

Introducere în programare 2013 - 2014

Corina Forăscu
corinfor@info.uaic.ro

<http://profs.info.uaic.ro/~corinfor/teach/IntroP/>

Curs 3: conținut

- Tablouri
- Pointeri

Tablouri

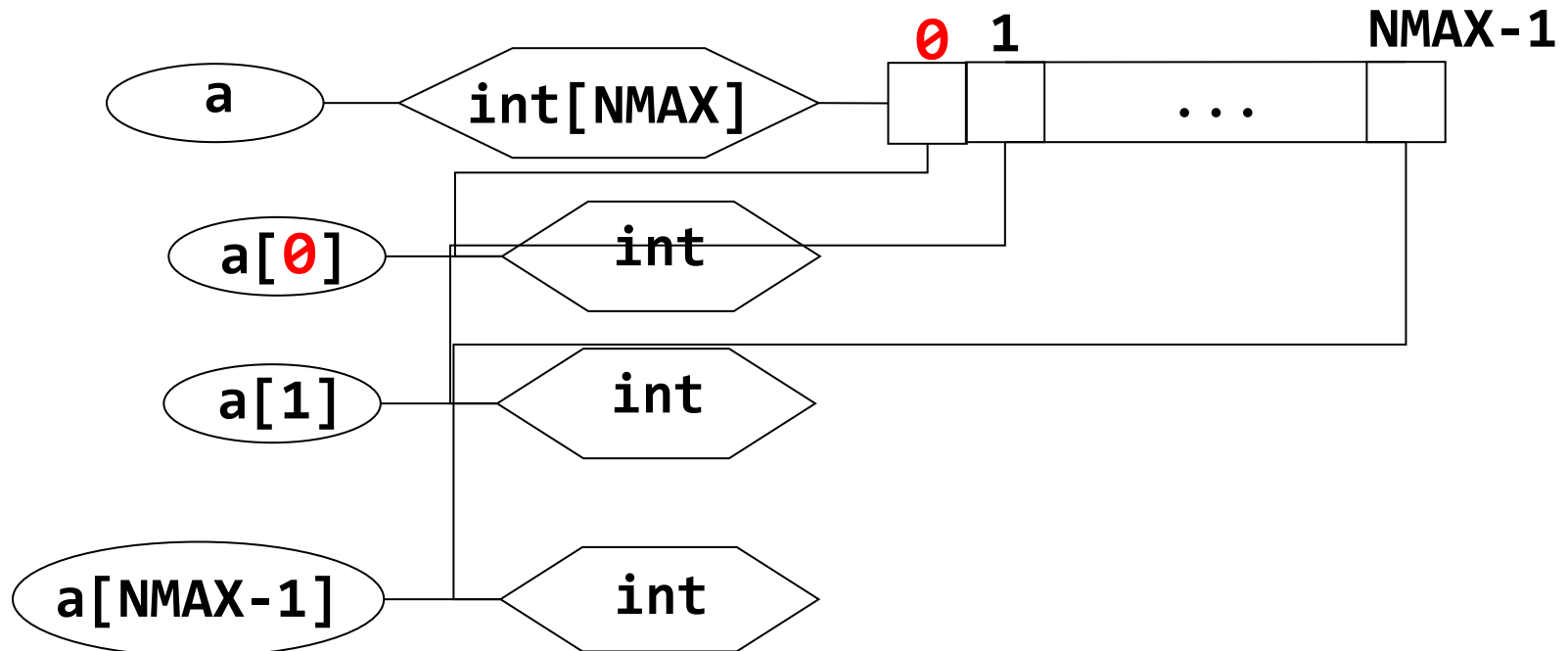
- Colecție de variabile de același tip, apelate cu același nume - componentele tabloului
- Componentele sunt identificate și accesate cu ajutorul indicilor
- Un tablou ocupă locații de memorie contigue, în ordinea indicilor
- Tablouri:
 - Unidimensionale (1 – dimensionale): șiruri de caractere
 - Bidimensionale (2 - dimensionale)

Tablouri unidimensionale

tip numeTablou[*dimensiune*];

octeți_necesari = sizeof(*tip*)* *dimensiune*

```
#define NMAX 25  
int a[NMAX]; // int a[25];
```



Tablouri unidimensionale

- Ordinea de memorare – ordinea indicilor
- Elementele tabloului – în zone de memorie contigue
- Operațiile se realizează prin intermediul componentelor, prin iterații **for** sau **while**,:

```
for (i=0; i<n; i++) a[i]=0;
```

```
for (i=0; i<n; i++) c[i]=a[i]+ b[i];
```

Tablouri unidimensionale

```
int hours[NO_OF_STUDS];  
int count;
```

```
for (count = 1 ; count <= NO_OF_STUDS ; count++)  
{cout << "Orele petrecute la laborator de  
studentul cu numărul: " << count << ": ";  
cin >> hours[count - 1];  
}
```

NO_OF_STUDS = 6

hours

	hours[0]
	hours[1]
	hours[2]
	hours[3]
	hours[4]
	hours[5]

```
Orele petrecute la laborator de studentul cu numărul: 1: 38  
Orele petrecute la laborator de studentul cu numărul: 2: 42  
Orele petrecute la laborator de studentul cu numărul: 3: 29  
Orele petrecute la laborator de studentul cu numărul: 4: 35  
Orele petrecute la laborator de studentul cu numărul: 5: 38  
Orele petrecute la laborator de studentul cu numărul: 6: 37  
Press any key to continue
```

hours

38	hours[0]
42	hours[1]
29	hours[2]
35	hours[3]
38	hours[4]
37	hours[5]

Tablouri unidimensionale

- Indexarea tablourilor începe de la **0** până la **(dim-1)**
- Dacă se depășește limita tabloului, apar rezultate neașteptate

```
for (count = 1 ; count <=
NO_OF_STUDS ; count++)
{cout << "Orele petrecute ...cu numărul:
" << count << ": ";
cin >> hours[count];
}
```

NO_OF_STUDS = 6

hours	
?	hours[0]
38	hours[1]
42	hours[2]
29	hours[3]
35	hours[4]
38	hours[5]
37

Numele unui tablou

- Numele unui tablou:
 - nume de variabilă;
 - pointer către primul element din tablou:

a echivalent cu **&a[0]**

a+1 echivalent cu **&a[1]**

a+2 echivalent cu **&a[2]**

a+i echivalent cu **&a[i]**

***a** echivalent cu **a[0]**

***(a+1)** echivalent cu **a[1]**

***(a+2)** echivalent cu **a[2]**

***(a+i)** echivalent cu **a[i]**

Inițializarea unui tablou unidimensional

```
int matr[4];
```

```
matr[0] = 6;
```

```
matr[1] = 0;
```

```
matr[2] = 9;
```

```
matr[3] = 6;
```

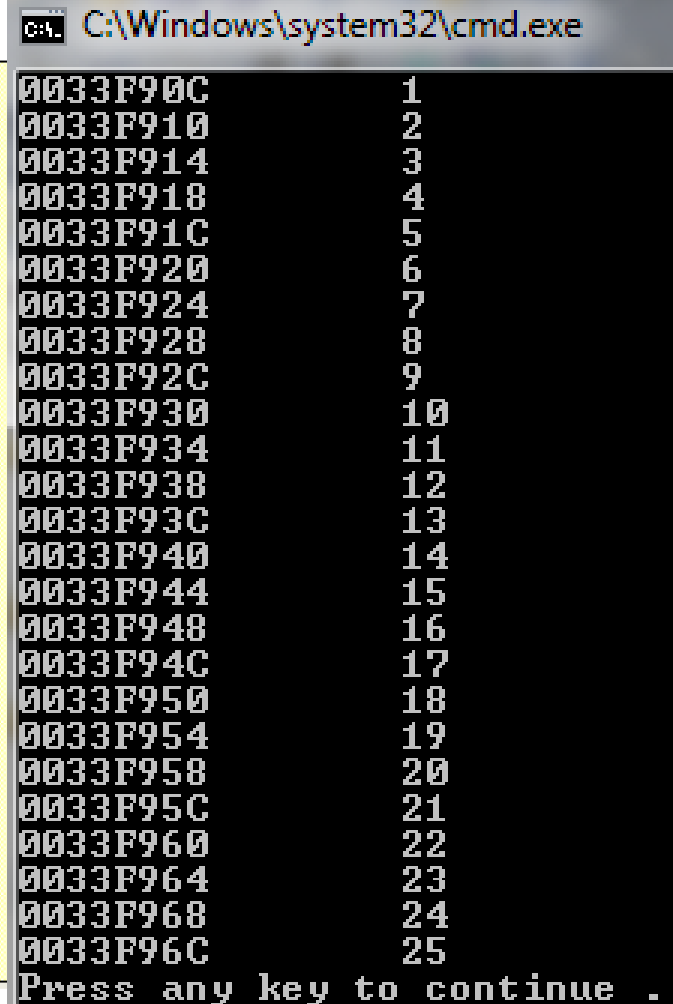
```
int matr[4] = { 6, 0, 9, 6 };
```

```
int matr[] = { 6, 0, 9, 6, 5, -9, 0 };
```

Memorarea unui tablou

```
#include <iostream>
using namespace std;

#define NMAX 25
int main(){
    int a[NMAX], i;
    for (i=0; i<NMAX; i++){
        a[i] = i+1;
        cout << &a[i] << "\t" <<
                *(a+i)<< "\n";
    }
    return 0;
}
```



```
C:\Windows\system32\cmd.exe
0033F90C      1
0033F910      2
0033F914      3
0033F918      4
0033F91C      5
0033F920      6
0033F924      7
0033F928      8
0033F92C      9
0033F930     10
0033F934     11
0033F938     12
0033F93C     13
0033F940     14
0033F944     15
0033F948     16
0033F94C     17
0033F950     18
0033F954     19
0033F958     20
0033F95C     21
0033F960     22
0033F964     23
0033F968     24
0033F96C     25
Press any key to continue .
```

Parcurgerea unui tablou

`/* Varianta 1 */`

```
for (i=0; i < n; ++i)  
    suma += a[i];
```

`/* Varianta 2 */`

```
for (i=0; i < n; ++i)  
    suma += *(a+i);
```

`/* Varianta 3 */`

```
for (p=a; p < &a[n]; ++p)  
    suma += *p;
```

`/* Varianta 4 */`

```
p=a;  
for (i=0; i < n; ++i)  
    suma += p[i];
```

Tablourile ca argumente ale funcțiilor

- Un tablou NU poate fi transmis ÎN ÎNTREGIME ca argument al unei funcții
- Tablourile se transmit ca argumente folosind un pointer la un tablou
- Parametrul formal al unei funcții ce primește un tablou poate fi declarat ca:
 - Pointer
 - Tablou cu dimensiune
 - Tablou fără dimensiune

```
int main(void){  
    int i[4];  
    functia(i);  
    return 0;  
}
```

```
void functia(int *x)
```

```
void functia(int tab[4])
```

```
void functia(int tab[])
```

Tablourile ca argumente ale funcțiilor

```
void insert_sort(int a[], int n)
{
    //...
}
```

```
/* utilizare */
int w[100];
// ...
insert_sort(w, 10);
```

Tablourile ca argumente ale funcțiilor

```
double suma(double a[], int n);  
// double suma(double *a, int n);
```

```
suma(v, 100);  
suma(v, 8);  
suma(&v[4], k-6);  
suma(v+4, k-6);
```

Șiruri de caractere

- Tablouri unidimensionale de tip char
 - Fiecare element al tabloului = un caracter
 - Ultimul caracter al șirului – caracterul nul "\0" = marchează sfârșitul șirului
- `char sir[10];`
 - Declară un șir de 9 caractere, la care se adaugă
 - Al 10-lea caracter = caracterul nul "\0"

Tablouri de char - Șiruri de caractere

- Declarare șiruri:

```
#define MAX_SIR 100 // suficient de mare  
...  
char sirCaract[MAX_SIR];
```

- Declarare cu inițializare:

```
char s[] = "Hi Mom!";  
  
char s[10] = "Hi Mom!";  
  
char s[10] = {'H', 'i', ' ', 'M', 'o', 'm', '!', '\0'};
```

s[0]	s[1]	s[2]	s[3]	s[4]	s[5]	s[6]	s[7]	s[8]	s[9]
H	i		M	o	m	!	\0	?	?

Asignare și comparare la șiruri de caractere

- Asignarea “=” doar la declarare:

```
char sir[10];  
sir = "Hello";// gresit  
  
char sir[10] = "Hello";// OK
```

- Compararea nu e posibilă prin operatorul “==”

```
char sir_unu[10] = "alba";  
char sir_doi[10] = "neagra";  
sir_unu == sir_doi;    // warning: operator has no effect
```

- Se folosește o funcție strcmp

Macroui și funcții pentru șiruri

- In fisierul **<cctype> (ctype.h)** (manipularea caracterelor)
isspace(c), isdigit(c), islower(c), tolower(c), toupper(c), ...

- In fisierul **<cstring> (string.h)** (manipularea șirurilor)

```
char * strcat ( char * destination, const char * source );
```

```
int strcmp(const char *s1,const char*s2);
```

```
char * strcpy ( char * destination, const char * source );
```

```
size_t strlen ( const char * str );
```

```
char mystr[100]="test string";  
sizeof(mystr); // 100  
strlen(mystr); // 11
```

```
const char * strchr ( const char * str, int character );
```

```
char * strchr ( char * str, int character );
```

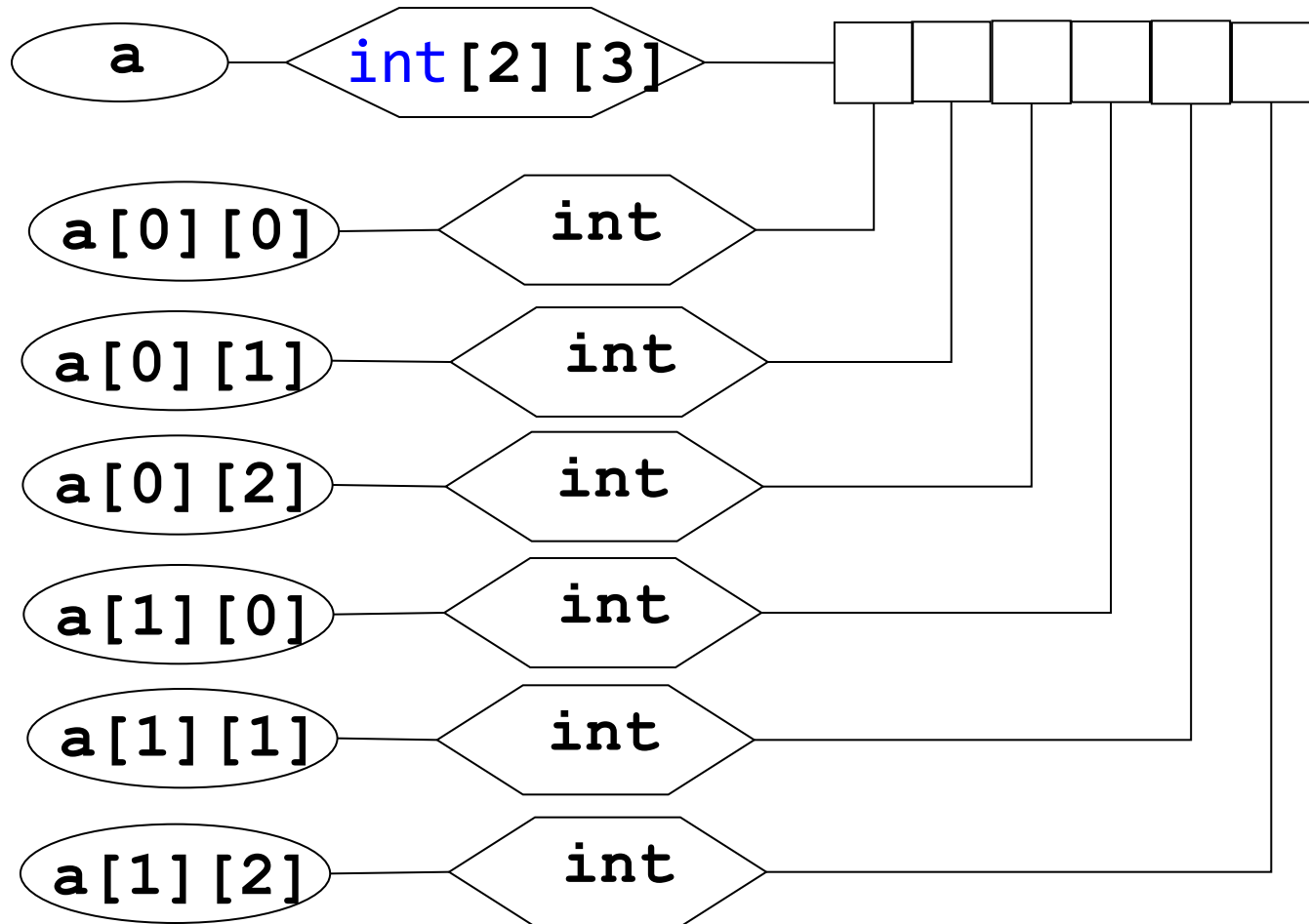
Tablouri bidimensionale

tip numeTablou[*m*][*n*]; *int* a[m][n];

- Memorie contiguă de $m \times n$ locații
- Componentele sunt identificate prin 2 indici:
 - Primul indice are valori $\{0, 1, \dots, m-1\}$
 - Al doilea indice are valori $\{0, 1, \dots, n-1\}$
 - Variabilele componente : $a[0][0], a[0][1], \dots, a[0][n-1],$
 $a[1][0], a[1][1], \dots, a[1][n-1], \dots, a[m-1][0], a[m-1][1], \dots,$
 $a[m-1][n-1]$
- Ordinea de memorare a componentelor este dată de ordinea lexicografică a indicilor

Tablouri bidimensionale

```
int a[2][3];
```



Parcurgerea unui tablou bidimensional

```
double a[MMAX][NMAX]; // declarare tablou bidim.
double suma; // suma elementelor din tablou

/* . . . */

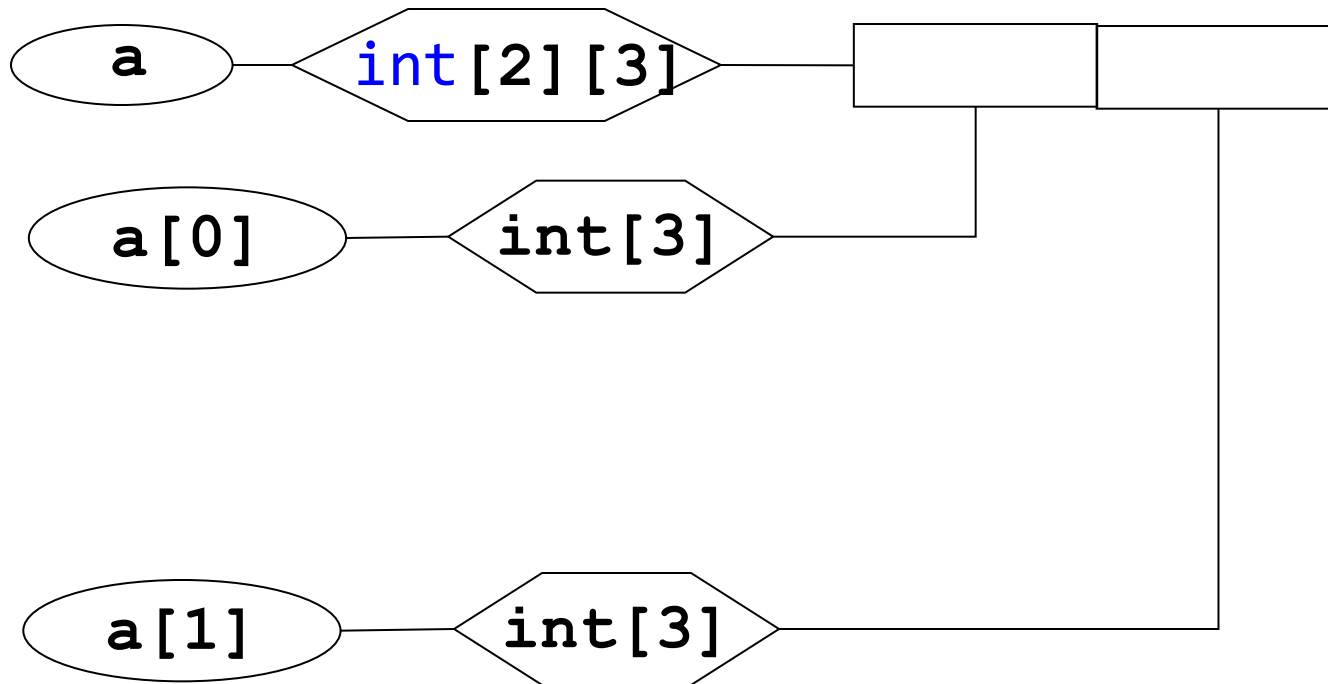
for (i = 0; i < m; i++)
    for (j = 0; j < n; j++)
        cin >> a[i][j];

suma = 0;
for (i = 0; i < m; i++)
    for (j = 0; j < n; j++)
        suma += a[i][j];
```

Tablouri bidimensionale

- Cu analogia de la matrici, un tablou 2-dimensional poate fi privit ca un tablou 1-dimensional în care fiecare componentă este un tablou 1-dimensional.
- Notăție:
 $a[0][0], a[0][1], \dots, a[0][n-1], \dots, a[m-1][0], a[m-1][1], \dots, a[m-1][n-1]$

Tablouri bi-dimensionale văzute ca tablouri uni-dimensionale



Tablouri bidimensionale

	coloana 0	coloana 1	...
linia 0	<code>a[0][0]</code>	<code>a[0][1]</code>	...
linia 1	<code>a[1][0]</code>	<code>a[1][1]</code>	...
...

Expresii echivalente cu `a[i][j]`

`*(a[i] + j)`

`*((*a + i)) + j)`

`(*(a + i))[j]`

`*(&a[0][0] + NMAX*i + j)`

Tablouri bidimensionale ca argumente

```
int minmax(int t[][NMAX], int i0, int j0,  
           int m, int n)  
{  
    //...  
}
```

```
/* utilizare */  
if (minmax(a,i,j,m,n))  
{  
    // ...  
}
```

Inițializarea tablourilor

```
int a[] = {-1, 0, 4, 7};  
/* echivalent cu */  
int a[4] = {-1, 0, 4, 7};
```

```
char s[] = "un sir";           /* echivalent cu */  
char s[7] = {'u', 'n', ' ', 's', 'i', 'r', '\0'};
```

```
int b[2][3] = {1,2,3,4,5,6}    /* echivalent cu */  
int b[2][3] = {{1,2,3},{4,5,6}} /*echivalent cu*/  
int b[][3] = {{1,2,3},{4,5,6}}
```

Pointeri

- Pointer = variabilă care conține o adresă din memorie, adresă care este localizarea unui obiect (de obicei altă variabilă)
- 👍 Oferă posibilitatea modificării argumentelor de apelare a funcțiilor
- 👍 Permit alocarea dinamică
- 👍 Pot îmbunătăți eficiența unor rutine
- 👎 Pointeri neinițializați
- 👎 Pointeri ce conțin valori inadecvate

Pointeri

- Declararea unei variabile pointer:

tip **nume_pointer*;

- *nume_pointer* este o variabilă ce poate avea valori adrese din memorie ce conțin valori cu tipul de bază *tip*.

- Exemple:

```
int *p, i; // int *p; int i;  
p = 0;  
p = NULL;  
p = &i;  
p = (int*) 232;
```

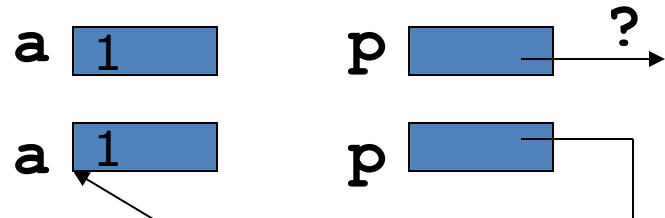
p = &i;

- p “pointează la i”, “conține / primește adresa lui i”, “referențiază la i”.

Pointeri

- Operatorul de dereferențiere (indirectare) ***** :
int *p;
 - **p** este pointer, ***p** este / primește valoarea variabilei ce are adresa p
 - Valoarea directă a lui **p** este adresa unei locații iar ***p** este valoarea indirectă a lui **p**: ceea ce este memorat în locația respectivă

```
int a = 1, *p;  
p = &a;
```



Pointeri

- pentru că memorează adrese, lungimile locațiilor de memorie nu depind de tipul variabilei asociate

`sizeof(int*) = sizeof(double*) = ...`

- afișarea unui pointer:

```
int *px, x = 0, *py, y = 0;  
px = &x;  py = &y;  
cout << "px= " << px << " , py = " << py << endl;
```

C:\Windows\system32\cmd.exe

```
px= 0043F8A4 , py = 0043F88C  
Press any key to continue . .
```

Pointeri

```
int i = 3, j = 5;
int *p = &i, *q = &j, *r;
double x;
```

Expresia	Echivalent	Valoare
<code>p == &i</code>	<code>p == (&i)</code>	002EFEA4
<code>**&p</code>	<code>*(*(&p))</code>	3
<code>r = &x</code>	<code>r = (&x)</code>	eroare!
	(cannot convert from double* to int*)	
<code>3**p/(*q)+2</code>	<code>((3*(*p)))/(*q)+2</code>	3
<code>*(r=&j)*=*p</code>	<code>(* (r=(&j)))*=(*p)</code>	15

Pointeri

```
int *i;      float *f;      void *v;
```

Expresii corecte

`i = 0;`

`i = (int*)1;`

`v = f; i=(int *)v;`

`i = (int*)f;`

`f = (float*)v;`

`*((int*)333);`

Expresii incorecte

`i = 1;`

`v = 1;`

`i = f;`

`&3;`

`&(k+8);`

`*333;`

Pointeri

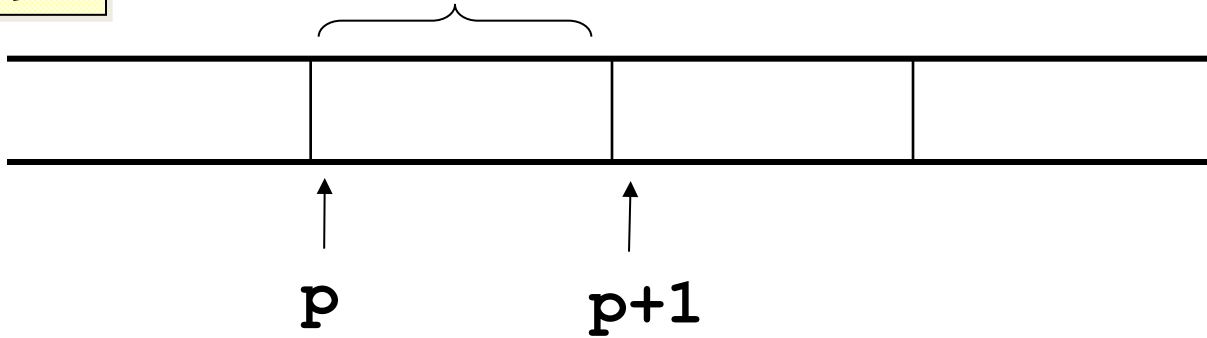
```
#include <iostream>
using namespace std;
int main(void){
    int i=5, *p = &i;
    float *q;
    void *v;
    q = (float*)p;
    v = q;
    cout << "p = " << p << ", *p = " << *p << "\n";
    cout << "q = " << q << ", *q = " << *q << "\n";
    cout << "v = " << v << ", *v = " << *((float*)v)<< "\n";
    return 0;
}
```

```
p = 0030F738, *p = 5
q = 0030F738, *q = 7.00649e-045
v = 0030F738, *v = 7.00649e-045
Press any key to continue . . .
```

Aritmetica pointerilor

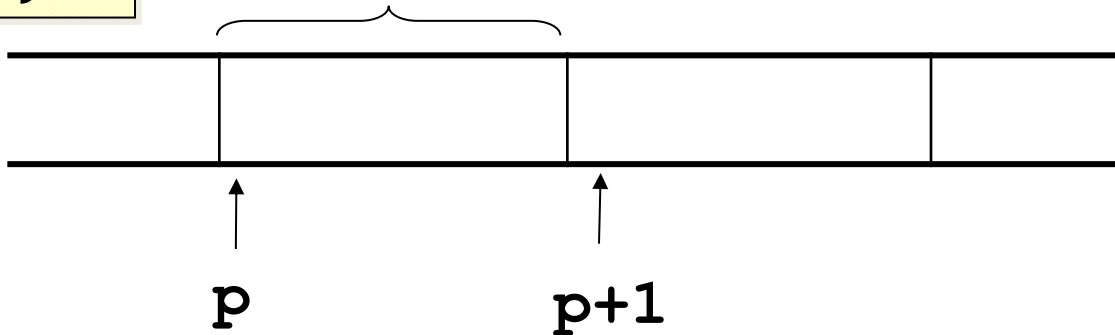
```
int *p;
```

`sizeof(int)`



```
double *p;
```

`sizeof(double)`



Aritmetica pointerilor

```
int a[2], *p1, *q1;
```

```
p1 = a;  
q1 = p1 + 1;  
cout << "q1-p1 " << q1-p1 << endl;  
cout << sizeof(int) << (int)q1 - (int)p1 << endl;
```

$q1 - p1 = 1$

$\text{sizeof}(\text{int}) = 4, (\text{int})q1 - (\text{int})p1 = 4$

Aritmetica pointerilor

```
double c[2], *p3, *q3;
```

```
p3 = c;  
q3 = p3 + 1;  
cout << q3 - p3;  
cout << sizeof(double), (int)q3 - (int)p3;
```

$q3 - p3 = 1$

$\text{sizeof}(\text{double}) = 8, (\text{int})q3 - (\text{int})p3 = 8$