

Programmazione e Laboratorio di Programmazione

Lezione VIII

I vettori

Vettori

- **Vettore (monodimensionale) di n elementi:**
definisce una corrispondenza biunivoca tra un multi-insieme omogeneo di n elementi e l'insieme di interi $\{0, 1, \dots, n-1\}$
- **Esempio:**

Vettore di 5 interi

0	5
1	-1
2	32
3	-4
4	27

Operatore sizeof()

- **Sintassi:**

`sizeof(tipo_di_dato)`

con `tipo_di_dato` identificatore di tipo
predefinito o non

- **Valore:**

numero delle locazione utilizzate per
rappresentare un valore di tipo `tipo_di_dato`

Operatore sizeof()

- **Esempio:**

```
/* sorgente: Sizeof.c */  
  
/*  
** programma che illustra il comportamento dell'operatore  
** sizeof()  
*/  
  
#include <stdio.h>  
  
int main ()  
{  
    printf ("\nDimensione int: %d\n", sizeof(int));  
    printf ("\nDimensione char: %d\n", sizeof(char));  
    printf ("\nDimensione double: %d\n", sizeof(double));  
    printf ("\nDimensione int *: %d\n", sizeof(int *));  
    printf ("\nDimensione char *: %d\n", sizeof(char *));  
    printf ("\nDimensione double *: %d\n", sizeof(double *));  
    return(1);  
}
```

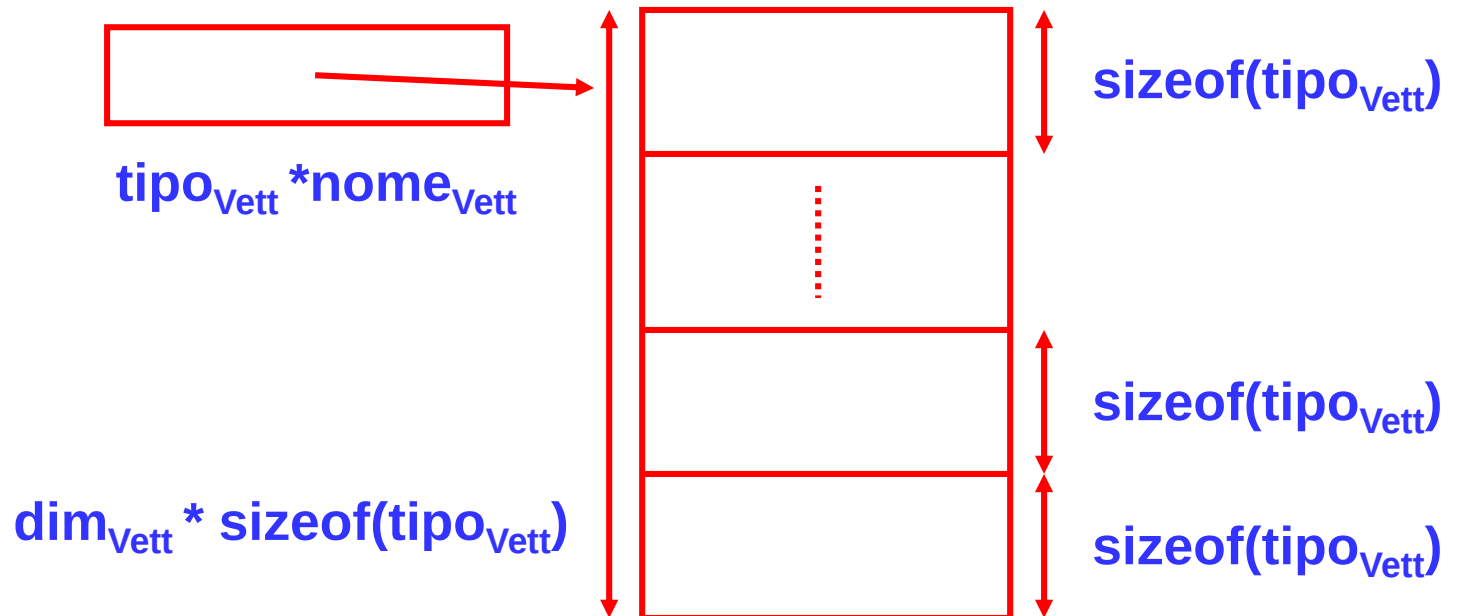
Definizione di un vettore

- Definizione:

Espressione costante intera

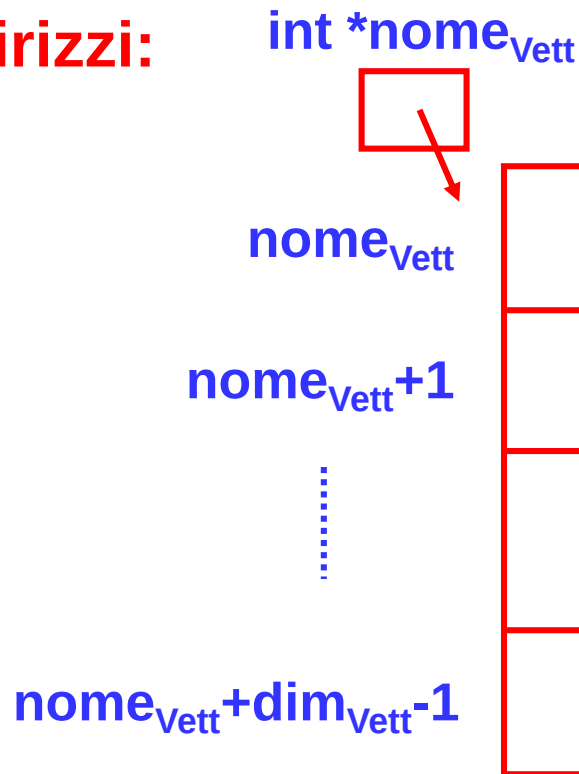
$\text{tipo}_{\text{vett}}$ $\text{nome}_{\text{vett}}$ [dim_{vett}]

- Modifiche allo stato della memoria:



Accesso agli elementi di un vettore

- Indirizzi:



- Contenuto:



- Accesso all'elemento i-esimo:

a) $*(\text{nome}_{\text{Vett}}+i)$, $0 \leq i \leq \text{dim}_{\text{Vett}}-1$

b) $\text{nome}_{\text{Vett}}[i]$, $0 \leq i \leq \text{dim}_{\text{Vett}}-1$

Accesso agli elementi di un vettore

- Esempio (nell'ipotesi che `sizeof(int)=2`):

```
int ArrInt[6];
```

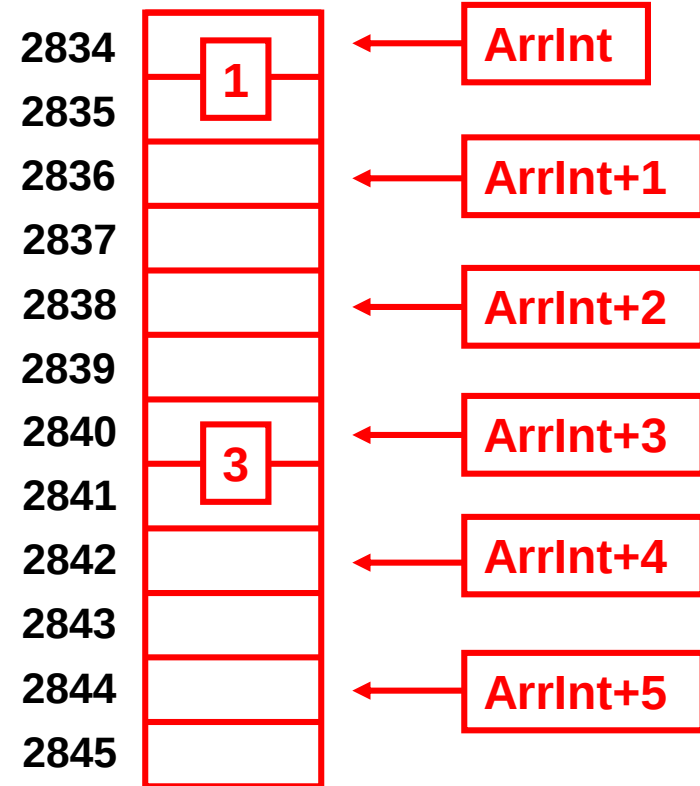
ArrInt 2834

```
*ArrInt = 1;
```

```
*(ArrInt+3) = *ArrInt+2;
```

```
ArrInt[0] = 1;
```

```
ArrInt[3] = ArrInt[0]+2;
```



I vettori e le funzioni

- I vettori come parametri formali:

a) `tipofun nomefun (... , tipoVett nomeVett[], int dimVett, ...)`
`{ ... };`

b) `tipofun nomefun (... , tipoVett *nomeVett, int dimVett, ...)`
`{ ... };`

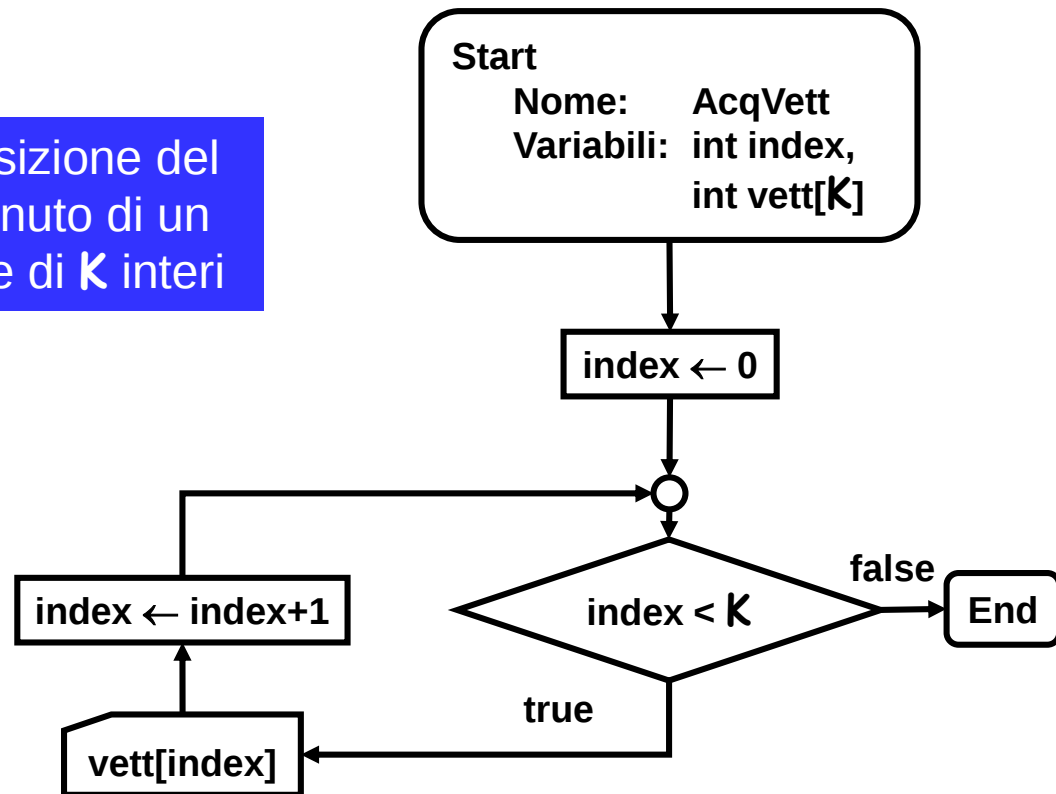
- I vettori come parametri attuali:

`nomefun (... , nomeVett, dimVett, ...)`

I/O di vettori

- I diagrammi di flusso:

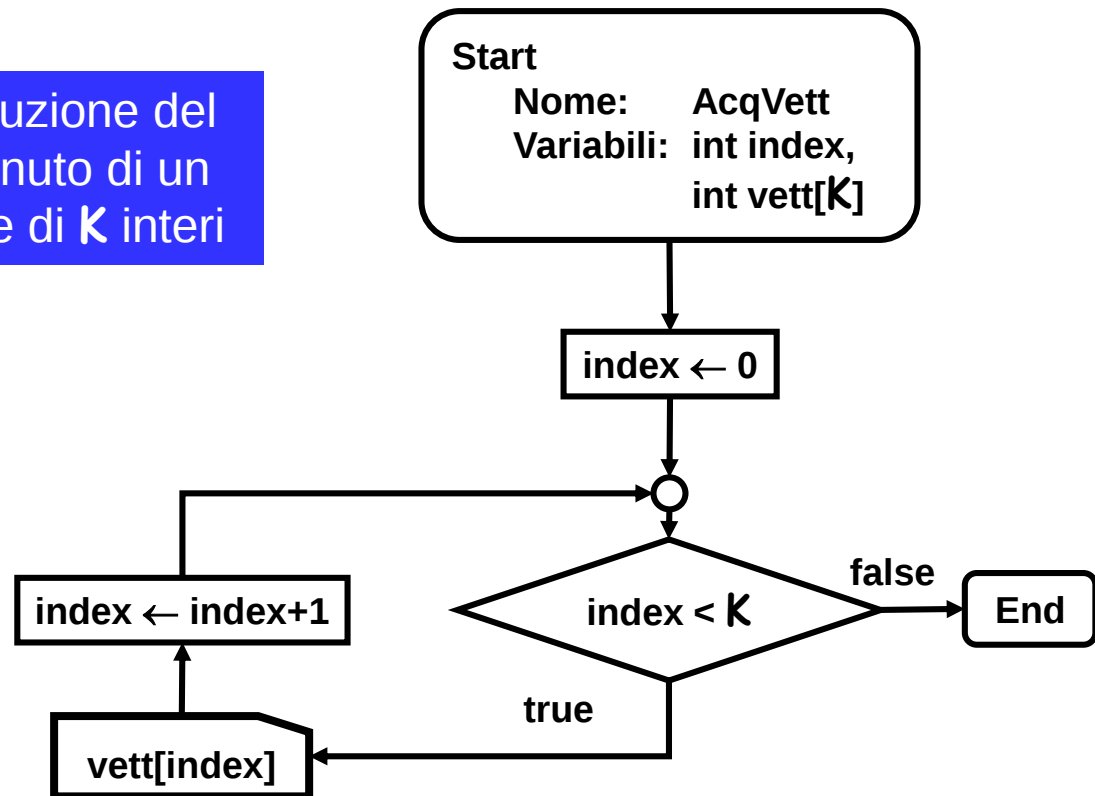
Acquisizione del
contenuto di un
vettore di **K** interi



I/O di vettori

- I diagrammi di flusso:

Restituzione del
contenuto di un
vettore di **K** interi



I/O di vettori

- Il codice:

```
/* sorgente: VettIOInd.c */
/*
** programma che illustra le modalita' di acquisizione
** e di restituzione del contenuto di un vettore di
** interi utilizzando l'indirizzo dei suoi elementi
*/

/* direttive per il preprocessore */
#include <stdio.h>
#define DIM_VETT 5

/* funzione per l'acquisizione del contenuto di un vettore di interi */
void AcqVettInt(int *Vett, int dim)
{
    /* definizione della variabile per la scansione del vettore */
    int pos;

    /* scansione del vettore e acquisizione del suo contenuto */
    for (pos = 0; pos < dim; pos++)
    {
        printf("\nVett[%d]? ", pos);
        scanf("%d", Vett+pos);
    }
}
```

I/O di vettori

```
/* funzione per la restituzione del contenuto di un vettore di interi */
void ResVettInt(int *Vett, int dim)
{
    /* definizione della variabile per la scansione del vettore */
    int pos;
    /* scansione del vettore e restituzione del suo contenuto */
    for (pos = 0; pos < dim; pos++)
        printf("\nVett[%d]: %d", pos, *(Vett+pos));
};

/* Chiamante */
int main()
{
    /* definizione di un vettore di interi */
    int prova[DIM_VETT];
    /* acquisizione del contenuto del vettore */
    AcqVettInt(prova, DIM_VETT);
    /* restituzione del contenuto del vettore */
    ResVettInt(prova, DIM_VETT);
    return(1);
}
```

I/O di vettori

- Il codice:

```
/* sorgente: VettIONome.c */
/*
** programma che illustra le modalita' di acquisizione e di restituzione del
** contenuto di un vettore di interi utilizzando il nome dei suoi elementi
*/
...

/* funzione per l'acquisizione del contenuto di un vettore di interi */
void AcqVettInt(int Vett[], int dim)
{
    /* definizione della variabile per la scansione del vettore */
    int pos;

    /* scansione del vettore e acquisizione del suo contenuto */
    for (pos = 0; pos < dim; pos++)
    {
        printf("\nVett[%d]? ", pos);
        scanf("%d", &Vett[pos]);
    }
}
```

I/O di vettori

- Il codice:

/* funzione per la restituzione del contenuto di un vettore di interi */

void ResVettInt(int Vett[], int dim)

{

/* definizione della variabile per la scansione del vettore */

int pos;

/* scansione del vettore e restituzione del suo contenuto */

for (pos = 0; pos < dim; pos++)

printf("\nVett[%d]: %d", pos, Vett[pos]);

}

...

// Chiamante

...

Dimensionamento a run-time di un vettore

- **La dimensione di un vettore può essere definita a run time?**

```
// definizione della variabile per la dimensione del vettore  
int dim;
```

```
// definizione di un vettore di dimensione nota a run-time  
int Vett [dim];
```

```
// acquisizione della dimensione di un vettore  
scanf("%d", &dim);
```

- **Assolutamente no!!!!**

nella definizione di un array la sua dimensione deve essere specificata tramite una espressione costante

- **Conseguenza:**

se le dimensioni dell'array cambiano il codice deve essere modificato e ricompilato

Dimensionamento a run-time di un vettore

**Esiste un soluzione a
questo problema?**