|  |  |  |
| --- | --- | --- |
| Universidad Centroamericana “José Simeón Cañas” | |  |
| Facultad de Ingeniería y Arquitectura | |
|  | |
|  | |
| Tutorías - Fundamentos de Programación | Ciclo 01/2021 |
| Tutor: Ernesto José Canales Guillén | |

**Introducción a la programación en C++**

**Operadores de incremento y disminución**

C ++ proporciona el operador de incremento, ++, que aumenta el valor de una variable en 1, y el operador de decremento, --, que reduce el valor de una variable en 1. Los operadores de incremento y decremento tienen cada uno dos formas, pre y post.

La sintaxis del operador de incremento es:

**Pre-incremento:** ++variable

**Post-incremento:** variable ++

La sintaxis del operador de decremento es:

**Pre-decremento:** --variable

**Post-decremento:** variable--

Debido a que los operadores de incremento y decremento están integrados en C ++, el valor de la variable se incrementa o decrementa rápidamente sin tener que usar la forma de una sentencia de asignación. Ahora, los operadores de incremento previo y posterior incrementan el valor de la variable en 1. De manera similar, los operadores de decremento previo y posterior disminuyen el valor de la variable en 1.

¿Cuál es la diferencia entre las formas pre y post de estos operadores?

La diferencia se hace evidente cuando la variable que utiliza estos operadores se emplea en una expresión: Suponga que x es una variable int. Si ++ x se usa en una expresión, primero el valor de x es incrementado en 1, y luego el nuevo valor de x se usa para evaluar la expresión. Por otro lado, si se usa x ++ en una expresión, primero se usa el valor actual de x en la expresión, y luego el valor de x se incrementa en 1.

El siguiente ejemplo aclara la diferencia entre los operadores previos y posteriores al incremento. Suponga que x e y son variables de tipo int. Considere las siguientes declaraciones:

x = 5;

y = ++ x;

La primera declaración asigna el valor 5 a x. Para evaluar la segunda declaración, que utiliza el operador de pre-incremento, primero el valor de x se incrementa a 6, y luego este valor, 6, se asigna a y. Después de que se ejecuta la segunda instrucción, tanto x como y tienen el valor 6.

Ahora, considere las siguientes declaraciones:

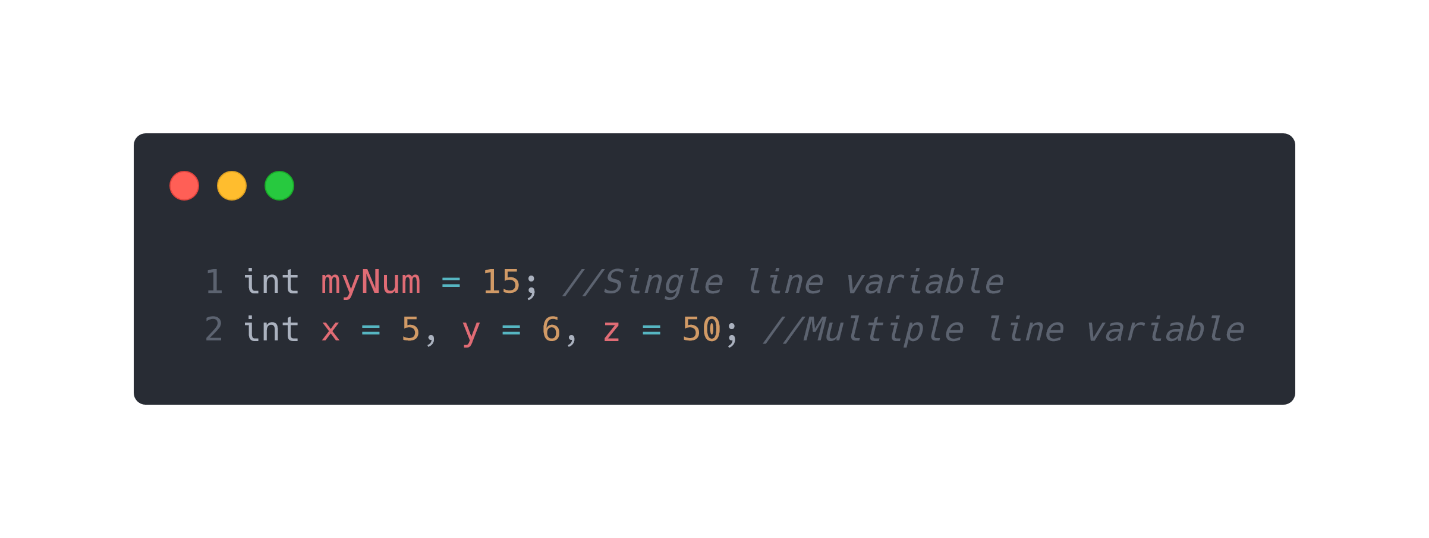
x = 5;

y = x ++;

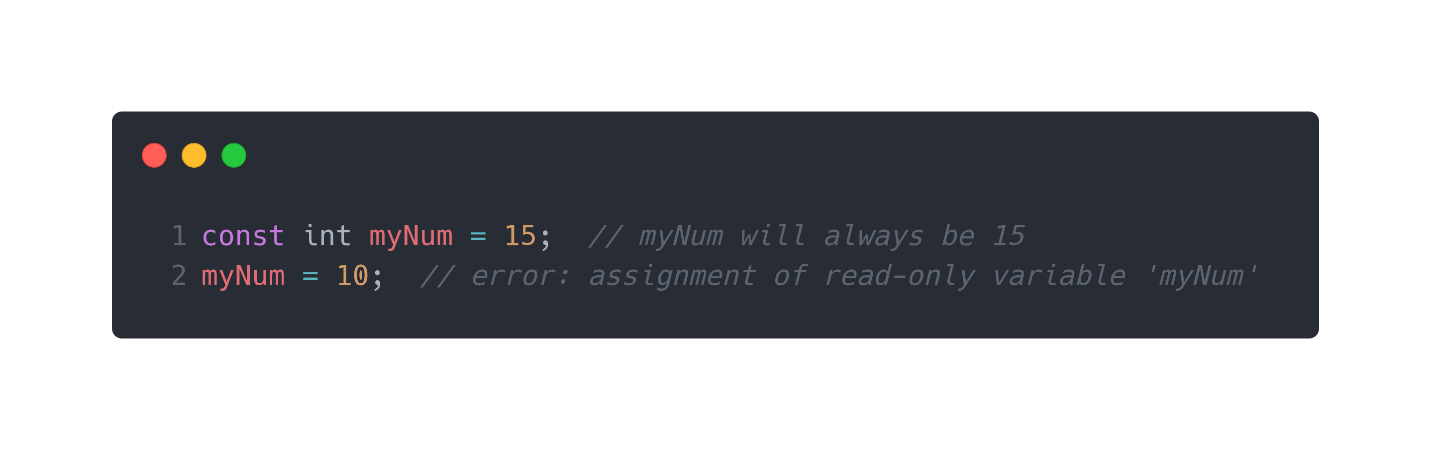
Como antes, el primer enunciado asigna 5 a x. En la segunda declaración, el incremento posterior el operador se aplica a x. Para ejecutar la segunda instrucción, primero el valor de x, que es 5, se usa para evaluar la expresión, y luego el valor de x se incrementa a 6. Finalmente, el valor de la expresión, que es 5, se almacena en y. Después de que se ejecuta la segunda declaración, el valor de x es 6 y el valor de y es 5.

**Variables, declaraciones de asignación, y declaraciones de entrada / salida**

Las variables son contenedores para almacenar valores de datos. Las variables pueden ser declaradas en una sola línea o en varias líneas:



Además de usar variables, C++ nos permite usar constantes; cuando no desee que otros (o usted mismo) anulen los valores de las variables existentes, use la palabra clave ***const*** (esto declarará la variable como "constante", lo que significa inmutable y de solo lectura).



**¿Cómo identificar nuestras variables, constantes, funciones, etc… en C++?**

Todas las variables de C ++ deben identificarse con nombres únicos.

* Las reglas generales para construir nombres para variables (identificadores únicos) son:
* Los nombres pueden contener letras, dígitos y guiones bajos.
* Los nombres deben comenzar con una letra o un guión bajo (\_).
* Los nombres distinguen entre mayúsculas y minúsculas (myVar y myvar son variables diferentes).
* Los nombres no pueden contener espacios en blanco ni caracteres especiales como !, #,%, etc…
* Las palabras reservadas no se pueden usar como nombres.

Para evitar confusiones y hacer mas legibles los bloques de código, los programadores han adaptado “Naming Conventions” para escribir variables, unas de las mas usadas son las siguientes:

* Camel Case:
  + La primera letra de cada palabra está en mayúscula sin espacios ni símbolos entre palabras.
* Pascal Case:
  + Popularizado por el lenguaje de programación Pascal, este es un subconjunto de Camel Case donde la palabra comienza con mayúsculas.
* Snake Case:
  + Las palabras dentro de frases o palabras compuestas se separan con un guión bajo.
* Kebab Case:
  + Como Snake Case, pero usando guiones en su lugar.
* Screaming Case:
  + Esto se refiere a nombres en mayúsculas.
* Hungarian Notation:
  + Los nombres comienzan con un prefijo en minúsculas para indicar la intención. El resto del nombre está en Pascal Case. Viene en dos variantes:
    - Sistemas húngaros, donde el prefijo indica el tipo de datos;
    - Aplicaciones húngaro, donde el prefijo indica un propósito lógico.

