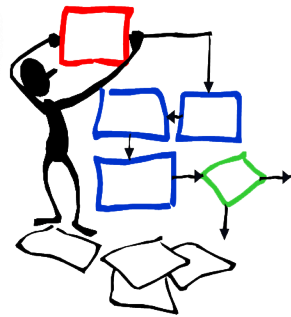


# Instituto Politécnico Nacional

## Escuela Superior de Cómputo



## Algoritmia y programación estructurada

### *Clase 22:* Apuntadores en C

Prof. Edgardo Adrián Franco Martínez

<http://computacion.cs.cinvestav.mx/~efranco>

[@efranco\\_escom](#)

[efranco.docencia@gmail.com](mailto:efranco.docencia@gmail.com)





# Contenido

- Introducción
- Dirección de memoria
- Definición de apuntador
- Declaración de un apuntador
- Inicialización de un apuntador
- Indirección de un apuntador
- Operadores para utilizar apuntadores
- Apuntadores NULL y void
- Apuntadores a apuntadores
- Aritmética de apuntadores
- Paso por referencia en funciones





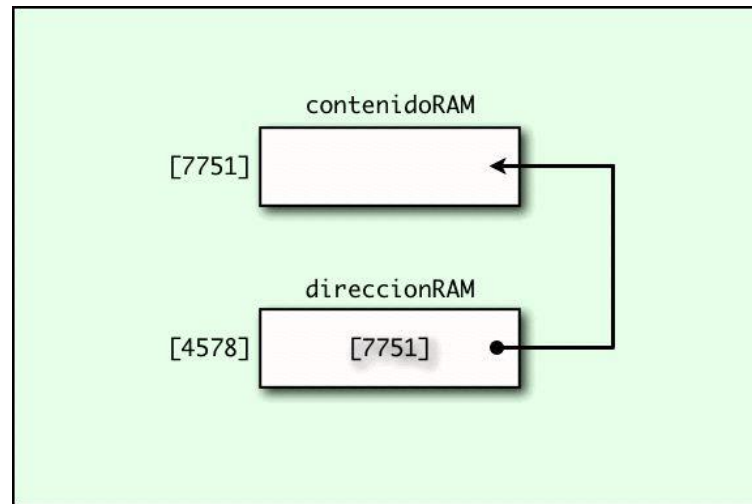
# Introducción

- Los apuntadores o punteros son otros tipos de dato que se utilizan en lenguaje de programación C.
- Su utilidad radica en que permiten a un programa ser más potente, dinámico y flexible.
- Además, su uso le da al lenguaje C su potencia y la popularidad que ha adquirido, debido a que permiten el acceso a memoria de manera más eficiente, sin embargo, una mala referencia a dicha memoria provocará en el programa una salida inesperada.





- Trabajar con apuntadores implica la **no manipulación** de variables en sí, sino manejar **direcciones de memoria** en la cuales residen los datos.
- Una **variable apuntador** o puntero, es una **variable que contiene direcciones** de otras variables, es decir, almacenan la dirección de memoria donde se encuentran los datos asociados a dichas variables.



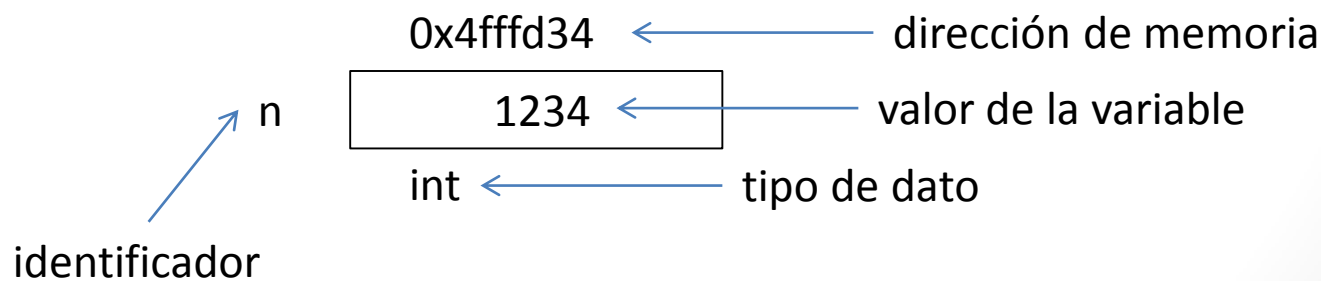


# Dirección de memoria

- Cuando una variable se declara, se le asocian tres atributos fundamentales a dichas variables: su *identificador*, su *tipo de dato* y su *dirección de memoria*.

por ejemplo:

```
int n;
```





# Definición de apuntador

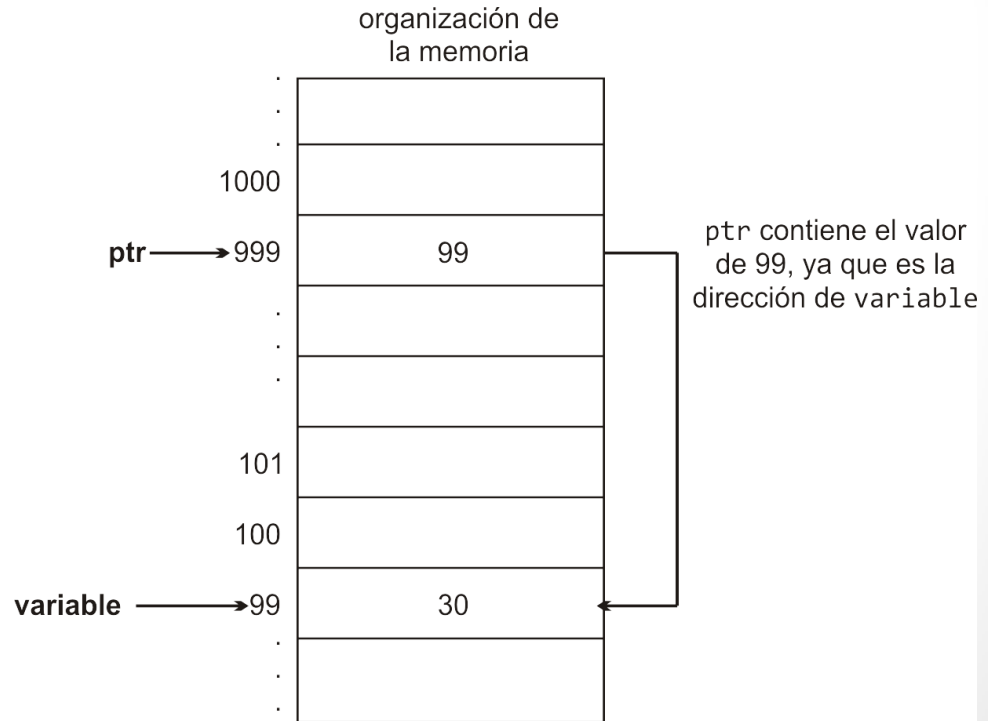
- Cuando se declara una variable en C, el compilador establece un área de memoria para almacenar el contenido de la variable, cuando se hace referencia a dicha variable, el compilador accede automáticamente a la dirección de memoria donde se almacena el contenido de la variable para poder utilizarla.
- Cuando se declara una variable de tipo apuntador, el compilador también le asigna una dirección de memoria





- Cuando se declara una variable de tipo apuntador, el compilador también le asigna una dirección de memoria, sin embargo, en esa localidad de memoria no se almacena un dato, sino la dirección de una variable del mismo tipo que se declaro el apuntador, es decir:

```
int variable = 30;  
int *ptr = &variable;
```





- Los apuntadores en C se rigen por la siguientes reglas básicas:

1. Un apuntador es una variable como cualquier otra;
2. Una variable apuntador contiene una *dirección* que apunta a otra posición de memoria;
3. En esa posición de memoria se almacenan los datos a los que apunta el apuntador;
4. Un apuntador apunta a una variable que se encuentra almacenada en alguna parte de la memoria física.







# Declaración de un apuntador

- Al igual que todas la variables, los apuntadores deben de ser declarados antes de ser utilizados, la forma de declarar un apuntador es el siguiente:

```
<tipo_de_dato_apuntado> * <identificador_apuntador> = & <variable_a_apuntar>;
```

donde:

- <tipo\_de\_dato\_apuntado>: Es el tipo de dato al cual va a apuntar el apuntador;
- \*: Es el elemento que dice que la variable va a ser un apuntador;
- <identificador\_apuntador>: Es el nombre del apuntador;
- &: Es el elemento para obtener la dirección de la variable apuntada;
- <variable\_a\_apuntar>: Identificador de la variable a apuntar.





# Inicialización de un apuntador

- Como sucede con las variables locales, C no inicializa un apuntador cuando se declara y es preciso inicializarlo antes de su uso ya que de lo contrario no se estará haciendo referencia a ninguna localidad de memoria, provocando un error en tiempo de compilación.
- Para asignar la dirección de memoria a un apuntador se utiliza el operador de referencia de dirección **&**, por ejemplo:

```
int *ptr = &x;
```

A este tipo de inicialización también se le llama **inicialización estática** ya que la dirección de memoria es fija y no puede desaparecer, es decir, la memoria para la variable siempre estará reservada hasta que termine el programa.






- Existe un segundo método para inicializar un apuntador y es mediante la asignación dinámica de memoria, para realizar este proceso se utilizan funciones de asignación de memoria como `malloc()`, `calloc()`, `realloc()` y `free()`. Se retomarán cuando se vea el tema de manejo de memoria dinámica.

- Para asignar un nuevo valor o recuperar dicho valor que se encuentra almacenado en localidad de memoria a la que apunta el apuntador se utiliza el operador de indirección `*`.

por ejemplo:  Se inicializa el apuntador **ptr**

`int variable2;`  Se cambia el valor de la localidad de memoria

`int variable = 30;`  Se obtiene el valor de la localidad de memoria a la que apunta **ptr**

`int *ptr = &variable;`

`*ptr = 45;`

`variable2 = *ptr;`





- Los apuntadores se enlazan a tipos de datos específicos y el compilador también verifica que las variables que se asignan al apuntador son del tipo correcto, ya que de otra forma se estará generando un error en tiempo de ejecución, es decir, C requiere que las variables apuntador direccionen realmente variables del mismo tipo de dato ligado en su declaración.

```
int n=1234;  
int * ap = &n;
```

tipo de dato	identificador	valor de la variable	dirección de memoria
int	n	1234	0x4fffd34
int *	ap	0x4fffd34	0x4fffd38



# Indirección de un apuntador

## Otro ejemplo:

```
int edad;  
int *ptr_edad;  
ptr_edad = &edad; ←  
*ptr_edad = 50;  
printf("%d", edad);  
printf("%d", *ptr_edad);
```

Otra forma de inicializar **ptr**, **nótese que en esta línea ya no es necesario utilizar el \*** ya que si se utiliza sería un error. Se utiliza el **\*** sólo en la declaración del apuntador

## Ejemplo:

```
#include <stdio.h>  
char c; //variable global  
  
int main( void ){  
    char *ptr_char;  
    ptr_char = &c; //apuntar a la variable global  
    for( c = 'A'; c <= 'Z'; c++ ){  
        printf( "%c", *ptr_char );  
    }  
    return 0;  
}
```



# Operadores para utilizar apuntadores

Operador	Propósito
<b>&amp;</b>	<b>Operador de dirección:</b> obtiene la dirección de la variable
<b>*</b>	<b>Operador de indirección:</b> permite modificar u obtener el contenido del dato que se encuentra en la localidad de memoria que apunta el apuntador.

\*

*Nota: En una declaración define la variable como apuntador  
P.e. **int \*apuntador;***





# Apuntadores NULL y void

- Cuando se declara un apuntador, el compilador no lo inicializa, es decir, tendrá un valor aleatorio, esto condiciona a que si se le asigna la dirección de una variable, el compilador no sepa a que tipo de dato se quiera apuntar.
- Si se requiere, existen dos tipos de apuntadores especiales: los apuntadores **void** y **NULL** (nulo).





# Apuntador NULL

- Un apuntador **nulo** no apunta a ninguna localidad de memoria o dato valido, este es muy útil cuando se requiere indicar al programa que valide cuando el apuntador no esta apuntando a nada (dato no valido). Es decir, es muy útil para propósitos de comparación en una estructura condicional o en una iterativa, por ejemplo:

```
if( ptr == NULL )
{
    . . .
}

while( ptr != NULL )
{
    . . .
}
```

NULL es una macro de tipo:  
**#define NULL 0**  
que se encuentra definida en las bibliotecas **stddef.h**, **stdio.h**, **stdlib.h** y **string.h**





- Otra forma de declarar un apuntador nulo es:

```
int *ptr = (int *) 0;
```

El *casting* **(int \*)** no es necesario ya que existe una conversión estándar de **0** a una variable apuntador.

- Nunca se utiliza un apuntador para referenciar un valor, solo se utiliza para saber si el apuntador se ha inicializado correctamente o para saber cuando el apuntador a dejado de apuntar a un dato valido, es decir, en un *test* de comparación.

```
char *ptr;  
ptr = malloc(121*sozeof(char));  
if( ptr == NULL )  
    printf("Error de asignacion de memoria...");
```





# Apuntador void

- Los apuntadores **void** son apuntadores que apuntan a cualquier tipo de dato, es decir, no se inicializa con un tipo de dato específico, también son llamados *apuntadores genéricos*. La forma de declarar un apuntador **void** es:

```
void *ptr;
```

- Un apuntador **void** permite a una función especificar un parámetro de entrada para recibir cualquier tipo de apuntador sin que se produzca un error de tipos.

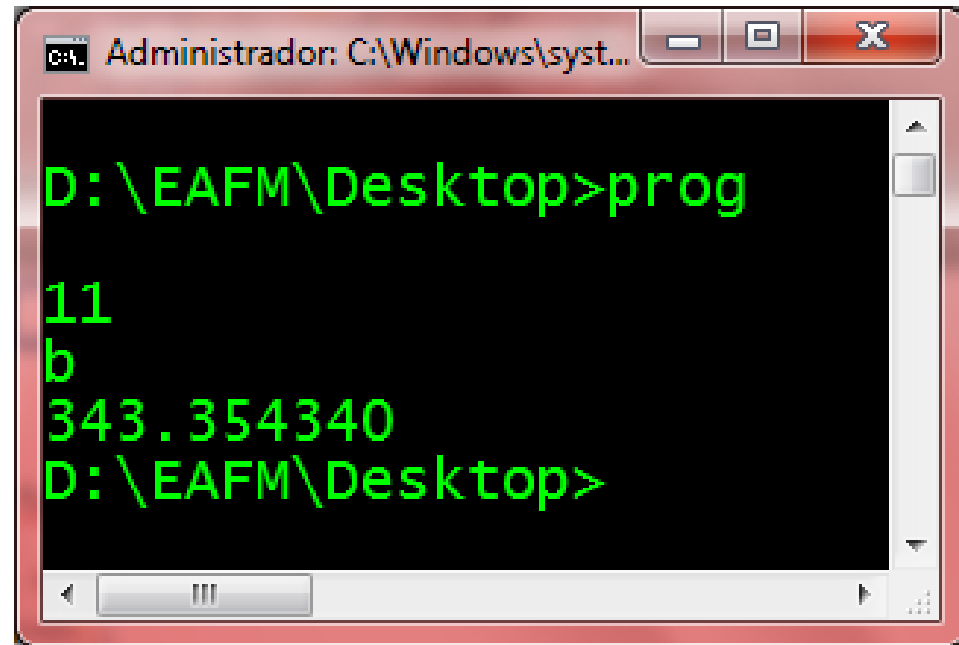


```
#include <stdio.h>

void funcion( void *p );
char c; //variable global

int main( void )
{
    char *ptr_char;
    float *pf;
    int *pd;
    int variable1 = 10;
    float variable2 = 343.3543;
    ptr_char = &c;
    pd = &variable1;
    pf = &variable2;
    funcion( pd );
    printf( "\n%d",variable1 );
    c = 'a';
    funcion( ptr_char );
    printf( "\n%c", c );
    funcion( pf );
    printf( "\n%f",variable2 );
    return 0;
}
```

```
void funcion( void *p )
{
    int *ptr = p;
    *ptr+=1;
}
```



```
C:\> Administrador: C:\Windows\syst...
D:\EAFM\Desktop>prog
11
b
343.354340
D:\EAFM\Desktop>
```

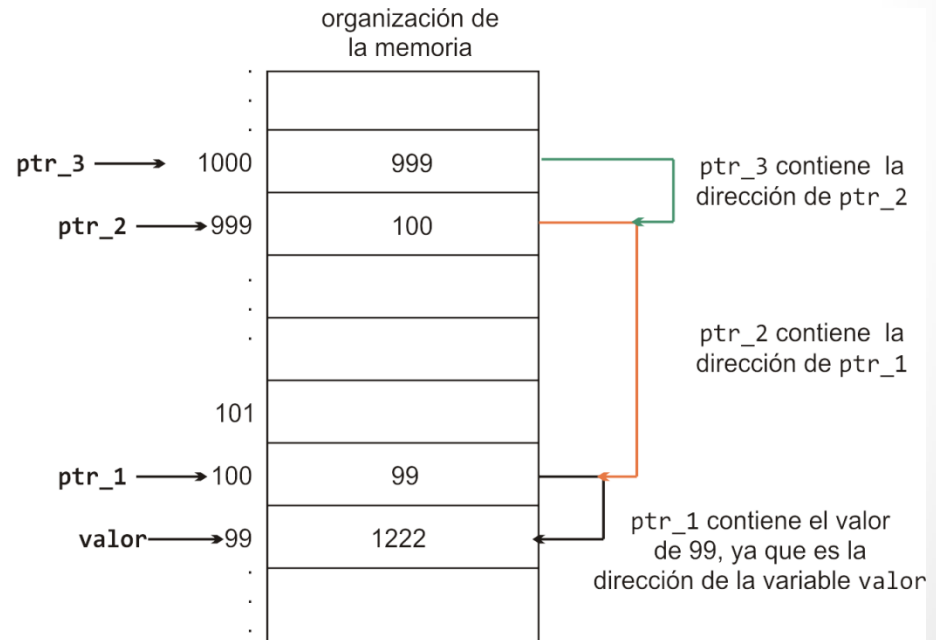


# Apuntadores a apuntadores

- Un apuntador puede apuntar a otro apuntador, para realizar esta operación se hace preceder a la variable apuntador el numero de asteriscos necesarios para direccionar a los apuntadores, es decir:

```
int valor = 1222;
int *ptr_1 = &valor;
int **ptr_2 = &ptr_1;
int ***ptr_3 = &ptr_2;
*ptr_1 = 105;
**ptr_2 = 3292;
***ptr_3 = 232;
```

Actualizan el valor de la variable valor





# Aritmética de apuntadores

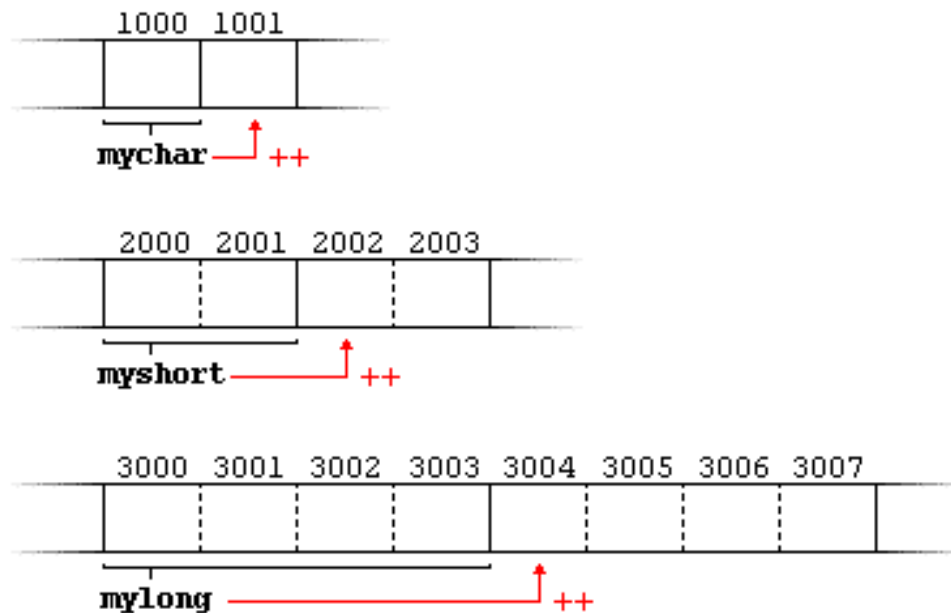
- Un apuntador es una variable que puede modificar el valor al que apunta, es decir, se puede realizar ciertas operaciones aritméticas con ellos.
- A un apuntador se le puede **sumar o restar un entero  $n$** ; esto hace que apunte  $n$  posiciones hacia adelante o hacia atrás de la dirección actual en la pila de memoria, sin embargo, a un apuntador **no se le puede sumar o restar un número de coma flotante, ni multiplicar, ni dividir.**

$$ptr = ptr + n;$$





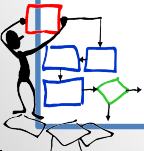
- A un apuntador se le puede aplicar el operador ++ y --, esto hace que obtenga la dirección del siguiente o anterior elemento en la memoria de datos, por ejemplo, si se declara un apuntador con un tipo de dato **int** entonces la siguiente dirección que apunte el apuntador será cuatro localidades después o antes de la referencia actual dependiendo del tipo de operador que se le haya aplicado.





# Aritmética de apuntadores

```
#include <stdio.h>
int main( void )
{
    //programa que utiliza un apuntador para modificar el
    //valor de diferentes variables de tipo entero
    int a,b,c,d,e,f,g,h,i,j;
    int n;
    int *ptr = &a;
    for( n = 0; n < 10; n++ )
    {
        *ptr = n;
        ptr--;
    }
    printf( "valor de las variables: \n" );
    printf( "%d, %d, %d, %d, %d, %d, %d, %d, %d, %d", a, b, c,d, e, f, g, h, i,j);
    printf( "\n\nvalor de sus direcciones: \n" );
    printf( "%d, %d, %d, %d, %d, %d, %d, %d, %d, %d", &a, &b,&c, &d, &e, &f, &g, &h, &i, &j);
    return 0;
}
```

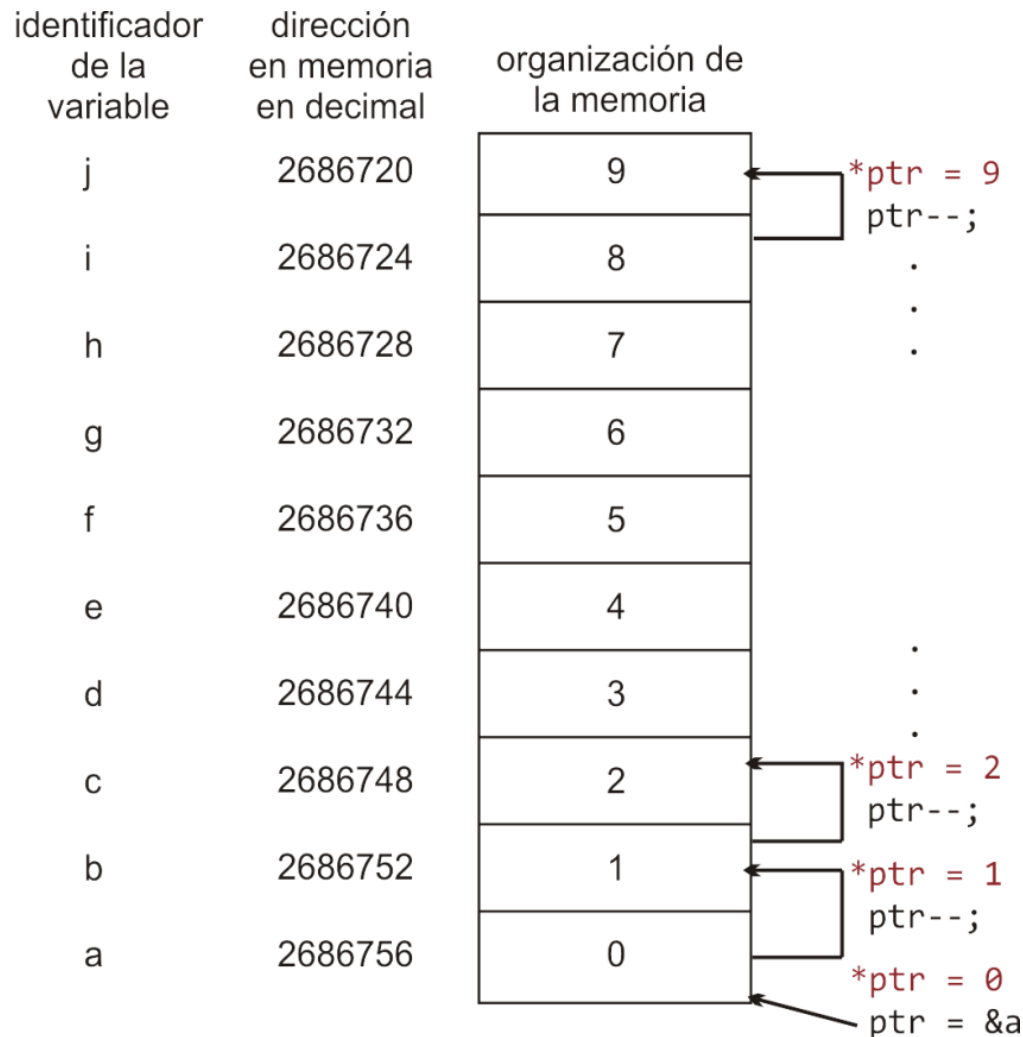


valor de las variables:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9

valor de sus direcciones:

2686756, 2686752, 2686748, 2686744, 2686740, 2686736, 2686732, 2686728, 2686724, 2686720







- En general, cada vez que se incrementa un apuntador, apuntará a la posición de memoria del siguiente elemento de su tipo base. Cada vez que se decrementa apuntará a la posición de memoria del anterior. Cuando se utilizan variables de tipo **char**, pareciera que la aritmética toma sentido, esto es debido a que un **char** ocupa un byte de espacio de memoria.
- El resto de los apuntadores aumentarán y decrementarán la longitud de memoria referente al tipo de dato a los que apuntan.





# Paso por referencia en funciones

- El convenio de paso de parámetros a una función de C sólo es el paso por valor (se pasa la copia del valor de la variable en cuestión), sin embargo, se puede realizar el paso de parámetros a una función por referencia, es decir, pasar la dirección de memoria de una variable a una función para que dicha función pueda trabajar sobre la localidad de memoria que se le asigno a la variable.
- En este caso, se dice que se esta pasando un apuntador a la función.
- Un apuntador se pasa a una función como cualquier otro argumento, para que esto funcione en la función se debe declarar los parámetros como tipo apuntador.





- Para pasar una variable por referencia, el operador de dirección **&** debe de anteceder el identificador de la variable y el parámetro que reciba la dirección de esa variable se deberá declarar como apuntador.

```
#include <stdio.h>
int funcion( int *, int * );

int main( void ){
    int x,y;
    int valorRetorno;
    x = 20;
    y = 34;
    printf( "Valor de x: %d y de y: %d",x,y );
    valorRetorno = funcion( &x, &y );
    printf( "\nNuevo valor de x: %d y de y: %d",x,y );
    printf( "\nEl valor de valorRetorno es: %d", valorRetorno );
    return 0;
}

int funcion( int *x, int *y ){
    int variable;
    *x = *x * 4;
    *y = *x - *y;
    variable = *x + *y;
    return variable;
}
```

