

PROGRAMAREA CALCULATORULOR

Andrei Patrascu

andrei.patrascu@fmi.unibuc.ro

Cursul 3

REGULAMENT DE EVALUARE ȘI NOTARE

$$Nota = \min(10, \underset{6p}{Curs} + \underset{4p}{Laborator} + \underset{1p}{Seminar})$$

Examenul final scris (daca ne permit conditiile)

- Eventual online

Pondere examen final: 60%

Partial (test de laborator): 40%

- Online (pe grupe)

Activitate seminar: 10%

Restanta: se reda doar examenul final!

PROGRAMA CURSULUI

❑ Introducere

- Algoritmi
- Limbaje de programare.

❑ Fundamentele limbajului C

- Introducere în limbajul C. Structura unui program C.
- Tipuri de date fundamentale. Variabile. Constante. Operatori. Expresii. Conversii.
- Tipuri derivate de date: tablouri, șiruri de caractere, structuri, uniuni, câmpuri de biți, enumerări, pointeri
- Instrucțiuni de control
- Directive de preprocesare. Macrodefiniții.
- Funcții de citire/scriere.
- Etapele realizării unui program C.

❑ Fișiere text

- Funcții specifice de manipulare.

❑ Funcții (1)

- Declarare și definire. Apel. Metode de transmitere a paramerilor. Pointeri la funcții.

❑ Tablouri și pointeri

- Legătura dintre tablouri și pointeri
- Aritmetica pointerilor
- Alocarea dinamică a memoriei
- Clase de memorare

❑ Șiruri de caractere

- Funcții specifice de manipulare.

❑ Fișiere binare

- Funcții specifice de manipulare.

❑ Structuri de date complexe și autoreferite

- Definire și utilizare

❑ Funcții (2)

- Funcții cu număr variabil de argumente.
- Preluarea argumentelor funcției main din linia de comandă.

CURSUL DE AZI

1. Pointeri
2. Tablouri. Șiruri de caractere.
3. Structuri, uniuni, câmpuri de biți, enumerări.

POINTERI

- **pointer** = tip de date derivat folosit pentru manipularea adreselor de memorie

- **sintaxa**

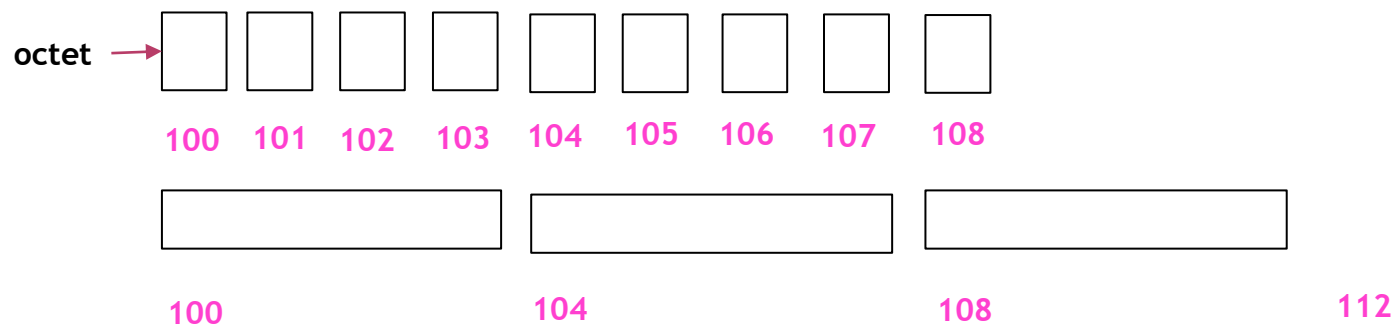
tip *nume_variabilă;

- **tip** = tipul de bază al variabilei de tip pointer `nume_variabilă`
 - ***** = operator de indirectare
 - **nume_variabila** = variabila de tip pointer care poate lua ca valori adrese de memorie
- **cel mai puternic mecanism de accesare a memoriei în C**

POINTERI

- **pointer** = tip de date derivat folosit pentru manipularea adreselor de memorie
- **operatori pentru manipularea adreselor de memorie:**
 - **&** - operatorul de referențiere
 - **&variabila** – furnizează adresa variabilei respective
 - ***** - operatorul de dereferențiere
 - ***variabila_de_tip_pointer** – furnizează valoarea aflată la adresa de memorie stocată în variabila_de_tip_pointer

MEMORIE



In principiu:

- Fiecare locatie de memorie are adresa unica
- Fiecare locatie de memorie contine o valoare

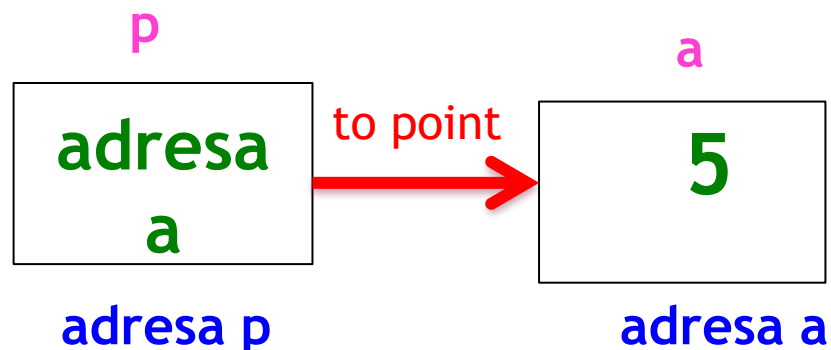
Nu putem lucra direct cu adrese de memorie!

De aceea, avem nevoie de **variabile de tip adresa!**

POINTERI

❑ exemplu:

```
int a=5  
int *p;  
p = &a;
```

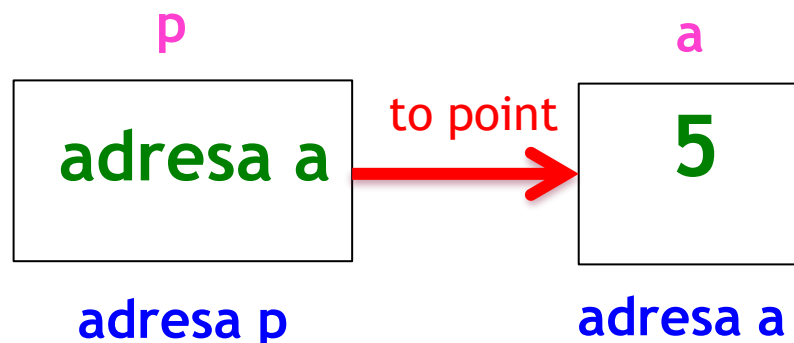


```
main.c x
1  #include <stdio.h>
2  #include <stdlib.h>
3
4
5  int main(){
6
7      int a=5,*p;
8      p = &a;
9
10     printf("Adresa lui a este: %d \n",&a);
11     printf("Valoarea stocata la adresa lui a este: %d \n",a);
12
13     printf("Adresa lui p este: %d \n",&p);
14     printf("Valoarea stocata la adresa lui p este: %d \n",p);
15     printf("Valoarea variabilei a carei adresa e stocata in p este %d \n",*p);
16
17     return 0;
18 }
19
```


POINTERI

❑ exemplu:

```
int a=5  
int *p;  
p = &a;
```



```
main.c x
1  #include <stdio.h>
2  #include <stdlib.h>
3
4
5  int main(){
6
7      int a=5,*p;
8      p = &a;
9
10     printf("Adresa lui a este: %d \n",&a);
11     printf("Valoarea stocata la adresa lui a este: %d \n",a);
12
13     printf("Adresa lui p este: %d \n",&p);
14     printf("Valoarea stocata la adresa lui p este: %d \n",p);
15     printf("Valoarea variabilei a carei adresa e stocata in p este %d \n",*p);
16
17     return 0;
18 }
19
```

Adresa lui a este: 1606416748
Valoarea stocata la adresa lui a este: 5
Adresa lui p este: 1606416736
Valoarea stocata la adresa lui p este: 1606416748
Valoarea variabilei a carei adresa e stocata in p este 5

Process returned 0 (0x0) execution time : 0.012 s
Press ENTER to continue.

POINTERI

- ❑ exemplu: modificarea valorii unei variabile prin dereferențiere

```
main.c x
1  #include <stdio.h>
2  #include <stdlib.h>
3
4
5  int main(){
6
7      int a=5,*p;
8      p = &a;
9
10     printf("Valoarea stocata la adresa lui a este: %d \n",a);
11
12     a += 10; //acces direct
13     printf("Valoarea stocata la adresa lui a este: %d \n",a);
14
15     *p += 20; //acces indirect
16     printf("Valoarea stocata la adresa lui a este: %d \n",a);
17
18     return 0;
19 }
20
```

POINTERI

- ❑ exemplu: modificarea valorii unei variabile prin dereferențiere

main.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4
5  int main(){
6
7      int a=5,*p;
8      p = &a;
9
10     printf("Valoarea stocata la adresa lui a este: %d \n",a);
11
12     a += 10; //acces direct
13     printf("Valoarea stocata la adresa lui a este: %d \n",a);
14
15     *p += 20; //acces indirect
16     printf("Valoarea stocata la adresa lui a este: %d \n",a);
17
18     return 0;
19 }
20
```

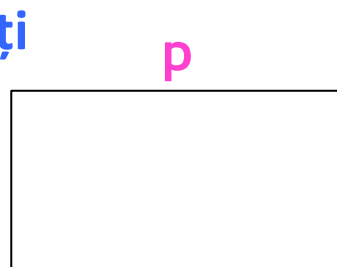
```
Valoarea stocata la adresa lui a este: 5
Valoarea stocata la adresa lui a este: 15
Valoarea stocata la adresa lui a este: 35
```

```
Process returned 0 (0x0)   execution time : 0.006 s
Press ENTER to continue.
```

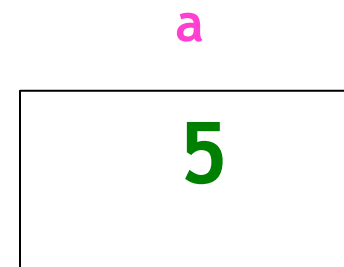
POINTERI

❑ exemplu: pointeri neinițializați

```
main.c ✕
1  #include <stdio.h>
2  #include <stdlib.h>
3
4
5  int main() {
6
7      int a=5, *p;
8      printf("a= %d\n", a);
9      *p = 10;
10     printf("a= %d \n", a);
11     return 0;
12 }
13
```



adresa p

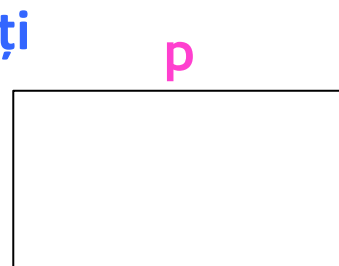


adresa a

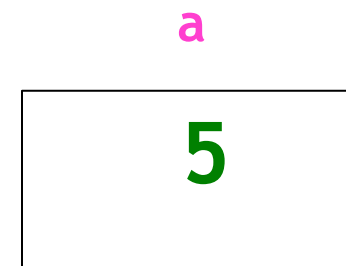
POINTERI

❑ exemplu: pointeri neinițializați

```
main.c ✕
1  #include <stdio.h>
2  #include <stdlib.h>
3
4
5  int main() {
6
7      int a=5,*p;
8      printf("a= %d\n",a);
9      *p = 10;
10     printf("a= %d \n",a);
11     return 0;
12 }
13
```



adresa p



adresa a

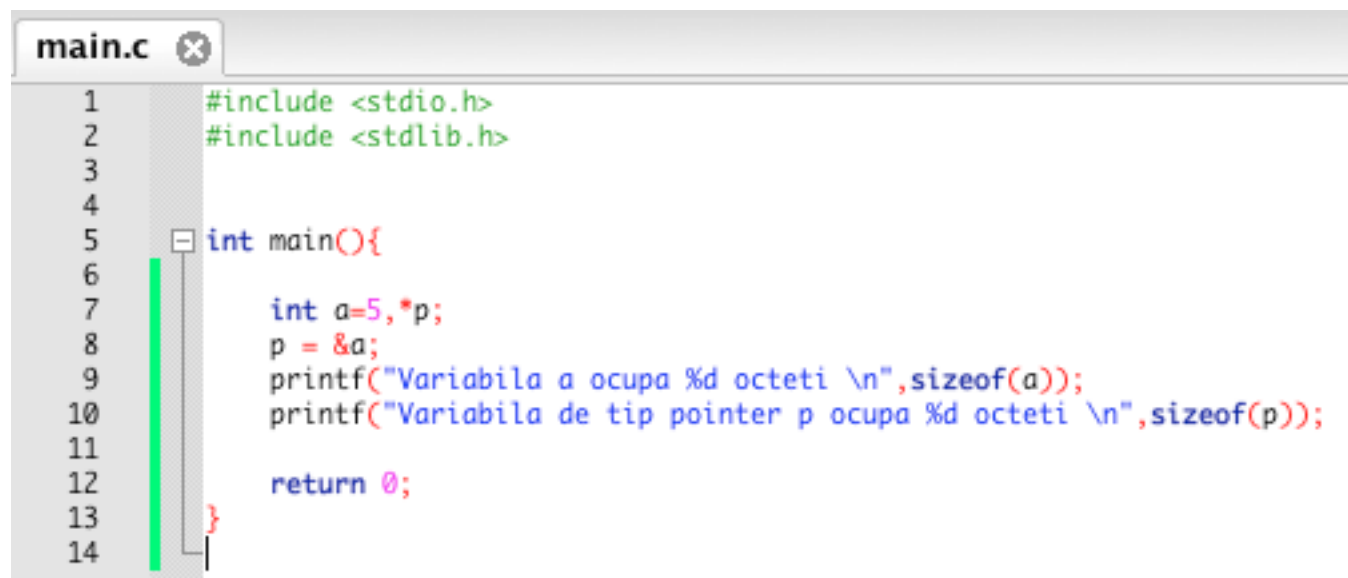
a= 5

Process returned -1 (0xFFFFFFFF)
Press ENTER to continue.

execution time : 0.027 s

POINTERI

- ❑ exemplu: spațiul de memorie ocupat de o variabilă de tip pointer



```
main.c ✕  
1  #include <stdio.h>  
2  #include <stdlib.h>  
3  
4  
5  int main()  
6  {  
7      int a=5,*p;  
8      p = &a;  
9      printf("Variabila a ocupa %d octeti \n",sizeof(a));  
10     printf("Variabila de tip pointer p ocupa %d octeti \n",sizeof(p));  
11  
12     return 0;  
13 }  
14
```

POINTERI

- ❑ exemplu: spațiul de memorie ocupat de o variabilă de tip pointer

```
main.c
1  #include <stdio.h>
2  #include <stdlib.h>
3
4
5  int main(){
6
7      int a=5,*p;
8      p = &a;
9      printf("Variabila a ocupa %d octeti \n",sizeof(a));
10     printf("Variabila de tip pointer p ocupa %d octeti \n",sizeof(p));
11
12     return 0;
13 }
14
```

Variabila a ocupa 4 octeti

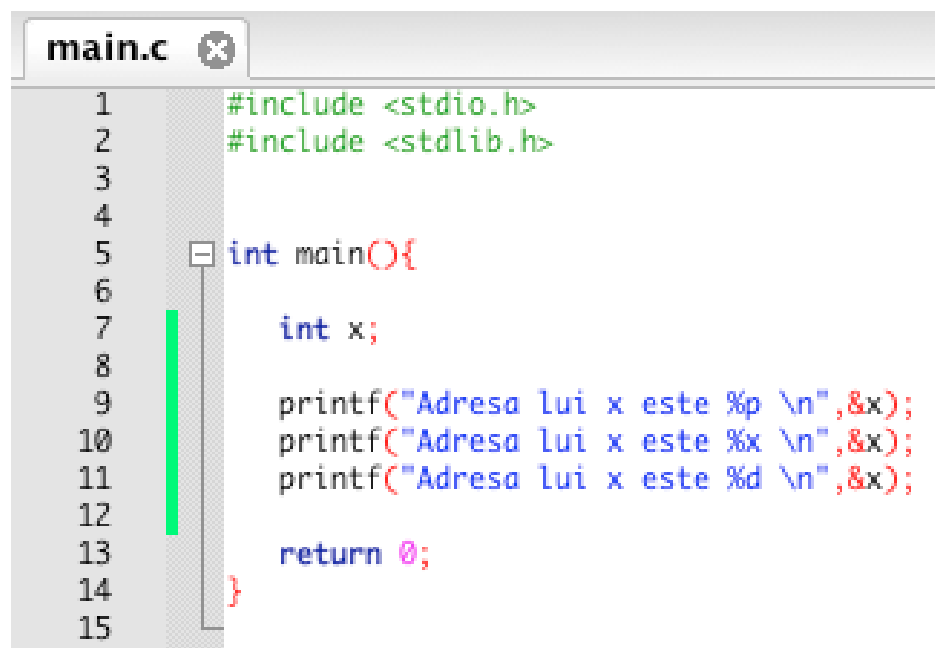
Variabila de tip pointer p ocupa 8 octeti

Process returned 0 (0x0) execution time : 0.005 s
Press ENTER to continue.

POINTERI

❑ adrese de memorie

- ❑ în C există un specificator de format special (**%p**) pentru tipărirea valorilor reprezentând adresele de memorie.



The screenshot shows a code editor window titled "main.c". The code is as follows:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4
5  int main(){
6
7      int x;
8
9      printf("Adresa lui x este %p \n",&x);
10     printf("Adresa lui x este %x \n",&x);
11     printf("Adresa lui x este %d \n",&x);
12
13     return 0;
14 }
15
```

A vertical green line is positioned at the start of line 7, and a mouse cursor is visible over the opening curly brace of the `main` function on line 5.

POINTERI

❑ adrese de memorie

- ❑ în C există un specificator de format special (**%p**) pentru tipărirea valorilor reprezentând adresele de memorie.

```
main.c x
1  #include <stdio.h>
2  #include <stdlib.h>
3
4
5  int main(){
6
7      int x;
8
9      printf("Adresa lui x este %p \n",&x);
10     printf("Adresa lui x este %x \n",&x);
11     printf("Adresa lui x este %d \n",&x);
12
13     return 0;
14 }
15
```

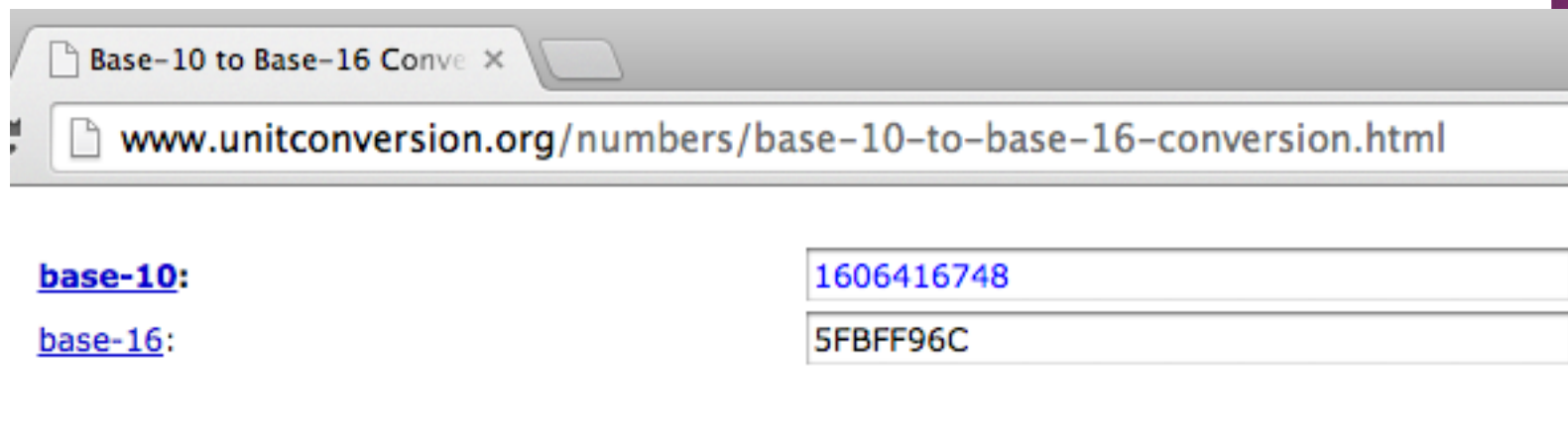
```
Adresa lui x este 0x7fff5fbff96c
Adresa lui x este 5fbff96c
Adresa lui x este 1606416748
```

```
Process returned 0 (0x0)   execution time : 0.008 s
Press ENTER to continue.
```

POINTERI

❑ adrese de memorie

- ❑ în C există un specificator de format special (**%p**) pentru tipărirea valorilor reprezentând adresele de memorie.



The screenshot shows a web browser window with a single tab titled "Base-10 to Base-16 Conve x". The address bar displays the URL "www.unitconversion.org/numbers/base-10-to-base-16-conversion.html". Below the address bar, there are two input fields. The first field is labeled "base-10:" and contains the value "1606416748". The second field is labeled "base-16:" and contains the value "5FBFF96C".

<u>base-10:</u>	1606416748
<u>base-16:</u>	5FBFF96C

CURSUL DE AZI

1. Pointeri
2. Tablouri. Șiruri de caractere.
3. Structuri, uniuni, câmpuri de biți, enumerări.

TABLOURI UNIDIMENSIONALE

□ sintaxa:

tip nume_tablou [dimensiune];

□ exemple:

double v[100];

0.3	-1.2	10	5.7	...	0.2	-1.5	1
0	1	2	3		9	9	9
					7	8	9

v[3] = 5.7;

int a[5];

3	-12	10	7	1
0	1	2	3	4

a[0] = 3;

char c[34];

A	&	*	+	...	c	M	#
0	1	2	3		3	3	3
					1	2	3

c[1] = '&';

□ în C, primul element al unui tablou are indicele 0.

TABLOURI UNIDIMENSIONALE

- definiție: set de valori de același tip memorat la adrese succesive de memorie.

tabloul.c

```
1  #include <stdio.h>
2
3  int main(){
4      int a[5],i;
5      for(i=0;i<5;i++)
6          printf("Adresa elementului %d din tabloul a este %p \n",i,&a[i]);
7      printf("*****\n");
8
9      double v[100];
10     for(i=0;i<3;i++)
11         printf("Adresa elementului %d din tabloul v este %p \n",i,&v[i]);
12     printf("*****\n");
13
14     char c[34];
15     for(i=0;i<4;i++)
16         printf("Adresa elementului %d din tabloul c este %p \n",i,&c[i]);
17     printf("*****\n");
18
19     return 0;
20 }
21
```

TABLOURI UNIDIMENSIONALE

- definiție: set de valori de același tip memorat la adrese succesive de memorie.

tabloul.c

```
1  #include <stdio.h>
2
3  int main() {
4      int a[5], i;
5      for(i=0; i<5; i++)
6          printf("Adresa elementului %d din tabloul a este %p \n", i, &a[i]);
7      printf("*****\n");
8
9      double v[100];
10     for(i=0; i<3; i++)
11         printf("Adresa elementului %d din tabloul v este %p \n", i, &v[i]);
12     printf("*****\n");
13
14     char c[34];
15     for(i=0; i<4; i++)
16         printf("Adresa elementului %d din tabloul c este %p \n", i, &c[i]);
17     printf("*****\n");
18
19     return 0;
20 }
21
```

Adresa elementului 0 din tabloul a este 0x7fff5fbff940
Adresa elementului 1 din tabloul a este 0x7fff5fbff944
Adresa elementului 2 din tabloul a este 0x7fff5fbff948
Adresa elementului 3 din tabloul a este 0x7fff5fbff94c
Adresa elementului 4 din tabloul a este 0x7fff5fbff950

Adresa elementului 0 din tabloul v este 0x7fff5fbff620
Adresa elementului 1 din tabloul v este 0x7fff5fbff628
Adresa elementului 2 din tabloul v este 0x7fff5fbff630

Adresa elementului 0 din tabloul c este 0x7fff5fbff960
Adresa elementului 1 din tabloul c este 0x7fff5fbff961
Adresa elementului 2 din tabloul c este 0x7fff5fbff962
Adresa elementului 3 din tabloul c este 0x7fff5fbff963

TABLOURI UNIDIMENSIONALE

- ❑ definiție: set de valori de același tip memorat la adrese succesive de memorie.
- ❑ memorie:
 - ❑ cantitatea de memorie necesară pentru stocarea unui tablou este direct proporțională cu tipul de date și mărimea sa.
 - ❑ tip nume [dimensiune] → $\text{sizeof}(\text{nume}) = \text{sizeof}(\text{tip}) * \text{dimensiune}$;

TABLOURI UNIDIMENSIONALE

memorie:

- cantitatea de memorie necesară pentru stocarea unui tablou este direct proporțională cu tipul de date și mărimea sa.
- tip nume [dimensiune] → $\text{sizeof}(\text{nume}) = \text{sizeof}(\text{tip}) * \text{dimensiune}$;

```
main.c x
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(){
5
6      double v[100];
7      int a[5];
8      char c[34];
9      printf("Stocarea tabloului v necesita %d octeti \n",sizeof(v));
10     printf("Stocarea tabloului a necesita %d octeti \n",sizeof(a));
11     printf("Stocarea tabloului c necesita %d octeti \n",sizeof(c));
12     return 0;
13 }
```


TABLOURI UNIDIMENSIONALE

memorie:

- cantitatea de memorie necesară pentru stocarea unui tablou este direct proporțională cu tipul de date și mărimea sa.
- tip nume [dimensiune] → $\text{sizeof}(\text{nume}) = \text{sizeof}(\text{tip}) * \text{dimensiune}$;

```
main.c x
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(){
5
6      double v[100];
7      int a[5];
8      char c[34];
9      printf("Stocarea tabloului v necesita %d octeti \n",sizeof(v));
10     printf("Stocarea tabloului a necesita %d octeti \n",sizeof(a));
11     printf("Stocarea tabloului c necesita %d octeti \n",sizeof(c));
12     return 0;
13 }
```

```
Stocarea tabloului v necesita 800 octeti
Stocarea tabloului a necesita 20 octeti
Stocarea tabloului c necesita 34 octeti

Process returned 0 (0x0)   execution time : 0.010 s
Press ENTER to continue.
```

TABLOURI UNIDIMENSIONALE

▣ exemplu de inițializare

```
1  #include <stdio.h>
2
3  int main() {
4
5      int a[3], x=100;
6      printf("x=%d\n", x);
7      a[0] = a[1] = a[2] = a[3] = 17;
8
9      printf("x=%d\n", x);
10
11     return 0;
12 }
```

TABLOURI UNIDIMENSIONALE

❑ exemplu de inițializare

```
1  #include <stdio.h>
2
3  int main() {
4
5      int a[3], x=100;
6      printf("x=%d\n", x);
7      a[0] = a[1] = a[2] = a[3] = 17;
8
9      printf("x=%d\n", x);
10
11     return 0;
12 }
```

x=100

x=17

TABLOURI UNIDIMENSIONALE

❑ exemplu de inițializare

```
1  #include <stdio.h>
2
3  int main() {
4
5      int a[3], x=100;
6      printf("x=%d\n",x);
7      a[0] = a[1] = a[2] = a[3] = 17;
8
9      printf("x=%d\n",x);
10
11     printf("Adresa lui a[2] este %p \n",&a[2]);
12     printf("Adresa lui x este %p \n",&x);
13
14     return 0;
15 }
```

TABLOURI UNIDIMENSIONALE

▣ exemplu de inițializare

```
1  #include <stdio.h>
2
3  int main()
4
5      int a[3], x=100;
6      printf("x=%d\n",x);
7      a[0] = a[1] = a[2] = a[3] = 17;
8
9      printf("x=%d\n",x);
10
11     printf("Adresa lui a[2] este %p \n",&a[2]);
12     printf("Adresa lui x este %p \n",&x);
13
14     return 0;
15 }
```

x=100
x=17
Adresa lui a[2] este 0x7fff5fbff988
Adresa lui x este 0x7fff5fbff98c

TABLOURI BIDIMENSIONALE

□ sintaxa

tip nume_variabila [dimensiune1][dimensiune2];

□ exemplu

int a[3][5];

a[1][4] = 41;

0	3	-12	10	7	1
1	10	2	0	-7	41
2	-3	-2	0	0	2
	0	1	2	3	4

TABLOURI BIDIMENSIONALE

- definiție: set de valori de același tip memorat la adrese succesive de memorie.

```
tablou3.c x
1  #include <stdio.h>
2
3  int main(){
4
5      int i,j;
6      int a[3][5];
7
8      for(i=0;i<3;i++)
9          for(j=0;j<5;j++)
10             printf("Adresa elementului [%d][%d] din tabloul a este %p \n",i,j,&a[i][j]);
11
12     return 0;
13 }
```

TABLOURI BIDIMENSIONALE

- definiție: set de valori de același tip memorat la adrese succesive de memorie.

```
tablou3.c x
1  #include <stdio.h>
2
3  int main() {
4
5      int i,j;
6      int a[3][5];
7
8      for(i=0;i<3;i++)
9          for(j=0;j<5;j++)
10             printf("Adresa elementului [%d][%d] din tabloul a este %p \n",i,j,&a[i][j]);
11
12     return 0;
13 }
```

Adresa elementului [0][0] din tabloul a este 0x7fff5fbff940
Adresa elementului [0][1] din tabloul a este 0x7fff5fbff944
Adresa elementului [0][2] din tabloul a este 0x7fff5fbff948
Adresa elementului [0][3] din tabloul a este 0x7fff5fbff94c
Adresa elementului [0][4] din tabloul a este 0x7fff5fbff950
Adresa elementului [1][0] din tabloul a este 0x7fff5fbff954
Adresa elementului [1][1] din tabloul a este 0x7fff5fbff958
Adresa elementului [1][2] din tabloul a este 0x7fff5fbff95c
Adresa elementului [1][3] din tabloul a este 0x7fff5fbff960
Adresa elementului [1][4] din tabloul a este 0x7fff5fbff964
Adresa elementului [2][0] din tabloul a este 0x7fff5fbff968
Adresa elementului [2][1] din tabloul a este 0x7fff5fbff96c
Adresa elementului [2][2] din tabloul a este 0x7fff5fbff970
Adresa elementului [2][3] din tabloul a este 0x7fff5fbff974
Adresa elementului [2][4] din tabloul a este 0x7fff5fbff978

TABLOURI BIDIMENSIONALE

- definiție: set de valori de același tip memorat la adrese succesive de memorie.

3	-12	10	7	1
10	2	0	-7	41
-3	-2	0	0	2
0	1	2	3	4



3	-12	10	7	1	10	2	0	-7	41	-3	-2	0	0	2
a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]	a[1][0]	...								a[2][4]

```
Adresa elementului [0][0] din tabloul a este 0x7fff5fbff940
Adresa elementului [0][1] din tabloul a este 0x7fff5fbff944
Adresa elementului [0][2] din tabloul a este 0x7fff5fbff948
Adresa elementului [0][3] din tabloul a este 0x7fff5fbff94c
Adresa elementului [0][4] din tabloul a este 0x7fff5fbff950
Adresa elementului [1][0] din tabloul a este 0x7fff5fbff954
Adresa elementului [1][1] din tabloul a este 0x7fff5fbff958
Adresa elementului [1][2] din tabloul a este 0x7fff5fbff95c
Adresa elementului [1][3] din tabloul a este 0x7fff5fbff960
Adresa elementului [1][4] din tabloul a este 0x7fff5fbff964
Adresa elementului [2][0] din tabloul a este 0x7fff5fbff968
Adresa elementului [2][1] din tabloul a este 0x7fff5fbff96c
Adresa elementului [2][2] din tabloul a este 0x7fff5fbff970
Adresa elementului [2][3] din tabloul a este 0x7fff5fbff974
Adresa elementului [2][4] din tabloul a este 0x7fff5fbff978
```

Reprezentarea în memoria calculatorului a unui tablou
bidimensional

TABLOURI BIDIMENSIONALE

- ❑ definiție: set de valori de același tip memorat la adrese succesive de memorie.
- ❑ memorie:
 - ❑ cantitatea de memorie necesară pentru stocarea unui tablou este direct proporțională cu tipul de date și mărimea sa.
 - ❑ tip nume [dimensiune1][dimensiune2] → $\text{sizeof}(\text{nume}) = \text{sizeof}(\text{tip}) * \text{dimensiune1} * \text{dimensiune2}$;

TABLOURI BIDIMENSIONALE

- cantitatea de memorie necesară pentru stocarea unui tablou este direct proporțională cu tipul de date și mărimea sa.
- tip nume [dimensiune1][dimensiune2] →
 $\text{sizeof}(\text{nume}) = \text{sizeof}(\text{tip}) * \text{dimensiune1} * \text{dimensiune2};$

```
main.c x
1  #include <stdio.h>
2  #include <stdlib.h>
3
4
5  int main() {
6
7      double v[100][50];
8      int a[5][25];
9      char c[34][10];
10     printf("Stocarea tabloului v necesita %d octeti \n", sizeof(v));
11     printf("Stocarea tabloului a necesita %d octeti \n", sizeof(a));
12     printf("Stocarea tabloului c necesita %d octeti \n", sizeof(c));
13     return 0;
14 }
15
```

TABLOURI BIDIMENSIONALE

- cantitatea de memorie necesară pentru stocarea unui tablou este direct proporțională cu tipul de date și mărimea sa.
- tip nume [dimensiune1][dimensiune2] →
 $\text{sizeof}(\text{nume}) = \text{sizeof}(\text{tip}) * \text{dimensiune1} * \text{dimensiune2};$

```
main.c x
1  #include <stdio.h>
2  #include <stdlib.h>
3
4
5  int main()
6
7      double v[100][50];
8      int a[5][25];
9      char c[34][10];
10     printf("Stocarea tabloului v necesita %d octeti \n",sizeof(v));
11     printf("Stocarea tabloului a necesita %d octeti \n",sizeof(a));
12     printf("Stocarea tabloului c necesita %d octeti \n",sizeof(c));
13     return 0;
14
15
```

Stocarea tabloului v necesita 40000 octeti
Stocarea tabloului a necesita 500 octeti
Stocarea tabloului c necesita 340 octeti

Process returned 0 (0x0) execution time : 0.00
Press ENTER to continue.

TABLOURI BIDIMENSIONALE

citire și afișare:

```
main.c x
1  #include <stdio.h>
2  #include <stdlib.h>
3
4
5  int main() {
6
7      int L, C, i, j;
8      printf("Nr de linii a matricei este L="); scanf("%d", &L);
9      printf("Nr de linii a matricei este C="); scanf("%d", &C);
10     int a[L][C];
11     //CITIRE
12     printf("CITIRE matrice a \n");
13     for(i=0; i<L; i++)
14         for(j=0; j<C; j++)
15         {
16             printf("a[%d][%d] = ", i, j);
17             scanf("%d", &a[i][j]);
18         }
19     //AFISARE
20     printf("AFISARE matrice a \n");
21     for(i=0; i<L; i++)
22     {
23         for(j=0; j<C; j++)
24             printf("a[%d][%d] = %d \t ", i, j, a[i][j]);
25         printf("\n");
26     }
27     return 0;
28 }
```

valabil in
standardul
C99

TABLOURI BIDIMENSIONALE

citire și afișare:

```
main.c x
1  #include <stdio.h>
2  #include <stdlib.h>
3
4
5  int main() {
6
7      int L, C, i, j;
8      printf("Nr de linii a matricei este L="); scanf("%d", &L);
9      printf("Nr de linii a matricei este C="); scanf("%d", &C);
10     int a[L][C];
11     //CITIRE
12     printf("CITIRE matrice a \n");
13     for(i=0; i<L; i++)
14         for(j=0; j<C; j++)
15         {
16             printf("a[%d][%d] = ", i, j);
17             scanf("%d", &a[i][j]);
18         }
19     //AFISARE
20     printf("AFISARE matrice a \n");
21     for(i=0; i<L; i++)
22     {
23         for(j=0; j<C; j++)
24             printf("a[%d][%d] = %d \t ", i, j, a[i][j]);
25         printf("\n");
26     }
27     return 0;
28 }
```

Nr de linii a matricei este L=2

Nr de linii a matricei este C=3

CITIRE matrice a

a[0][0] = 1

a[0][1] = 2

a[0][2] = 3

a[1][0] = 4

a[1][1] = 5

a[1][2] = 6

AFISARE matrice a

a[0][0] = 1 a[0][1] = 2 a[0][2] = 3

a[1][0] = 4 a[1][1] = 5 a[1][2] = 6

Process returned 0 (0x0)

execution time : 4.9

Press ENTER to continue.

ȘIRURI DE CARACTERE

- ❑ **un șir de caractere (string)** este o zonă de memorie ocupată cu caractere/char-uri (un char ocupă un octet) terminată cu un octet de valoare 0 (caracterul `'\0'` are codul ASCII egal cu 0).
- ❑ o variabilă care reprezintă un șir de caractere este un pointer (adresa) la primul octet.
- ❑ se poate reprezenta ca:
 - ❑ tablou de caractere (pointer constant):
 - ❑ `char sir1[10];`
 - ❑ `char sir2[10] = "exemplu";`
 - ❑ pointer la caractere:
 - ❑ `char *sir3;`
 - ❑ `char *sir4 = "exemplu";`

CODURILE ASCII

- ❑ **cod ASCII** = reprezentarea numerică a unui caracter.
(ASCII - American Standard Code for Information Interchange)

```
main.c
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main() {
5
6      int i;
7
8      for(i = 33; i <= 127; i++)
9      {
10         printf("%c ", i);
11         if ((i-2) % 10 == 0)
12             printf("\n");
13     }
14     return 0;
15 }
16
```

Ce afișează programul?

CODURILE ASCII

- ❑ **cod ASCII** = reprezentarea numerică a unui caracter.
(ASCII - American Standard Code for Information Interchange)

```
main.c
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main() {
5
6      int i;
7
8      for(i = 33; i <= 127; i++)
9      {
10         printf("%c ", i);
11         if ((i-2) % 10 == 0)
12             printf("\n");
13     }
14     return 0;
15 }
16
```

Ce afișează programul?

```
! " # $ % & ' ( ) *
+ , - . / 0 1 2 3 4
5 6 7 8 9 : ; < = >
? @ A B C D E F G H
I J K L M N O P Q R
S T U V W X Y Z [ \
] ^ _ ` a b c d e f
g h i j k l m n o p
q r s t u v w x y z
{ | } ~
```

```
Process returned 0 (0x0)
Press ENTER to continue.
```

execution time :

CODURILE ASCII

❑ **cod ASCII** = reprezentarea numerică a unui caracter.

(ASCII - American Standard Code for Information

Interchange)

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040	 	Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	\$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	((72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051))	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[[123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174	|	
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135]]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177		DEL

CODURILE ASCII

❑ **cod ASCII** = reprezentarea numerică a unui caracter.
(ASCII - American Standard Code for Information

Int **Extended ASCII Codes**

128	Ç	144	É	160	á	176	░	192	Ł	208	⌚	224	α	240	≡
129	ü	145	æ	161	í	177	▒	193	ł	209	⌞	225	β	241	±
130	é	146	Æ	162	ó	178	▓	194	ṽ	210	⌘	226	Γ	242	≥
131	â	147	ô	163	ú	179		195	ṽ	211	⌚	227	π	243	≤
132	ä	148	ö	164	ñ	180	┆	196	—	212	↳	228	Σ	244	∫
133	à	149	ò	165	Ñ	181	┆	197	+	213	ℱ	229	σ	245	∫
134	â	150	û	166	°	182		198	↳	214	ℱ	230	μ	246	÷
135	ç	151	ù	167	°	183	π	199		215		231	τ	247	≈
136	ê	152	ÿ	168	¿	184	┆	200	↳	216	≠	232	Φ	248	°
137	ë	153	Ö	169	┆	185		201	ℱ	217	┆	233	⊙	249	.
138	è	154	Û	170	┆	186		202	↳	218	┆	234	Ω	250	.
139	ï	155	◊	171	½	187	┆	203	⌞	219	■	235	δ	251	√
140	î	156	£	172	¼	188	┆	204	┆	220	■	236	∞	252	π
141	ì	157	¥	173	¡	189	┆	205	=	221	■	237	φ	253	²
142	Ä	158	ℳ	174	«	190	┆	206	┆	222	■	238	ε	254	■
143	Å	159	ƒ	175	»	191	┆	207	↳	223	■	239	∩	255	

CITIREA ȘI AFIȘAREA ȘIRURILOR DE CARACTERE

❑ citire:

- ❑ funcția `scanf` cu modelatorul de format `%s`:
 - ❑ atenție: dacă inputul este un șir de caractere cu spațiu citește până la spațiu
- ❑ Funcția `fgets`
 - ❑ citește și spațiile

❑ afișare:

- ❑ funcția `printf` cu modelatorul de format `%s`:
- ❑ funcția `puts`
 - ❑ trece pe linia următoare

```
unsigned long long int l;  
unsigned char *ch;  
fflush(stdin);  
scanf("%d",&l); //prin citire cu %d se scriu 4 octeti (un int) in l
```

```
printf("sizeof(int) = %d \n\n",sizeof(int));
```

```
ch = &l;  
printf("Valoare data de ultimii 4 octeti: %u \n",l);  
printf("Ultimii 4 octeti: %d %d %d %d\n\n",*(ch+3),*(ch+2),*(ch+1),*ch);  
printf("8 octeti: %d %d %d %d %d %d %d %d\n",*(ch+7),*(ch+6),*(ch+5),*(ch+4),*(ch+3),*(ch+2),*(ch+1),*ch);  
printf("Valoare data de toti cei 8 octeti: %lld \n\n\n",l);
```

```
l = l/10;  
printf("Valoare lui l/10 data de ultimii 4 octeti = %u \n",l);  
printf("Ultimii 4 octeti: %d %d %d %d\n",*(ch+3),*(ch+2),*(ch+1),*ch);  
printf("8 octeti: %d %d %d %d %d %d %d %d\n",*(ch+7),*(ch+6),*(ch+5),*(ch+4),*(ch+3),*(ch+2),*(ch+1),*ch);  
printf("Valoare lui l/10 data de toti cei 8 octeti = %lld \n",l);
```

```
unsigned long long int l;  
unsigned char *ch;  
fflush(stdin);  
scanf("%d",&l); //prin citire cu %d se scriu 4 octeti (un int) in l
```

```
printf("sizeof(int) = %d \n\n",sizeof(int));
```

```
ch = &l;
```

```
printf("Valoare data de ultimi  
printf("Ultimii 4 octeti: %d %  
printf("8 octeti: %d %d %d %d  
printf("Valoare data de toti c
```

```
l = l/10;
```

```
printf("Valoare lui l/10 data  
printf("Ultimii 4 octeti: %d %  
printf("8 octeti: %d %d %d %d  
printf("Valoare lui l/10 data de toti cei 8 octeti = %lld \n",l);
```

```
234  
sizeof<int> = 4  
Valoare data de ultimii 4 octeti: 234  
Ultimii 4 octeti: 0 0 0 234  
8 octeti: 0 0 0 121 0 0 0 234  
Valoare data de toti cei 8 octeti: 519691043050  
Valoare lui l/10 data de ultimii 4 octeti = 429496753  
Ultimii 4 octeti: 25 153 153 177  
8 octeti: 0 0 0 12 25 153 153 177  
Valoare lui l/10 data de toti cei 8 octeti = 51969104305  
Process returned 0 (0x0) execution time : 1.767 s  
Press any key to continue.
```

```
unsigned long long int l;
```

```
unsigned char *ch;
```

```
fflush(stdin);
```

```
scanf("%lld",&l); //prin citire cu %d se scriu 4 octeti (un int) in l
```

```
printf("sizeof(int) = %d \n\n",sizeof(int));
```

```
ch = &l;
```

```
printf("Valoare data de ultimii 4 octeti: %u \n",l);
```

```
printf("Ultimii 4 octeti: %d %d %d %d\n\n",*(ch+3),*(ch+2),*(ch+1),*ch);
```

```
printf("8 octeti: %d %d %d %d %d %d %d %d\n",*(ch+7),*(ch+6),*(ch+5),*(ch+4),*(ch+3),*(ch+2),*(ch+1),*ch);
```

```
printf("Valoare data de toti cei 8 octeti: %lld \n\n\n",l);
```

```
l = l/10;
```

```
printf("Valoare lui l/10 data de ultimii 4 octeti = %u \n",l);
```

```
printf("Ultimii 4 octeti: %d %d %d %d\n",*(ch+3),*(ch+2),*(ch+1),*ch);
```

```
printf("8 octeti: %d %d %d %d %d %d %d %d\n",*(ch+7),*(ch+6),*(ch+5),*(ch+4),*(ch+3),*(ch+2),*(ch+1),*ch);
```

```
printf("Valoare lui l/10 data de toti cei 8 octeti = %lld \n",l);
```

```
unsigned long long int l;
```

```
unsigned char *ch;
```

```
fflush(stdin);
```

```
scanf("%lld",&l); //prin citire cu %d se scriu 4 octeti (un int) in l
```

```
printf("sizeof(int)
```

```
234  
sizeof<int> = 4
```

```
ch = &l;
```

```
printf("Valoare data
```

```
Valoare data de ultimii 4 octeti: 234  
Ultimii 4 octeti: 0 0 0 234
```

```
printf("Ultimii 4 oc
```

```
printf("%8 octeti: %c
```

```
8 octeti: 0 0 0 0 0 0 0 234  
Valoare data de toti cei 8 octeti: 234
```

```
printf("Valoare data
```

```
Valoare lui 1/10 data de ultimii 4 octeti = 23  
Ultimii 4 octeti: 0 0 0 23  
8 octeti: 0 0 0 0 0 0 0 23  
Valoare lui 1/10 data de toti cei 8 octeti = 23
```

```
l = l/10;
```

```
Process returned 0 (0x0) execution time : 2.639 s  
Press any key to continue.
```

```
printf("Valoare lui
```

```
printf("Ultimii 4 octeti: %d %d %d %d\n",*(ch+3),*(ch+2),*(ch+1),*ch);
```

```
printf("%8 octeti: %d %d %d %d %d %d %d %d\n",*(ch+7),*(ch+6),*(ch+5),*(ch+4),*(ch+3),*(ch+2),*(ch+1),*ch);
```

```
printf("Valoare lui 1/10 data de toti cei 8 octeti = %lld \n",l);
```


- Citire numar sub forma de sir de caractere

```
int nr[10], count = 0;
char ch[10];
scanf("%s", ch);

while (count < strlen(ch))
{
    nr[count] = ch[count] - '0';
    count++;
}
printf("\n\n");

for (count=0 ; count<strlen(ch) ; count++)
    printf(" %d \n", nr[count]);
```

```
int nr[10], count = 0;
char ch[10];
scanf("%s", ch);

while (count < strlen(ch))
{
    nr[count] = ch[count] - '0';
    count++;
}
printf("\n\n");

for (count=0 ; count<10 ; count++)
    printf(" %d", nr[count]);
```

123

1
2
3

Process returned 0 (0x0) execution time : 4.372 s
Press any key to continue.