

Programmazione e Laboratorio di Programmazione

Lezione X

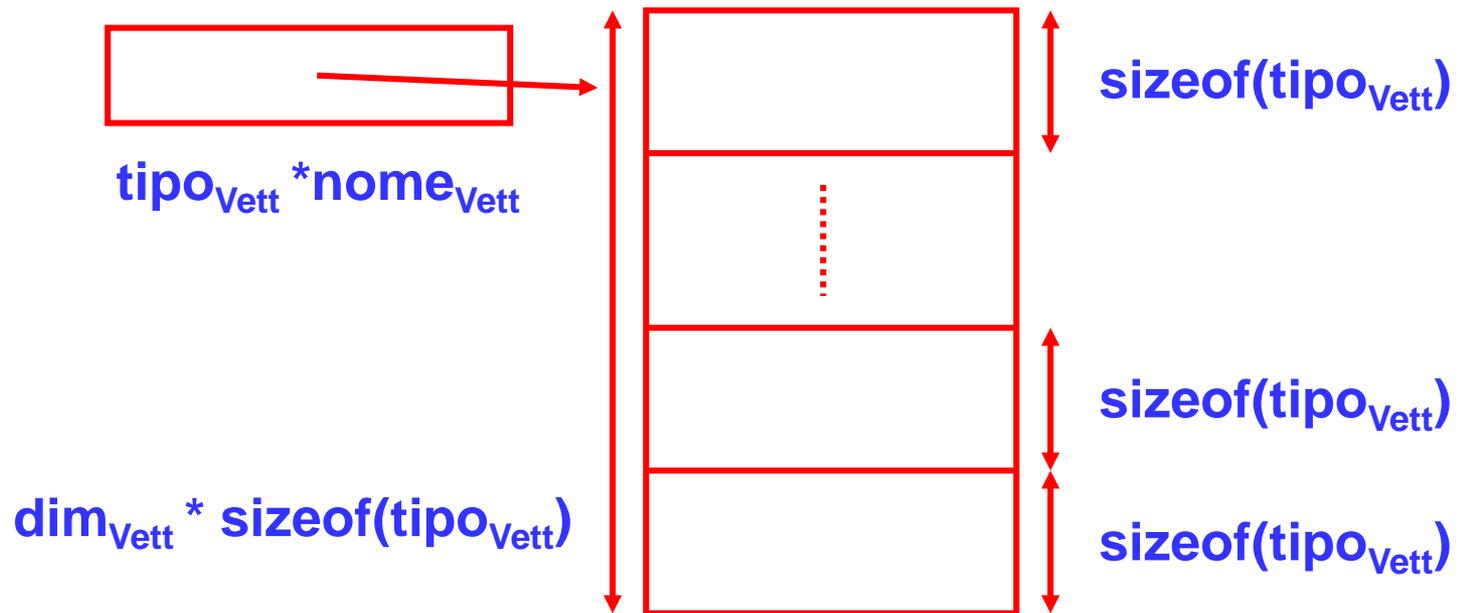
Definizione di vettori a run-time

Definizione statica di vettori

- **Definizione:** espressione intera costante

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

- **Modifiche allo stato della memoria:**



Definizione statica di vettori

- **Problemi:**

- non riesco a gestire situazioni nelle quali la dimensione del vettore è nota, o varia, a run-time

- **Soluzione:**

riprodurre le modifiche allo stato della memoria “innescate” dalla definizione statica di un vettore attraverso le funzioni di gestione della memoria rese disponibili dal C

Definizione di vettori a run-time

- **Modifiche allo stato della memoria:**

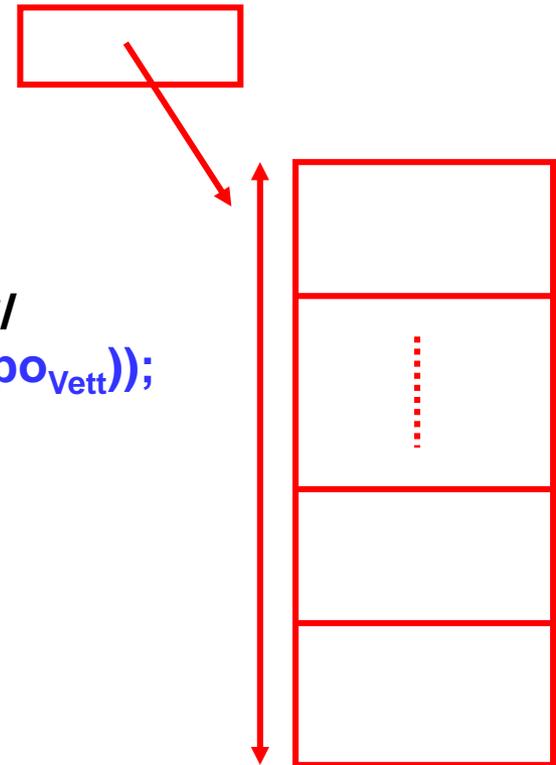
/* definizione del "nome" del vettore */

tipo_{Vett} *nome_{Vett};

/* allocazione della memoria per il vettore */

nome_{Vett} = (tipo_{Vett} *) malloc(dim_{Vett} * sizeof(tipo_{Vett}));

tipo_{Vett} *nome_{Vett}



dim_{Vett} * sizeof(tipo_{Vett})

Esempio: I/O di vettori

```
/* sorgente: VettIO.c */
/* programma che illustra le modalita' di definizione di un vettore
** a run-time */
/* inclusione del file di intestazione della libreria standard
** che contiene definizioni di macro, costanti e dichiarazioni
** di funzioni e tipi funzionali alle varie operazioni di I/O */
#include <stdio.h>
/* inclusione del file di intestazione della libreria standard
** che contiene definizioni di macro, costanti e dichiarazioni
** di funzioni di interesse generale */
#include <stdlib.h>
/* funzione per l'acquisizione del contenuto di un vettore di interi */
void AcqVettInt(int *Vett, unsigned 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);
    };
}
continua ...
```

Esempio: I/O di vettori

Continua ...

```
/* funzione per la restituzione del contenuto di un vettore di interi */  
void ResVettInt(int *Vett, unsigned 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));  
}  
  
/* funzione per l'allocazione di un buffer la cui dimensione è espressa  
** in numero di interi */  
int * AllBuffInt(unsigned int dim)  
{  
    /* definizione della variabile per l'indirizzo iniziale del buffer */  
    int *ptr;  
    /* allocazione della memoria per il buffer */  
    ptr = (int *) malloc(dim * sizeof(int));  
    /* se l'allocazione e' andata a buon fine restituisce l'indirizzo di  
    ** inizio del buffer; NULL altrimenti */  
    return(ptr);  
}
```

Continua ...

Esempio: I/O di vettori

Continua ...

```
/* Chiamante */
int main()
{
    /* definizione del "nome" del vettore */
    int * prova;
    /* definizione e inizializzazione della dimensione del vettore */
    unsigned int dim;
    printf("\nDimensione del vettore? ");
    scanf("%u", &dim);
    /* allocazione della memoria necessaria */
    if ((prova = AllBuffInt(dim)) == NULL)
    {
        /* se l'allocazione fallisce, termina */
        printf("\nEsito dell'allocazione negativo");
        return(0);
    };
    /* altrimenti, acquisisce e restituisce il contenuto del vettore */
    printf("\nEsito dell'allocazione positivo");
    printf("\nAcquisizione del contenuto del vettore\n");
    AcqVettInt(prova, dim);
    printf("\nRestituzione del contenuto del vettore");
    ResVettInt(prova, dim);
}
```

Continua ...

Esempio: I/O di vettori

Continua ...

```
/* rilascia la memoria allocata per il vettore e termina */  
free(prova);  
return(1);  
}
```

Esempio: concatenazione di vettori

```
/* sorgente: VettConc.c */  
/* Programma che acquisisce due vettori di interi (sorgenti), la cui  
** dimensione e' nota a run-time, per concatenarli in un terzo vettore  
** (destinazione) */  
/* inclusione del file di intestazione della libreria standard  
** che contiene definizioni di macro, costanti e dichiarazioni  
** di funzioni e tipi funzionali alle varie operazioni di I/O */  
#include <stdio.h>  
/* inclusione del file di intestazione della libreria standard  
** che contiene definizioni di macro, costanti e dichiarazioni  
** di funzioni e tipi di interesse generale */  
#include <stdlib.h>  
/* inclusione del file di intestazione della libreria standard  
** che contiene definizioni di macro, costanti e dichiarazioni  
** di funzioni e tipi per la gestione delle stringhe e della memoria */  
#include <string.h>
```

Continua ...

Esempio: concatenazione di vettori

Continua ...

```
/* funzione per l'acquisizione del contenuto di un vettore di interi */  
void AcqVettInt(int *Vett, unsigned 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);
```

```
    };
```

```
}
```

```
/* funzione per la restituzione del contenuto di un vettore di interi */  
void ResVettInt(int *Vett, unsigned int dim)
```

```
{  
    /* definisce la variabile per la scansione del vettore */  
    unsigned int pos;
```

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

```
    for (pos = 0; pos < dim; pos++)  
        printf("\nVett[%d]: %d", pos, *(Vett+pos));
```

```
}
```

Continua ...

Esempio: concatenazione di vettori

Continua ...

```
/* funzione per l'allocazione di un buffer la cui dimensione è espressa
** in numero di interi */
int * AllBuffInt(unsigned int dim)
{
  /* definizione della variabile per l'indirizzo iniziale del buffer */
  int *ptr;
  /* allocazione della memoria per il buffer */
  ptr = (int *) malloc(dim * sizeof(int));
  /* se l'allocazione e' andata a buon fine restituisce l'indirizzo di
  ** inizio del buffer; NULL altrimenti */
  return(ptr);
}
```

Continua ...

Esempio: concatenazione di vettori

Continua ...

```
/* funzione che concatena due vettori di interi in un terzo vettore */
int *ConcVettInt(int *sorg1, unsigned int dim1, int *sorg2,
                unsigned int dim2)
{
    /* definizione del "nome" del vettore destinazione */
    int *result;
    /* alloca la memoria per il vettore risultante */
    if ((result = AllBuffInt(dim1+dim2)) == NULL)
    {
        // se l'allocazione non ha esito positivo restituisce NULL
        return(result);
    };
    /* copia il I vettore sorgente in testa al vettore destinazione */
    memcpy ((void *) result, (void *) sorg1, dim1*sizeof(int));
    /* copia il II vettore sorgente in coda al I nel vettore destinazione */
    memcpy ((void *) (result+dim1), (void *) sorg2, dim2*sizeof(int));
    /* restituisce l'indirizzo del vettore risultante */
    return(result);
}
```

Continua ...

Esempio: concatenazione di vettori

Continua ...

```
/* Chiamante */
int main()
{
    /* definizione delle variabili per i "nomi" dei tre vettori */
    int *src1, *src2, *trg;
    /* definizione delle variabili per la dimensione dei due vettori sorgente */
    unsigned int dim_src1, dim_src2;
    /* acquisizione della dimensione del I vettore sorgente */
    printf("\nDimensione del I vettore? ");
    scanf("%u", &dim_src1);
    /* allocazione della memoria per il I vettore sorgente */
    if ((src1 = AllBuffInt(dim_src1)) == NULL)
    {
        /* se l'allocazione fallisce, termina */
        printf("\nEsito dell'allocazione del I sorgente negativo");
        return(0);
    }
}
```

Continua ...

Esempio: concatenazione di vettori

Continua ...

```
/* acquisizione della dimensione del II vettore sorgente */
printf("\nDimensione del II vettore? ");
scanf("%u", &dim_src2);
/* allocazione della memoria per il II vettore sorgente */
if ((src2 = AllBuffInt(dim_src2)) == NULL)
{
    /* se l'allocazione fallisce, rilascia la memoria per il I vettore e termina */
    printf("\nEsito dell'allocazione del II sorgente negativo");
    free(src1);
    return(0);
};
/* acquisizione del contenuto dei due vettori sorgente */
printf("\nAcquisizione del I vettore"); AcqVettInt(src1, dim_src1);
printf("\nAcquisizione del II vettore"); AcqVettInt(src2, dim_src2);
/* concatenazione dei vettori sorgente nel vettore destinazione: se
** l'esito è negativo rilascia la memoria allocata per i sorgenti e termina
*/
if ((trg = ConcVettInt(src1, dim_src1, src2, dim_src2)) == NULL)
{
    printf("\nEsito dell'allocazione del vettore destinazione negativo");
    free(src1);
    free(src2);
    return(0);
};
```

Continua ...

Esempio: concatenazione di vettori

Continua ...

```
/* restituzione del contenuto del vettore destinazione */  
printf("\nContenuto del vettore destinazione:");  
ResVettInt(trg, dim_src1+dim_src2);  
/* recupero della memoria allocata per i tre vettori */  
free(src1);  
free(src2);  
free(trg);  
return(1);  
}
```