

Algoritmos y Estructuras de Datos II

# PUNTEROS

---

2019

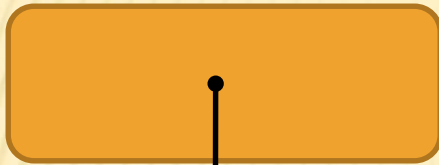
# ESTRUCTURAS DE DATOS DINÁMICAS

- ✖ Son estructuras flexibles que crecen a medida que se ejecuta un programa.
- ✖ Esta formada por una colección de elementos llamados nodos.
- ✖ La definición y manipulación de estos objetos se efectúa mediante un mecanismo conocido como puntero o apuntador, que permite al programa referirse directamente a la memoria.

# PUNTEROS

*Es un tipo de variable que en lugar de almacenar valores, almacena direcciones de memoria.*

PUNTERO



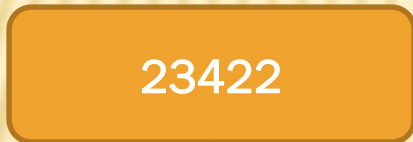
*Variable Referenciada*

DATO



v\_puntero

23422



Dirección: 23422

Ana






# PUNTEROS

---


- ✖ Las variables puntero contienen valores que son direcciones de memoria donde se almacenan datos.
- ✖ Tiene dos componentes:
  - + La dirección de memoria a la que apunta(contenido del puntero).
  - + El elemento referido (contenido de la celda de memoria cuya dirección esta contenida en el puntero).
- ✖ Mediante una variable de tipo puntero, se puede manipular tanto direcciones de memoria como valores.

# DECLARACIÓN DE PUNTEROS

*<tipo de dato apuntado> \* <identificador de puntero>*



*Hace referencia al tipo de datos  
al que apuntará nuestro puntero.*



*Nombre que asignaremos a  
nuestro puntero.*

## Ejemplos:

```
int *v_puntero;  
int *v_dni;  
char * v_opcion;
```

# OPERADOR & Y OPERADOR DE INDIRECCIÓN

- ✖ El **operador &** da la dirección de un objeto. Solo se aplica a objetos que están en memoria.
- ✖ El **operador de indirección**, cuando se aplica a un puntero, da acceso al objeto al que señala el apuntador.

## Ejemplo:

```
int v_x = 1, int v_y = 2, v_z[10];
```

```
int *v_apunta; //v_apunta es un apuntador a un entero
```

```
v_apunta = &v_x; //v_apunta ahora apunta a v_x
```

```
v_y = *v_apunta; //v_y es ahora igual a 1
```

```
*v_apunta = 0; //v_x es ahora igual a 0
```

```
v_apunta = &v_z[0]; //v_apunta ahora apunta a z[0]
```



# INICIALIZACIÓN

---

- ✗ Es preciso inicializar el puntero antes de su uso.
- ✗ Método estático:  
    int edad;  
    int \*apuntaEdad;  
    apuntaEdad = &edad;
- ✗ Método dinámico: se utilizan funciones de asignación de memoria como malloc() y free().

# MALLOC()

---

- ✖ Asigna un bloque de memoria que es el número de bytes pasados como argumentos.  
`v_puntero = malloc(tamaño en bytes);`

Ejemplo: `int *var;`  
`var = (int*)malloc(sizeof(int));`



# FREE()

---

- ✗ Permite liberar espacio de memoria y dejarlo disponible para otros usos, una vez que se ha terminado de utilizar un bloque de memoria previamente asignado por malloc().

**free(puntero)**

# EJEMPLO

```
#include <stdio.h>
void cambiar (int *p_x, int *p_y);
int main(void){
    int a, b;
    a = 3;
    b = 5;
    printf(" \nAntes de intercambiar: a = %d, y b = %d",a,b);
    cambiar(&a, &b); /* Se pasa la dirección de las dos variables */
    printf("\nDespués de intercambio: a = %d, y b = %d",a,b);
    return 0;
}

void cambiar(int *p_x, int *p_y){
    int aux;
    aux = *p_x; /* A través del puntero x, puedo acceder al valor original */
    *p_x = *p_y;
    *p_y = aux;
}
```

## **Resultado:**

- Antes de intercambiar: a = 3, y b = 5
- Después de intercambio: a = 5, y b = 3

# LINKS DE INTERÉS

- ✖ Programación en C – Apuntadores, parte 1.  
<https://www.youtube.com/watch?v=6GbGutblvb0>
- ✖ Programación en C – Uso básico de punteros.  
[https://www.youtube.com/watch?v=iW\\_zzg2ppvq&t=276s](https://www.youtube.com/watch?v=iW_zzg2ppvq&t=276s)
- ✖ Los punteros en C.  
[https://lsi.vc.ehu.eus/pablogn/docencia/manuales/C/Punteros\\_en\\_C.pdf](https://lsi.vc.ehu.eus/pablogn/docencia/manuales/C/Punteros_en_C.pdf)
- ✖ Diseño de algoritmos y su codificación en lenguaje C. María A. C. Nakamura, María de los A. A. Valdez.  
[https://docs.google.com/file/d/0B2yZ\\_Q2q5m6scEJDc25veGNWeVE/edit](https://docs.google.com/file/d/0B2yZ_Q2q5m6scEJDc25veGNWeVE/edit)