***Apuntador***

Un **apuntador** es la dirección en memoria de una variable, es decir la dirección que se

usa para nombrar dicha variable, se llama apuntador porque podemos pensar que la

dirección apunta a la variable. Un apuntador se puede guardar en una variable, por

ejemplo p, cuyo tipo de dato debe de ser del tipo del dato que guarda, por ejemplo si va a

guardar un número entonces p debe ser del tipo int o float. Para señalar que es un

apuntador se usa un asterisco \*.

Tipo \_dato \* nombre\_variable ;

int \*p;

Se usa el operador & para determinar la dirección de una variable, por ejemplo p contiene

la dirección de var.

int var;

int \*p;

p=&var;

Para el manejo de la memoria dinámica en **C** usamos malloc, calloc, realloc y

free; mientras que en **C++** se usa new y delete. Malloc nos pemite reservar la

cantidad de bytes especificada y nos regresa un apuntador genérico (void) y se encuentra

en la librería stdlib.h

Apuntador\_genérico malloc (número\_bytes);

void\* malloc ( unsigned int);

En el siguiente ejemplo podemos observar el uso de la memoria Heap, la cual es

expandible.

main(){

int \*ptr;

ptr = (int\*)malloc(sizeof(int));

\*ptr = 8;

}

Otra forma de hacerlo es declarar la variable y después pasar la dirección de memoria de

esa variable a una apuntador con el operados &.

int variable;

int \* apuntador = &variable

\*apuntador = 20; //Asignamos un valor al contenido en esa posición de memoria.

***Paso por Valor***

Una variable solo existe en la función en la que se crea y nada más, a través de los

parámetros podemos pasar el valor de una determinada variable a otra función, pero lo

que pasamos es el valor y no la variable en sí.

***Paso por Referencia***

Hay veces que nos interesa que una función modifique una variable que no

pertenece a ella, es decir, fuera de su ámbito; lo cual se hace a través de punteros

(o referencias). La idea es que como solo se puede pasar el valor de una variable

a una función lo que hacemos es pasar la dirección de una variable la cual puede ser reservada con malloc o a través de del operador de indirección & con el que

podemos acceder al contenido de la variable original.