



ARRAY E STRINGHE

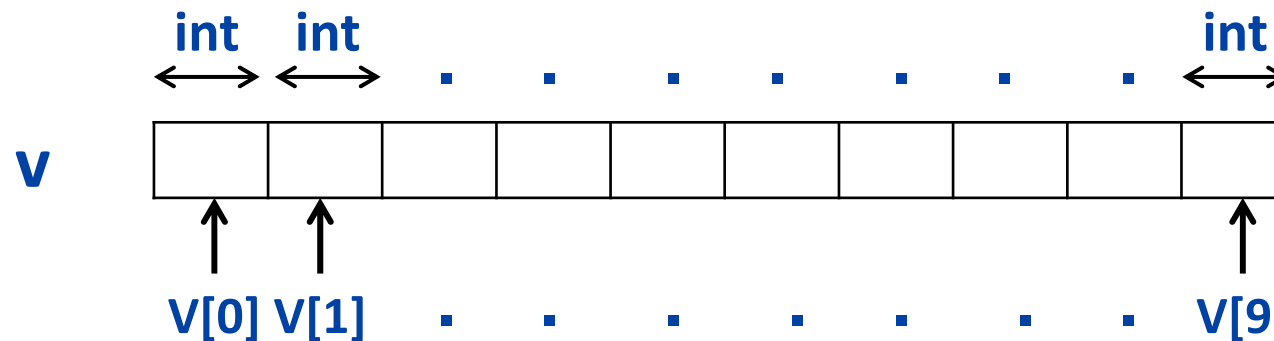
Gli Array nel C++

Roberto Nardone, Luigi Romano

- ☐ Concetto di array
- ☐ Array monodimensionale
- ☐ Array di caratteri
- ☐ Stringhe
- ☐ Array multidimensionale
- ☐ Esercizi

- ❑ Un **array** è una struttura dati complessa la cui nozione è ispirata ai concetti matematici di vettore (array monodimensionale) e matrice (array multidimensionale).
- ❑ Un **array monodimensionale** è una sequenza di variabili dello stesso tipo, allocate in aree contigue della memoria, a cui è possibile accedere con un unico identificativo e un indice intero.
- ❑ Funzioni per l'ordinamento, il calcolo della media e la ricerca, vengono tipicamente applicate su set di variabili memorizzate all'interno di un array.

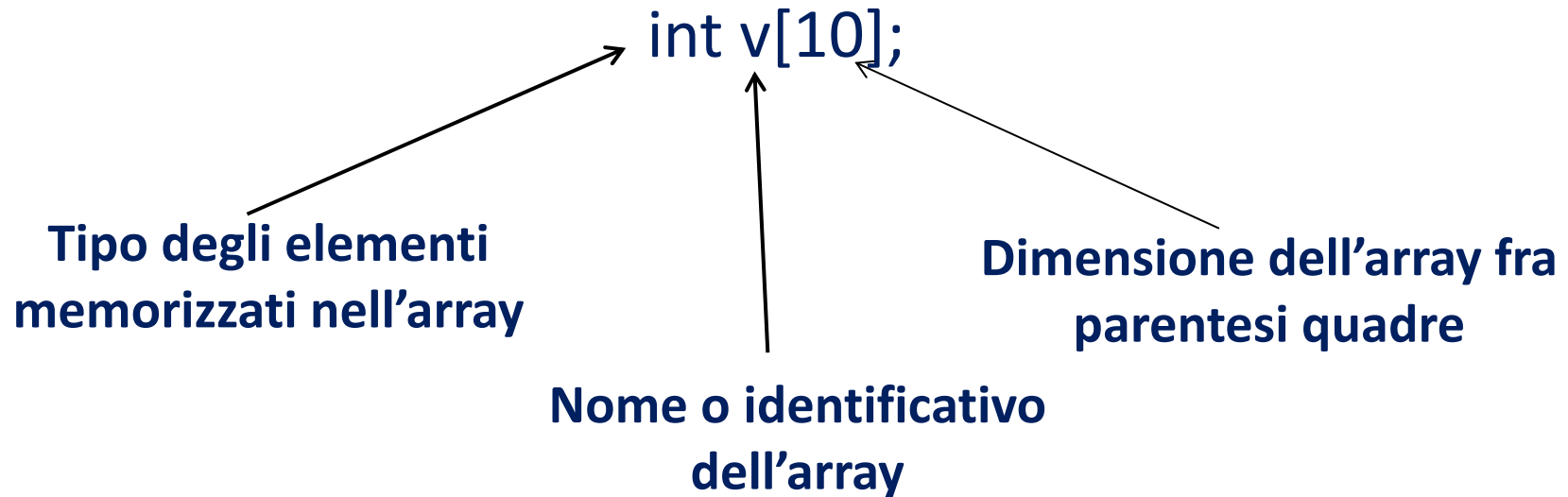
- Un array di nome **v**, in grado di contenere **10** variabili di tipo **int**, può essere rappresentato graficamente come segue:



- Ogni cella rappresenta un **elemento** dell'array, ogni elemento è contrassegnato da un indice intero.
- L'**indice del primo elemento** è **0**, quindi *un array da 10 elementi avrà indici compresi fra 0 e 9*.
- Il **nome dell'array coincide con il suo indirizzo** e con l'indirizzo del suo primo elemento $\Rightarrow v == \&v[0]$

DICHIARAZIONE DI UN ARRAY

- ❑ Come avviene per le variabili, anche gli array per essere utilizzati vanno prima dichiarati.
- ❑ Riprendendo l'esempio grafico della precedente slide, la sintassi per dichiarare l'array **v** sarà:



- ❑ Il campo dimensione all'interno della dichiarazione deve essere un valore costante.
- ❑ Il motivo è legato al fatto che gli elementi di un array vengono allocati in locazioni di memoria fisicamente contigue, quindi il compilatore deve conoscere esattamente quanta memoria serve per memorizzare l'array prima che il programma venga eseguito.
- ❑ IL C++ **non** effettua alcun controllo sugli indici che si utilizzano durante le operazioni su di un array con il rischio di accedere ad aree di memoria del tutto scollegate all'array.

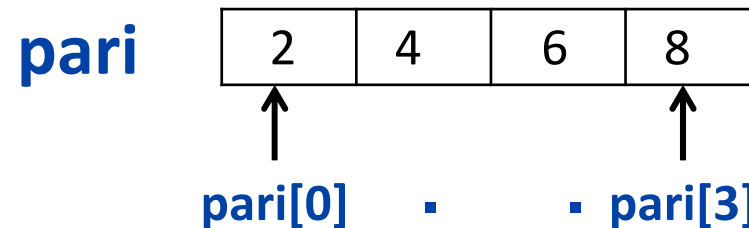
- ❑ Un array può essere inizializzato in due modi:
 - ❑ esplicitamente durante la creazione dell'array;
 - ❑ durante l'esecuzione del programma, assegnando o copiando dati nell'array.

- ❑ Esempi di dichiarazioni esplicite sono riportati di seguito:

```
char vocali[5]={'a', 'e', 'i', 'o', 'u'};  
int pari[4]={2,4,6,8};
```

- ❑ Nell'inizializzazione esplicita i valori degli elementi, che devono essere in numero uguale alla dimensione dell'array, vanno posti tra le parentesi {} e separati dalla virgola.

- ❑ Consideriamo l'array di interi 'pari' inizializzato alla slide precedente:



- ❑ Supponiamo di voler memorizzare l'intero **22** al posto del terzo elemento dell'array, l'assegnazione da effettuare sarà:

pari[2] = 22;

- ❑ Per copiare un elemento di un array in una variabile (dello stesso tipo), l'assegnazione sarà:

int p = pari[1];

- ❑ Per inizializzare un array durante l'esecuzione di un programma si deve accedere ad ogni elemento dell'array e assegnargli un valore.
- ❑ Tipicamente l'accesso agli elementi di un array avviene attraverso l'uso di un costrutto iterativo **for**.

```
const int len = 4;  
int pari[len] ;  
for(int i=0; i<len; i++)  
{  
    pari[i] = (i+1)*2;  
}
```

□ Vediamo ad ogni iterazione l'assegnazione che viene effettuata:

1. $i=0$, accesso alla prima posizione dell'array (`pari[0]`), assegnazione del valore $(i+1)*2 = 2$ a `pari[0]`;
2. $i=1$, accesso alla seconda posizione dell'array (`pari[1]`), assegnazione del valore $(i+1)*2 = 4$ a `pari[1]`;
3. $i=2$, accesso alla terza posizione dell'array (`pari[2]`), assegnazione del valore $(i+1)*2 = 6$ a `pari[2]`;
4. $i=3$, accesso alla quarta posizione dell'array (`pari[3]`), assegnazione del valore $(i+1)*2 = 8$ a `pari[3]`;

- ❑ Cosa accade se accedo ad un indice che eccede la dimensione dell'array?

Il compilatore non ritornerà alcun messaggio di errore, ma il programma farà accesso ad un'area di memoria il cui contenuto è del tutto sconosciuto al programmatore.

- ❑ Definire all'inizio del programma una costante che rappresenta la dimensione dell'array è una valida soluzione per evitare l'accesso ad aree di memoria esterne all'array.

```
#include<iostream>

using namespace std;

int main()
{
    int dispari[5] = {1, 3, 5, 7, 9};

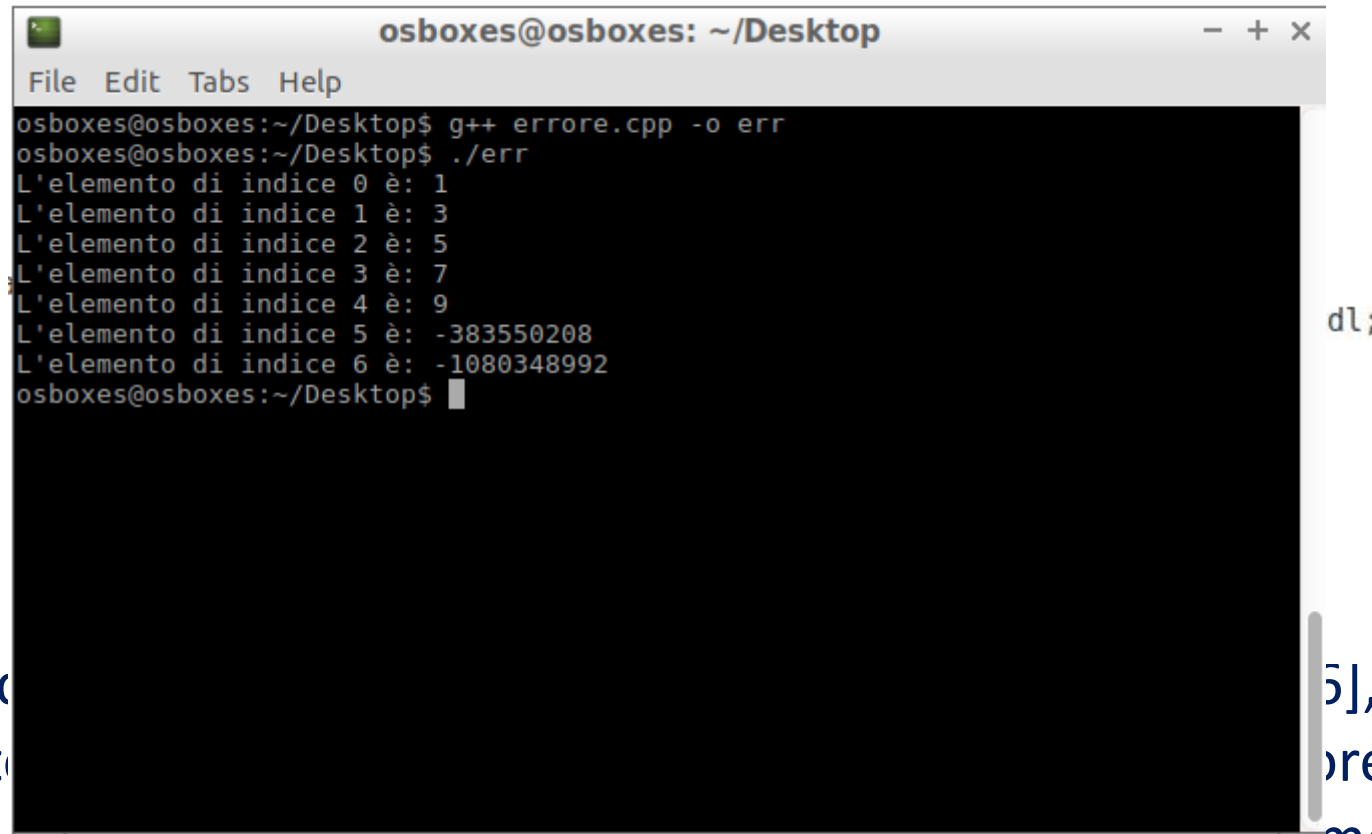
    for (int i=0; i<7; i++)
    {
        cout<<"L'elemento di indice "<<i<<" è: "<<dispari[i]<<endl;
    }
    return 0;
}
```

- ❑ Quando accedo agli elementi `dispari[5]` e `dispari[6]`, sono all'esterno dell'array, il compilatore non ritorna un errore, ma i valori stampati sono quelli contenuti nella locazioni di memoria in quel momento.

ESEMPIO DI «BUFFER OVERFLOW»

13

```
#include<iostream>
```



```
osboxes@osboxes: ~/Desktop
File Edit Tabs Help
osboxes@osboxes:~/Desktop$ g++ errore.cpp -o err
osboxes@osboxes:~/Desktop$ ./err
L'elemento di indice 0 è: 1
L'elemento di indice 1 è: 3
L'elemento di indice 2 è: 5
L'elemento di indice 3 è: 7
L'elemento di indice 4 è: 9
L'elemento di indice 5 è: -383550208
L'elemento di indice 6 è: -1080348992
osboxes@osboxes:~/Desktop$
```

- ❑ Quando l'array è pieno, i valori memorizzati all'esterno dell'array (in quel momento) sono sovrascritti, ma i valori originali rimangono in memoria.



USO DI UNA COSTANTE COME DIMENSIONE DELL'ARRAY

14

```
#include<iostream>

#define array_dimension 5

using namespace std;

int main()
{
    int dispari[5]={1, 3, 5, 7, 9};

    for (int i=0; i<array_dimension; i++)
    {
        cout<<"L'elemento di indice "<<i<<" è: "<<dispari[i]<<endl;
    }

    return 0;
}
```

```
#include<iostream>

using namespace std;

const int array_dimension=5;

int main()
{
    int dispari[5]={1, 3, 5, 7, 9};

    for (int i=0; i<array_dimension; i++)
    {
        cout<<"L'elemento di indice "<<i<<" è: "<<dispari[i]<<endl;
    }

    return 0;
}
```

- ❑ Definita con una costante la dimensione dell'array, e utilizzata questa all'interno del ciclo, si preclude l'accesso all'esterno dell'array.
- ❑ #define VS const int



- ☐ Dato un array di dieci elementi inizializzato da programma, contare il numero di elementi pari nell'array
- ☐ Dato un array di dieci elementi chiedere all'utente i valori da usare per inizializzare l'array e stampare a video l'array finale
- ☐ Dato un array di dieci elementi inseriti dall'utente, stamparlo al contrario
- ☐ Dato un array di dieci elementi inseriti dall'utente, presentare all'utente un menù che chieda l'operazione da eseguire tra: media, somma, stampa

- ❑ Un **array multidimensionale** è un array i cui elementi sono a loro volta un array.
- ❑ La dichiarazione è del tutto analoga a quella di un array monodimensionale, con la differenza che si dovranno specificare tante coppie di parentesi quadre quante sono le dimensioni dell'array multidimensionale.
- ❑ Concettualmente un array multidimensionale è una matrice e per accedere ad un suo elemento devo conoscere gli indici della riga e colonna corrispondenti alla posizione dell'elemento.


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

	Col0	Col1	Col2
Row0	1	2	3
Row1	4	5	6
Row2	7	8	9

- ❑ L'inizializzazione di un array multidimensionale si effettua specificando tra parentesi graffe i valori di ciascuna riga.
- ❑ L'accesso agli elementi dell'array, richiederà tanti cicli for innestati quanti sono gli array monodimensionali che costituiscono l'array multidimensionale.

```
#include<iostream>

const int DIM_1 = 3;
const int DIM_2 = 4;

using namespace std;

int main()
{
    int A[DIM_1][DIM_2];
    int num;

    for(int i=0; i<DIM_1; i++)
    {
        for (int j=0; j<DIM_2; j++)
        {
            cout<<"Inserire elemento ("<<i<<", "<<j<<)"<<endl;
            cin>>num;
            A[i][j]=num;
        }
    }

    //additional code

    return 0;
}
```

ACCESSO AD UN ARRAY M

```
#include<iostream>

const int DIM_1 = 3;
const int DIM_2 = 4;

using namespace std;

int main()
{
    int A[DIM_1][DIM_2];
    int num;

    for(int i=0; i<DIM_1; i++)
    {
        for (int j=0; j<DIM_2; j++)
        {
            cout<<"Inserire elemento ("<<i<<", "<<j<<)"<<endl;
            cin>>num;
            A[i][j]=num;
        }
    }

    //additional code

    return 0;
}
```

```
osboxes@osboxes: ~/Desktop
File Edit Tabs Help
osboxes@osboxes:~/Desktop$ g++ Multi.cpp -o multi
osboxes@osboxes:~/Desktop$ ./multi
Inserire elemento (0, 0)
2
Inserire elemento (0, 1)
3
Inserire elemento (1, 0)
5
Inserire elemento (1, 1)
1
Inserire elemento (2, 0)
3
Inserire elemento (2, 1)
4
osboxes@osboxes:~/Desktop$
```

1. Dato un vettore di 10 interi, scrivere un programma che acquisisca il vettore da terminale, e ne stampi a video somma e media degli elementi.
2. Dato il seguente vettore $v=[1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10]$, scrivere un programma che cerca uno degli elementi scelto dall'utente stampando a video la posizione corrispondente.
3. Inserire un vettore di interi allocato staticamente da tastiera, cercare gli elementi pari stampandoli a video.
4. Scrivere un programma che acquisisca una matrice 3x3 da terminale e stampa a video la somma degli elementi sulla diagonale principale.