

## SEMANA #3: PUNTEROS

Para iniciar con nuestro tema, recordemos que a nivel de Memoria, la memoria de nuestro equipo guarda la información en BITS, es decir solo puede contener 0 o 1. El conjunto de bits (8 bits) forman un byte. Cada grupo de byte va a tener una dirección de memoria asignada. Es aquí donde entran los punteros.

**Los punteros a nivel de C++ son variables que guardan la dirección de memoria asignada a una variable. Si declaramos una variable entera**

`int num= 20;` ya sabemos que su valor es 20 , pero estará guardada en una dirección de memoria.

Para saber cuál es la dirección de memoria donde esta guardada nuestra variable solicitamos

```
cout<< &num<<endl;
```

**El símbolo & es un operador de referencia que nos indica que dirección en memoria va a tener nuestra variable número.** Veamos el siguiente ejemplo

```
#include<iostream>
#include<stdlib.h>
using namespace std;

int main() {

    int num;
    num = 20;

    cout << "Numero: " << num << endl;
    cout << "Direccion de memoria: " << &num << endl;

    system("pause");
    return 0;
}
```

Al ejecutar este programa nos mostrara que el valor de la variable num es 20 y que su dirección en memoria es \_\_\_\_\_(en HEXADECIMAL)

C:\Users\Usuario\Desktop\Ejercicios Punteros\Ejercicio1\Debug\Ejercicio1.exe

```
Numero: 20
Direccion de memoria: 008FF7FC
Presione una tecla para continuar . . .
```

Retomando un puntero es una variable que va a almacenar la dirección de memoria. Para que un puntero guarde esa dirección vamos a declararlo **\*dir\_num** que en nuestro caso va a almacenar una variable de tipo entero llamada num.

```
Ejercicio2.cpp X
Ejercicio2 (Ámbito global) main()
// DECLARACION DE PUNTEROS
//      *n = la variable cuya variable esta almacena en n

#include<iostream>
#include<stdlib.h>

using namespace std;
int main() {
    int num, * dir_num; //asigno una variable puntero
    num = 20;
    dir_num = &num; //agregamos la direccion de memoria del numero

    cout << "Numero: " << *dir_num << endl;
    //muestra la posicion de memoria que se le asigno al puntero "20"

    cout << "Direccion de memoria: " << dir_num << endl;
    //muestra la direccion donde esta almacenada la variable

    system("pause");
    return 0;
}
```

C:\Users\Usuario\Desktop\Ejercicios Punteros\Ejercicio2\Debug\Ejercicio2.exe

```
Numero: 20
Direccion de memoria: 0058F7BC
Presione una tecla para continuar . . .
```

Al ejecutar nos dará como resultado:

## Correspondencia entre ARREGLOS y PUNTEROS

```
Ejercicio3 (Ámbito global)

//CORRESPONDENCIA ENTRE UN ARREGLO Y UN PUNTERO
#include<iostream>
#include<stdlib.h>

using namespace std;

int main() {
    int numeros[] = { 1,2,3,4,5 };//declaramos el arreglo
    int* dir_numeros;//variable puntero

    dir_numeros = numeros;//igualamos la direccion numeros a la
    //primera posicion de nuestro arreglo

    for (int i = 0;i < 5;i++) {
        cout << "Elemento del arreglo[" << i << "]:" << *dir_numeros++ << endl;

        /*dir_numeros++me permite mostrar todos los elementos del arreglo
        // *dir_numeros mostrara solo el primer elemento del arreglo
    }

    system("pause");
    return 0;
}
```

C:\Users\Usuario\Desktop\Ejercicios Punteros\Ejercicio3\Debug\Ejercicio3.exe

```
Elemento del arreglo[0]:1
Elemento del arreglo[1]:2
Elemento del arreglo[2]:3
Elemento del arreglo[3]:4
Elemento del arreglo[4]:5
Presione una tecla para continuar . . .
```

En cada arreglo el paso de un elemento a otro consume un espacio de memoria de 4 bytes, es decir cada vez que el arreglo recorre de la posición 1 a la 5 consume 4 bytes de memoria. Confirmemos esta teoría

```
Ejercicio4 (Ámbito global)

//RESPONDENCIA ENTRE UN ARREGLO Y UN PUNTERO
#include<iostream>
#include<stdlib.h>

using namespace std;

int main() {
    int numeros[] = { 1,2,3,4,5 };//declaramos el arreglo
    int* dir_numeros;//variable puntero

    dir_numeros = numeros;//igualamos la direccion numeros a la
    //primera posicion de nuestro arreglo

    for (int i = 0;i < 5;i++) {
        cout << "Posicion de memoria" << numeros[i] << "es" << dir_numeros++ << endl;

        //imprimimos la posicion de memoria dir_numeros, mostrara todas las
        //direcciones de memoria de cada uno de los elementos dl arreglo
    }

    system("pause");
    return 0;
}
```

C:\Users\Usuario\Desktop\Ejercicios Punteros\Ejercicio4\Debug\Ejercicio4.exe

```
Posicion de memoria1es001FF930
Posicion de memoria2es001FF934
Posicion de memoria3es001FF938
Posicion de memoria4es001FF93C
Posicion de memoria5es001FF940
Presione una tecla para continuar . . .
```

```
Ejercicio5.cpp* X
Ejercicio5 (Ámbito global)

/*Ejercicio 5 Comprobar si un número es par o impar, y señalar la posición de memoria
donde se está guardando el número. Con punteros.*/

#include<iostream>
#include<stdlib.h>
using namespace std;

int main() {
    int numero, * dir_numero;

    cout << "Digite un numero: "; cin >> numero; //Pedimos el numero al usuario

    dir_numero = &numero;

    if (*dir_numero % 2 == 0) {
        cout << "El numero: " << *dir_numero << " es par" << endl;
        cout << "Posicion: " << dir_numero << endl;
    }
    else {
        cout << "El numero: " << *dir_numero << " es impar" << endl;
        cout << "Posicion: " << dir_numero << endl;
    }

    system("pause");
    return 0;
}
```