



Introducción a la programación en C++

Declaración e inicialización de variables

Cuando se declara una variable, es posible que C ++ no le asigne automáticamente un valor significativo. En otras palabras, es posible que C ++ no inicialice variables automáticamente. Por ejemplo, `int` y es posible que las variables dobles no se inicialicen a 0, como sucede en algunos lenguajes de programación. Sin embargo, esto no significa que no haya ningún valor en una variable después de su declaración. Cuando se declara una variable, se le asigna memoria.

Recuerde que la memoria principal es una secuencia ordenada de celdas, y cada celda es capaz de almacenar un valor. Además, recuerde que el lenguaje de máquina es una secuencia de ceros y 1 o bits. Por lo tanto, los datos de una celda de memoria son una secuencia de bits. Estos bits no son nada, pero señales eléctricas, por lo que cuando la computadora está encendida, algunos de los bits son 1 y algunos son 0. El estado de estos bits depende de cómo funciona el sistema. Sin embargo, cuando instruir a la computadora para almacenar un valor particular en una celda de memoria, los bits se establecen según los datos almacenados.

Durante la manipulación de datos, la computadora toma el valor almacenado en celdas particulares y realiza un cálculo. Si declara una variable y no almacena un valor en ella, la memoria la celda todavía tiene un valor, generalmente el valor de la configuración de los bits de su último uso, y no tiene forma de saber cuál es este valor.

Si solo declara una variable y no le indica a la computadora que coloque datos en la variable, el valor de esa variable es basura. Sin embargo, la computadora no nos advierte, cualquier valor que esté en la memoria como legítimo, y realiza cálculos utilizando esos valores en memoria. Usar una variable en una expresión sin inicializarla produce errores resultados. Para evitar estos errores, C++ le permite inicializar variables mientras se están declarado.

Tipo de dato: string


La cadena de tipo de datos es un tipo de datos definido por el programador. No está disponible directamente para utilizarlo en un programa como los tipos de datos simples discutidos anteriormente. Para utilizar este tipo de datos, necesidad de acceder a los componentes del programa desde la biblioteca, que se discutirá más adelante en este. La cadena de tipo de datos es una característica del estándar C ++ ANSI / ISO.

Antes del estándar de lenguaje ANSI / ISO C ++, la biblioteca estándar de C ++ no proporcionaba un tipo de datos de cadena. Los proveedores de compiladores a menudo proporcionaban su propia cadena definida por el programador tipo, y la sintaxis y semántica de las operaciones de cadena a menudo variaba de un proveedor a otro.

Una cadena es una secuencia de cero o más caracteres. Las cadenas en C ++ se encierran en doble comillas. Una cadena que no contiene caracteres se denomina cadena nula o vacía. La los siguientes son ejemplos de cadenas. Tenga en cuenta que "" es la cadena vacía.

Cada carácter de una cadena tiene una posición relativa en la cadena. La posición del primero carácter es 0, la posición del segundo carácter es 1, y así sucesivamente. La longitud de una cuerda es el número de caracteres que contiene.

El tipo de cadena es muy poderoso y más complejo que los tipos de datos simples. Proporciona muchas operaciones para manipular cadenas. Para usar cadenas, debe incluir un archivo de encabezado adicional en el código fuente, la biblioteca **<string>**:



```
1 #include <string> // Include the string library
2
3 string greeting = "Hello"; // Create a string variable
```

El operador + se puede usar entre cadenas para sumarlas y crear una nueva cadena. Esto se llama *concatenación*.

```
1 string firstName = "John ";
2 string lastName = "Doe";
3
4 string fullName = firstName + lastName; //Basic concatenation
5
6 cout << fullName;
```

String en C ++ es en realidad un objeto, que contiene funciones que pueden realizar ciertas operaciones en strings. Por ejemplo, también puede concatenar cadenas con la función *append()*.

```
1 string firstName = "John ";
2 string lastName = "Doe";
3
4 string fullName = firstName.append(lastName); //Using append
5
6 cout << fullName;
```

Depende de usted si desea usar + o *append()*. La principal diferencia entre los dos es que la función *append()* es mucho más rápida. Sin embargo, para pruebas y demás, podría ser más fácil usar +.

(C ++ usa el operador + tanto para la suma como para la concatenación. Se agregan números. Las cadenas están concatenadas.)

Para obtener la longitud de una cadena, use la función *length()*:

```
1 string txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";  
2 cout << "The length of the txt string is: " << txt.length();
```

Es posible que vea algunos programas en C++ que usan la función *size()* para obtener la longitud de una cadena. Este es solo un alias de *length()*. Depende completamente de usted si desea usar *length()* o *size()*

```
1 string txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";  
2 cout << "The length of the txt string is: " << txt.size();
```

Puede acceder a los caracteres de una cadena consultando su número de índice entre corchetes []. Este ejemplo imprime el primer carácter en myStrings: (Los índices de cadena comienzan con 0: [0] es el primer carácter. [1] es el segundo carácter, etc.)

```
1 string myString = "Hello";  
2 cout << myString[0]; // Outputs H
```

Para cambiar el valor de un carácter específico en una cadena, consulte el número de índice y use comillas simples:



```
1 string myString = "Hello";  
2  
3 myString[0] = 'J';  
4  
5 cout << myString; // Outputs Jello instead of Hello
```

Directivas del pre-procesador (Preprocessor Directives)

Sólo un pequeño número de operaciones, como operaciones aritméticas y de asignación, son definidos explícitamente en C ++. Muchas de las funciones y símbolos necesarios para ejecutar C++ El programa se proporciona como una colección de bibliotecas. Cada biblioteca tiene un nombre y es referida a por un archivo de encabezado.

Si desea utilizar funciones de E/S o matemáticas, necesita para decirle a la computadora dónde encontrar el código necesario. Usas directivas de preprocesador y los nombres de los archivos de encabezado para indicarle a la computadora las ubicaciones del código proporcionado en Bibliotecas. Las directivas de preprocesador son procesadas por un programa llamado preprocesador.

Las directivas de preprocesador son comandos suministrados al preprocesador que hacen que el preprocesador modifique el texto de un programa C ++ antes de compilarlo. Todo preprocesador los comandos comienzan con #. No hay punto y coma al final de los comandos del preprocesador. porque no son declaraciones de C ++. Para usar un archivo de encabezado en un programa C ++, use la directiva del preprocesador *include*.

#include <headerFileName>

Las directivas del preprocesador para incluir archivos de encabezado se colocan como la primera línea de un programa, por lo que los identificadores declarados en esos archivos de encabezado se pueden usar en todo el programa. (Recuerde que en C ++, los identificadores deben declararse antes de que puedan usarse).

namespace y Using cin y cout en un programa

Anteriormente, aprendió que tanto cin como cout son identificadores predefinidos. En ANSI / ISO C ++ estándar, estos identificadores se declaran en el archivo de encabezado iostream, pero dentro de un espacio de nombres. El nombre de este espacio de nombres es std.

Hay varias formas de utilizar un identificador declarado en el espacio de nombres std. Una forma de usar cin y cout es referirse a ellos como *std::cin* y *std::cout* a lo largo del programa. Otra opción es incluir la siguiente declaración en su programa:

using namespace std;



```
1 //with using namespace std;
2 using namespace std;
3 string myVariable = "Bye"
4
5 cout << "Hi";
6 cin >> myVariable;
7
8 //////////////////////////////////////
9
10 //without using namespace std;
11 std::string myVariable = "Bye"
12
13 std::cout << "Hi";
14 std::cin >> myVariable;
```