**La memoria puo essere allocata dinamicamente:** quindi nella RAM durante l'esecuzione del programma: nello stack, nell'heap o nel data segment

*In ugual modo essa puo essere anche deallocata*

Essa puo essere inoltre allocata:

- In run-time
- In compile-time

Quando il programma viene eseguito, l'os alloca uno spazio per l'heap e per lo stack

- Lo spazio di memoria per lo stack può esaurirsi per effetto di ripetute chiamate annidate a funzione.
- Lo spazio di memoria per l'heap può esaurirsi per ripetute allocazioni dinamiche.

Si chiama **allocazione statica** quando viene definita una variabile globale o locale con specificatole *static*.

Essa avviene in compile-time nel data segment

Questo blocco non è riutilizzabile

```
void Incrementa()
{
    int x=0;
    x++;
    printf("%d\n",x);
}

int main()
{
    Incrementa();
    Incrementa();
    Incrementa();
}
```

Produce come output:

1
1
1

```
void Incrementa()
{
    static int x=0;
    x++;
    printf("%d\n",x);
}

int main()
{
    Incrementa();
    Incrementa();
    Incrementa();
}
```

Produce come output:

1
2
3

L'**allocazione dinamica** invece avviene in run-time nell'heap:

Allocable e disallocabile solo tramite opportune funzioni di libreria
L'heap è gestito dall'os

Prototipo della funzione (dichiarato in stdlib.h):
void * malloc(size_t size);
Alloca un blocco di byte = size e restituisce un puntatore alla prima cella del blocco
In caso di blocchi non disponibili restituisce NULL

*Nota: Non si può applicare l'operatore sizeof a un blocco di memoria allocato dinamicamente in quanto sizeof viene valutato dal compilatore.*

Per rilasciare la memoria allocata dinamicamente utilizzeremo la funzione free().
void free(void * p);

**Esempio:**

```
#include <sdtlib.h>
...
int * p = (int *) malloc(sizeof(int));
...
...
// Alcuni compilatori  (in particolare i compilatori
// C++) richiedono un cast di p a (void *).
free((void *)p);
```

*Una volta chiamata free per rilasciare un blocco di memoria, i valori dei puntatori al blocco dovrebbero essere impostati a NULL per eliminare la possibilità che il programma faccia riferimento alla memoria che è stata liberata e che potrebbe già essere stata allocata per un altro scopo.*