Punteros

Parte 1: Declaración y uso

Definición

Un puntero es una variable que contiene una dirección de memoria, normalmente esa dirección es una posición de memoria de otra variable, por lo cual se suele decir que el puntero “apunta” a la otra variable.

Declaración de un puntero

En la declaración sólo le indicamos al compilador que reserve una posición de memoria para albergar la dirección de una variable , del **tipo** indicado.

```
tipo *  punteroTipo ;

int *   punteroInt ;

char *  punteroChar;

float * punteroFloat;
```

Uso de punteros

```
int*    punteroInt = NULL;
char* punteroChar = NULL;
float*  punteroFloat = NULL;
```

Uso de punteros

```
int*    punteroInt = NULL;
char* punteroChar = NULL;
float*  punteroFloat = NULL;
```

Nombre de la variable	Dirección de Memoria	Valor
punteroInt	0x00B0	NULL
punteroChar	0x00B2	NULL
punteroFloat	0x00B4	NULL

Uso de punteros

```
int*    punteroInt = NULL;
char* punteroChar = NULL;
float*  punteroFloat = NULL;

int     auxInt = 10;
char   auxChar = ‘a’;
float  auxFloat = 3.14;
```

Nombre de la variable	Dirección de Memoria	Valor
punteroInt	0x00B0	NULL
punteroChar	0x00B2	NULL
punteroFloat	0x00B4	NULL

Uso de punteros

```
int*    punteroInt = NULL;
char* punteroChar = NULL;
float*  punteroFloat = NULL;

int     auxInt = 10;
char   auxChar = ‘a’;
float  auxFloat = 3.14;
```

Nombre de la variable	Dirección de Memoria	Valor
punteroInt	0x00B0	NULL
punteroChar	0x00B2	NULL
punteroFloat	0x00B4	NULL

Nombre de la variable	Dirección de Memoria	Valor
auxInt	0x00C0	10
auxChar	0x00C2	‘a’
auxFloat	0x00C3	3.14

Operador de Dirección (&)

Este operador permite obtener la dirección de memoria de una variable.

```
int *    punteroInt ;    // Declaro un puntero a Int
int      variableInt ;   // Declaro una variable del tipo Int

punteroInt = &variableInt ; // Obtengo la posición de memoria

printf(“Posicion de Memoria: %p”, punteroInt)
```

Uso de punteros

```
int*    punteroInt = NULL;
char* punteroChar = NULL;
float*  punteroFloat = NULL;

int     auxInt = 10;
char   auxChar = ‘a’;
float  auxFloat = 3.14;

punteroInt = &auxInt ;
punteroChar = &auxChar;
punteroFloat = &auxFloat;
```

Nombre de la variable	Dirección de Memoria	Valor
punteroInt	0x00B0	0x00C0
punteroChar	0x00B2	0x00C2
punteroFloat	0x00B4	0x00C3

Nombre de la variable	Dirección de Memoria	Valor
auxInt	0x00C0	10
auxChar	0x00C2	‘a’
auxFloat	0x00C3	3.14

Operador de Indirección: (*)

Este operador devuelve el contenido en la posición apuntada por el puntero.

```
int *    punteroInt ;    // Declaro un puntero a Int
int      variableInt ;   // Declaro una variable del tipo Int
variableInt = 44;        // Asigno un valor a la variable
punteroInt = &variableInt ; // Obtengo la posición de memoria

printf(“Valor: %i”, *punteroInt)
```

Uso de punteros

```
int*    punteroInt = NULL;
char* punteroChar = NULL;
float*  punteroFloat = NULL;

int     auxInt = 10;
char   auxChar = ‘a’;
float  auxFloat = 3.14;

punteroInt = &auxInt ;
punteroChar = &auxChar;
punteroFloat = &auxFloat;

*punteroInt = 22;
*punteroChar = ‘b’;
*punteroFloat = 22.55;
```

Nombre de la variable	Dirección de Memoria	Valor
punteroInt	0x00B0	0x00C0
punteroChar	0x00B2	0x00C2
punteroFloat	0x00B4	0x00C3

Nombre de la variable	Dirección de Memoria	Valor
auxInt	0x00C0	22
auxChar	0x00C2	‘b’
auxFloat	0x00C3	22.55