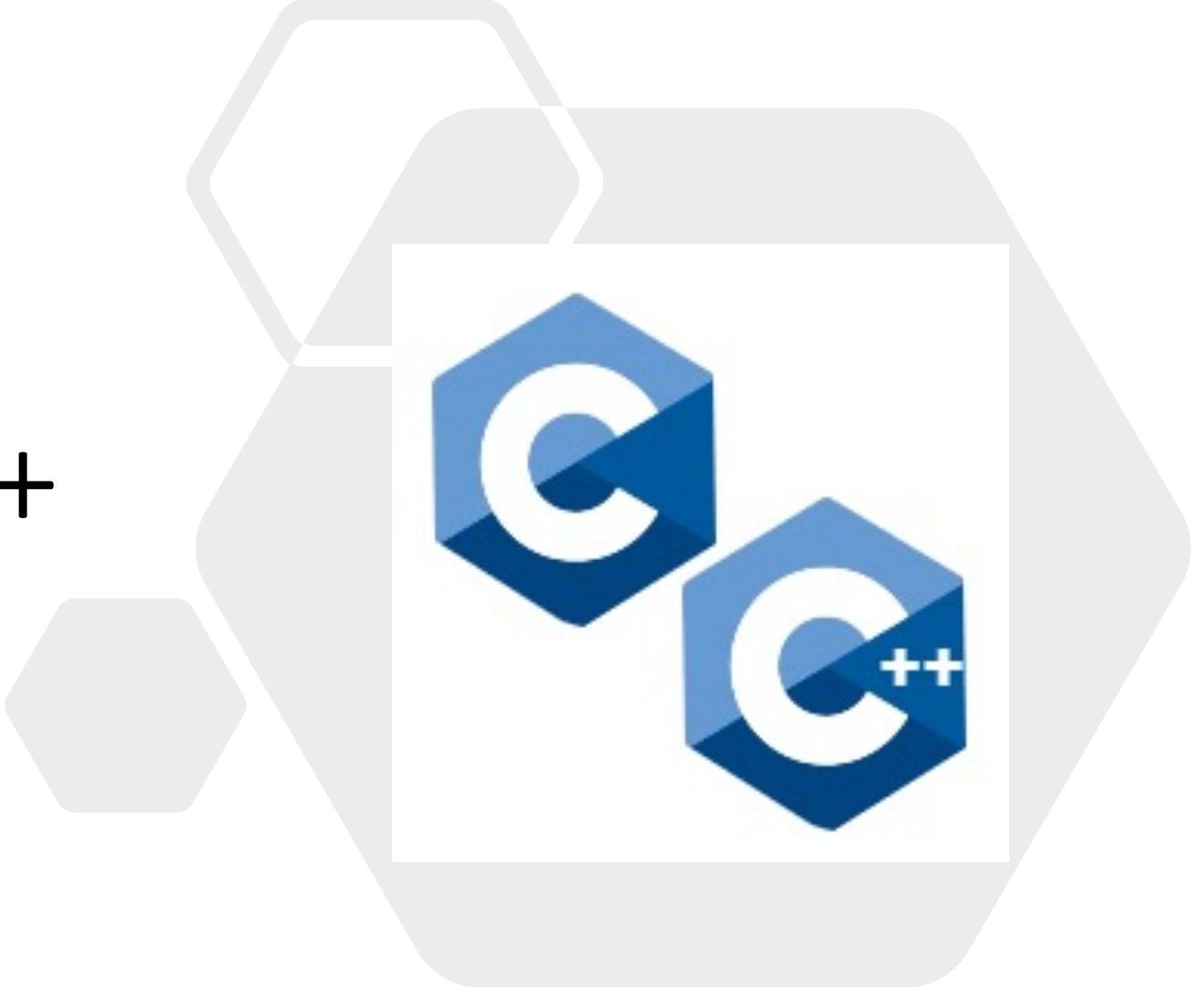


Programare II

Limbajul C/C++

CURS 3-4



Curs anterior

❑ Funcții

❑ Prototip

❑ Definire

❑ Convenția de apel

❑ Macro-definiții

❑ Bibliografie

❑ Kernighan B. and D. Ritchie - The C Programming Language - capitolul 4

Curs curent

- ❑ Pointeri constanți
 - ❑ Tablouri
- ❑ Siruri de caractere
- ❑ Alocarea dinamică a memoriei
- ❑ Pointeri la funcții
- ❑ Bibliografie
 - ❑ Kernighan B. and D. Ritchie - The C Programming Language - capitolul 5

Tablouri de elemente

- ❑ Ce este un tablou (array)?

- ❑ Colecție **indexată** de elemente de **același tip**

- ❑ Indexat

- ❑ Elementele tabloului sunt **numerotate** (începând cu **indexul 0**)
 - ❑ Elementele pot fi accesate prin intermediul indexului

- ❑ Același tip

- ❑ Elementele sunt **stocate** în **locații successive** de memorie de același tip (au aceeași lungime)

Declarare

❑ Declarare

❑ Sintaxă: tip nume [dimensiune] ;

❑ Exemplu

❑ int a[10];
❑ double b[20];

❑ Tipuri

❑ Unidimensionale

❑ int a[5];

Numele tabloului
indică spre
primul element
din tabou

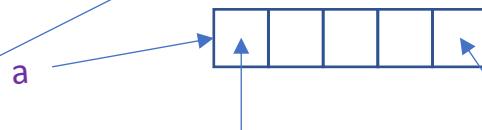
❑ Bidimensionale (matrici)

❑ int b[2][3];

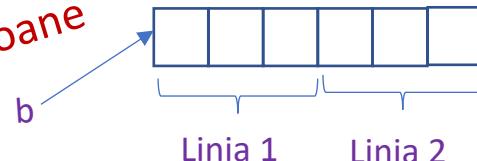
❑ Multidimensionale

❑ int a[4][6][7];

Declare unui tablou cu 5 elemente de tip întreg
Reprezentare în memorie



Declare unui tablou unui tablou bidimensional
cu 2 linii și 3 coloane



Initializare

- Elementele tabloului se specifică între acolade separate prin virgulă
- Exemplu
 - `int a[3] = {3, 2, 1};` Initializarea tabloului a cu trei elemente întregi
 - `double m[2][2] = { {3.2, 4.5}, {3.6, 8} };` Initializarea matrice m cu trei elemente întregi

- `int c[] = { 4, 5, 9, 8 };`

Initializarea tabloului c cu patru elemente întregi
Cum calculăm dimensiunea tabloului?
`int n = sizeof(a)/sizeof(int);`

Returnează dimensiunea unui
obiect în bytes

Accesare elemente

❑ Folosirea operatorului de indexare []

❑ Nu validează corectitudinea indexului

❑ Exemplu

```
int a [] = { 3, 4, 6 }
printf("%d", a[1]);
a[0]=9;
int b = a[2] + 9;
```

Citirea/afişarea unui tablou

Să se definească funcții care citesc elementele unui tablou de n numere întregi și le afișeză

Copierea unui tablou

❑ Tablourile sunt **pointeri constanți**

❑ Nu pot schimba zona de memorie la care referă

❑ Tablourile se copiază element cu element

❑ Exemplu

```
main ()  
{  
    int list1[5];  
    list1={3, 5, 7, 9, 11};  
  
    int list2[5] = {3, 5, 7, 9, 11};  
    list1 = list2;  
}
```

Eroare de compilare: tablourile se pot inițializa
doar în momentul declarării

Eroare de compilare: tablourile nu sunt asignabile

Caractere. Siruri de caractere

❑ Caractere

- ❑ Tipul de date **char**
- ❑ Specificate intre **apostroafe**

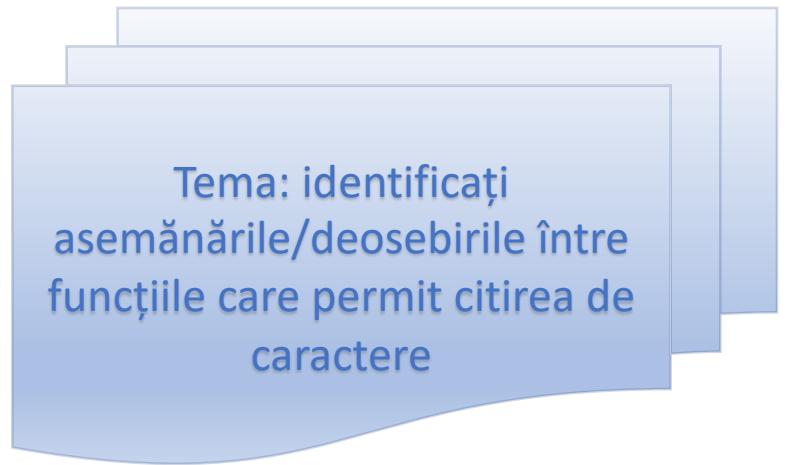
- ❑ Exemplu: `char c='Z' ;`

❑ Citire

- ❑ `scanf ("%c", &c)`
- ❑ `getc ()`
- ❑ `getchar ()`
- ❑ `getch ()`
- ❑ `getche ()`

❑ Scriere

- ❑ `printf ("%c", c);`



Tema: identificați
asemănările/deosebirile între
funcțiile care permit citirea de
caractere

Caractere. Siruri de caractere

- ❑ Sirurile de caractere sunt tip special de tablouri
- ❑ Terminatorul de sir '\0'
 - ❑ Trebuie alocat/rezervat spatiu si pentru el
- ❑ Declarare
 - ❑ `char c[10];`
- ❑ Initializare
 - ❑ `char a[] = "ana are mere";`
 - ❑ `char b[] = {'a', 'n', 'a', '\0'};`
- ❑ Citire
 - ❑ `scanf("%s", c);` //dacă sirul de caractere citit nu conține caractere albe
 - ❑ `gets(c);` //dacă sirul de caractere citit conține caractere albe
- ❑ Scriere
 - ❑ `printf("%s", c);`

Manipularea sirurilor de caractere

- ❑ Biblioteca `string.h` (<http://www.cplusplus.com/reference/cstring/>)
- ❑ Exemple de funcții
 - ❑ Aflarea **lungimii** unui sir de caractere – `strlen(sir)`
 - ❑ Concatenarea a două siruri de caractere – `strcat(to, from)`
 - ❑ Copierea conținutului unui sir în alt sir `strcpy(to, from)`
 - ❑ Compararea conținutului a două siruri de caractere `strcmp(sir1, sir2)`,
`strncmp(sir1, sir2, numar_de_caractere)`
 - ❑ Împărțirea unui sir de caractere în tokenuri – `strtok(sir, delimitator)`
 - ❑ Găsirea unui subșir într-un sir – `strstr(sir, subsir_cautat)`

Caractere. Siruri de caractere

Citiți un sir de caractere de la tastatură și numărați numărul de vocale din sir.

Gestiunea memoriei

❑ Tablouri - pointeri constanți

- ❑ Nu se poate modifica zona de memorie spre care pointează, se poate modifica doar conținutul acestei

❑ Pointeri

- ❑ Se poate modifica zona de memorie spre care pointează, se poate modifica doar conținutul acestei

❑ Provocări în folosirea pointerilor

- ❑ Când folosim un pointer?
- ❑ Când diferențiem un pointer?
- ❑ Când transmitem adresa unei variabile și când transmitem valoarea variabilei ca parametru al unei funcții?
- ❑ Cum folosim aritmetică pointerilor?
- ❑ Cum folosim pointeri astfel încât programul să rămână ușor de înțeles?

Pointeri

- ❑ Un pointer este o **adresă** unde este stocată o dată
 - ❑ Pointeri au tipul datei la care referă
- ❑ Declararea unui pointer este prefixată de caracterul *****
 - ❑ La declararea unui pointer se alocă spațiu pentru a reține adresa zonei de memorie spre care pointează nu se alocă (rezervă) spațiu pentru valoarea variabilei referite de pointer
- ❑ **Deferențierea** – accesarea valorii la care referă pointerul se realizează prin operatorul *****
- ❑ Operatorul **&** - întoarce **adresa de memorie** a unei variabile

Pointeri

```
int    i1;  
int    i2;  
int *ptr1;  
int *ptr2;
```

Declararea a două variabile
de tip pointer la întreg ptr1 și
ptr2

Stiva de apel

0x1010
0x1008
0x1004
0x1000

ptr2:
ptr1:
i2:
i1:

Pointeri

```
int    i1;  
int    i2;  
int *ptr1;  
int *ptr2;  
  
i1 = 1;  
i2 = 2;
```

Declararea a două variabile
de tip pointer la întreg ptr1 și
ptr2

Asignarea de valori pentru
variabile întregi i1 și i2

Stiva de apel

0x1010	ptr2:
0x1008	ptr1:
0x1004	i2: 2
0x1000	i1: 1

Pointeri

```
int i1;  
int i2;  
int *ptr1;  
int *ptr2;
```

```
i1 = 1;  
i2 = 2;
```

```
ptr1 = &i1;  
ptr2 = ptr1;
```

Declararea a două variabile de tip pointer la întreg ptr1 și ptr2

Asignarea de valori pentru variabile întregi i1 și i2

În variabila ptr1 se va stoca adresa de memorie a variabilei i1
În variabila ptr2 se va stoca valoarea variabilei ptr1, adică adresa de memorie a variabilei i1

0x1010
0x1008
0x1004
0x1000

Stiva de apel

ptr2 : 0x1000
ptr1 : 0x1000
i2: 2
i1: 1

Pointeri

```
int    i1;  
int    i2;  
int *ptr1;  
int *ptr2;  
  
i1 = 1;  
i2 = 2;  
  
ptr1 = &i1;  
ptr2 = ptr1;  
  
*ptr1 = 3;  
i2 = *ptr2;
```

Declararea a două variabile de tip pointer la întreg ptr1 și ptr2

Asignarea de valori pentru variabile întregi i1 și i2

În variabila ptr1 se va stoca adresa de memorie a variabilei i1

În variabila ptr2 se va stoca valoarea variabilei ptr1, adică adresa de memorie a variabilei i1

În conținutul variabilei ptr1 (care este i1) se va pune valoare 3 – referențiere

În conținutul variabilei i2 se va pune valoarea dereferentiată de variabila ptr2 adică 3

0x1010
0x1008
0x1004
0x1000

Stiva de apel

ptr2 : 0x1000
ptr1 : 0x1000
i2: 23
i1: 13

ptr2 : 0x1000

ptr1 : 0x1000

i2: 23

i1: 13

Pointerul NULL

- ❑ O constantă specială
 - ❑ Nu pointează spre nimic
 - ❑ Deferențierea ei duce la excepții de tip –**segmentation fault**
 - ❑ Inclusă în bibliotecile <stdlib.h> sau <stdio.h>
- ❑ Exemplu
 - ❑ `int *c = NULL; //pointerul nu referă la o zonă de memorie`

Transmiterea argumentelor prin referință

- ❑ Transmiterea argumentelor

- ❑ Prin valoare

- ❑ Copie a parametrilor actuali

- ❑ Referință

- ❑ Adresa parametrilor actuali

- ❑ Modificările se păstrează la ieșirea din funcție

Transmiterea argumentelor prin referință

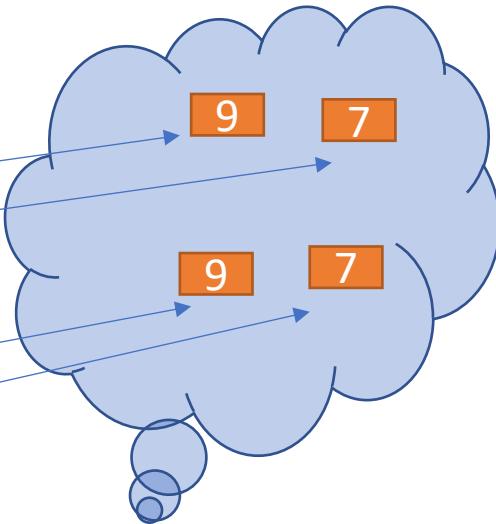
Prin valoare

```
#include <stdio.h>
#include <stdlib.h>

void swap(int, int);

int main()
{
    int x = 9, y =7;
    swap (x, y);
}

void swap(int a, int b)
{
    int aux = a;
    a = b;
    b=aux;
}
```



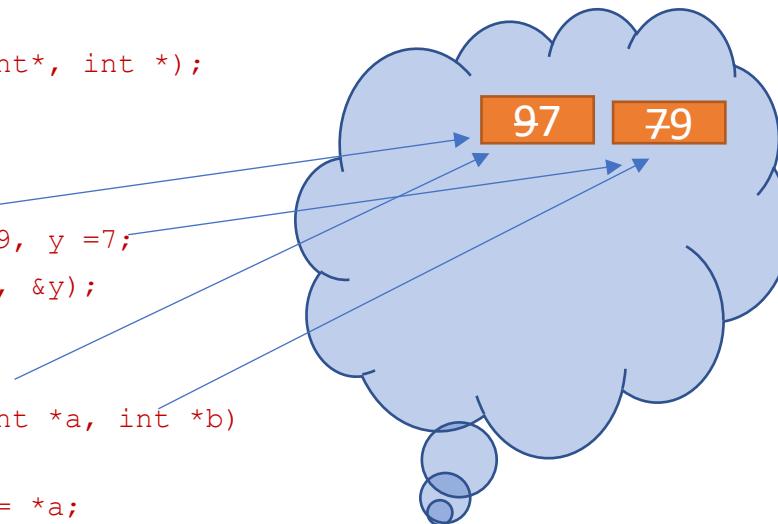
Prin referință

```
#include <stdio.h>
#include <stdlib.h>

void swap(int*, int *);

int main()
{
    int x = 9, y =7;
    swap (&x, &y);
}

void swap(int *a, int *b)
{
    int aux = *a;
    *a = *b;
    *b=aux;
}
```



Transmiterea argumentelor prin referință

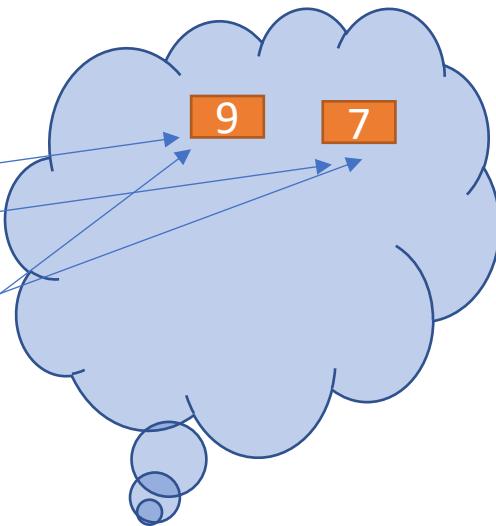
Prin referință

```
#include <stdio.h>
#include <stdlib.h>

void swap(int*, int*);

int main()
{
    int x = 9, y =7;
    swap (&x, &y);
}

void swap(int *a, int *b)
{
    int *aux = a;
    a = b;
    b=aux;
}
```



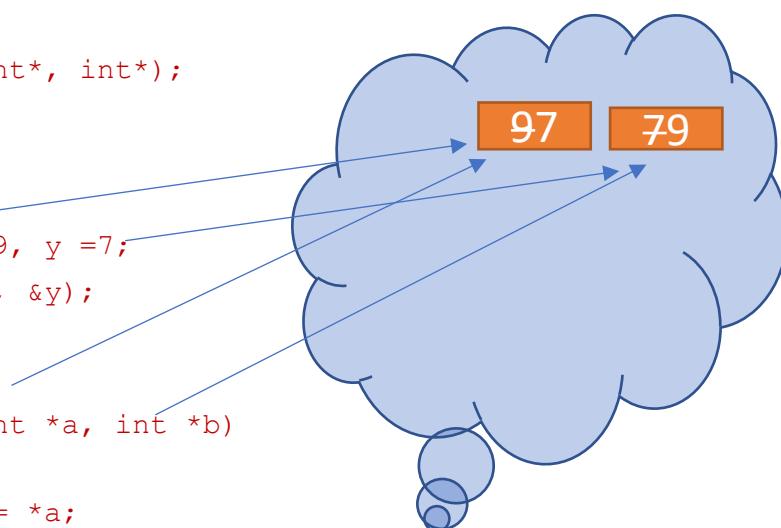
Prin referință

```
#include <stdio.h>
#include <stdlib.h>

void swap(int*, int*);

int main()
{
    int x = 9, y =7;
    swap (&x, &y);
}

void swap(int *a, int *b)
{
    int aux = *a;
    *a = *b;
    *b=aux;
}
```



Pointeri și tablouri

- ❑ Numele unui tablou referă la adresa primului element al tabloului
 - ❑ $a[i]$ echivalent cu $*(a+i)$
- ❑ Transmiterea tablourilor ca parametrii ai funcțiilor
 - ❑ Se trimit ca pointeri => pierderea informației despre lungimea zonei de memorie alocate
- ❑ Observație
 - ❑ La transmiterea parametrilor păstrați tipul argumentelor
 - ❑ Nu schimbați tablouri cu pointeri și invers

La funcții evitați specificarea primei dimensiuni a tabloului astfel încât să poată fi apelată cu tablouri de dimensiune diferită corect în cazul tablourilor multidimensionale trebuie specificate exact următoarele dimensiuni

```
int foo(int array[],  
        unsigned int size)  
{  
    ... array[size - 1] ...  
}           sizeof (array) = ?  
  
int main(void)  
{  
    int a[10], b[5];  
    ... foo(a, 10)... foo(b, 5) ...  
}           sizeof (a) = ?
```

Transmiterea argumentelor prin referință

Scrieti o functie care returneaza valoarea maxima si minima dintr-un tablou.

Aritmetica pointerilor

Pointer + număr

- ❑ Operatori
 - ❑ +, +=, ++

❑ Exemplu

- ❑ Afişarea unui tablou

```
int a[]={1, 3, 5, 6};  
int *p =a; // int *p= &a[0]  
for(int i=0;  
    i<sizeof(a)/sizeof(int);  
    p++, i++)  
printf("%d ", *p);
```

Pointer - număr

- ❑ Operatori
 - ❑ -, -=, --

❑ Exemplu

- ❑ Roarea unui tablou cu un element la dreapta

```
int a[]={1, 3, 5, 6};  
int n = sizeof(a)/sizeof(int)-1;  
int *p = a + n;  
int aux=a[n];  
do{  
    p--; *(p+1) = *(p);  
} while (p!=a);  
*p = aux;
```

Aritmetica pointerilor

Verificați dacă un sir de caractere este palindrom folosind aritmetică pointerilor
Ex: radar cojoc caiac abba aibohphobia tacocat

Pointeri void

- ❑ Un pointer care nu este asociat cu un tip de date
 - ❑ Poate referi la orice tip de date
 - ❑ Se diferențiază doar prin cast

```
int a = 10;  
char b = 'x';
```

```
void *p = &a;  
p = &b;
```

Alocarea dinamică a memoriei

❑ Utilitate?

- ❑ Odată declarat la un tablou (int a[100]) nu se mai poate modifica dimensiunea lui (alocare statică), chiar dacă o parte din elementele tabloului nu mai sunt necesare sau este nevoie de o zonă mai mare de memorie

❑ Alocare memorie

- ❑ void * malloc(int dimensiune_bloc_memorie)
- ❑ void * calloc(int dimensiune_bloc_memorie)
 - ❑ Alocă memorie și o initializează cu zero
- ❑ void *realloc (void *pointer, int dimensiune_bloc_memorie)
 - ❑ Realocă un nou bloc de memorie

❑ Eliberare memorie

- ❑ void free(void *pointer)

Alocarea dinamică a memoriei

Definiți o funcție care primește ca parametru un sir de numere întregi și dimensiunea sirului și returnează un nou sir care conține pătratul numerelor citite în funcție.

Gestionarea memoriei

❑ Stivă

- ❑ Contextul de apel al funcției
 - ❑ Alocare spațiu pentru parametrii și variabile locale
 - ❑ Când se termină apelul memoria alocată se dealocă automat

❑ Heap

- ❑ Folosită pentru date care trebuie să persiste și după terminare apelului unei funcții
 - ❑ Variabile globale
 - ❑ Memorie alocată dinamic

Pointeri la pointeri

- ❑ Pointeri la pointeri sunt variabile care stochează adrese de memorie care referă la locațiile informațiilor
- ❑ Exemplu
 - ❑ Alocare spațiului de memorie pentru o matrice
 - ❑ Statică:

```
int a[10][10];
```
 - ❑ Dinamică:

```
int **p;
p = (int**) malloc(10*sizeof(int*));
for(int i=0; i<10; i++) p[i] = (int*) malloc(sizeof(int)*10);
```

Pointeri la pointeri

- ❑ Folosind alocarea dinamică realizați un program care realizează adunarea a două matrici.

Argumente pe linia de comandă

- ❑ int main(int argc, char ** argv) { ... }
 - ❑ argc
 - ❑ dimensiunea vectorului argc
 - ❑ argv
 - ❑ un tablou de elemente de tip char * (șiruri de caractere)
 - ❑ conține numele programului
- ❑ Exemplu apel
 - ❑ Foo.exe hello 1 2 3

Pointeri la funcții

- Utilitate
 - Modificarea dinamică a funcției apelate
- Sintaxă
 - tip_de_return (*var_pointer) (lista_argumente);
- Exemplu

```
int numarul_de_cifre_pare (int x, int y);
int crescator (int x, int y);
int suma_cifrelor(int x, int y);

int aux;
for(int i=0; i<n; i++) {
    for(int j=0; j<n-1-i; j++)
        if (criteriu(a[j], a[j+1]))
            aux = a[j+1], a[j+1]=a[j], a[j]=aux;
}
```

```
int main() {
    ...
    sort(a, n, &numarul_de_cifre_pare);
    ...
    sort(a, n, &crescator);
    ...
}
```

Despre ce vom discuta?

- ❑ Structuri

- ❑ Uniuni

- ❑ Bibliografie

- ❑ Kernighan B. and D. Ritchie - The C Programming Language - capitolul 6



ÎNTREBĂRI