul 5



Schema logică FOR

Să se scrie un program care afișază pe ecran cuvântul „Informatica” de *n* ori, valoarea n fiind introdusă de la tastatură.

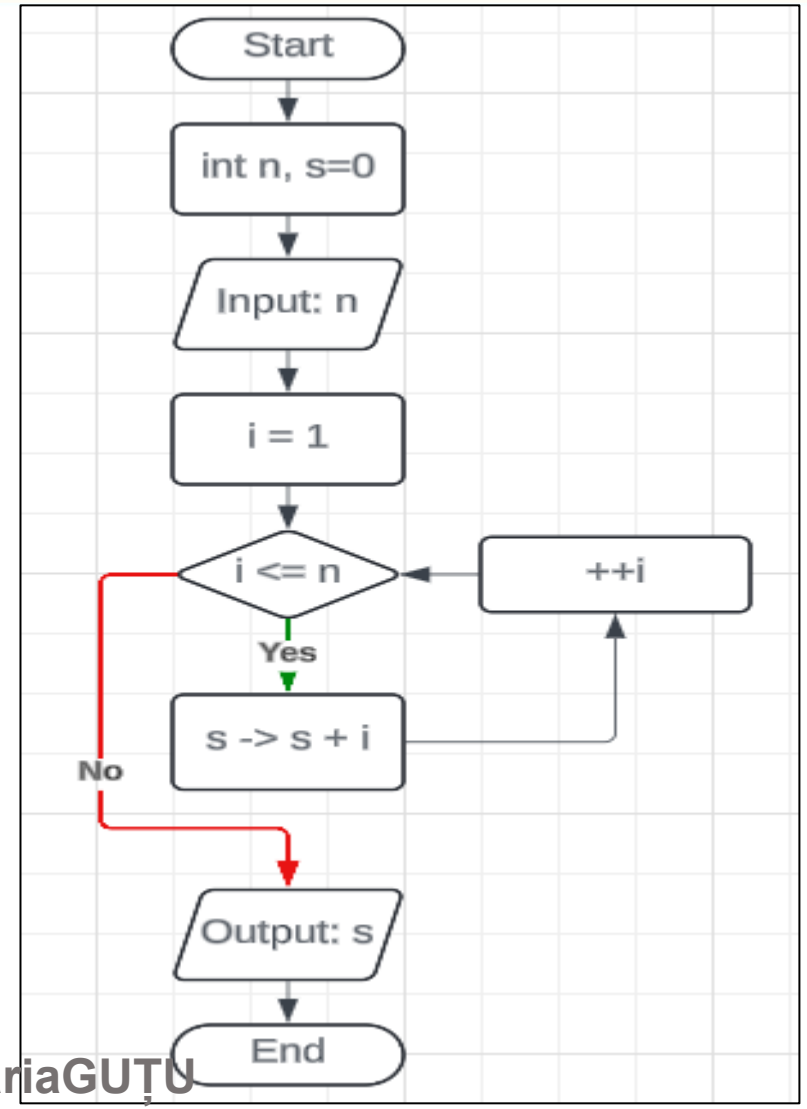
```
#include <stdio.h>
int main(){

int n;
scanf("%d",&n); // n=5
for (int i=1; i<n; ++i)
printf("Informatica\n");

return 0;
}
```

n	i	Consola
5	1	Informatica
	2	Informatica
	3	Informatica
	4	Informatica
	5	

Exemplul 6



Schema logică

Să se scrie un program care calculează suma numerelor naturale mai mici sau egale cu *n*.

```
#include <stdio.h>
int main(){

int n, s = 0;
scanf("%d",&n); // n=5

 for (int i = 1; i <= n; ++i)
            s += i;
    printf("s = %d", s);
    return 0;
}
```

Consolă
Sum=15

n	i	Sum
5		0
	1	1
	2	3
	3	6
	4	10
	5	15

Exemple

1.De la tastieră se introduce o secvență de *n* numere întregi. Să se scrie un program care calculează **media aritmetică** a numerelor introduse.

```
#include <stdio.h>
int main()
{
   int k, n, sum = 0;
   scanf("%d", &n);
   for (int i = 0; i<n; ++i){
       scanf("%d", &k);
       sum = sum + k;
   }
   float avg = (float) sum / n;
   printf("Avg: %f", avg);
return 0;
}
```

2. De la tastieră se introduce un număr natural *N*. Să se scrie un program care afișează divizorii numărului introdus.

```
#include <stdio.h>
int main()
{
   unsigned int n;
   scanf("%d", &n);
   for (int i = 1; i<=n; ++i)
       if (n % i == 0)
           printf("%d ", i);
return 0;
}
```

3. De la tastieră se introduce un număr natural *N*. Să se scrie un program care ar determina dacă acest **număr** este **prim**.

```
#include <stdio.h>
int main() {
  unsigned int n, prime = 0;
      scanf("%d", &n);
      for (int i = 2; i<=n/2; ++i)
          if (n % i == 0) {
              prime++;
              break;
          }
      if (prime == 0)
          printf("%d este numar prim", n);
          else
        printf("%d nu este numar prim", n);
return 0;
}
```

4. Să se scrie un program care afișează în consolă următorul triunghi:

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

```
#include <stdio.h>
int main()
{
   for (int i = 1; i<=4; ++i) {
     for (int j = 1; j<=i; ++j)
     printf("%d ",j);
     printf("\n");
   }
return 0;
}
```

Instrucțiuni Repetitive:
Instrucțiunea While
VS
Instrucțiunea Do...While

Sintaxa generală

Instrucțiunea While

conține o expresie booleană, numită și condiție, care controlează execuția repetată a unei instrucțiuni. Această instrucțiune se execută repetat atât timp cât condiția este adevărată.

Instrucțiunea WHILE

Sintaxa generală

While <Expresie booleană>
<Instrucțiune>

Instrucțiunea Do …While

indică repetarea unei secvențe de instrucțiuni în funcție de valoarea unei expresii booleene. Secvența de instrucțiuni se va executa repetat atât timp cât condiția este adevărată.

Instrucțiunea Do … While

Sintaxa generală

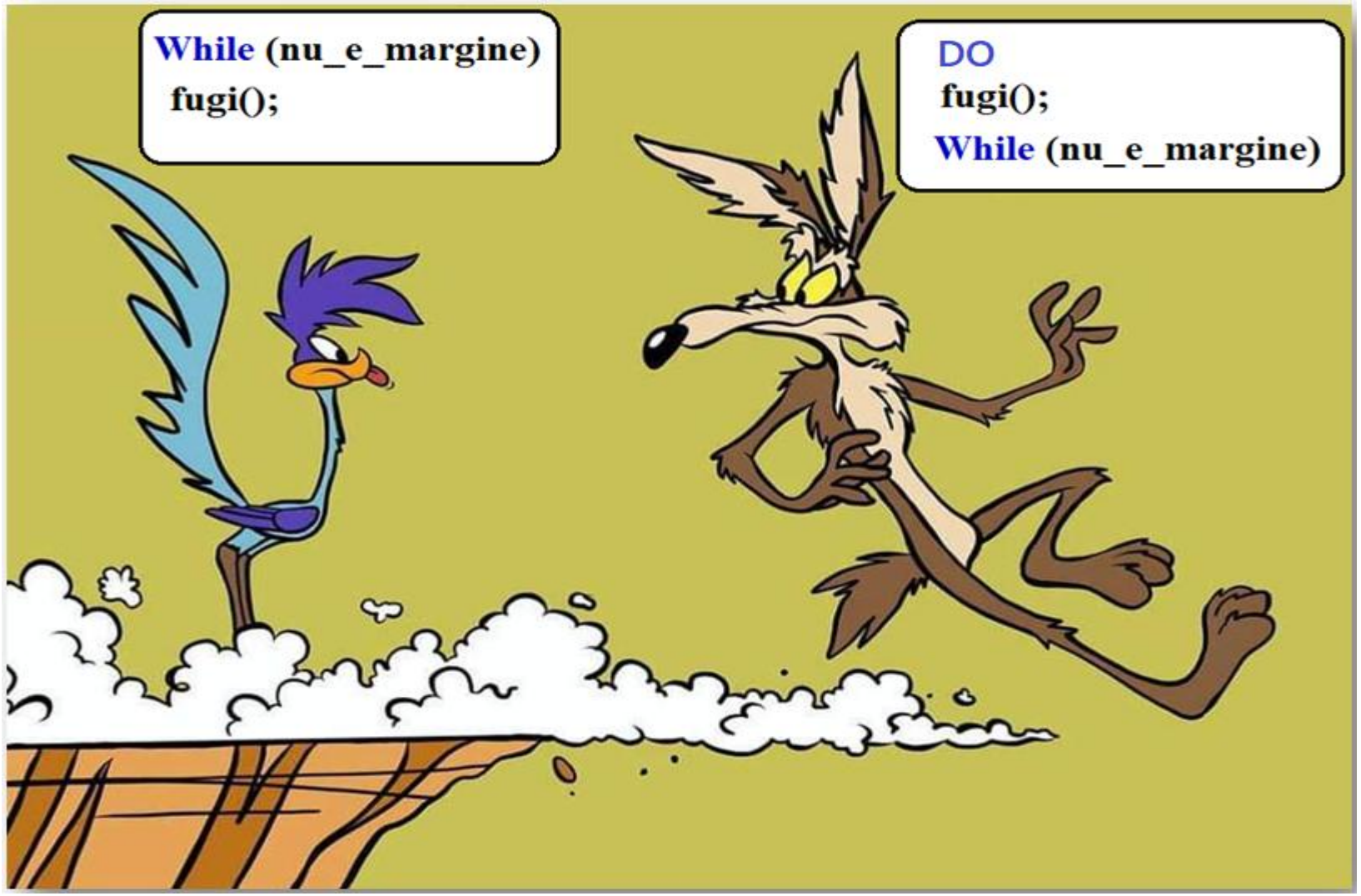
Do
{<Instrucțiune>}
While <Expresie booleană>

FACULTATEA CALCULATOARE, INFORMATICĂ ȘI MICROELECTRONICĂ

FCIM

Reprezentare vizuală

Instrucțiunile repetitive **While** & **Do…while** sunt utile atunci când **numărul de repetiții** este **nedefinit**.



@MariaGUȚU

Modul de execuție

1. Se evaluează condiția logică;
2. Dacă valoarea condiției este *adevărat* (true), se excută instrucțiunea aflată între { }, apoi se revine la pasul 1;
3. Dacă valoarea condiției este *fals* (false), se conținuă cu următoarea instrucțiune aflată după instrucțiunea while.

Sintaxa generală

While <Expresie booleană>
<Instrucțiune>

```
scanf("%d", &n);
While (n!=0) { scanf("%d", &n); }
```

Dacă valoarea condiției este *fals* (false) de la început atunci instrucțiunea ce se află după { } nu se va executa.

While se numește instrucțiune cu **test inițial** sau condiționată **anterior.**

Modul de execuție

1. Se execută secvența de instrucțiuni;
2. Se evaluează condiția logică;
3. Dacă valoarea condiției este *true* , se revine la pasul 1;
4. Dacă valoarea condiției este *False* , se continuă cu următoarea instrucțiune aflată după instrucțiunea DO, adică după punct și virgulă.

Sintaxa generală

```
Do
{<Instrucțiune>}
while <Expresie booleană>
```

```
Do
{ scanf("%d", &n);
} while (n=0);
```

Indiferent de valoarea condiției, secvența de instrucțiuni dintre Do și While se va executa cel puțin o dată.

Do…while se numește instrucțiune cu **test final** sau condiționată **posterior.**

Exemplul #1

n	rest	sum	count
32541		0	0
3254	1	1	1
325	4	5	2
32	5	10	3
3	2	12	4
0	3	15	5

Consolă

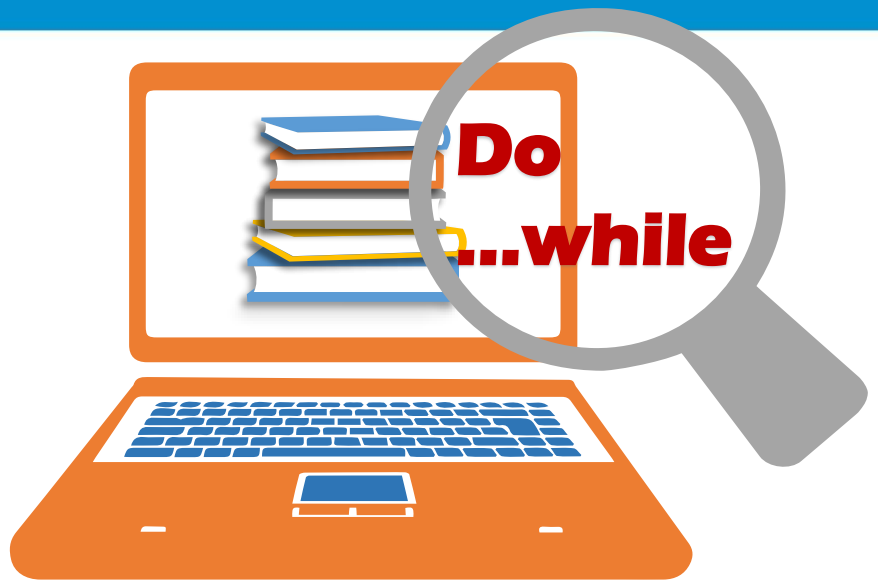
count=5; sum=15

câț ← N / 10
rest ← N % 10

#2 Se consideră un număr natural *N*. Să se scrie un program care numără câte cifre conține acest număr și calculează suma lor.
Input: Valoarea numărului *N*.
Output: Numărul de cifre din componența numărului dat (k) și suma lor (sum).

```
#include <stdio.h>
int main()
{
   int n, count = 0, sum = 0;
   scanf("%d", &n);
   while (n != 0) {
      int rest = n % 10;
      sum += rest;
      count++;
      n /= 10;
   }
   printf("%d cifre; sum = %d", count, sum);
return 0;
}
```

Exemplul #1



Să se scrie un program care să determine suma cifrelor unui număr natural citit de la tastatură.

```
#include <stdio.h>
int main() {
    int nr, sum = 0, rest;
    scanf("%d", &nr);
    do{
        rest = nr % 10;
        sum += rest;
        nr /= 10;
    } while (nr != 0);
    printf("Suma cifrelor este %d", sum);
    return 0; }
```

Cerințe:
Modificați codul astfel încât să se numere câte cifre are numărul citit de la tastatură.
Modificați astfel încât numărul să fie generat random.
Calculați media aritmetică a cifrelor numărului generat random.

FACULTATEA CALCULATOARE, INFORMATICĂ ȘI MICROELECTRONICĂ

Exemplul #2

FCIM

#2 Se citesc numere de la tastatură până la apariția lui zero. Să se determine câte dintre numerele citite sunt pare.

Input: Valoarea numărului *N*.

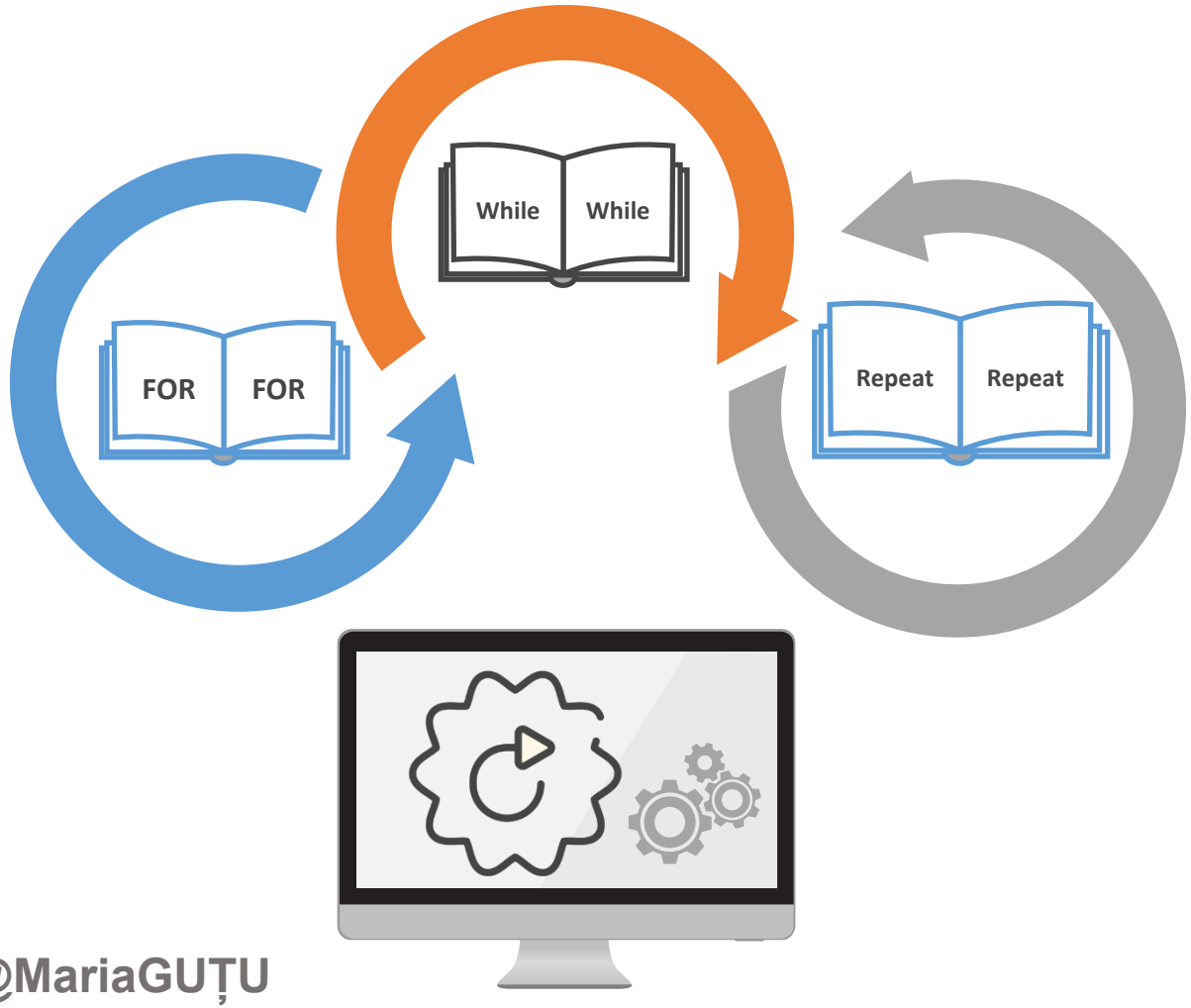
Output: Numărul de cifre din componența numărului dat (k) și suma lor (sum).

```
#include <stdio.h>
 int main(void){
   int nr, count = 0;
 scanf("%d", &nr);
   while (nr != 0) {
      if (nr % 2 == 0) {
            count++;
         }
         scanf("%d", &nr);
   }
   printf("Sunt %d numere pare", count);
   return 0; }
```

Cerințe: Modificați codul astfel încât să se numere doar numerele impare.

- Calculați suma tuturor numerelor citite de la tastatură.
- Calculați media aritmetică doar a numerelor pare citite de la tastatură.

Sinteză



FOR este o instrucțiune repetitivă cu un număr cunoscut de pași, adică numărul de repetiții este definit.

While este o instrucțiune repetitivă cu cu test inițial. Numărul de repetări este nedefinit.

Do … While este o instrucțiune repetitivă cu cu test final. Numărul de repetări este nedefinit.

Aplicații Practice!!!

(maria.gutu@iis.utm.md)