COMENTARIOS, IDENTIFICADORES Y TIPOS DE DATOS EN C++.

Autores: MsC. Marcos Antonio León Fonseca. mleonf@udg.co.cu
MsC. Noralys Muñiz Maldonado. nmunizm@udg.co.cu
MsC. Virgilio Herrera Rondón. vherrerah@udg.co.cu

Resumen:

En este artículo el autor describe como se utiliza en la programación Dev-C++ los comentarios, identificadores y los tipos de datos que le permiten al programador hacer declaraciones simplificadas de variables y nuevas formas de realizar las operaciones de entrada y salidas.

Introducción:

El lenguaje Dev-C++, mantiene ventajas en cuanto a riqueza de operadores y expresiones, flexibilidad, concisión y eficiencia. Además, ha eliminado algunas de las dificultades y limitaciones del C original. La evolución de C++ ha continuado con la aparición de Java, un lenguaje creado simplificando algunas cosas de C++ y añadiendo otras, que se utiliza para realizar aplicaciones en Internet.

La tarea de la programación es utilizar las facilidades de los lenguajes de programación, de forma que se simulen los conceptos del problema. Es por ello que, a pesar de que a nivel de equipamiento y de código de máquina los datos son representados por secuencias de dígitos binarios, los lenguajes de programación permiten el uso de abstracciones, e ignoran los detalles de representación de los datos de modo que pueda reducirse la distancia con el problema.

Este trabajo tiene como propósito describir como se utiliza los comentarios, identificadores y tipos de datos en el lenguaje de programación Dev-C++.

Desarrollo:

Comentarios

Es importante reconocer algunas de las características que intervienen en este programa, una de ellas es el uso de explicaciones que independientemente que no son interpretadas por el programa le permiten al programador leer y comprender el mismo.

Los programadores insertan comentarios para documentar los programas y mejorar la legibilidad de los mismos. Al ejecutarse el programa, los comentarios no hacen que la computadora realice ninguna acción. Los comentarios serán ignorados por el compilador C y no harán que se genere ningún código objeto en la máquina. El comentario en C simplemente describe el objetivo del programa. Los comentarios también ayudan a otras personas a leer y comprender su programa, pero demasiados comentarios podrían hacer que un programa sea difícil de leer.

Ejemplos:

C++ soporta dos tipos de comentarios:

1. Viejo estilo C: Entre **/*** y ***/**.

/* Esto es un comentario simple. */
/* Esto es un comentario más largo, distribuido en varias líneas. El texto se suele alinear por la izquierda. */

/**********************************
* Esto es un comentario de varias *
* líneas, encerrado en una caja para *
* llamar la atención. *
**********************************/
La ventaja de este nuevo método es que no se pueden comentar inadvertidamente varias líneas de un programa abriendo un indicador de comentario que no se cierre en el lugar adecuado.

2. Nuevo estilo C++: Comienzan con **//**. Su efecto termina cuando se alcanza el final de la línea actual.

// Esto es un comentario simple.
// Esto es un comentario más largo,
// Distribuido en varias líneas. El
// Texto se suele intentar por la izquierda.

Identificadores

Son los nombres elegidos para las variables, constantes, funciones, clases y similares. El primer carácter debe ser una letra o un subrayado. El resto del nombre puede contener dígitos. Los identificadores que comienzan con dos subrayados están reservados para uso interno del compilador C++.

Estos identificadores permiten que el programa se entienda más fácilmente, dando un significado a cada valor de la variable entera. Las variables tipo **enum** son adecuadas para representar de distintas formas valores binarios (SI o NO; VERDADERO o FALSO; EXITO o FRACASO, etc.), los días de la semana (LUNES, MARTES, MIERCOLES, ...), los meses del año (ENERO, FEBRERO, MARZO, ...), y cualquier conjunto análogo de posibles valores. En C las variables de tipo **enum** se hacían corresponder con enteros, y por tanto no hacían nada que no se pudiera hacer también con enteros.

En C++ las variables **enum** son verdaderos tipos de variables, que necesitan un **cast** para que un valor entero les pueda ser asignado (ellas son promovidas a enteros cuando hace falta de modo automático). Esto quiere decir que si una función espera recibir como argumento un tipo **enum** sólo se le puede pasar un entero con un **cast**. Por el contrario, si espera recibir un entero se le puede pasar un valor **enum** directamente.

Ejemplos

Constantes

C++ permite utilizar varios tipos de constantes:

1. Constantes enteras **44 0 -345**
2. Constantes enteras muy grandes. Se identifican situando una **L** al final de la constante entera **33L -105L**
3. Constantes octales o hexadecimales. Un **0** a la izquierda indica una constante octal y un **0x** o bien **0X** indican una constante hexadecimal 0 02 077 0123 equivalen a 0 2 63 83 en octal 0x0 0x2 0x3F 0x53 equivalen a 0 2 63 83 en hexadecimal
4. Constantes reales (coma flotante) 0.0 3.1416 -99.2 C++ permite especificar constante de coma flotante de simple precisión (sufijo f o F) y doble precisión larga (sufijo l o L). 32.0f 3.1416L
5. Constantes carácter 'z' '5'
6. Constantes cadena "hola" "hoy es lunes"

Tipos de datos

C++, igual que C, contiene tipos fundamentales y tipos derivados o estructurados.

Los fundamentales son: **int**, **char**, **long int**, **float**, **double**, **long double**.

- **Tipo vacío**. El tipo vacío (void) se utiliza principalmente para especificar:

* Funciones que no devuelven valores.

* Punteros void, que referencian a objetos cuyo tipo es desconocido.

- **Tipos enumerados**. Un tipo enumerado o enumeración está construido por una serie de constantes simbólicas enteras. Los tipos enumerados se tratan de modo ligeramente diferente en C++ que en ANSI C. El nombre de la etiqueta **enum** se considera como un nombre de tipo igual que las etiquetas de **struct** y **union**. Por tanto se puede declarar una variable de enumeración, estructura o **union** sin utilizar las palabras **enum**, **strcut** o **union**.

C define el tipo de **enum** de tipo **int**. En C++, sin embargo, cada tipo enumerado es su propio tipo independiente. Esto significa que C++ no permite que un valor **int** se convierta automáticamente a un valor **enum**. Sin embargo, un valor enumerado se puede utilizar en lugar de un int.

Ejemplo:

```
enum lugar {primero, segundo, tercero};

lugar pepe =primero; //correcto

int vencedor = pepe; //correcto

lugar Juan =1; //incorrecto
```

La última sentencia de asignación es aceptable en C pero no en C++, ya que 1 no es un valor definido en lugar.

- **Tipos referencia**. Las referencias son como alias. Son alternativas al nombre de un objeto. Se define un tipo referencia haciéndole preceder por el operador de dirección **&**. Un objeto referencia, igual que una constante debe ser inicializado.

int a=50;

int &refa=a; //correcto

int &ref2a; //incorrecto: no inicializado

Todas las operaciones efectuadas sobre la referencia se realizan sobre el propio objeto:

refa+=5; equivale a sumar 5 a a, que vale ahora 55

int *p=&refa; inicializa p con la dirección de a

Conclusiones:

El conocimiento de la forma de describir como se utiliza en la programación Dev-C++ los comentarios, identificadores y los tipos de datos le permiten al programador hacer declaraciones simplificadas de variables, nuevas formas de realizar las operaciones de entrada y salidas y construir estructuras de datos flexibles y complejas.

Bibliografía:

Katrib mora, miguel . Lenguajes de programación y Técnicas de compilación/ Miguel Katrib Mora.-- Ciudad de la Habana: Editorial Pueblo y Educación, 1986.-- 421 p.

GOTTFRIED, BYRON S. Programación en Pascal/ Byron S Gottfried.—Ciudad de la Habana: Editorial Pueblo y Educación, 1989.—397 p.

LIPSCHUTZ, SEYMOUR. Estructura de datos/ Syemour Lipschutz.—Ciudad de la Habana: Edición Revolucionaria, 1989.—390 p.