

# PROGRAMACIÓN ESTRUCTURADA

UNIDAD 5

# Punteros

# ¿Qué son?

---

- Permiten simular el pasaje de parámetros por referencia.
- Permiten crear y manipular estructuras de datos dinámicas.
- Su manejo es de fundamental importancia para poder programar en C.
- Un puntero es una variable que contiene una dirección de memoria.
- Por lo general, una variable contiene un valor y un puntero a ella contiene la dirección de dicha variable.
- Es decir que la variable se refiere **directamente** a un valor mientras que el puntero lo hace **indirectamente**.

# ¿Para qué se usan?

---

- Utilizamos punteros para acceder a la información a través de su dirección de memoria

# Punteros

---

- Una dirección de memoria y su contenido no es lo mismo

```
int x = 25;
```



Dirección     1502    1504    1506    1508

...	...	25	...	...	...	...	
-----	-----	----	-----	-----	-----	-----	--

La **dirección** de la variable `x` es 1502

El **contenido** de la variable `x` es 25

# Punteros

---

- Cada puntero debe llevar su nombre precedido por \*.

```
int *countPtr; //puntero a un entero  
int count; //es un entero, no un puntero
```



- El \* no se aplica a todos los nombres de variables de una declaración. Cada puntero debe llevar su nombre precedido por \*

```
int *countPtr, count;
```

# Operadores de punteros

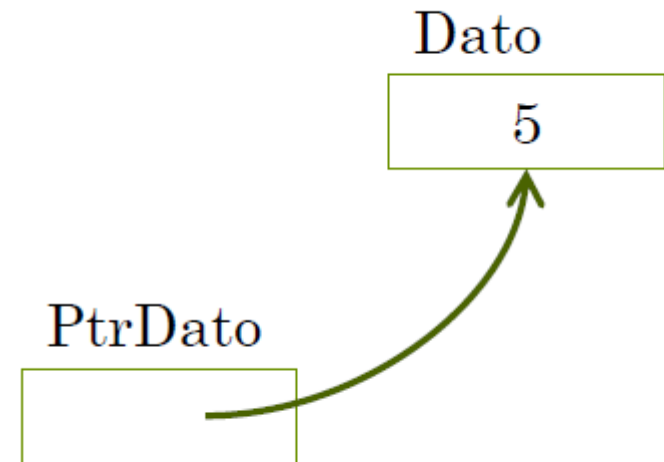
---

- El operador & u operador de dirección, es un operador unario que retorna la dirección de su operando

```
datoPtr = (int *) 0x1  
dato = 5
```

```
int *datoPtr, dato;  
  
dato=5;  
datoPtr=&dato;
```



# Operadores de punteros

---

- El operador \*, también llamado operador de indirección, retorna el valor del objeto hacia el cual apunta su operando

```
int *datoPtr, dato;  
  
dato=5;  
datoPtr=&dato;  
  
printf("%d",*datoPtr);
```



IMPRIME 5



# Operadores de punteros

---

- El puntero debe contener una dirección a un elemento del mismo tipo que la variable apuntada

```
int *datoPtr, dato;
```

→ Declaro un puntero a un entero

```
dato=5;  
datoPtr=&dato;
```

→ Obtengo la dirección de memoria de la variable dato

```
*datoPtr=10;
```

```
printf("%d", *datoPtr);
```

→ IMPRIME 10

# Operadores de punteros

---

- ¿Qué imprime?

```
int a,b,c, *p1, *p2;

p1=&a; //en p1 pongo la direccion de a
*p1=1; // en el puntero p1 pongo el valor 1

p2=&b; //en p2 pongo la direccion de b
*p2=2; // en puntero p2 pongo el valor 2

p2=&c; //en p2 pongo la direccion de c
*p2=3; // en el puntero p2 pongo el valor 3

printf("a-%d, b-%d, c-%d",a,b,c);
```

# Operadores de punteros

- ¿Qué imprime?

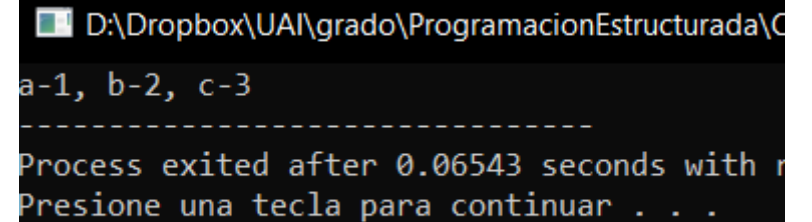
```
int a,b,c, *p1, *p2;

p1=&a; //en p1 pongo la direccion de a
*p1=1; // en el puntero p1 pongo el valor 1

p2=&b; //en p2 pongo la direccion de b
*p2=2; // en puntero p2 pongo el valor 2

p2=&c; //en p2 pongo la direccion de c
*p2=3; // en el puntero p2 pongo el valor 3

printf("a-%d, b-%d, c-%d",a,b,c);
```



```
D:\Dropbox\UAL\grado\ProgramacionEstructurada\C
a-1, b-2, c-3
-----
Process exited after 0.06543 seconds with r
Presione una tecla para continuar . . .
```

# Operadores de punteros

- Puede utilizarse printf con la especificación de conversión %p para visualizar el valor de una variable puntero en forma de entero hexadecimal

```
int dato=10, *datoPtr;  
datoPtr=&dato;  
printf("%p",datoPtr);
```



# Operadores de punteros

---

```
int dato=10, *datoPtr;  
  
datoPtr=&dato;  
  
printf("La direccion de dato es %p\n",&dato);  
printf("El valor de datoPtr es %p\n",datoPtr);  
  
printf("El valor de dato es %d\n",dato);  
printf("El valor de *datoPtr es %d\n",*datoPtr);
```

# Operadores de punteros

---

```
int dato=10, *datoPtr;  
  
datoPtr=&dato;  
  
printf("La direccion de dato es %p\n",&dato);  
printf("El valor de datoPtr es %p\n",datoPtr);  
  
printf("El valor de dato es %d\n",dato);  
printf("El valor de *datoPtr es %d\n",*datoPtr);
```

```
La direccion de dato es 00000000070fe14  
El valor de datoPtr es 00000000070fe14  
El valor de dato es 10  
El valor de *datoPtr es 10
```

# PARÁMETROS POR REFERENCIA

# Parámetros por referencia

---

- En C los parámetros de las funciones siempre se pasan por valor.
- Para simular el pasaje de parámetro por referencia se utiliza la dirección de la variable, es decir, que lo que se envía es un puntero a su valor.
- El puntero es un parámetro sólo de entrada que permite modificar el valor de la variable a la que apunta.



# Parámetros por referencia

```
1  #include <iostream>
2
3  void suma (int a, int b, int *resultado); |
4
5  int main(int argc, char** argv) {
6
7      int res=0;
8
9      printf("El valor de res es %d\n",res);
10     suma(10,20,&res);
11     printf("El de res es %d\n",res);
12
13 }
14
15 void suma (int a, int b, int *resultado){
16
17     *resultado= a+b;
18 }
```

Envío la dirección de la variable res

Recibe un puntero a un entero