1. ¿Cuál es la diferencia entre int *p; **y** *p = 5;**?**

- int *p; → estás declarando un puntero a entero. Todavía no apunta a nada.
- *p = 5; → estás usando el puntero para escribir un valor en la dirección a la que apunta.

Importante: si no inicializaste el puntero (por ejemplo, p = &algo;), hacer *p = 5; es un error grave (puntero colgante).

---

2. ¿Por qué tengo que usar & **con** scanf**, pero no con** printf**?**

- scanf necesita modificar tu variable, por lo tanto necesita saber dónde está ubicada en memoria. Por eso se usa & (el operador de dirección).
- printf solo necesita leer el valor, no modificarlo, por eso no se usa &.

Ejemplo:

int edad;

scanf("%d", &edad); // correcto

printf("%d\n", edad); // correcto

---

3. ¿Un array siempre se comporta como puntero?

Sí, en la mayoría de los contextos un array se convierte en un puntero al primer elemento (es decir, arr == &arr[0]).

Sin embargo, hay algunas diferencias importantes, como por ejemplo con sizeof. Si arr es un array real, sizeof(arr) da el tamaño total del array; pero si es un puntero, sizeof(arr) da el tamaño del puntero.

En funciones:

void imprimir(int *arr, int size);

es equivalente a:

void imprimir(int arr[], int size);

---

4. ¿Puedo declarar una función que reciba un puntero y modificar el valor original?

Sí. Esa es una de las principales ventajas de usar punteros: permiten modificar directamente los valores originales de las variables pasadas.

Ejemplo:

void cambiar(int *x) {

***x = 100;**

}

Llamada:

int a = 5;

cambiar(&a);

// a ahora vale 100

---

5. ¿Qué pasa si paso un puntero sin inicializar?

Es un error muy común y peligroso.

Un puntero sin inicializar apunta a una dirección aleatoria en memoria, y acceder a ella con *p puede causar un error de ejecución o resultados impredecibles.

Ejemplo incorrecto:

int *p;

*p = 5; // ERROR: p no apunta a memoria válida

---

6. ¿char *str **es lo mismo que un string?**

Sí. En C, una cadena de caracteres (string) es un puntero a char que apunta al primer carácter de un array terminado en el carácter nulo '\0'.

Ejemplo:

char *str = "Hola"; // apunta a una cadena constante “LITERAL DE CADENA”

También válido:

char str[] = "Hola"; // array local con espacio editable

Diferencia importante: char *str = "Hola" apunta a una cadena de solo lectura (no deberías modificarla). En cambio, char str[] = "Hola" crea un array que sí podés editar.