

Lezione 4

Appunti di Davide Scarlata 2024/2025


 **Prof:** Michele Garetto

 **Mail:** michele.garetto@unito.it

 **Corso:** C

 [Moodle corso C](#)

 [Moodle Lab matricole dispari](#)

 **Data:** 03/03/2025

ALLOCAZIONE DELLA MEMORIA

Nel modello di memoria della macchina astratta C esistono tre tipi di memoria per allocare le variabili:

1. Memoria statica

- Usata per memorizzare le variabili globali e le variabili locali `static`.

2. Memoria automatica (stack)

- Usata per memorizzare i record di attivazione delle funzioni.
- Ogni record di attivazione è costituito da:
 1. L'indirizzo di codice dell'istruzione successiva a quella che ha invocato la funzione (indirizzo di rientro).
 2. I parametri della funzione.
 3. Le variabili locali (non `static`) della funzione.

3. Memoria dinamica (heap)

- Usata per allocare variabili (spazi di memoria) la cui dimensione si conosce solo in fase di esecuzione.
 - Esempio: in un programma che, dopo aver acquisito il numero di rilevazioni disponibili da una sonda di temperatura, le memorizzi in un vettore.
-

◆ Allocazione dinamica in C

In C, l'allocazione nella memoria heap e la sua liberazione sono esplicite, ovvero si usano apposite funzioni della libreria `stdlib.h` :

- `malloc` : Alloca uno spazio di memoria contiguo nell'heap e restituisce un puntatore al primo byte allocato. Se fallisce, restituisce `NULL` (indirizzo 0).
- `free` : Libera lo spazio di memoria precedentemente allocato con `malloc` .

Sintassi

```
#include <stdlib.h>

void *malloc(size_t size); // size è il numero di byte da allocare
void free(void *ptr);      // passo il puntatore della zona da deallocare
```

La funzione `malloc` restituisce un puntatore `void *`, che deve essere **castato** nel tipo corretto.

Attenzione!

Se `malloc` restituisce `NULL` , significa che la memoria non è stata allocata.

```
ptr = (cast-type *) malloc(byte-size); // Cast del puntatore
free(ptr); // Dopo free la memoria è disponibile per nuove allocazioni
```

◆ Esempio di allocazione statica vs dinamica

Statica	Dinamica
<code>int x; x = 0;</code>	<code>int *x; x = (int *) malloc(sizeof(int)); *x = 0; free(x);</code>

Statica	Dinamica
Accesso più rapido	Accesso meno rapido
Occupava memoria anche per variabili inutili	Occupava solo la memoria necessaria

Allocazione dinamica di un array

```
int *d;
d = (int *)malloc(sizeof(int) * ESPR);
free(d);
d = NULL; // Non obbligatorio, ma consigliato
```



Allocazione dinamica di una struct

```
typedef struct {
    char titolo[MAXT];
    int pagine;
    char autore[MAXN];
    float prezzo;
} Libro;

Libro *L;
L = (Libro *) malloc(sizeof(Libro));
```



Passaggio di struct come parametro

```
struct prova {
    int x;
    int y;
};

void stampa(struct prova p) {
    printf("\n x=%d\n", p.x);
    printf(" y=%d\n", p.y);
}
```

Variabili locali static

Una variabile locale (cioè dichiarata all'interno di una funzione)

static è una variabile che mantiene il suo valore dopo che la funzione è terminata. Ad esempio:

```
int nextNumber()
{
    static int counter = 1;
```

```
return counter++;  
}
```

restituisce:

- 1 la prima volta che è chiamata
- 2 la seconda volta che è chiamata
- n (a meno di overflow) l'n-esima volta che è chiamata