Fundamentos de Programación 101

By Ernie

Ernesto José Canales Guillén

Círculos de estudio UCA

Ciclo Virtual 01/2021

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.

- 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

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.

- La cadena de tipo de datos es una característica del estándar C++ ANSI/ISO.

- 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.

string

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
```

Concatenar con +

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;
```

Concatenar con append()

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;
```

Largo de una cadena con length()

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();
```

Largo de una cadena con size()

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();
```

Acceder a un caracter específico

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
```

Modificar a un caracter específico

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
```

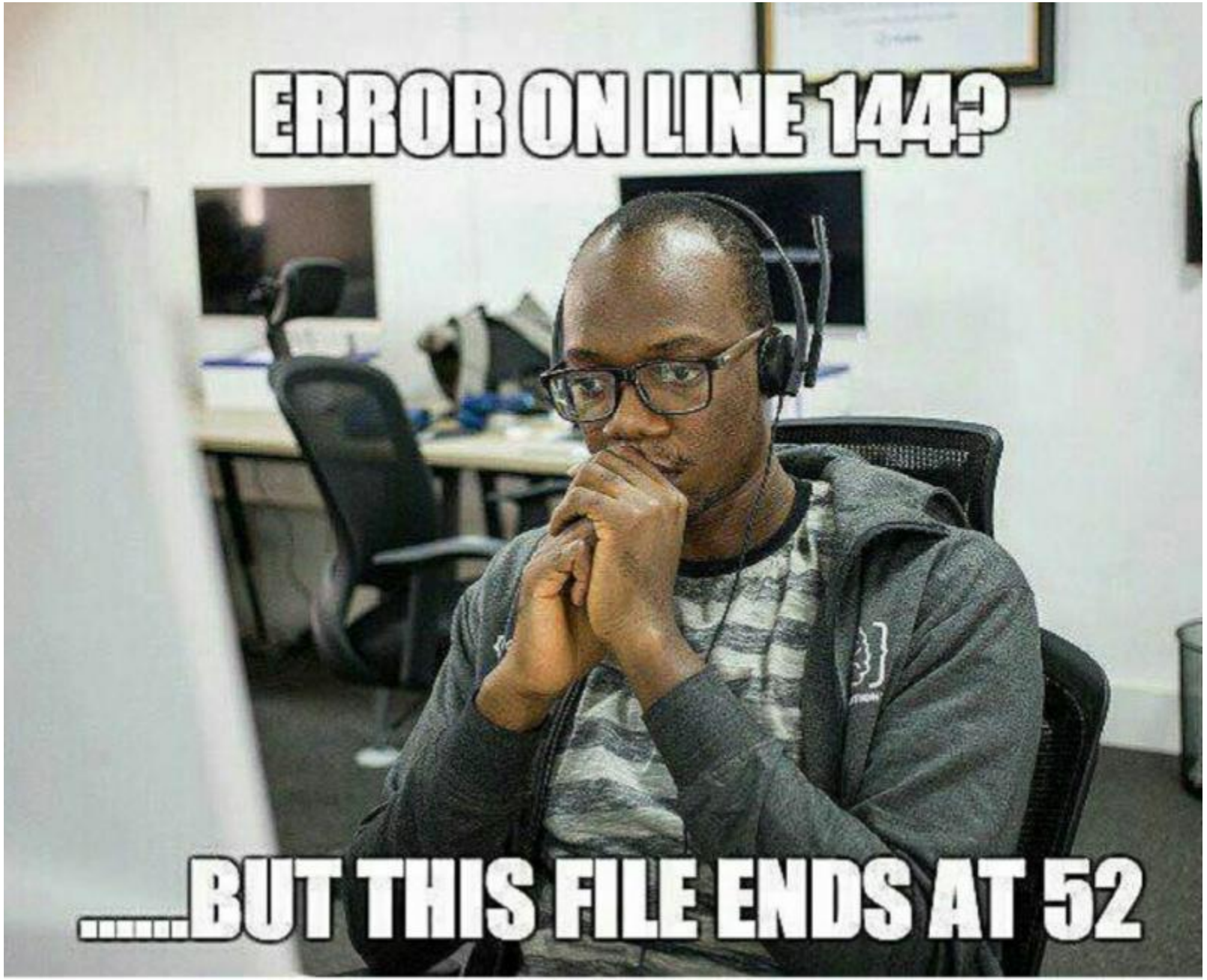
Preprocessor Directives

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.

- 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>


ERROR ON LINE 144?
.....BUT THIS FILE ENDS AT 52

namespace y Using cin y cout en un programa

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;
```

```cpp
//with using namespace std;
using namespace std;
string myVariable = "Bye"

cout << "Hi";
cin >> myVariable;

////////////////////////////////

//without using namespace std;
std::string myVariable = "Bye"

std::cout << "Hi";
std::cin >> myVariable;
```

Referencias para los “Preprocessor Directives”

https://en.cppreference.com/w/

https://www.cplusplus.com/

Bibliografía

- L. J. Aguilar, Programación en C++. Algoritmos, estructuras de datos y objetos, Aravaca (Madrid): McGRAW-HILL, 2006.

- D. Malik, C++ Programming: From Problem Analysis to Program Design, Boston, MA: Cengage Learning, 2003.