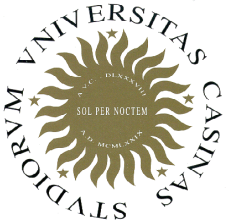


Università degli Studi di Cassino

Corso di Fondamenti di Informatica

Tipi strutturati: Array

Anno Accademico 2011/2012

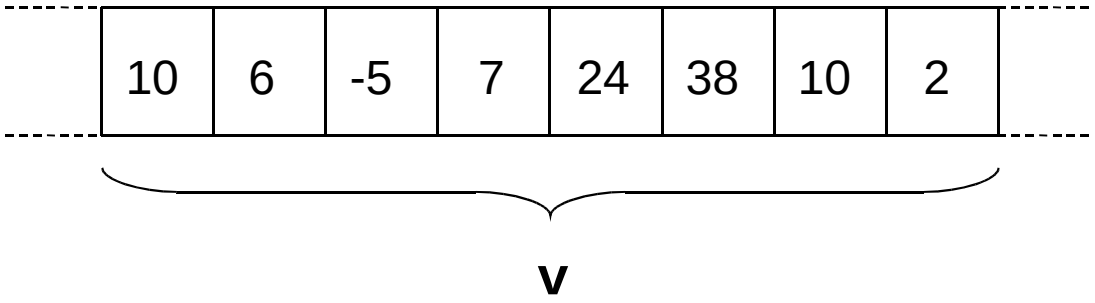
Francesco Tortorella

Gli array

- In alcuni casi, l'informazione che bisogna elaborare consiste di un'aggregazione di valori, piuttosto che di un valore solo.
- Questo significa che sarebbe conveniente indicare l'insieme di valori di interesse con una sola variabile piuttosto che con tante variabili quante sono i valori da considerare: una variabile di *tipo strutturato*.
- In C++ (come nella maggior parte dei linguaggi), questa possibilità è offerta dagli ***array***.

Gli array

- Un array è un insieme di variabili, tutte dello stesso tipo, identificato da un nome unico. Gli elementi dell'array sono disposti in memoria in posizioni consecutive.



Definizione di un array

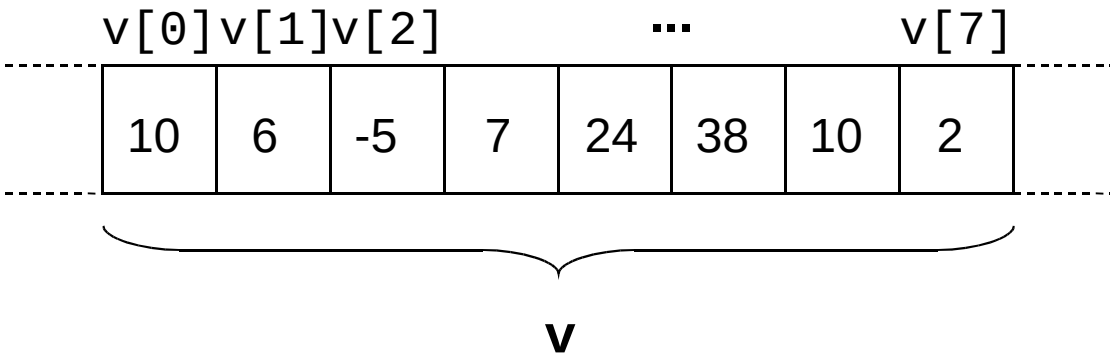
- Per definire una variabile array, è necessario specificare:
  - il nome della variabile array
  - il tipo degli elementi
  - il numero degli elementi presenti (cardinalità dell'array)

Esempio

- Definizione di una variabile array **v** contenente 20 interi:
  `int v[20];`
- Definizione di una variabile array **w** contenente 10 float:
  `float w[10];`

Accesso agli elementi dell'array

- Per accedere ai singoli elementi di un array, è necessario specificare il nome della variabile array e la posizione dell'elemento di interesse tramite un valore intero (variabile o costante) che si definisce *indice.*



Accesso agli elementi dell'array

- Si noti che l'indice parte da 0; quindi **`v[0]`** sarà il primo valore dell'array, mentre l'N-mo sarà **`v[N-1]`.**
- Va quindi ricordato che, se si definisce un array con N elementi, l'indice dovrà essere limitato tra 0 ed N-1.
- Questo controllo è a cura dell'utente, in quanto non ci sono controlli automatici della correttezza dell'indice. Nel caso si considera un indice errato (es. **`v[N]`**), sarà effettuato un accesso ad una zone della memoria che non appartiene all'array, con effetti imprevedibili a runtime.

Accesso agli elementi dell'array

- Ogni elemento di un array è, a tutti gli effetti, una variabile del tipo costituente l'array e quindi può essere impiegato come tale

```
int a,b,i;
int v[10];

v[2]=3;
v[7]=0;
cout << "Valore: " << v[7];
i=2;
a=v[i]*4+6;
b=v[i+5];
```

Assegnazione tra array

- Diversamente dalle variabili di tipo atomico, non è possibile fare assegnazioni dirette tra array.
- L'unica possibilità per assegnare i valori degli elementi di un array agli elementi di un altro array è quindi fare una serie di assegnazioni tra elementi corrispondenti:

```
int a[]={7,9,6,3};
int b[4];
```

```
b=a;
```

errata

```
b[0]=a[0];
b[1]=a[1];
b[2]=a[2];
b[3]=a[3];
```

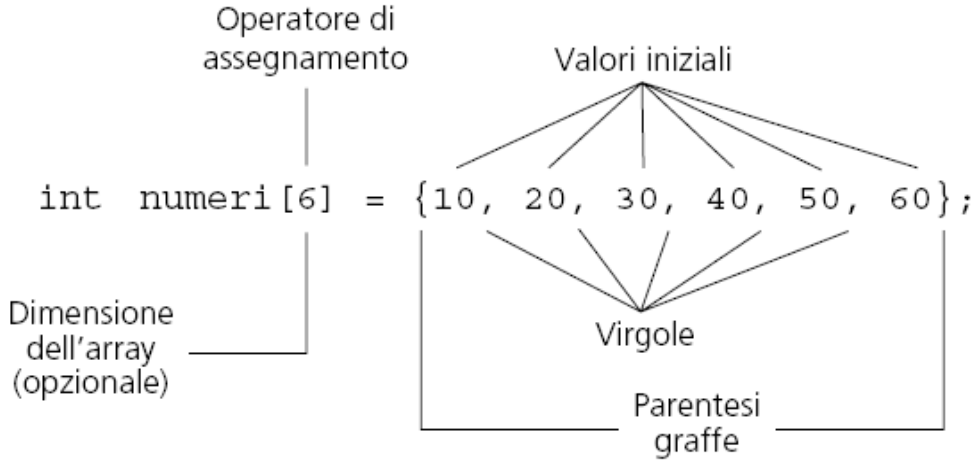
Inizializzazione di un array

- Un array può essere inizializzato in fase di definizione:

```
int numeri[6] = {10,20,30,40,50,60};
```

- La dimensione dell'array può essere anche implicita:

```
int numeri[] = {10,20,30,40,50,60};
```



Lettura e stampa degli elementi di un array

- Per inizializzare da input una variabile array, è necessario realizzare un'operazione di input per ciascuno degli elementi
- Analogamente, per stampare il contenuto di un array, è necessario fare la stampa di ognuno degli elementi.
- Qual è il costrutto da utilizzare ?
- Problema:
  - leggere da input la dimensione e gli elementi di un array e stampare il risultato della lettura

Array come parametri

- Gli array sono passati unicamente per riferimento (perché ?).
- Come parametro formale, un array si indica con tipo, identificatore e parentesi quadre: **`int vet[]`**. Non si inserisce l'&.
- Come parametro effettivo va fornito il nome dell'array, senza specificare altro.

Array come parametri

```
void stamparray (int vet[], int num) {
  for (int i=0; i<num; i++)
    cout << vet[i] << " ";
  cout << "\n";
}

int main ()
{
  int vet1[] = {5, 10, 15};
  int vet2[] = {2, 4, 6, 8, 10};

  stamparray(vet1,3);
  stamparray(vet2,5);

  return (EXIT_SUCCESS);
}
```

Array come parametri: errori frequenti

```
void stamparray (int vet[], int num) {
  for (int i=0; i<num; i++)
    cout << vet[i] << " ";
  cout << "\n";
}

int main ()
{
  int vet1[] = {5, 10, 15};
  int vet2[] = {2, 4, 6, 8, 10};

  stamparray(vet1[3],3);
  stamparray(vet2,5);

  return (EXIT_SUCCESS);
}
```

ACHTUNG !!

Si passa un intero e non un array

korrekt !!

Problema: ricerca del massimo

- Leggere da input la dimensione e gli elementi di un array; fornire in uscita valore e posizione dell'elemento di valore massimo.
- Definire opportuni sottoprogrammi.

Problema: ricerca del massimo e del minimo

- Leggere da input la dimensione e gli elementi di un array; fornire in uscita valori e posizioni dell'elemento di valore massimo e dell'elemento di valore minimo.
- Definire opportuni sottoprogrammi:
  - Quanti ?
  - Quali ?

Problema: calcolo della media e della deviazione standard

- Leggere da input la dimensione e gli elementi di un array; fornire in uscita la media μ e la deviazione standard σ degli elementi presenti nell'array, dove:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(v_i - \mu)^2}$$

Array bidimensionali

- Finora abbiamo considerato *array monodimensionali*, i quali richiedono un solo indice per l'individuazione di un elemento.
- Il C++ permette di definire anche *array bidimensionali*, in cui l'organizzazione degli elementi è di tipo matriciale.
- In questo caso, sono necessari due indici per identificare un elemento nell'array.
- Questo tipo strutturato permette di affrontare tutte quelle situazioni in cui è necessario lavorare con matrici, tabelle, ecc.

Definizione di un array bidimensionale

- Per definire un array bidimensionale, è necessario specificare:
  - il nome della variabile array
  - il tipo degli elementi
  - il numero degli elementi presenti nelle due dimensioni (cardinalità di riga e cardinalità di colonna dell'array)

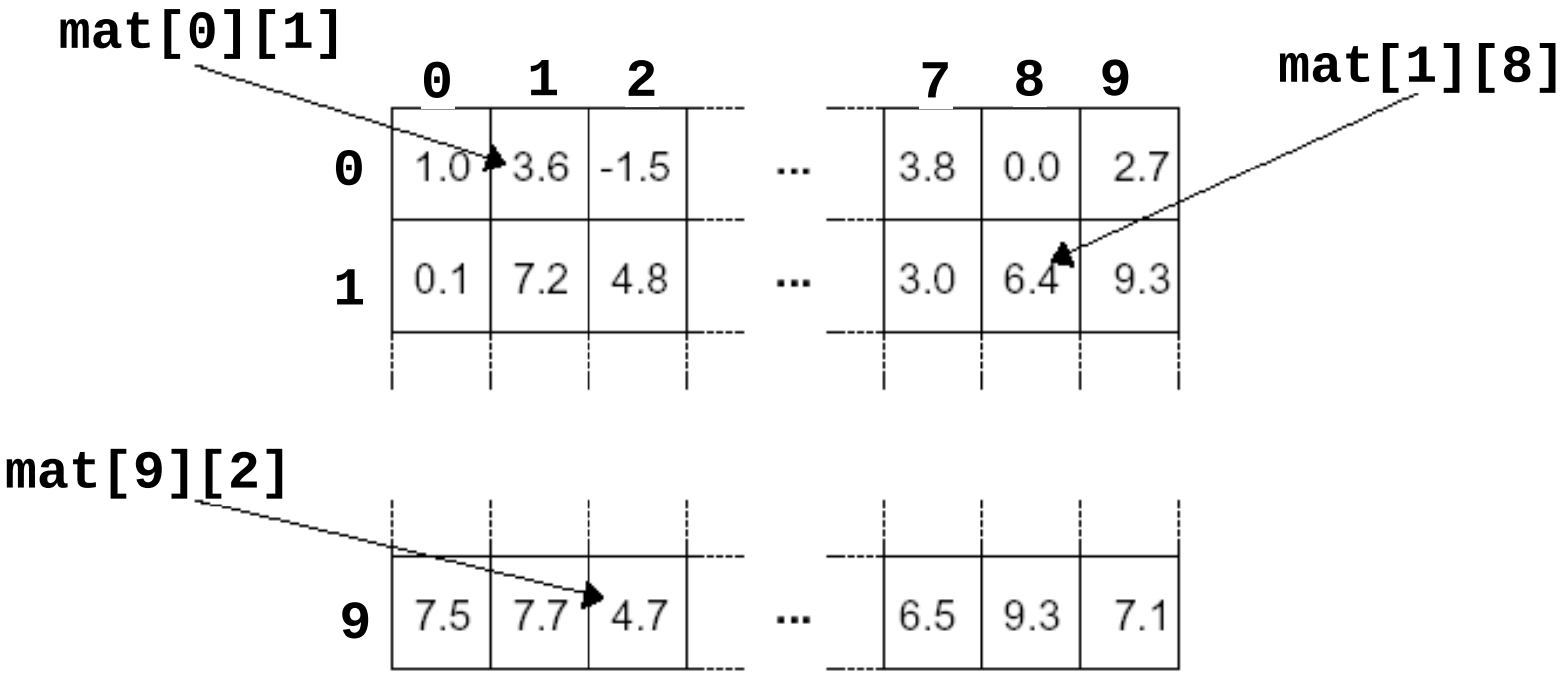
Esempio

- Definizione di una variabile array `mat` contenente 10x10 elementi `double`:

```
double mat[10][10];
```

  - Che differenza c'è rispetto ad un array monodimensionale di 100 elementi?

Organizzazione di un array bidimensionale



Accesso agli elementi dell'array

- Per accedere ai singoli elementi di un array bidimensionale, è necessario specificare il nome della variabile array e gli indici di riga e di colonna che individuano l'elemento desiderato.
- Esempi:

```
mat[2][1]=3;
cout<<"il valore è: " << mat[2][7];
i=3;
j=5;
x=mat[i][j]*4+6;
```

Inizializzazione di un array bidimensionale

- Un array bidimensionale può essere inizializzato in fase di definizione:

```
int mat[2][3] = {{10,20,30},{40,50,60}};
```

- Altre inizializzazioni equivalenti:

```
int mat[2][3] = {10,20,30,40,50,60};

int mat[][3] = {{10,20,30},{40,50,60}};
```

Lettura e stampa degli elementi di un array bidimensionale

- Anche nel caso bidimensionale, l'inizializzazione da input di una variabile array va realizzata realizzare tramite un'operazione di input per ciascuno degli elementi
- Analogamente, per stampare il contenuto di un array, è necessario fare la stampa di ognuno degli elementi.
- Qual è il costrutto da utilizzare ?
- Esempio:
  - leggere da input le dimensioni e gli elementi di un array bidimensionale e stampare il risultato della lettura

Array multidimensionali

- Il C++ permette la definizione di array multidimensionali con più di due indici:
  `int mat3[5][10][5];`
- Con le dovute modifiche valgono le considerazioni sulla definizione, inizializzazione, assegnazione, accesso fatte per gli array monodimensionali e bidimensionali.

Array multidimensionali come parametri

- A differenza degli array monodimensionali, nella definizione come parametro formale di un array N-dimensionale vanno specificate almeno N-1 dimensioni (dalla seconda in poi).
- Esempio (array bidimensionale):

```
float det(float mat[][20], int rig, int col);
…
d=det(matrix,num_rig,num_col);
```