# Resumen teórico
# Punteros

**Punteros:**

Un puntero es una variable que contiene una dirección de memoria. Por ejemplo, un puntero a entero es una variable que se utiliza para guardar la dirección de una variable tipo int.

```
int cuenta;
int *m;
int q;
m = &cuenta;  ← en m se guarda la dirección de "cuenta", no el valor.
q = *m;       ← q contiene el valor de cuenta. O sea el valor de la dirección
que apunta m.
```

**Declaración:**

```
int *punt;// declaración de un puntero a entero llamado "punt"
char *punt1;          // declaración de un puntero a char llamado "punt1"
float *punt2; // declaración de un puntero a float llamado "punt2"
```

Estas declaraciones reservan espacio para un puntero, pero no inicializan el puntero, es decir, están apuntando a cualquier parte.

**Operadores implicados:**

- **Operador de indirección "*"** Situado a la izquierda de una variable (puntero) devuelve la variable a la que apunta, o lo que es lo mismo, el contenido de la dirección que contiene.
- **Operador de dirección-de: "&"** Situado a la izquierda de una variable devuelve su dirección.

**Ejemplos:**

```
int *punt;//puntero a entero sin inicializar llamado punt
int x;// variable entera llamada x
int y;// variable entera llamada y
punt=&x;// Escribimos en punt la dirección de x, es decir, punt"apunta" a x
*punt=4;//Escribimos un 4 en donde apunta punt,es decir, escribimos un 4 en x
punt=&y;// Escribimos en punt la dirección de y, es decir, punt "apunta" a y
*punt=8; //Escribimos un 8 en donde apunta punt,es decir, escribimos un 8 en y
```

**La relación entre punteros y arrays**

El nombre de un array es la dirección del primer elemento del array. Esto se expresa mediante la siguiente fórmula:

**X [i] ≡ *(X + i)**

Esta fórmula es válida tanto si X es un array como si X es un puntero al primer elemento del array. La única diferencia de un lado de la ecuación al otro es diferencia de notación, es decir, en C podemos usar para arrays la notación de array (en el lado de la izquierda de la ecuación) ó la notación puntero (en el lado de la derecha), de forma totalmente indistinta.

**Operaciones con punteros**

Los elementos de un array están en posiciones contiguas de memoria, por este motivo, se permiten hacer operaciones con punteros, es decir, sumarle a un puntero un número, equivale a adelantarlo ese número de posiciones de memoria, lo mismo sucede con la resta.

Veamos algunos ejemplos:

```
int miarray[7];
int *punt;
punt=miarray;
*(punt +1)=3; // idem que miarray[1]=3 ;
punt=punt+3;//Ahora punt apunta al cuarto elemento del array.
*punt=5; // idem que miarray[3]=5;
miarray[6]=*(punt-2);// idem que miarray[6]= miarray[1];
```

**Paso de parámetros por referencia a funciones**

Hasta ahora las funciones solamente podían devolver un único valor, es decir, una función sólo podía modificar una única variable del ámbito desde el que se la llamaba (a la función). Los punteros nos permiten un nuevo uso de las funciones, podemos decirles dónde se encuentran nuestras variables (las variables del ámbito de la llamada) para que la función pueda modificarlas. A esto se le llama comúnmente "paso de parámetros por referencia", y consiste en dar como parámetro a una función, en vez de la variable, su dirección, es decir, un puntero a dicha variable.

Ejemplo:

```
Void suma_dos(int *x, int *y, int *z)
{
        *x=*x+2;
        *y=*y+2;
```

```
        *z=*z+2;
}

Void main(void){
        int x;
        printf("Introduzca tres numeros:");
        scanf("%d %d %d %d",&x, &y, &z);
        suma_dos (&x, &y, &z);
        printf("%d %d %d %d",x, y, z);// qué imprimirá??
}
```

**Aritmética de punteros:**

| | |
|---|---|
| `int *p;` | Es un puntero del tipo entero |
| `p++;` | Incrementa su valor y apunta al siguiente elemento. |
| `p--` | Apunta al elemento anterior |
| `p = p + 9` | Apunta al noveno elemento después de él. |

**NO PODEMOS:**

- Multiplicar punteros
- Dividir punteros
- Sumar punteros
- Aplicar desplazamiento a nivel de bits.

**Punteros y Arrays:**

```
charcad[80];
char *pl;
pl = cad;
```

**cad[4]**es lo mismo que ***(pl+4).** Ambas sentencias devuelven el quinto elemento de la cadena cad.

Tenemos que tener en cuenta que el primer elemento es cero, por eso con cuatro apuntamos al quinto elemento.

Algo equivalente pasa con pl, que ya apunta al primer elemento, por eso al sumarle cuatro ya estamos en el quinto.

Tener en cuenta que el nombre de un array sin índice devuelve la dirección de su primer elemento, al igual que un puntero.