

# **Bazele programării I**

## **Abstracția în programare.**

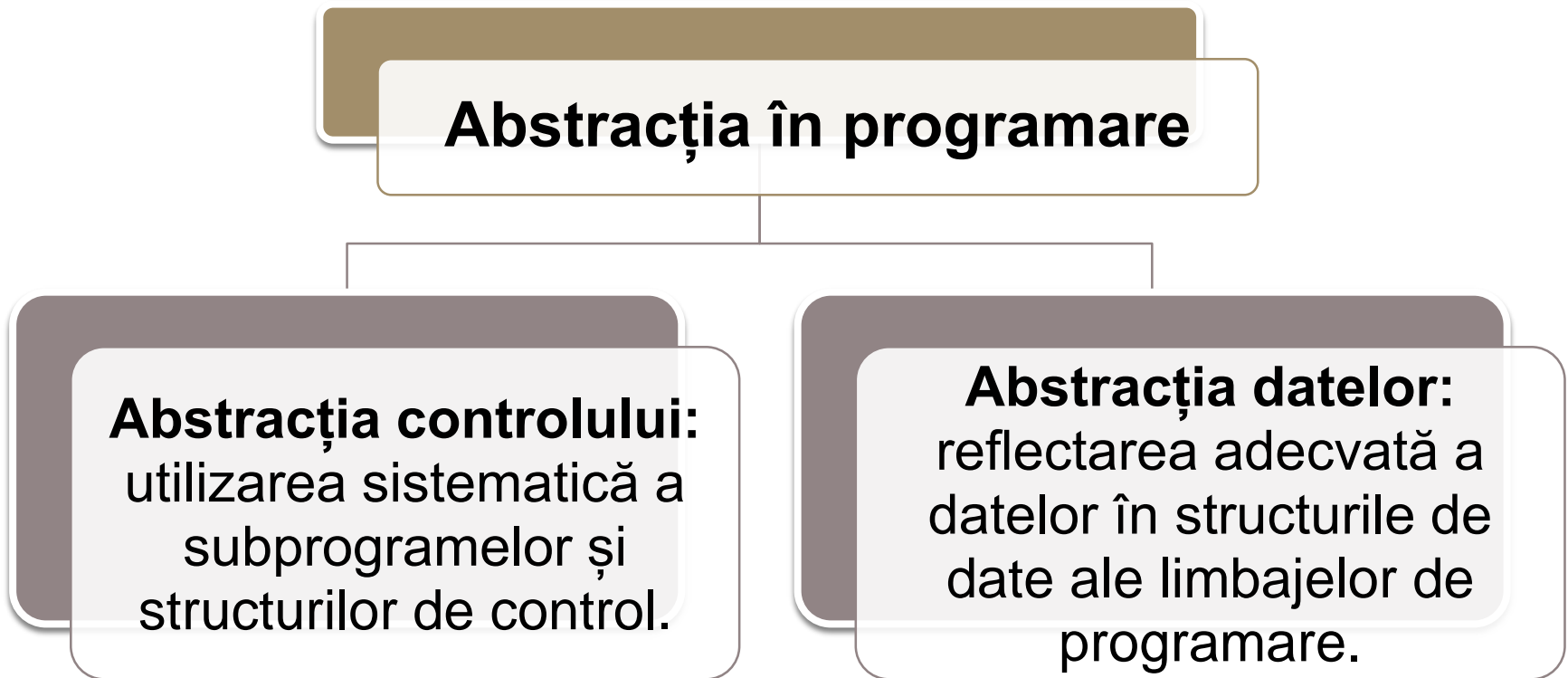
### **Proceduri. Funcții**

**lector univ. D.V. Moglan**  
**mogdiana@gmail.com**

# Abstracția în programare

---

**Nivel de abstracție** este o modalitate de „ascundere” a detaliilor de realizare a unei mulțimi de posibilități funcționale.



Abstracția în programare se realizează cu ajutorul subprogramelor.

# Subprograme

---

În rezolvarea problemelor apar următoarele situații care necesită o rezolvare:

- o secvență dintr-un algoritm se repetă;
- există mai multe algoritme care au nevoie de un anumit calcul.

Un **subprogram** reprezintă un ansamblu alcătuit din tipuri de date, variabile și instrucțiuni scrise în vederea unei anumite prelucrări (calcul, citiri, scrieri), care se identifică printr-un nume și care poate fi executat doar dacă este apelat.

# Exemplu

Să se calculeze aria totală a două loturi dreptunghiulare.

```
Algoritm Aria_totala_1
Var
  S, A,B, S_total: real
Begin
  //calcularea ariei primului lot
  ReadReal(a)
  ReadReal(b)
  S:=a*b
  S_total:=S
  //calcularea ariei lotului 2
  ReadReal(a)
  ReadReal(b)
  S:=a*b
  S_total:=S_total+s
End
```

poate fi structurat  
ca subalgoritm:

```
Algoritm Aria_1_lot
Begin
  ReadReal(a)
  ReadReal(b)
  S:=a*b
End
```

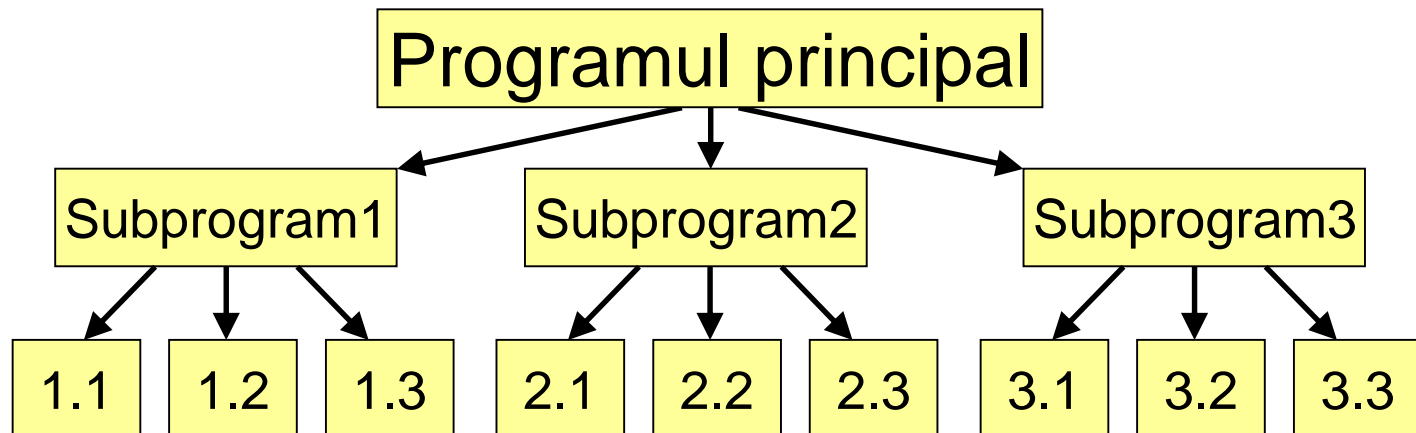
**Fragment  
comun**



# Avantajele folosirii subprogramelor

---

- reducerea dimensiunilor textului sursă al programelor;
- scurtarea timpului de elaborare a programelor, prin utilizarea aceluiași subprogram în programe diferite;
- economisirea spațiului de memorie, la execuția programului;
- sistematizarea elaborării programelor, prin divizarea unei probleme mai complicate în probleme mai simple, tratate prin subprograme separate.



# Subprograme

---

În limbajele de programare subprogramele se realizează cu ajutorul **procedurilor** și **funcțiilor**.

Utilizarea subprogramelor presupune determinarea a două elemente esențiale:

**1**

**Precizarea grupului de instrucțiuni comune;**

**2**

**Specificarea locului sau locurilor din program în care grupul de instrucțiuni trebuie executat.**

# Definirea și apelul subprogramelor

---

**Definirea subprogramului** este inclusă în partea declarativă a algoritmului, prin identificarea conținutului instrucțiunilor comune.

**Apelul subprogramului** reprezintă instrucțiunile din grupul comun, care urmează să fie executat.

# Definiția și apelul unei proceduri simple

## Formatul procedurii simple:

**Procedure** <nume>

Antetul

**Var**

<variabile locale>

Partea declarativă

**Begin**

<corpul procedurii>

Partea executabilă

**End**

**Antetul** precizează numele procedurii, care este utilizat în program, în fiecare din apelurile acesteia.

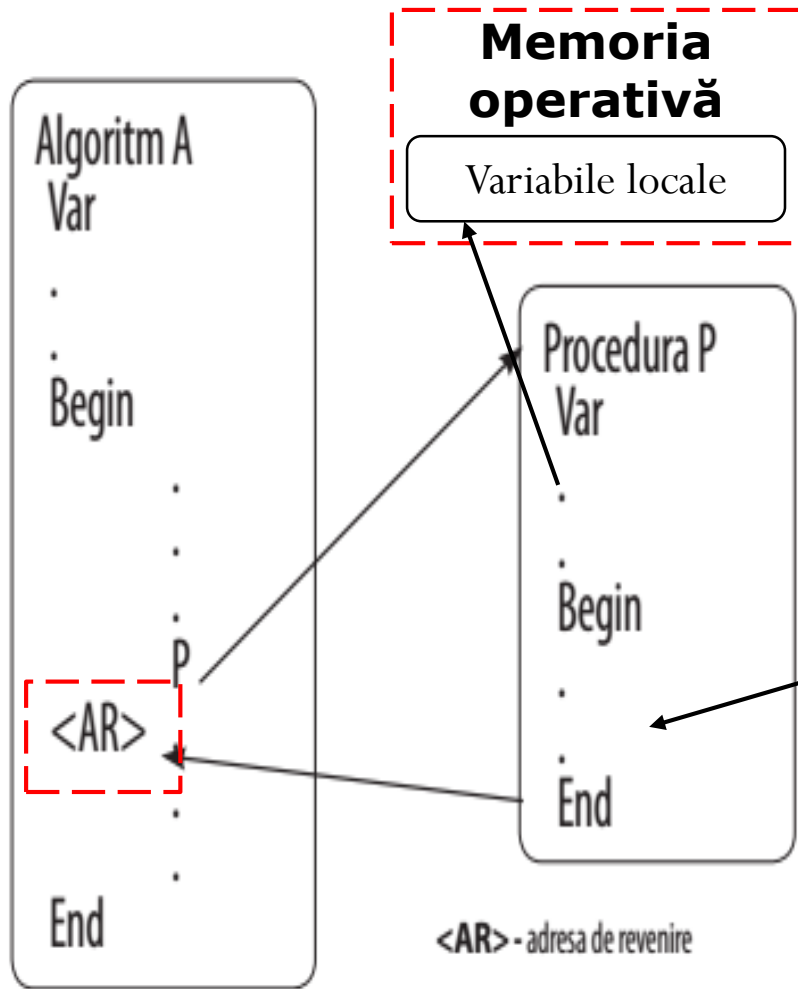
**Corpul procedurii** are o parte declarativă și una executabilă.

**Partea declarativă** – declararea datelor (variabilelor).

**Partea executabilă** – instrucțiunile care se vor executa la apelul procedurii.



# Execuția apelului de procedură



## Execuția unei proceduri simple:

- 1. Se memorizează adresa de revenire (adresa instrucțiunii care urmează după instrucțiunea de apel);*
- 2. Se alocă memorie pentru variabilele locale ale procedurii;*
- 3. Se execută corpul procedurii;*
- 4. Se eliberează memoria, ocupată de variabilele locale;*
- 5. Se reia execuția procedurii apelante, de la instrucțiunea imediat următoare apelului.*

# Variabile locale și variabile globale

---

Variabilele declarate în procedură se numesc **variabile locale**.

Ele pot fi folosite numai în această procedură și nu pot fi accesate în exteriorul ei.

Variabilele declarate în algoritmul principal se numesc **variabile globale**.

Domeniul de vizibilitate al variabilelor globale este întregul algoritm, inclusiv procedurile algoritmului.

# Observații

---

1. O variabilă locală există **numai în timpul execuției** procedurii în care este declarată.
2. Variabilele locale sunt **nedefinite** până la începutul execuției procedurii, nu sunt inițializate la apelul procedurii și nici nu-și păstrează valoarea de la un apel la altul.
3. În corpul procedurii pot fi accesate variabilele locale ale procedurii date și variabilele globale.

# Variabile locale și variabile globale

Algoritm A

Var

A: Integer

B: Real

C: Char

variabile  
globale

Procedure P1

Var

X: Natural

Y: Boolean

variabile  
locale

Begin

...

End

Procedure P2

Var

A: Boolean

Z: Integer

variabile  
locale

Begin

...

End

Begin

... ①

P1 ②

... ③

P2 ④

... ⑤

End

# Procedură cu parametri

## Formatul procedurii cu parametri:

**Procedure** <nume> (lista parametrilor formali)

**Var**

<variabile locale>

**Begin**

<corpul procedurii>

**End**

Antetul

Partea declarativă

Partea executabilă



Procedura calculează oricâte valori.

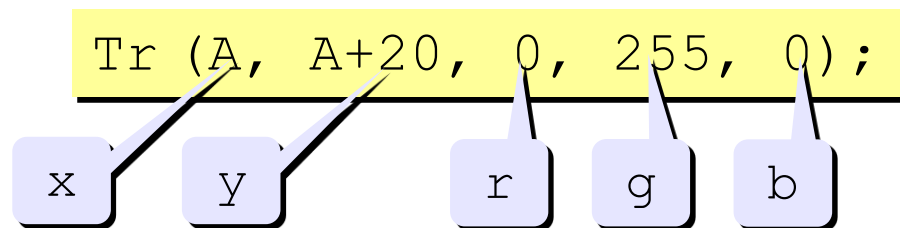
**Lista parametrilor formali** va conține descrierile tuturor intrărilor și a tuturor ieșirilor procedurii.

## Apelul procedurii cu parametri:

<nume\_procedură> (lista parametrilor actuali)

# Proceduri. Particularități

- toate procedurile se află mai sus de programul principal;
- numărul parametrilor actuali trebuie să coincidă cu numărul parametrilor formali;
- în antetul procedurii se enumeră parametrii **formali**, care se indică prin nume, deoarece se pot modifica; `procedure Tr( x, y, r, g, b: integer)`
- parametrii formali sunt cunoscuți în procedură și nu pot fi utilizați în afara procedurii;
- la apelul procedurii în paranteze se indică parametrii **actuali** (constantă, variabilă, expresie) **în aceeași ordine**.

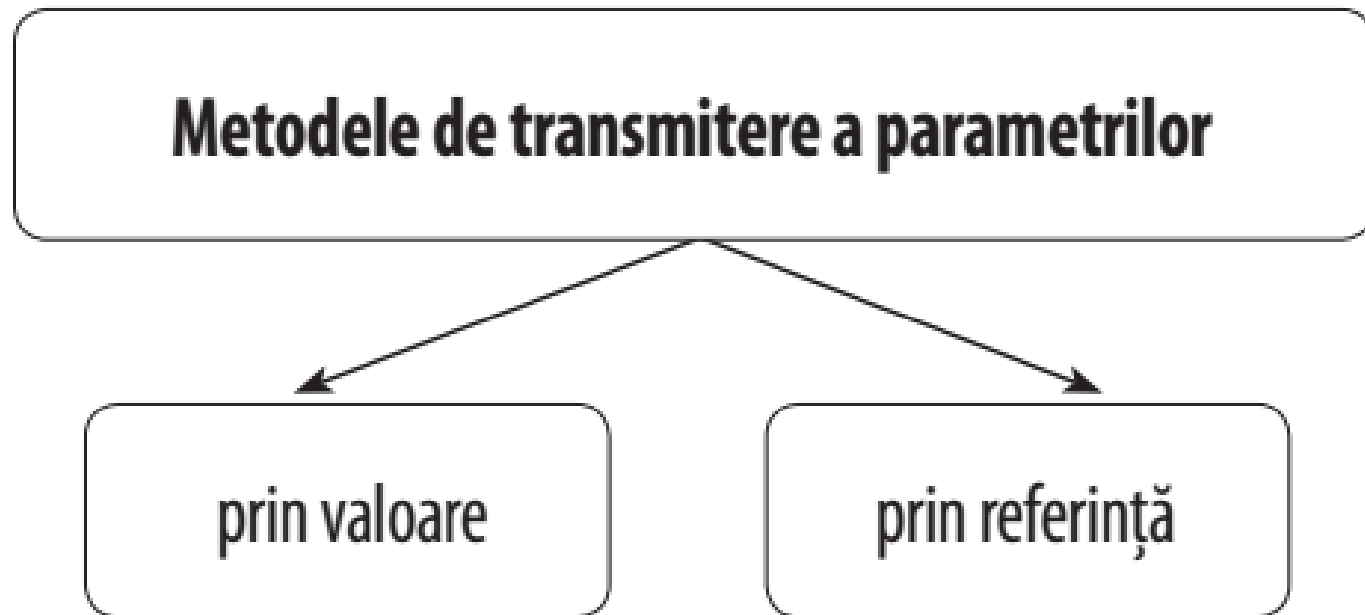


# Metodele de transmitere a parametrilor

---

Caracteristicile parametrilor formali:

- **numele parametrului** - se recomandă să fie sugestiv, să sugereze natura datelor;
- **tipul de date** - se alege în funcție de natura datelor;
- **metoda de transmitere** a parametrului.

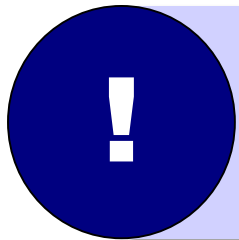


# Transmiterea parametrilor prin valoare

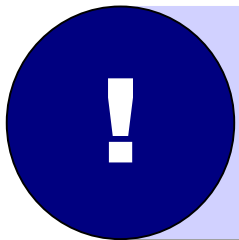
---

## Descrierea parametrilor formali:

<nume\_parametru>: <tip\_date>



Parametrii actuali pot fi: **constante**, **variabile** sau **expresii**, tipul de date al cărora este compatibil cu tipul parametrului formal corespunzător.



Metoda de transmitere prin valoare poate fi utilizată numai în cazul **parametrilor de intrare** ai procedurii.



# Transmiterea parametrilor prin valoare

---

## **Execuția unei proceduri dacă parametrii sunt transmiși prin valoare:**

- *se memorizează adresa de revenire (adresa instrucțiunii care urmează după instrucțiunea de apel);*
- *se calculează valoarea fiecărui parametru actual;*
- *se alocă memorie pentru parametrii formali, transmiși prin valoare și pentru variabilele locale ale procedurii; valorile parametrilor actuali calculate sunt atribuite parametrilor formali, inițializându-le;*
- *se execută corpul procedurii;*
- *se eliberează memoria, ocupată de parametrii formali, transmiși prin valoare și de variabilele locale;*
- *se reia execuția procedurii apelante de la instrucțiunea imediat următoare apelului.*

# Proceduri cu parametri

**Exemplu.** Afișați pe ecran scrierea binară a numărului întreg (0..255) pe 8 biți.

de multe ori!

**Algoritm:**

$$178 \Rightarrow 10110010_2$$

**?** Cum afișăm prima cifră?

$n :=$ 

7	6	5	4	3	2	1	0	
1	0	1	1	0	0	1	0	$_2$

 ordinele

$n \text{ div } 128$

$n \text{ mod } 128$

**?** Cum afișăm a doua cifră?

$n1 \text{ div } 64$

# Proceduri cu parametri

**Exemplu.** Afişaţi pe ecran scrierea binară a numărului întreg (0..255) pe 8 biţi.

**Rezolvare:**

```
k := 128
while k > 0 do
  WriteInt(n div k)
  n := n mod k
  k := k div 2
end
```

178  $\Rightarrow$  10110010

n	k	Afişare
178	128	1

# Descrierea algoritmului

Algoritm **binCode**

```
procedure printBin (n: integer)
```

```
var k: integer
```

```
begin
```

variabile locale

```
  k := 128
```

```
  while k > 0 do
```

```
    WriteInt(n div k)
```

```
    n := n mod k
```

```
    k := k div 2
```

```
  end
```

```
end
```

```
Begin
```

```
  printBin (99)
```

```
End
```

**Parametrii formali** –  
datele, care modifică  
lucrul procedurii.

Valoarea  
parametrului actual

# Exemplu. Aria totală

---

*Să se calculeze aria totală a două loturi dreptunghiulare.*

## Algoritm Aria\_totala

**Var**

A1, B1, A2, B2, S\_total: Real

**Procedure** **Aria\_1\_lot2** (Latura1: Real, Latura2: Real)

**Var**

S: Real

**Begin**

S:=Latura1 \* Latura2

S\_total:= S\_total+ S

**End**

**Begin**

S\_total:=0

ReadReal(A1, B1)

**Aria\_1\_lot2(A1,B1)**

ReadReal(A2, B2)

**Aria\_1\_lot2(A2,B2)**

WriteReal(S\_total)

**End**

# Traseul executării

Instructiunea	Tast.	Ecran	A1	B1	A2	B2	S_ total	Lat_1	Lat_2	S
			X	X	X	X	X			
ReadReal(A1)	2		2							
ReadReal(B1)	3			3						
S_total:=0							0			
Aria_1_lot2(A1,B1)								2	3	X
S:=Latura1*Latura2										6
S_total:=S_total+S							6			
ReadReal(A2)	4				4					
ReadReal(B2)	5					5				
Aria_1_lot2(A2,B2)								4	5	X
S:=Latura1*Latura2										20
S_total:=S_total+S							26			
WritReal(S_total)		26								

# Exemplu. Divizorii

*Să se afișeze divizorii numerelor naturale din intervalul  $[A..B]$ .*

## Algoritm Afisare\_divizori

```
Var
    A, B, k: Natural
Procedure Divizor (N: Natural)
    Var I: Natural
    Begin
        For I:=1 to N step 1
            If N mod I =0 Then
                WriteNat(I)
            End
        End
    End
Begin
    ReadNat(A, B)
    For k:=A to B step 1
        Divizor(k)
        Writeln
    End
End
```

# Transmiterea parametrilor prin referință

## Descrierea parametrilor formali:

**Var** <nume\_parametru>: <tip\_date>



**Parametrii de ieșire** se transmit numai prin referință.



Parametrul actual corespunzător parametrului formal, transmis prin referință, poate fi numai **variabilă**.



Modificările parametrilor formali, transmiși prin referință ai unei proceduri persistă și după terminarea execuției procedurii. O schimbare a parametrului formal **se reflectă în parametrul actual** corespunzător.



Dacă parametrul de intrare reprezintă o **structură voluminoasă**, va fi mai rațională transmiterea lui **prin referință**.



# Transmiterea parametrilor prin referință

---

## **Execuția unei proceduri dacă parametrii sunt transmiși și prin referință:**

- *se memorizează adresa de revenire (adresa instrucțiunii care urmează după instrucțiunea de apel);*
- *se calculează valoarea fiecărui parametru actual transmis prin valoare;*
- *se alocă memorie pentru parametrii formali, transmiși prin valoare și pentru variabilele locale ale procedurii;*
- *valorile parametrilor actuali calculate sunt atribuite parametrilor formali, inițializând-le;*

# Transmiterea parametrilor prin referință

---

## Execuția unei proceduri dacă parametrii sunt transmiși și prin referință:

- *se instalează corespondența dintre parametrul formal transmis prin referință și parametrul actual corespunzător (adresarea la parametrul formal va accesa zona de memorie care aparține parametrului actual);*
- *se execută corpul procedurii;*
- *se eliberează memoria, ocupată de parametrii formali, transmiși prin valoare și de variabilele locale;*
- *se reia execuția procedurii apelante, de la instrucțiunea imediat următoare apelului.*

# Exemplu. Interschimbare a două variabile

Să se scrie o procedură care să interschimbe valorile a două variabile.

Algorithm **Exchange**

var **x, y**: **integer**

procedure **Swap**(**a, b**: **integer**)

var **c**: **integer**

begin

**c** := **a**   **a** := **b**   **b** := **c**

end

begin

**x** := 2

**y** := 3

**Swap**(**x, y**)

WriteInt(**x, y**)

end.

transmiterea  
prin valoare



Procedură funcționează cu copiile parametrilor transmiși prin valoare!

2 3



De ce nu funcționează?

# Transmiterea parametrilor prin referință

variabilele se pot  
schimba

```
procedure Swap ( var a, b: integer)
var c: integer
begin
    c:=a a:=b b:=c
end
```

transmiterea  
prin referință

## Apelul:

```
var a, b: integer
...
Swap (a, b) { corect }
Swap (2, 3) { incorect }
Swap (a, b+3) { incorect }
```

# Exemplu

Algoritm Afisare1

Var x, y: integer

Procedure P (a: integer, var b: integer)

Begin

a := a + 3

b := b + a

End

Begin

x := 0 y := 0

P(x, y) WriteInt(x, y)

P(y, x) WriteInt(x, y)

P(x, x) WriteInt(x, y)

P(y, y) WriteInt(x, y)

End

x	y	
	b	a
0	3	3

Ecran

# Exemplu

Algoritm Afisare1

Var x, y: integer

Procedure P (a: integer, var b: integer)

Begin

a := a + 3

b := b + a

End

Begin

x := 0 y := 0

P(x, y) WriteInt(x, y)

P(y, x) WriteInt(x, y)

P(x, x) WriteInt(x, y)

P(y, y) WriteInt(x, y)

End

x	y	
0	3	

Ecran

03

# Exemplu

Algoritm Afisare1

Var x, y: integer

Procedure P (a: integer, var b: integer)

Begin

a := a + 3

b := b + a

End

Begin

x := 0 y := 0

P(x, y) WriteInt(x, y)

P(y, x) WriteInt(x, y)

P(x, x) WriteInt(x, y)

P(y, y) WriteInt(x, y)

End

x	y	
b		a
6	3	6

Ecran

03

# Exemplu

Algoritm Afisare1

Var x, y: integer

Procedure P (a: integer, var b: integer)

Begin

a := a + 3

b := b + a

End

Begin

x := 0 y := 0

P(x, y) WriteInt(x, y)

P(y, x) WriteInt(x, y)

P(x, x) WriteInt(x, y)

P(y, y) WriteInt(x, y)

End

x	y	
6	3	

Ecran

0363



# Exemplu

Algoritm Afisare1

Var x, y: integer

Procedure P (a: integer, var b: integer)

Begin

a := a + 3

b := b + a

End

Begin

x := 0 y := 0

P(x, y) WriteInt(x, y)

P(y, x) WriteInt(x, y)

P(x, x) WriteInt(x, y)

P(y, y) WriteInt(x, y)

End

x	y	
b		a
15	3	9

Ecran

0363

# Exemplu

Algoritm Afisare1

Var x, y: integer

Procedure P (a: integer, var b: integer)

Begin

a := a + 3

b := b + a

End

Begin

x := 0 y := 0

P(x, y) WriteInt(x, y)

P(y, x) WriteInt(x, y)

P(x, x) WriteInt(x, y)

P(y, y) WriteInt(x, y)

End

x	y	
15	3	

Ecran

0363153

# Exemplu

Algoritm Afisare1

Var x, y: integer

Procedure P (a: integer, var b: integer)

Begin

a := a + 3

b := b + a

End

Begin

x := 0 y := 0

P(x, y) WriteInt(x, y)

P(y, x) WriteInt(x, y)

P(x, x) WriteInt(x, y)

P(y, y) WriteInt(x, y)

End

x	y	
	b	a
15	9	6

Ecran

0363153

# Exemplu

Algoritm Afisare1

Var x, y: integer

Procedure P (a: integer, var b: integer)

Begin

a := a + 3

b := b + a

End

Begin

x := 0 y := 0

P(x, y) WriteInt(x, y)

P(y, x) WriteInt(x, y)

P(x, x) WriteInt(x, y)

P(y, y) WriteInt(x, y)

End

x	y	
15	9	

Ecran

0363153159

# Exemplu

Algoritm Afisare2

Var a, b, c, d: integer

Procedure P (var b: integer, c: integer)

begin

a:=2\*a

b:=2\*b

c:=2\*c

d:=2\*d

WriteInt(a,b,c,d)

end

Begin

a:=1 b:=1 c:=1 d:=1

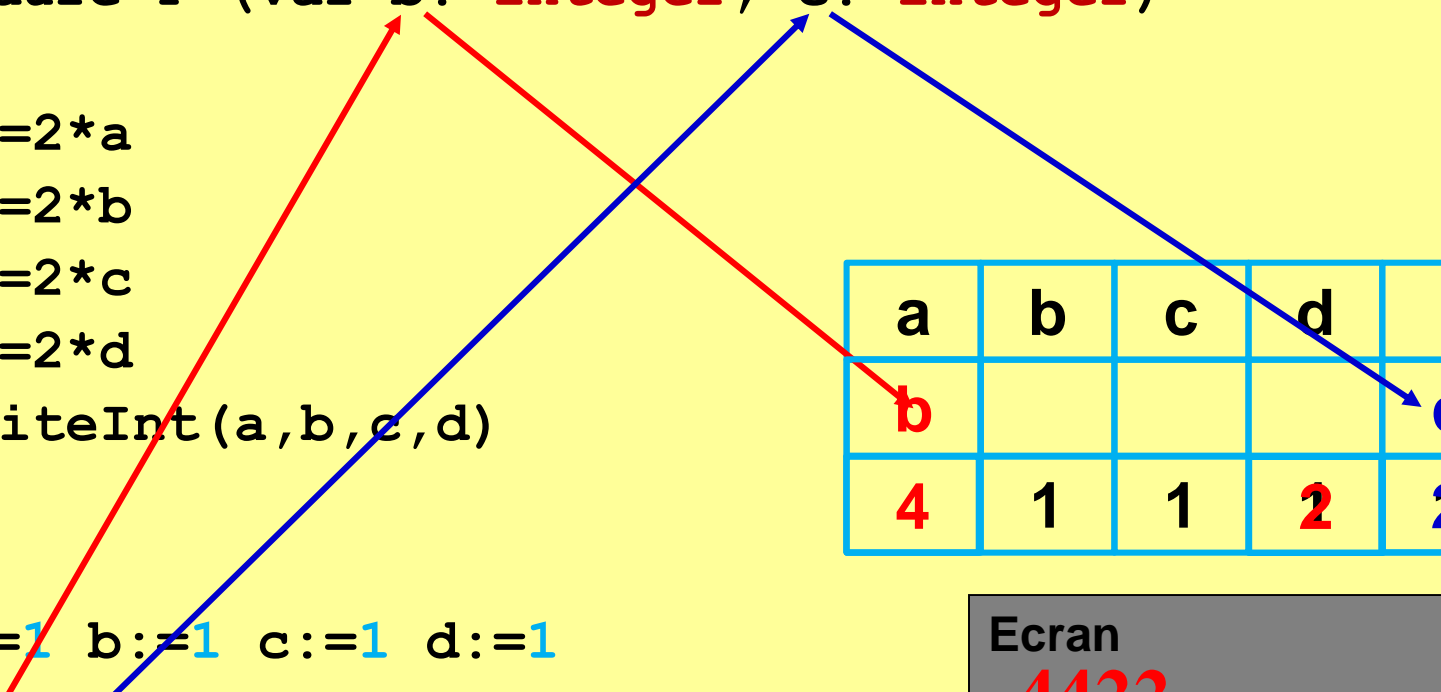
P(a, b)

WriteInt(a,b,c,d)

P(c, d)

WriteInt(a,b,c,d)

End



a	b	c	d	
	<b>b</b>			<b>c</b>
<b>4</b>	1	1	<b>2</b>	<b>2</b>

Ecran

**4422**

# Exemplu

Algoritm Afisare2

Var a, b, c, d: integer

Procedure P (var b: integer, c: integer)

begin

a:=2\*a

b:=2\*b

c:=2\*c

d:=2\*d

WriteInt(a,b,c,d)

end

Begin

a:=1 b:=1 c:=1 d:=1

P(a, b)

WriteInt(a,b,c,d)

P(c, d)

WriteInt(a,b,c,d)

End

a	b	c	d	
4	1	1	2	

Ecran

4422

4112

# Exemplu

Algoritm Afisare2

Var a, b, c, d: integer

Procedure P (var b: integer, c: integer)

begin

a:=2\*a

b:=2\*b

c:=2\*c

d:=2\*d

WriteInt(a,b,c,d)

end

Begin

a:=1 b:=1 c:=1 d:=1

P(a, b)

WriteInt(a,b,c,d)

P(c, d)

WriteInt(a,b,c,d)

End

a	b	c	d	
		<b>b</b>		<b>c</b>
<b>8</b>	1	<b>2</b>	<b>2</b>	<b>2</b>

Ecran

4422

4112

8244

# Exemplu

Algoritm Afisare2

Var a, b, c, d: integer

Procedure P (var b: integer, c: integer)

begin

a:=2\*a

b:=2\*b

c:=2\*c

d:=2\*d

WriteInt(a,b,c,d)

end

Begin

a:=1 b:=1 c:=1 d:=1

P(a, b)

WriteInt(a,b,c,d)

P(c, d)

WriteInt(a,b,c,d)

End

a	b	c	d	
8	1	2	4	

Ecran

4422

4112

8244

8124



# Exemplu. Aria totală

*Să se calculeze aria totală a două loturi dreptunghiulare.*

## Algoritm Aria\_totala

**Var**

A1, B1, A2, B2: Real

S1, S2, S\_total: Real

**Procedure** **Aria\_1\_lot3** (Latura1, Latura2: Real, **Var** S:Real)

**Begin**

S:=Latura1 \* Latura2

**End**

**Begin**

ReadReal(A1, B1)

**Aria\_1\_lot3(A1, B1, S1)**

ReadReal(A2, B2)

**Aria\_1\_lot3(A2, B2, S2)**

S\_total := S1+S2

WriteReal(S\_total)

**End**

# Tehnologia elaborării unei proceduri

---

- 1** Se alege un nume sugestiv în calitate de nume al procedurii;
- 2** Se identifică parametrii de intrare și cei de ieșire;
- 3** Se alege metoda de transmitere a parametrilor: parametrii de intrare pot fi transmiși prin valoare; parametrii de ieșire pot fi transmiși numai prin referință;
- 4** Se descrie lista parametrilor formali ai procedurii;
- 5** Se elaborează corpul procedurii, presupunând că valorile parametrilor de intrare sunt cunoscute.

# Funcții

---

**Funcția** – este un subalgoritm (subprogram), rezultatul căruia este o careva valoare.

## Exemple:

- **calcularea**  $\sin x$ ,  $\cos x$ ,  $\sqrt{x}$
- **efectuarea calculelor după formule complexe**
- **răspuns la întrebare (este număr prim sau nu?)**

## Pentru ce?

- **pentru efectuarea calculelor similare în diferite locuri a programului;**
- **pentru crearea bibliotecilor de funcții**



Prin ce se deosebesc de proceduri?

# Funcții

## Forma generală a definiției unei funcții:

```
Function <nume> (lista_parametri_formali): tip_rezultat  
Var  
    <variabile locale>  
Begin  
    <corpul funcției>  
End
```

**Lista parametrilor formali** conține descrierea tuturor parametrilor de intrare. Parametrii funcției pot fi transmiși atât prin valoare, cât și prin referință.

**Rezultatul** calculat de o funcție este de tip elementar sau String specificat în antet. Rezultatul calculat de funcție se întoarce programului apelant cu ajutorul operatorului **Return <expresie>**.

# Funcții. Particularități

- antetul se începe cu cuvântul **function**

```
function Max (a, b: integer): integer
```

- parametrii formali se descriu în același mod ca și pentru proceduri

```
function qq( a, b: integer, x: real ): real
```

- parametrii funcției pot fi transmiși atât prin valoare, cât și prin **referință**

```
function Max ( var a:integer, b: integer): integer
```

- la sfârșitul antetului după semnul **:** se indică **tipul rezultatului**

```
function Max (a, b: integer): integer
```

- toate funcțiile se descriu **mai sus** de programul principal

# Funcții. Particularități

---

- pot fi declarate și utilizate **variabile locale**

```
function qq (a, b: integer): real  
  var x, y: real  
begin  
  ...  
end
```

- rezultatul calculat de funcție se întoarce programului apelant cu ajutorul operatorului **Return <expresie>**

```
function Max (a, b: integer): integer  
var N_max: integer  
begin  
  ...  
  Return N_max  
end
```

# Apelul funcției

---

## Sintaxa apelului unei funcții:

... < nume\_funcție > (lista\_parametri\_actuali)

**Apelul funcției** se face chiar în expresia care utilizează valoarea calculată de funcție. Apelul funcției poate să apară în orice expresie, ca operand care are tipul rezultatului funcției.

**Funcțiile sunt tratate ca și variabile.** Ele pot fi folosite în cadrul:

- expresiilor aritmetice și logice;
- în cadrul apelului altor funcții pe post de parametri actuali;
- în cadrul instrucțiunilor Return.

# Apelul funcției

---

## Mecanismul de apel al unei funcții:

- *se face transferul parametrilor actuali funcției;*
- *se calculează valoarea fiecărui parametru actual transmis prin valoare;*
- *se alocă memorie pentru parametrii formali, transmiși prin valoare și pentru variabilele locale ale funcției;*
- *valorile parametrilor actuali calculate sunt atribuite parametrilor formali, inițializându-le;*
- *se instalează corespondența dintre parametrul formal transmis prin referință și parametrul actual corespunzător;*
- *se execută corpul funcției; ca urmare, este calculată valoarea funcției;*
- *această valoare este valoarea operandului, folosită, în continuare, la reluarea evaluării expresiei ce conține apelul funcției.*



# Exemplu. Suma cifrelor

*Să se calculeze suma cifrelor a unui număr natural.*

## Algoritm Sum

```
function sumDigits (n: natural) : natural
var sum: natural
begin
    sum := 0
    while n <> 0 do
        sum := sum + n mod 10
        n := n div 10
    end
    Return sum
end
```

tipul rezultatului

transmiterea  
rezultatului

```
begin
    WriteNat (sumDigits (12345))
end
```

# Utilizarea funcțiilor

---

```
x := 2 * sumDigits (n + 5)

z := sumDigits (k) + sumDigits (m)

if sumDigits (n) mod 2 = 0 then
  WriteString ('Suma cifrelor este pară')
  WriteString ('Ea este egală cu ')
  WriteNat (sumDigits (n))
end
```



Funcția care întoarce valoarea întreagă poate fi utilizată în locurile unde se pot folosi valori întregi!

# Exemplu. Maximul dintre 3 numere

---

*Să se calculeze  $S = \max(a, b, c) + \max(d, e, f)$ .*

**Algorithm** Suma\_max

**Var**

A, B, C, D, E, F: Integer

S: Integer

**Function** Max(N1, N2, N3: Integer): Integer

**Var** N\_max: Integer

**Begin**

**If** N1 > N2 **Then** N\_max := N1

**Else** N\_max := N2

**End**

**If** N3 > N\_max **Then** N\_max := N3

**End**

Return N\_max

**End**

# Exemplu. Maximul dintre 3 numere

---

**Begin**

ReadInt(A)

ReadInt(B)

ReadInt(C)

ReadInt(D)

ReadInt(E)

ReadInt(F)

$S := \text{Max}(A, B, C) + \text{Max}(D, E, F)$

WriteInt(S)

**End**

# Tehnologia elaborării unei funcții

---

**1**

Se alege un nume sugestiv în calitate de nume al funcției;

**2**

Se identifică parametrii de intrare;

**3**

Se alege metoda de transmitere a parametrilor;

**4**

Se descrie lista parametrilor formali ai funcției;

**5**

Se determină tipul de date ale rezultatului funcției;

**6**

Se elaborează corpul funcției, presupunând că valorile parametrilor de intrare sunt cunoscute.

# Sarcini pentru lucrul independent

---

1. Scrieți o procedură care determină suma divizorilor unui număr natural.
2. Scrieți o funcție logică care determină dacă un caracter este cifră.
3. Scrieți o procedură care calculează suma cifrelor impare ale unui număr și produsul cifrelor pare ale acestuia.
4. Scrieți o funcție care calculează distanța dintre două puncte în plan  $A(x_1, y_1)$  și  $B(x_2, y_2)$ .