Universidad Centroamericana “José Simeón Cañas”
Facultad de Ingeniería y Arquitectura

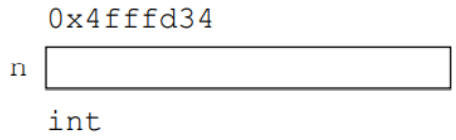
Tutorías - Fundamentos de Programación Ciclo 01/2021
Tutor: Ernesto José Canales Guillén

Referencias y Punteros en C++

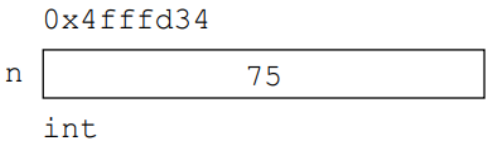
Direcciones y referencias

Cuando una variable se declara, se asocian tres atributos fundamentales con la misma: *su nombre, su tipo* y *su dirección en memoria*.

```
int n;      // asocia al nombre n, el tipo int y la dirección de
            // alguna posición de memoria donde se almacena el
            // valor de n
```



Esta caja representa la posición de almacenamiento en memoria. El nombre de la variable está a la izquierda de la caja, la dirección de variable está encima de la caja y el tipo de variable está debajo en la caja. Si el valor de la variable se conoce, se representa en el interior de la caja.



Al valor de una variable se accede por medio de su nombre.

```
cout << n;
```

A la dirección de la variable se accede por medio del operador de dirección &.

```
cout << &n;
```

¿Y por qué es útil conocer la dirección de la memoria?

Las referencias y punteros son importantes en C ++, porque le brindan la capacidad de manipular los datos en la memoria de la computadora, lo que puede reducir el código y mejorar el rendimiento.

Estas dos características son una de las cosas que hacen que C ++ se destaque de otros lenguajes de programación, como Python y Java.

Referencias

Una referencia es un alias de otra variable. Se declara utilizando el operador de referencia (&) que se añade al tipo de la referencia.

Las variables n y r tienen la misma dirección de memoria.

```
1 int n = 75;
2 int& r = n; // r es una referencia para n
3
4 cout << "&n = " << &n << ", &r =" << &r << endl; //&n = 0x4fffd34, &r = 0x4fffd34
```

Los dos identificadores n y r son nombres diferentes para la misma variable.

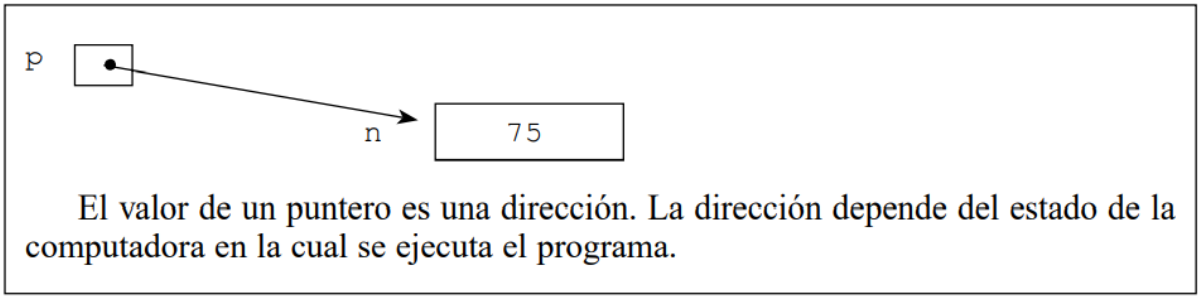
```
1 int n = 75;
2 int& r = n; // r es una referencia para n
3
4 cout << "n = " << n << ", r = " << r << endl; //n = 75, r = 75
```

Puntero (Apuntador)

Cada variable que se declara en C++ tiene una dirección asociada con ella. Un puntero es una dirección de memoria. El concepto de punteros tiene correspondencia en la vida diaria. Cuando se envía una carta por correo, su información se entrega basada en un puntero que es la dirección de esa carta.

Un puntero en C o en C++, también indica dónde encontrar algo, ¿dónde encontrar los datos que están asociados con una variable? Un puntero C++ es la dirección de una variable. Los punteros se rigen por estas reglas básicas:

- Un puntero es una variable como cualquier otra;
- Una variable puntero contiene una dirección que apunta a otra posición en memoria;
- En esa posición se almacenan los datos a los que apunta el puntero;
- Un puntero apunta a una variable de memoria.



El valor de un puntero es una dirección. La dirección depende del estado de la computadora en la cual se ejecuta el programa.

Declaración de punteros

Si se utiliza cuando se declara un puntero: (indirección)

```
<tipo de dato apuntado> *<identificador de puntero>
```

Indica que dicha variable se utilizará como un puntero (y no como una variable normal), es decir, su uso será almacenar la dirección en memoria de otra variable.

Si se utiliza junto a un puntero ya declarado: (desreferencia)

```
cout << (*puntero) << endl;
```

```
*puntero = 8;
```

Devuelve el valor contenido (o asigna uno nuevo) en la dirección apuntada por dicho puntero.

Cuando ya se ha definido un puntero, el asterisco delante de la variable puntero indica «el contenido de» de la memoria apuntada por el puntero y será del tipo dado. Este tipo de inicialización es estática, ya que la asignación de memoria utilizada para almacenar el valor es fijo y no puede desaparecer. Una vez que la variable se define, el compilador establece suficiente memoria para almacenar un valor del tipo de dato dado. La memoria permanece reservada para esta variable y no se puede utilizar para otra cosa durante la ejecución del programa. En otras palabras, no se puede liberar la memoria reservada para una variable. El puntero a esa variable puede ser cambiado, pero la cantidad de memoria reservada permanecerá.

```
1 string food = "Pizza";
2 string* ptr = &food;
3
4 cout << food << "\n"; // Output the value of food (Pizza)
5 cout << &food << "\n";// Output the memory address of food (0x6dfed4)
6
7 //Dereference
8 cout << *ptr << "\n";// Access the memory address of food and output its value (Pizza)
9
10 //Dereference
11 *ptr = "Hamburger";// Change the value of the pointer
12
13 //Dereference
14 cout << *ptr << "\n";// Output the new value of the pointer (Hamburger)
15 cout << food << "\n";// Output the new value of the food variable (Hamburger)
```

Operador	Propósito
&	Obtiene la dirección de una variable.
*	Declara una variable como puntero.
*	Obtiene el contenido de una variable puntero.

Punteros null y void

Normalmente, un puntero inicializado adecuadamente apunta a alguna posición específica de la memoria. Sin embargo, un puntero no inicializado, como cualquier variable, tiene un valor aleatorio hasta que se inicializa el puntero. En consecuencia, será preciso asegurarse que las variables puntero utilicen direcciones de memoria válida.

Existen dos tipos de punteros especiales muy utilizados en el tratamiento de sus programas: Los punteros void y null (nulo).

Un puntero nulo no apunta a ninguna parte —objeto válido— en particular, es decir, «un puntero nulo no direcciona ningún dato válido en memoria». Un puntero nulo se utiliza para proporcionar a un programa un medio de conocer cuándo un puntero apunta a una dirección válida. Los punteros nulos se utilizan con frecuencia en programas con arrays de punteros.

```
int* puntero = NULL;
```

En C++ se puede declarar un puntero de modo que apunte a cualquier tipo de dato, es decir, no se asigna a un tipo de dato específico. El método es declarar el puntero como un puntero void *.

```
void* puntero;
```

NOTA: No confundir punteros void y null. Un puntero nulo no direcciona ningún dato válido. Un puntero void direcciona datos de un tipo no especificado. Un puntero void se puede igualar a nulo si no se direcciona ningún dato válido. Nulo es un valor; void es un tipo de dato.