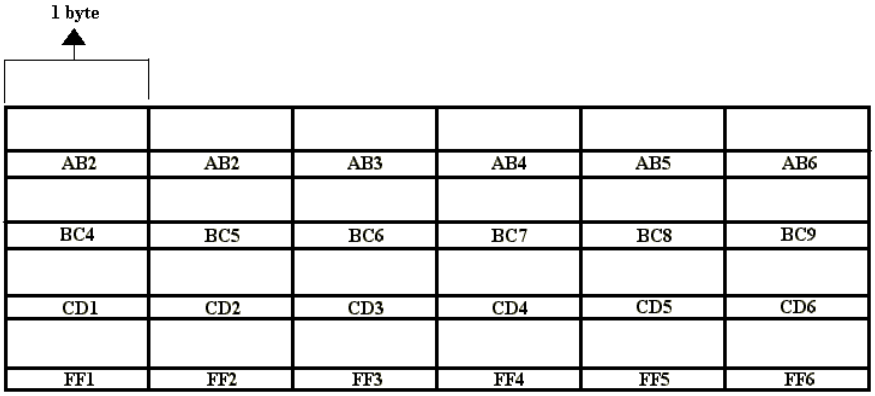
APUNTADORES.

La memoria de una máquina esta ordenada en forma de celdas numeradas consecutivamente y que se pueden manipular individualmente o en grupos contiguos. La siguiente figura muestra una representación de un segmento de la memoria RAM de una computadora.



La memoria de una computadora esta constituida en base a celdas de memoria. De la figura anterior, cada cuadro (incluyendo el cuadro etiquetado con un número hexadecimal) representa una celda de memoria, la cual puede almacenar información del tamaño de un byte de memoria. Cada celda de memoria se etiqueta con un número en formato hexadecimal, conocido como la dirección de la celda. En la memoria RAM no se repite ninguna dirección.

Un apuntador puede tener acceso a la información contenida en una dirección de memoria, la cual puede representar a una variable. El apuntador puede ser capaz de modificar el valor que contiene esa variable.

En lenguaje C, para tener acceso a la dirección de memoria de una variable se debe de hacer uso del operador unario &.

Para utilizar un apuntador, este se debe declarar anteponiendo a su nombre el operador unario *. Este operador se conoce como operador de indirección o desreferencia, este da acceso al contenido de la dirección que señala el apuntador. El siguiente ejemplo muestra dos declaraciones de apuntadores:

```
int *ip;
char *cp;
```

La primer declaración se lee de la siguiente manera: “ip” es un apuntador a datos de tipo entero. La segunda declaración se lee de la siguiente manera “cp” es un apuntador a datos de tipo char.

Para trabajar con apuntadores las siguientes reglas son necesarias:

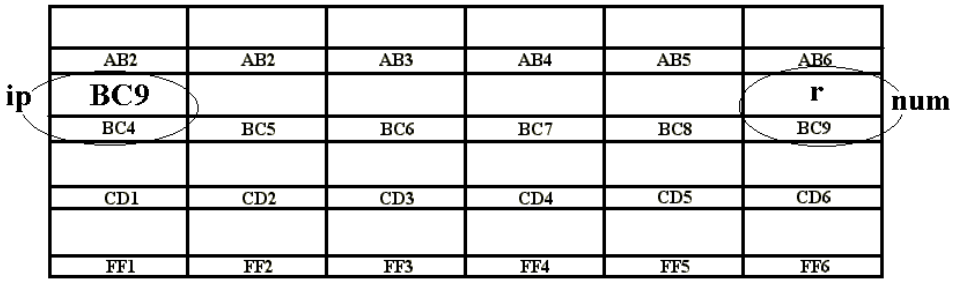
1. Un apuntador antes de utilizarse, debe de inicializarse hacia alguna dirección de memoria, por lo general es una dirección donde se encuentra una variable.
2. Un apuntador únicamente almacena direcciones a diferencia de las variables que almacenan valores.

3. Un apuntador solo puede apuntar a una sola dirección en un instante de tiempo. En otras palabras, un apuntador solo almacena una dirección en un instante de tiempo. La dirección a la que se apunta el apuntador, generalmente representa la ubicación de una variable.
4. Varios apuntadores si pueden apuntar a la misma dirección de memoria en un instante de tiempo.
5. Aunque no siempre, un apuntador debe de apuntar al mismo tipo de dato con el cual fue declarado. Por ejemplo si el apuntador se declaro como apuntador a datos de tipo entero, entonces deberá apuntar únicamente a variables de tipo entero.

Un ejemplo de inicialización se da a continuación:

```
char num='r', *ip;
ip = &num;
```

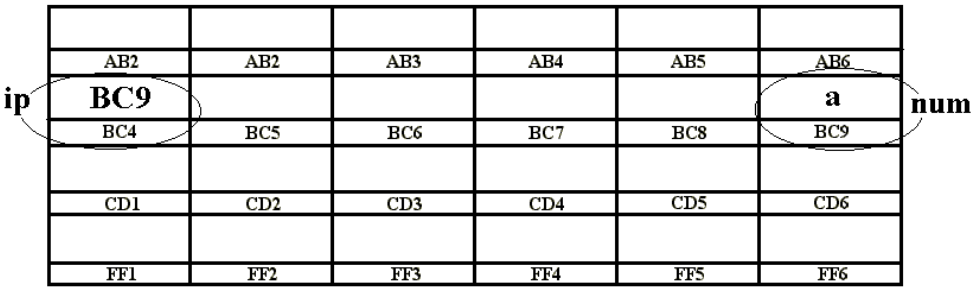
El apuntador fue inicializado con la dirección de la variable "num". La operación anterior se podría representar en memoria como lo muestra la siguiente figura:



La variable "num", por ser declarada como dato de tipo carácter requiere de un byte de memoria, el cual se le asigna a la variable en la dirección BC9. El apuntador "*ip" al ser declarado, se le asigna una dirección de memoria en BC4. Al inicializar el apuntador en la dirección de la variable "num", este guarda como dato la dirección de BC9, que es donde se encuentra la variable "num". La siguiente operación modifica el valor de "num" a través del apuntador "*ip":

```
*ip =  'a';
```

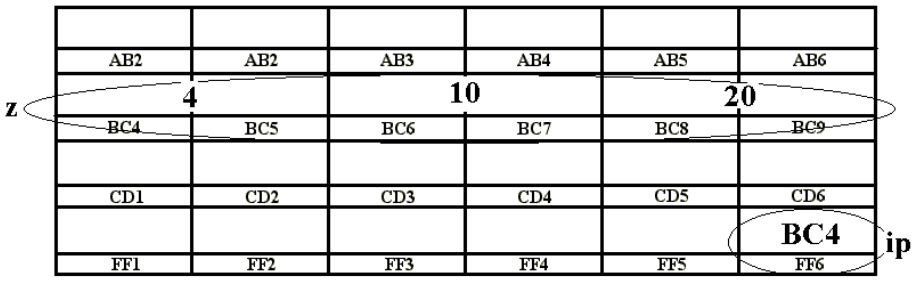
La operación anterior se lee como, el contenido de la dirección a la que apunta "ip" es igual al carácter 'a'. El resultado de esta operación se representa en memoria por la siguiente figura:



Un ejemplo de inicialización de un apuntador hacia un arreglo:

```
int *ip, z[3]={4,10,20};
ip=&z[0];
```

Al inicializar el apuntador con la dirección del arreglo “z”, este toma la dirección más pequeña encontrada en el espacio cero del arreglo como se muestra en la siguiente figura:



El tamaño de memoria requerido para el apuntador, se adapta al valor numerico que se requiera almacenar como dirección.

El valor de una variable o expresión de tipo arreglo es la dirección del elemento cero del arreglo. esto es ip = &z[0]; /* ip y z, tienen valores idénticos en dirección*/

El nombre de un arreglo es sinónimo para la localidad del elemento inicial (dirección), la asignación ip = &z[0] se puede escribir como

```
ip=z;
```

Una referencia a z[i] puede ser escrita como *(z+i). Al ser evaluada z[i], el compilador de C la convierte inmediatamente como *(z+i), esta equivalencia se puede representar de la siguiente forma:

z[i] <==> *(z+i)

si aplicamos el operador & a estas dos expresiones

&z[i] <==> &*(z+i)

la relación “&*” puede eliminarse pues el resultado de esta relación es la unidad. Por lo cual se tiene lo siguiente

&z[i] <==> z+i

si se busca la dirección inicial del arreglo, entonces i = 0 y se tiene que

&z[0] <==> z

Esto implica que el nombre de un arreglo da como resultado la dirección inicial del arreglo.

Si ip es un apuntador a un arreglo, las expresiones pueden usarlo como subíndice, esto es

*(ip+i) <==> ip[i]

Se debe tener siempre presente que un apuntador es una variable, por esto ip = z y ip++ son legales, pero un nombre de arreglo no es una variable, por lo tanto construcciones como z = ip y z++ son ilegales.

Un apuntador puede conocer el contenido total del arreglo si tiene la dirección inicial de este. Para conocer lo que existe en el siguiente espacio del vector solo Será necesario sumarle a la dirección guardada por el apuntador el espacio correspondiente al tipo de dato del arreglo, por ejemplo de la figura anterior, para utilizar el valor numérico de 10 que tiene el vector en z[1], a través del apuntador, se puede hacer de dos formas:

1) con *(ip+1)
2) con ip[1]

Arreglos de apuntadores.

Las declaraciones

char saludo[]=”hola que tal”; /*arreglo*/

char *saludo = ”hola que tal”; /*apuntador*/

En el arreglo se pueden modificar caracteres individuales dentro de este, pero en el apuntador no ya que es inicializado para apuntar a una cadena constante, se puede modificar solo para que apunte a otra cadena.

Dada las siguientes declaraciones

int a[0][20];
int *b[0];

“a” es un arreglo de dos dimensiones, pero “b” indica que solo se asigna 10 apuntadores y no los inicializa; siempre deben de inicializarse con código o estáticamente. Suponga que cada elemento de “b” apunta a un arreglo de veinte elementos, entonces existirán 200 enteros reservados, más diez celdas para los apuntadores. La ventaja importante del arreglo de apuntadores es que los renglones del arreglo pueden ser de longitudes diferentes.

ejemplo:

char *meses[]={”mes ilegal”,”enero”,”marzo”,”abril”,”mayo”,”junio”};

Ejemplo de uso de direccionamiento de apuntadores, para un mejor entendimiento utilizar el depurador:

```
#include <stdio.h>


void main(void)
{
int x=1, y=2;
int z[]={1,2,3,4,5,6,7,8,9,10,19};
int *ip;                    /* ip es un apuntador a entero */
ip=&x;                          /* ip ahora apunta a x */
printf(“El contenido de *ip = %d”,*ip);
y=*ip;                              /* y es ahora 1 */
printf(“El contenido de y = %d”,y);
*ip=0;                              /* x es ahora cero */
ip=&z[0];           /* ip ahora apunta a z[0] esto es ip=z */
printf(“El contenido de *ip = %d”,*ip);
}
```