**¿Cómo funciona la memoria en lenguaje C?**

El lenguaje C implementa el tipo de gestión manual, es decir que cada programa debe explícitamente pedir memoria y liberarla.

La función malloc definida en stdlib.h permite reservar memoria de un tamaño dado. Su contrato es:

void \*malloc(size\_t tamanio);

Y se lee así:

* Reserva la cantidad de memoria especificada por parámetro.
* Retorna la dirección de la porción de memoria que reservó. Es decir que retorna un puntero. De ahí que primer vimos [acá](https://sites.google.com/site/programacioniiuno/temario/unidad-1---manejo-de-memoria-dinmica/punteros-referencias-y-direcciones) algunas definiciones iniciales de qué son los 1punteros.

La llamada a malloc podría "fallar" si por algún motivo el sistema operativo no puede darle la cantidad de memoria especificada a nuestro programa. Cuando este sucede malloc retorna NULL.

**¿Qué es un apuntador?**

Un Apuntador es una variable que contiene una dirección de memoria, la cual corresponderá a un dato o a una variable que contiene el dato.

Cada variable que se utiliza en una aplicación ocupa una o varias posiciones de memoria. Estas posiciones de memoria se accesan por medio de una dirección.

El Operador de Dirección ( &) regresa la dirección de una variable.

El Operador de Indirección ( \* ), toma la dirección de una variable y regresa el dato que contiene esa dirección.

¿Cómo se declara un apuntador?

La declaración de un puntero de manera general es:

Tipo\_dato \*nombre de apuntador;

Tipo\_dato : Especifica el tipo de objeto apuntado y puede ser cualquier tipo (int, float, char, etc).

Nombre de apuntador: Es el identificador (nombre asignado) del apuntador.

**¿Cómo conocer la dirección de memoria de una variable?**

Toda variable tiene una dirección de memoria la cual podemos obtener programáticamente (es decir desde el mismo programa).

Las direcciones de memoria se pueden almacenar a su vez en variables, data una dirección de memoria, podemos acceder a su contenido o modificarlo y las variables que contienen direcciones a memoria se denominan de tipo puntero.

**¿Cómo mostrar el valor de un apuntador?**

Printf(“%p”, nombreapuntador);

**¿Cómo pasar parámetros por referencia a funciones?**

Si queremos que los cambios realizados en los parámetros dentro de la función se conserven al retornar de la llamada, deberemos pasarlos por referencia. Esto se hace declarando los parámetros de la función como referencias a objetos.

**#include** <iostream>

**using** **namespace** std;

**int** funcion(**int** &n, **int** &m);

**int** *main*() {

**int** a, b;

a = 10; b = 20;

*cout* << "a,b ->" << a << ", " << b << *endl*;

*cout* << "funcion(a,b) ->" << funcion(a, b) << *endl*;

*cout* << "a,b ->" << a << ", " << b << *endl*;

/\* *cout* << "funcion(10,20) ->"

<< funcion(10, 20) << *endl*; // (1)

es ilegal pasar constantes como parámetros cuando

estos son referencias \*/

**return** 0;

}

**int** funcion(**int** &n, **int** &m) {

n = n + 2;

m = m - 5;

**return** n+m;

}

En este caso, los objetos "a" y "b" tendrán valores distintos después de llamar a la función. Cualquier cambio de valor que realicemos en los parámetros dentro de la función, se hará también en los objetos referenciadas.

Esto quiere decir que no podremos llamar a la función con parámetros constantes, como se indica en (1), ya que aunque es posible definir referencias a constantes, en este ejemplo, la función tiene como parámetros referencias a objetos variables.

Y si bien es posible hacer un *casting* implícito de un objeto variable a uno constante, no es posible hacerlo en el sentido inverso. Un objeto constante no puede tratarse como objeto variable.