Memoria Dinamica C

venerdì 23 dicembre 2022 08:04

LA MEMORIA DINAMICA

LA SEZIONE "HEAP" è l'area dove posso chiedere memoria!

HEAP

Viene utilizzata per supportare le allocazioni dinamiche della memoria A RUN-TIME, perché mentre il programma esegue potrebbe avere bisogno di memoria aggiuntiva;

Questo è possibile perché esistono delle funzioni di libreria per gestire l'heap!

#INCLUDE "STDLIB.H"

1 MALLOC

- `void *malloc(size_t size)`: alloca una quantità di memoria di dimensione size byte e restituisce un puntatore alla memoria allocata, o NULL in caso di errore. La memoria non è inizializzata. Se `size` è zero, `malloc()` restituisce NULL o un puntatore che può essere successivamente passato a `free()`.

2 FREE

- `void free(void *ptr)`: libera l'area di memoria puntata da `ptr`, che deve essere stato precedentemente restituito da `malloc()` (o varianti).

↳ invalida logicamente l'area di memoria puntata da free!

MA i leggere i dati dopo aver eseguito free! UNDEFINED BEHAVIOUR;

Se la heap è piena e chiamo la malloc, essa mi restituirà NULL!
NULL significa che la malloc non ha potuto allocare memoria;
E se ci scrivi, il programma va in crash!
Bisogna sempre controllare il valore del puntatore!

```c
int *ptr = malloc(10 * sizeof(int));
if (!ptr) {
    /* Manage the error here */
} else {
    /* Allocation successful. Do whatever you want! */
    free(ptr);  /* When memory is not needed anymore, you free it. */
    ptr = NULL; /* Set the pointer to NULL, to avoid a "dangling" pointer */
}
```

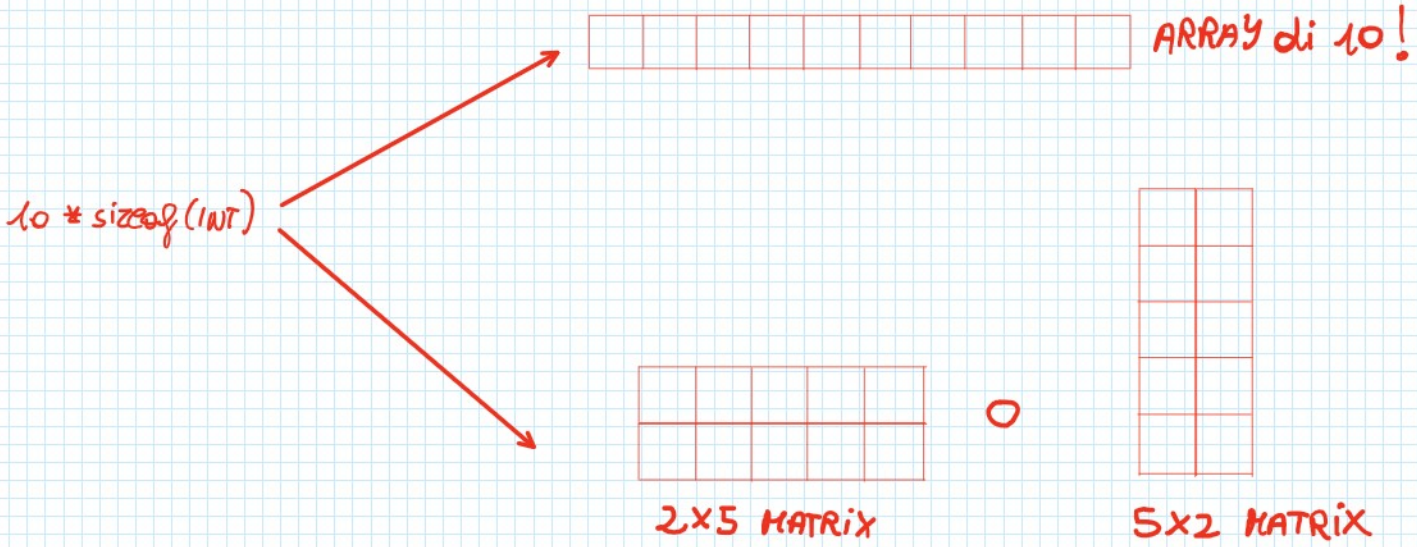
chiedo a malloc di riservarmi un buffer di memoria grande 10 * 4 = 40 byte

controlla se pointer è NULL!

- Un'alternativa all'allocazione iniziale:

```c
int *ptr = malloc(10 * sizeof(*ptr));
```

dimensione dato puntato → questo tutto * allora l'indirizzo sarebbe stato a 64 bit – 8 byte!




```
INT *PTR = (INT *) MALLOC (10*SIZEOF (*PTR))
```

MAI USARE UN CAST SUL VALORE DI RITORNO DELLA MALLOC;

Come fa il mio compilatore a sapere che la firma è `void *malloc(size_t size)` ?
Legge la libreria! E che trova? Il prototipo della funzione! La sua firma!

Però magari io sto sviluppando in fretta e mi dimentico di includere stdlib.h, come diventa per il compilatore la firma di questa malloc?

`INT MALLOC (void)` → FIRMA PREDEFINITA!

Restituisce un intero;
Un intero è grande 4 byte!
↳ quindi prenderebbe 4 byte e li scriverebbe nel puntatore;

Ma un puntatore è grosso 8 byte! Quindi significa che sto troncando a metà l'indirizzo!

Ma effettuando quel CAST, diciamo al compilatore che siamo consapevoli del fatto che noi vogliamo usare esclusivamente 4 byte per questo facciamo il CAST!
Il compilatore GENEREREBBE WARNING, ma noi lo vogliamo col CAST!
quindi se effettuiamo il CAST e ci dimentichiamo di includere stdlib.h, SILENZIAMO il warning del compilatore, ENTRIAMO IN Undefined Behaviour!