



PUNTEROS EN EL LENGUAJE C

ING. GIANKARIS G. MORENO R., M.SC.

DEFINICIÓN DE PUNTEROS

DECLARACIÓN DE PUNTEROS

OPERACIONES CON PUNTEROS

**PASO DE PUNTEROS A FUNCIONES
(PARÁMETROS POR REFERENCIA)**

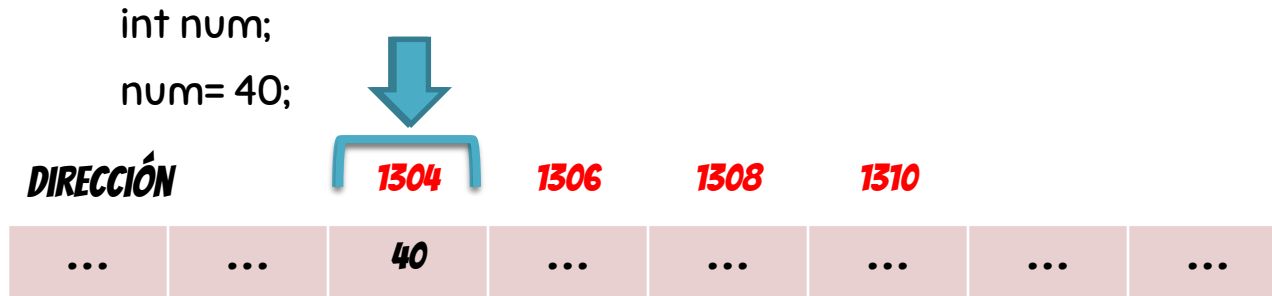
PUNTEROS Y ARREGLOS

ASIGNACIÓN DINÁMICA DE MEMORIA

PUNTEROS EN C

DEFINICIÓN DE PUNTEROS

- × Se define como una variable que permite almacenar una dirección de memoria.
- × Una dirección de memoria y su contenido no es lo mismo, por ejemplo:



La **dirección** de num es **1304**, y el **contenido** de num es **40**.

DEFINICIÓN DE PUNTEROS

- × Importancia de utilizar punteros:
 - × Proporcionan soporte para asignación dinámica de memoria.
 - × Crean código eficiente y rápido ya que el compilador puede traducir más fácilmente la operación en código máquina.
 - × Protegen datos pasados como parámetros a las funciones.
 - × Capacidad de pasar estructuras de datos sin ocasionar un overhead (exceso de código)

NOTA: Un sólido conocimiento de punteros y la habilidad de utilizarlos eficientemente hace la diferencia entre un programador novato y uno experto.

DECLARACIÓN DE PUNTEROS

- ✕ Formato de declaración:

```
tipo_dato * nombre_identificador;
```

Tipo de dato: se refiere al tipo de dato del objeto referenciado por el puntero.

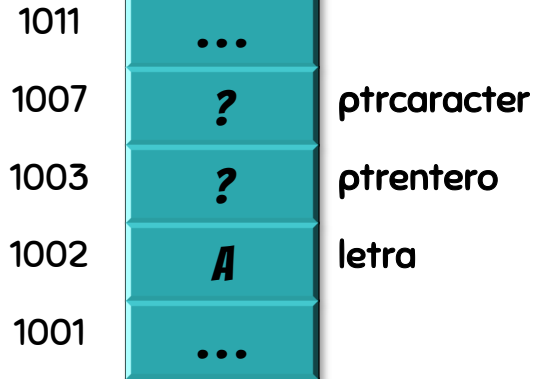
Nombre de identificador: es el nombre de la variable de tipo puntero.

DECLARACIÓN DE PUNTEROS

- ✗ El espacio de memoria reservado para almacenar un puntero es el mismo independientemente del tipo de dato al que apunte.
- ✗ Un puntero ocupa en memoria 4 bytes.

Ejemplo:

```
char letra;  
letra = 'A';  
char *ptrcaracter;  
int *ptrentero;
```



OPERACIONES CON PUNTEROS

- ✕ Cuando trabajamos con punteros es posible realizar operaciones tales como:



OPERACIONES CON PUNTEROS

DIRECCIÓN

- Para llevar a cabo esta operación se emplea el operador `&`.
- El operador `&` devuelve la dirección de memoria de la variable que precede.
- El operador `&` se utiliza para asignar valores a datos de tipo puntero

```
int var;  
int *ptr;  
...  
ptr = &var;
```


OPERACIONES CON PUNTEROS

INDIRECCIÓN

- Para llevar a cabo esta operación se emplea el operador `*`.
- El operador `*` devuelve el contenido de la variable referenciada por el puntero.
- El operador `*` se utiliza para acceder a los datos de las variables a los que apunta un puntero.

```
char var;  
char *ptr;  
...  
ptr = &var;  
*ptr = 'A'; //var = 'A'
```

OPERACIONES CON PUNTEROS

ASIGNACIÓN

- Para llevar a cabo esta operación se emplea el operador `=`.
- A un puntero se le puede asignar:
 - 0 o NULL (para indicar que no apunta a nada)
 - la dirección de una variable
 - el contenido de otro puntero.

NOTA: Como todas las variables, los punteros también contienen “basura” cuando se declaran, por lo que es una buena costumbre inicializarlos.

OPERACIONES CON PUNTEROS

ASIGNACIÓN (INICIALIZACIÓN)

Los punteros deben ser inicializados:

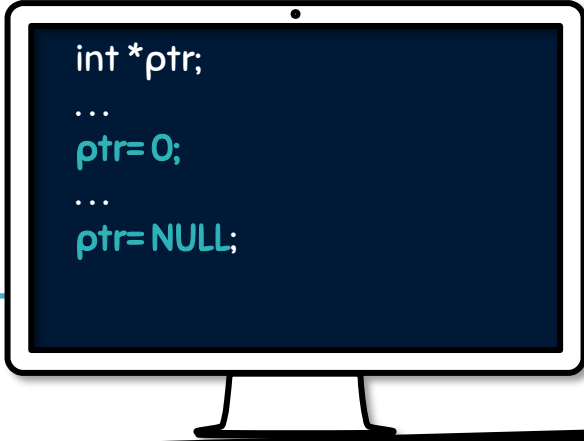
- Se utiliza NULL para indicar que el puntero no apunta a nada.
- El 0 es el único valor entero que puede asignarse directamente a un puntero y es equivalente a NULL.

NOTA: NULL está definido en `<stdio.h>`.

OPERACIONES CON PUNTEROS

ASIGNACIÓN

- Asignación para indicar que el puntero no apunta a nada:

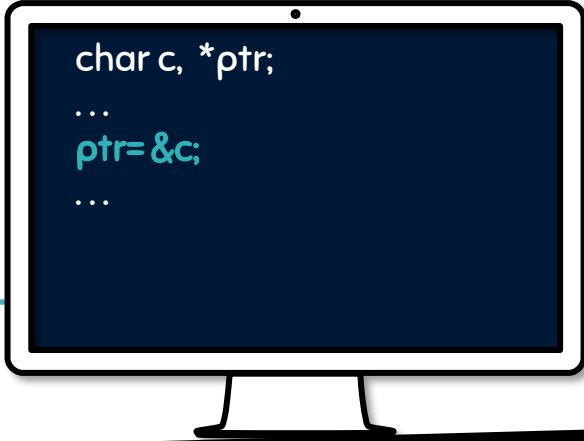


```
int *ptr;  
...  
ptr= 0;  
...  
ptr= NULL;
```

OPERACIONES CON PUNTEROS

ASIGNACIÓN

- Asignación de la dirección de una variable del tipo al que apunta el puntero:

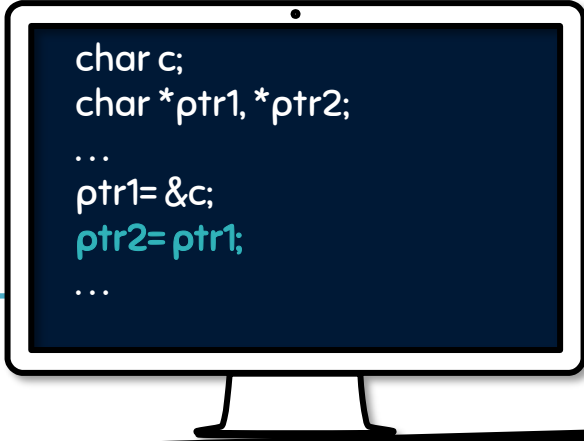


```
char c, *ptr;  
...  
ptr = &c;  
...
```

OPERACIONES CON PUNTEROS

ASIGNACIÓN

- Asignación del contenido de otro puntero:



```
char c;  
char *ptr1, *ptr2;  
...  
ptr1= &c;  
ptr2= ptr1;  
...
```


PREGUNTAS?

