Array Mono-multi Dimensionali, Struct

Informatica - CdL Triennale in Fisica

Carlo Mereghetti

`Sito del corso:  mereghetti.di.unimi.it/inf`

`Pagina personale: mereghetti.di.unimi.it`

Array monodimensionali

`int X[100]`  `X[0], X[1], ..., X[99]`

ATTENZIONE!

- I limiti degli array non vengono automaticamente controllati!
- Istruzioni tipo `cin >> X[100]` o `cout << X[100]` o `X[100] = 5` non è detto che diano errore
- E' responsabilità del programmatore controllare i limiti degli array

Dichiarazione di array con inizializzazione

- `int X[5] = {1, 2, 3, 4, 5}` → inizializzazione
- `int X[] = {1, 2, 3}` → inizializzazione con dimesionamento
- `int X[5] = {1, 2}` → mette le altre componenti a `0`
- `int X[5] = {0}` → azzeramento veloce di `X`

Gli array vanno usati componente per componente

```
int X[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
int Y[10];
```

- **Copiare** `X` **in** `Y` → `Y = X` **NO!**

```
for ( int i = 0; i < 10; i++ )
   Y[i] = X[i];
```

- **Leggere o stampare** `X` → `cin >> X` **o** `cout << X` **NO!**

```
for ( int i = 0; i < 10; i++ )
   cin >> X[i] o cout << X[i];
```

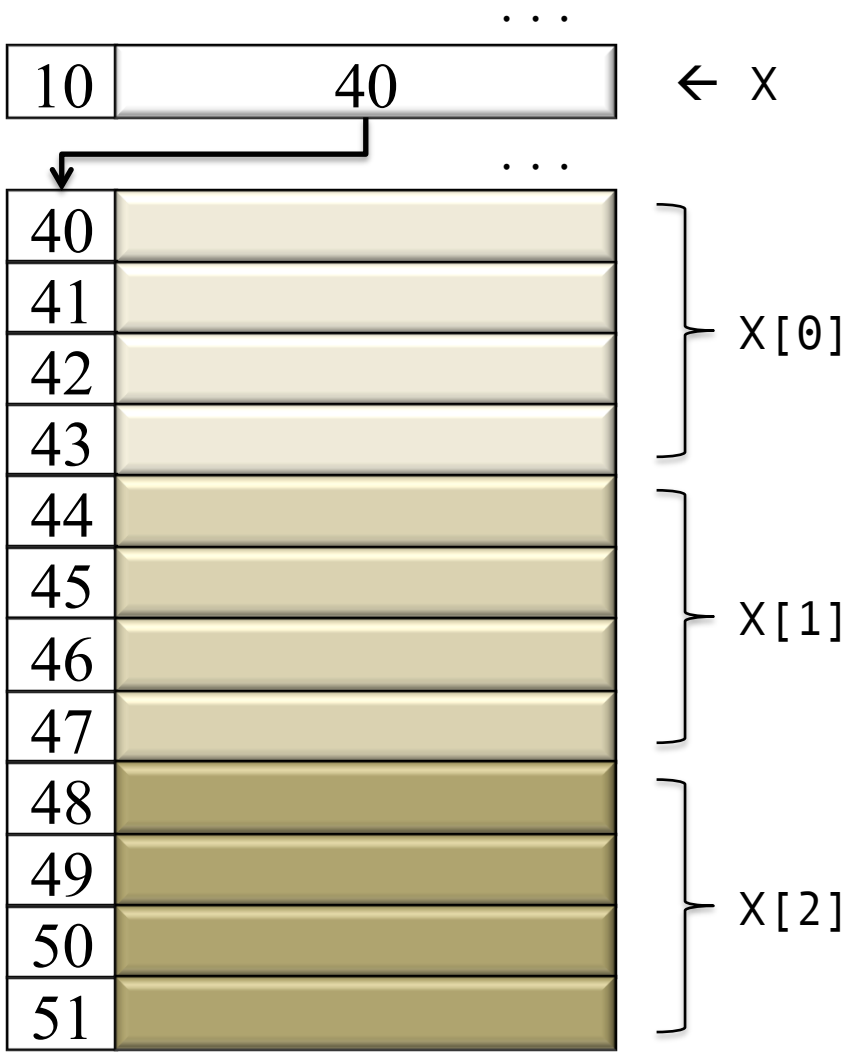
- **Testare l'uguaglianza tra** `X` **e** `Y` → `X == Y` **NO!**

```
bool uguali = true;
for ( int i = 0; i < 10 && uguali; i++ )
   if ( X[i] != Y[i] )
      uguali = false;
```

Allocazione in memoria degli array

`int X[3];` 

- Sequenza di tre variabili `int` contigue: `X[0]`, `X[1]`, `X[2]`
- Una variabile `int` occupa 4 byte



 Nella RAM

X è un *puntatore*: variabile che contiene l'indirizzo di una cella, "punta" alla cella

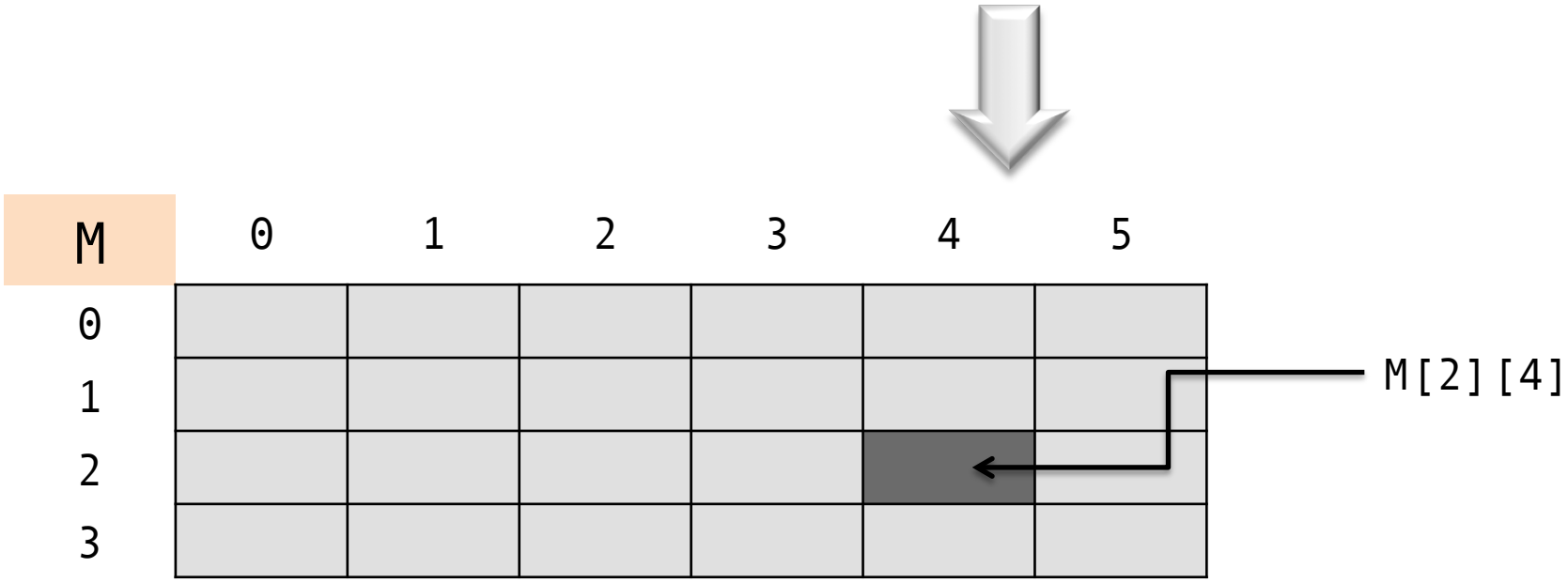
- `cout << X:` stampa `40` in esadecimale (`0x...`)
- `X == Y:` `false`
- `X < Y:` `true/false`

Array multidimensionali

Come memorizzare strutture multidimensionali? Ad esempio, come memorizzare una matrice o una tabella che sono strutture bidimensionali?

Posso usare array di array. Per memorizzare una matrice 4 x 6 di interi definisco la struttura

```
int M[4][6];
```



Array multidimensionali

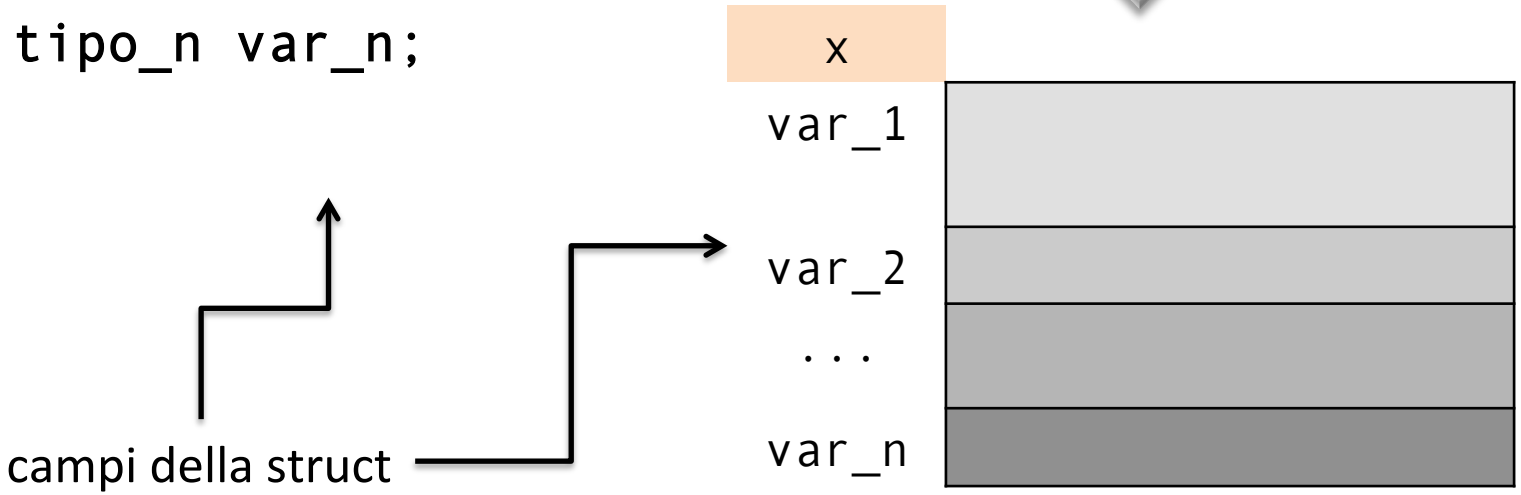
Problema: leggi una matrice 4 x 6 di interi e calcola e stampa la somma riga per riga

```
int M[4][6];
int s;

for ( int i = 0; i < 4, i++ )
   for ( int j = 0; j < 6; j++ )
      cin >> M[i][j];

for ( int i = 0; i < 4, i++ ) {
   s = 0;
   for ( int j = 0; j < 6; j++ )
      s = s + M[i][j];
   cout << "Somma riga " << i << " = " << s << ebdl;
}
```

Aggregare tipi eterogenei: `struct`

```
struct nome_struct {
   tipo_1 var_1;
   tipo_2 var_2;
   ...
   tipo_n var_n;
};
```

```
nome_struct x;
```



- Accesso ai campi: `x.var_1, x.var_2, ..., x.var_n`
- Occupazione di `x`: `dim(tipo_1) + . . . + dim(tipo_n)`
- Ovviamente `nome_struct` può essere usato per definire array

Aggregare oggetti con struttura eterogenea

Problema: Una persona è denotata da nome, cognome ed età. Acquisire un gruppo di 20 persone e stampare nome e cognome delle persone più anziane

```
struct persona {
    string nome;
    string cognome;
    int eta;
};
persona X[20];

for ( int i = 0; i < 20; i++ )
    cin >> X[i].nome >> X[i].cognome >> X[i].eta;

int maxEta = X[0].eta;
for ( int i = 0; i < 20, i++ )
    if ( X[i].eta > maxEta )
        maxEta = X[i].eta;

for ( int i = 0; i < 20; i++ )
    if ( X[i].eta == maxEta )
        cout << X[i].nome << " " << X[i].cognome << endl;
```