I Vettori (Array)

Prof. Francesco Accarino
IIS Altiero Spinelli Sesto San Giovanni

Variabili

- I tipi di variabili che abbiamo usato finora hanno la caratteristica comune di non essere strutturate: ogni variabile è una singola cella di memoria.
- Se il programma deve trattare collezioni di dati, anche se sono dello stesso tipo, a ognuno deve essere associato un identificatore.
- Supponendo di dover gestire le paghe in una ditta di 3000 dipendenti sarebbe necessario definire 3000 variabili diverse, del tipo: *operaio1*, *operaio2*, ..., *impiegato1*, *impiegato2*, ..., ecc.

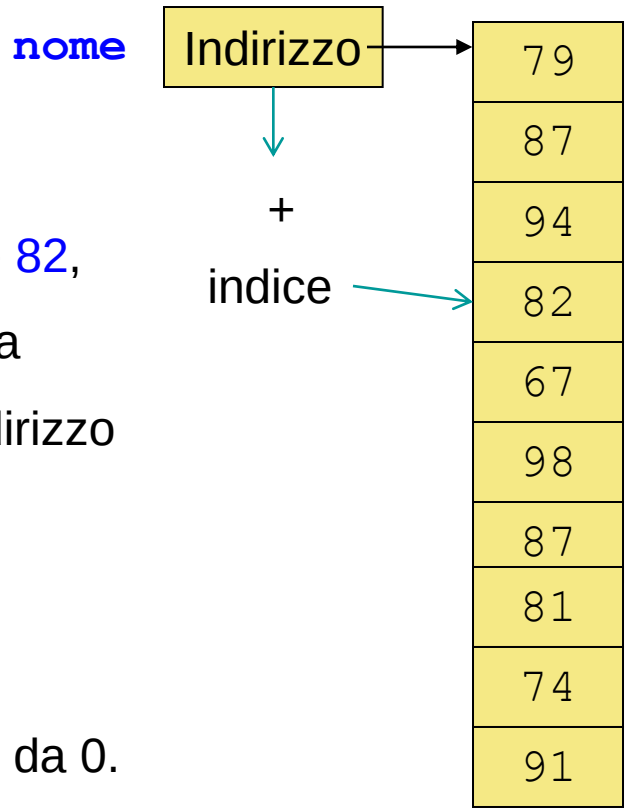
Vettori

- Il *vettore* è una collezione di variabili tutte dello stesso tipo di lunghezza prefissata.

- Ogni elemento del vettore è detto *componente* ed è individuato dal nome del vettore seguito da un *indice* posto tra parentesi quadre.

- L'*indice* può essere solo di tipo *intero* o *enumerato* e determina la posizione dell'elemento nel vettore. Il primo elemento del vettore ha sempre indice 0

Vettori



Int nome[10];

Scrivendo nome[3] facciamo riferimento al valore 82, perché in pratica il compilatore traduce la scrittura nome[3] mediante l'operazione di somma tra l'indirizzo associato a nome e l'indice 3

Ed è per questo motivo che l'indice parte sempre da 0.

Indirizzo + 0 = prima cella

Vettori

- Un array viene dichiarato scrivendo, nell'ordine, il **tipo degli elementi**, il **nome** dell'array, e la sua **dimensione.**

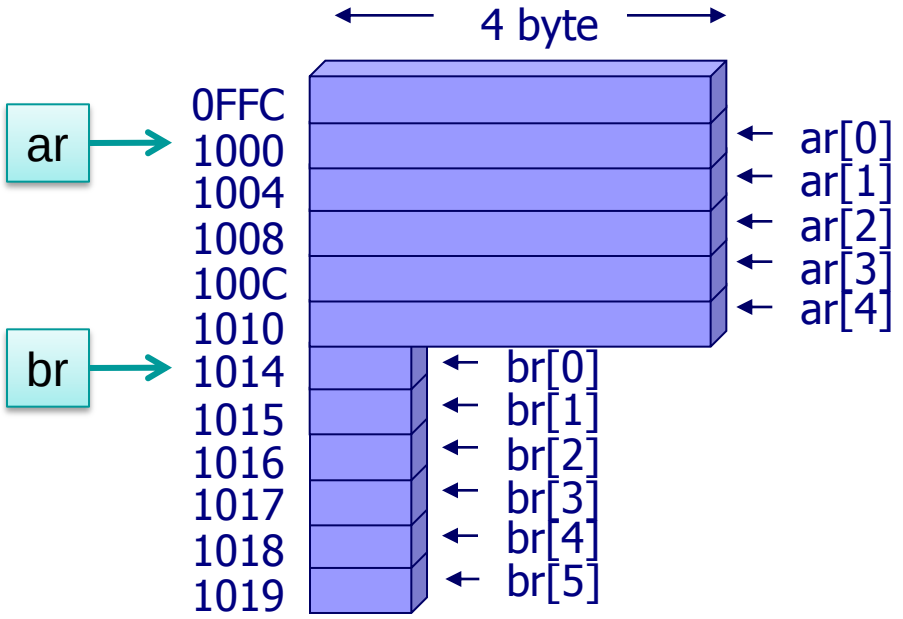
tipo nome [dimensione];

- dove **tipo** è il tipo degli elementi (**int**, **float** …) detto anche tipo base dell'array , **nome** è un identificatore e **dimensione**, che deve essere racchiuso tra parentesi quadre **[ ]**, è la dimensione, ossia il numero di elementi, dell'array. La dimensione viene anche chiamata Numerosità e deve essere un valore costante in quanto gli array sono blocchi di memoria di dimensione prefissata

Vettori

Si noti che all'interno delle parentesi quadre **non è possibile**, in fase di dichiarazione dell'array, utilizzare **nomi di variabili** .
E' possibile, per specificare le dimensioni di un array, usare delle **costanti definite**, come ad esempio:

```
#define UNO_MAX 5
#define DUE_MAX 6
……..
int  ar[UNO_MAX];

char br[DUE_MAX];
```



Questo ritorna utile in fase di progettazione perchè si testa il programma con una numerosità piccola e solo quando funziona si imposta la numerosità effettiva cambiando solo il valore della costante nella define.

Vettori

Come per le variabili semplici, anche per gli **array** è possibile specificare un valore iniziale. Ad esempio, con la dichiarazione:

int pippo [5] = { 16, 2, 77, 40, 12071 };

l'array viene inizializzato come segue:

pippo [0]=16;
pippo [1]=2;
pippo [2]=77;
pippo [3]=40;
pippo [4]=12071;

Il numero di valori usati per l'inizializzazione (quelli posti tra le parentesi grafe **{ }**) deve essere esattamente uguale alla dimensione dell'array. In C++ è possibile anche usare la notazione:

int pippo [ ] = { 16, 2, 77, 40, 12071 };

ed in questo caso viene assunto implicitamente come **dimensione dell'array** il **numero di valori della lista** di inizializzazione

Vettori

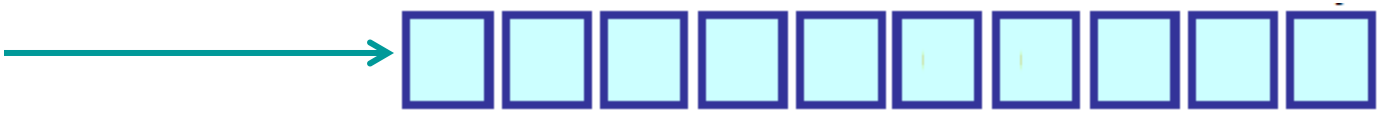
- altri esempi di inizializzazione esplicita al momento della creazione:

```
int numeri[3] = {12, 0, 4};

char vocali[5] = {'a','e','i','o','u'};

float decimali[2] = {1.329, 3.34};
```

- Per inizializzare un array durante l'esecuzione del programma occorre accedere, generalmente con un **ciclo**, ad ogni elemento dell'array stesso ed assegnargli un valore.

Vettori

int a[10];



Int N;

Leggi N


```
for (i=0; i <= N; i++)
      scanf("%d",&a[i]);
```

In **C non è possibile assegnare un array ad un altro.**

```
int a[10],b[10];
a=b; /* ERRORE!!! */
```

Per trasferire il contenuto di un array in un altro è necessario assegnare individualmente ogni valore per esempio utilizzando un ciclo **for.**

```
for (i=0; i <= 10; i++)
      a[i]=b[i]
```

Passaggio di Arrays alle funzioni

- Il passaggio di un array ad una funzione avviene solo per indirizzo perché in questo modo si passa un solo dato, cioè l'indirizzo e non tutta la collezione di dati contenuti nel vettore (ipotesi assurda)
- Quindi quando si passa un vettore come parametro ad una funzione, in realta` si sta passando l'indirizzo dell'elemento di indice 0.
- Questo indirizzo è rappresentato dal nome del vettore
- Il parametro formale deve essere di tipo puntatore al tipo degli elementi del vettore. Di solito si passa la dimensione del vettore in un ulteriore parametro.

Esempio

```
Void stampa(double *, int)

int main(void)
{
 ...
double vet[5] = {1.1, 2.2, 3.3, 4.4, 5.5};

stampa(vet, 5);
...
}

void stampa(double *v, int dim)
 {
int i;
for (i = 0; i < dim; i++)
printf("%lf: %g\n", i, *(v+i));
}
```

Prototipo della funzione

Dichiarazione ed inizializzazione

Chiamata alla funzione

Testata della funzione

Corpo della funzione

Esempio notazione esplicita

- Per evidenziare che il parametro formale e` in realtà un vettore (ovvero l'indirizzo dell'elemento di indice 0), di solito si usa la notazione:

  nome-parametro[] invece di *nome-parametro.

```
Void stampa(double v[ ], int)

int main(void)
{
 ...
double vet[5] = {1.1, 2.2, 3.3, 4.4, 5.5};

stampa(vet, 5);
...
}

void stampa(double v[ ], int dim)
 {
int i;
for (i = 0; i < dim; i++)
printf("%lf: %g\n", i, v[i]);
}
```

Prototipo della funzione

Dichiarazione ed inizializzazione

Chiamata alla funzione

Testata della funzione

Corpo della funzione

Notazione Esplicita

- Si può anche specificare la dimensione nel parametro, ma viene ignorata.

Esempio: void stampa(int v[5], int dim) { ... }

- Nel prototipo della funzione può anche mancare il nome del vettore.

Esempio: void stampa(int [], int);