

Malloc y free

¿Qué son y cómo los uso?



La Memoria Dinámica

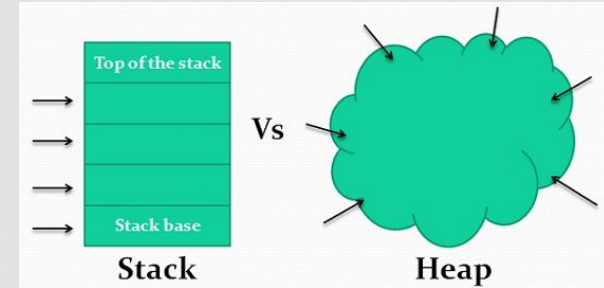
La memoria dinámica es aquella que sería reservada y utilizada únicamente en **tiempo de ejecución**.

La memoria dinámica es reservada en el **heap** del programa.

Para utilizar este tipo de memoria el programador debe hacerlo de forma **explícita** solicitándoselo al sistema operativo.

Para ello en el lenguaje de programación C se utilizan dos **funciones** de la biblioteca estándar llamadas:

- malloc()
- free()





Biblioteca stdlib.h

```
#include <stdlib.h>
```

Para utilizar tanto malloc como free debemos incluir la biblioteca `stdlib.h` (igual que para usar printf y scanf debemos incluir `stdio.h`).

`stdlib.h` tiene algunas otras funciones, como `realloc` por ejemplo, pero hoy nos centraremos en `malloc` y `free` únicamente.

¿Qué son malloc y free?

Malloc y Free son funciones de la biblioteca estándar de C `stdlib.h`.

La primera nos permite **reservar memoria** del heap (dinámica) y la segunda es utilizada para **liberar** dicha memoria una vez utilizada.



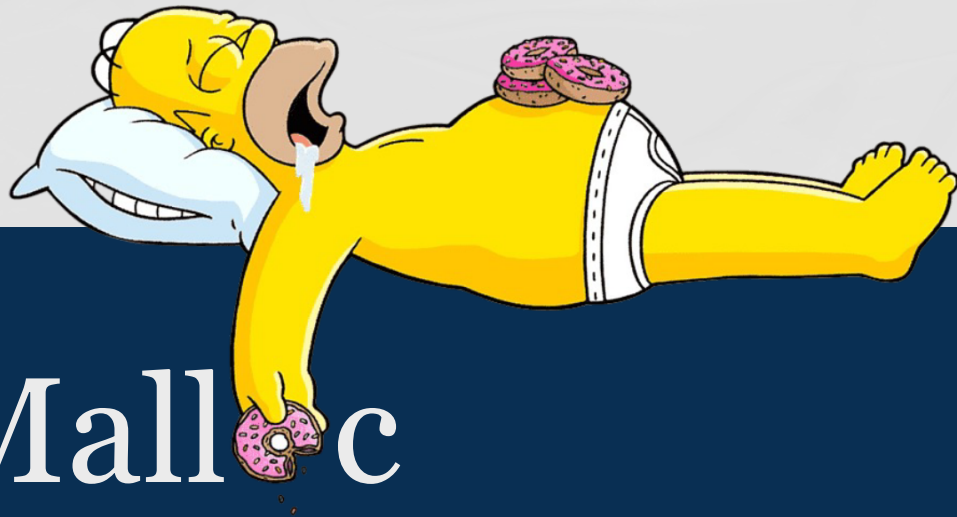
Los pasos para utilizar memoria dinámica son dos:

1. en primer lugar se reserva utilizando **malloc()** y;
2. cuando se termina de utilizar y no se necesita más tener ese espacio de memoria reservado, se libera usando **free()**



01

Mallc





1. malloc()

Malloc : Memory Allocation (Asignación de memoria)

La función malloc() recibe como parámetro la **cantidad de bytes** que se quieren **reservar** en el **heap**.

Asigna un bloque de memoria acorde al tamaño especificado en el heap.

Si funciona **correctamente**, devuelve un **puntero** al primer byte de la dirección de esa memoria reservada. Caso contrario devuelve **NULL**.

```
#include <stdlib.h>
void *malloc(size_t size);
```

Entonces ¿Cómo puedo usar malloc?

Si yo quisiera **reservar espacio** para un entero (int) en el heap debería hacer:

```
int* puntero = malloc(4);
```



Puedo utilizar el operador **sizeof()** para no tener que poner un 4 literal como parámetro:

```
int* puntero = malloc(sizeof(int));
```



Genial, entonces si quiero reservar memoria solo pongo malloc y listo, ¿no?





No!!!! malloc es una función que **puede fallar** por lo tanto, **siempre** que la use, debo asegurarme que se haya **reservado correctamente la memoria**.


Por suerte malloc nos devuelve **NULL**.
Entonces puedo hacer:

```
int* puntero = malloc(sizeof(int));  
...
```

¿cómo manejo el error?





 **No!!!!** malloc es una función que **puede fallar** por lo tanto, **siempre** que la use, debo asegurarme que se haya **reservado correctamente la memoria**.

Por suerte malloc nos devuelve **NULL**.
Entonces puedo hacer:

```
int* puntero = malloc(sizeof(int));  
if (puntero == NULL){  
    printf("..mensaje descriptivo del error");  
    return -1;  
}
```



¿Qué pasa si solo uso malloc?

Si solo hago malloc() , entonces queda **memoria reservada en el heap** que nadie libera.

```
int* puntero = malloc(sizeof(int));
```

Si hago esto, entonces quedan 4 bytes de memoria en el heap sin liberar. Para evitar esto es que debo **utilizar free**.



Free nos permite **recuperar** esa memoria cuando ya no la uso.





¿Si quisiera reservar espacio para un vector de enteros?

A dark gray square box with a thin black border, containing the text 'int' in a white, monospaced font.

Si para reservar 1 entero hacemos de forma implícita :

```
int* vector = malloc(1* sizeof(int));
```



¿Si quisiera reservar espacio para un vector de enteros?



```
int cantidad = 7;  
int* vector = malloc(cantidad* sizeof(int));
```

02

Free





2. free()

La función `free()` libera espacio de memoria apuntado por el puntero que debe haber sido devuelto previamente por una llamada a la función `malloc()`, `calloc()` o `realloc()`.

De otra forma o si hago `free(puntero)` de un puntero que ya ha sido liberado anteriormente, ocurría un **comportamiento no definido**.

Si puntero es `NULL`, la operación **no se realiza**.

¿Cómo puedo usar free?

Utilizar `free()` es muy sencillo, solo debo pasarle como parámetro un **puntero** que **apunte** a **memoria** en el **heap**.

Si reservé memoria (ya sea para cualquier tipo de dato) y la almacene la dirección de esa memoria en **puntero**, entonces para liberar esa memoria debo hacer:



```
free(puntero) ;
```


A tener en cuenta:

Cuando llamo a `free()` sobre un puntero, si este **no apunta** a memoria en el **heap** entonces va a **fallar**, por ende, si hago:

```
free(puntero);  
free(puntero);
```

Se va a **romper**.

Por otro lado, luego de hacer un **free()** sobre un puntero este no necesariamente apunta a **NULL**, queda con **contenido basura**.





Recapitulando...

El siguiente código reserva y libera memoria para un entero, ¿lo hace bien?

```
int* puntero = malloc(sizeof(int));  
free(puntero);
```

Recapitulando...

Falta chequear el malloc!!!

```
int* puntero = malloc(sizeof(int));  
if (puntero == NULL) {  
    //maneja el error;  
}  
free(puntero);
```



Algunos consejos a la hora de usar malloc() y free()



Para **evitar** posibles **errores** a la hora de usar memoria dinámica en C, hay algunas recomendaciones que podemos seguir:

- **Siempre** después de hacer un **malloc()**, hacer un **free()**.
- **Siempre** **chequear** que el malloc haya **reservado correctamente** la memoria.
- Usar el operador **sizeof()** para el malloc() y no el valor literal.
- **Evitar** liberar **dos** veces el mismo puntero.
- Si voy a seguir usando un puntero luego del free(), asignarle **NULL**.

03

Ejemplos

