Manuel Arturo Chinchilla Sánchez - 2013009344

Resumen #1

Introduction/Basics of C++, Control Structures, Compound data Types - Tutorial C++

Páginas: 5-82

Podemos definir el término “variable” como una porción de memoria que almacena un determinado valor, cada variable necesita un nombre por identificador, que las distinga de otras variables, una buena práctica de programación es usar nombres representativos para nombrar a las variables.

Un identificador válido se una secuencia de una o más letras, dígitos o guiones bajos, no pueden haber espacios o signos de puntuación, ni símbolos. Los identificadores de variables tienen que iniciar siempre con una letra, o también pueden iniciar con guiones bajos. Una regla muy importante, es que ningún identificador de variable puede coincidir con palabras reservadas del lenguaje C++.

En C++ podemos encontrar diversos tipos de datos, como por ejemplo: los de tipo Integer, utilizados para almacenar datos de tipo numérico, los String, para almacenar conjuntos de letras, etc. Cada tipo difiere en la cantidad de memoria que se reserva para almacenar el valor y en el tipo de dato que puede almacenar.

Para declarar una variable en C++, primero debemos declarar el tipo de dato que va a almacenar la variable, para reservar el campo en la memoria, seguido del identificador de la variable, por ejemplo, vamos a declarar una variable de tipo Integer con el identificador “primero” y terminamos la línea con un punto y coma: *int primero;.*

Además podemos clasificar las variables en dos grandes grupos, de acuerdo al alcance de las mismas, las de tipo global, se declaran fuera de los métodos de nuestro código y pueden ser accedidas desde cualquier parte del código, y las de tipo local, las cuales se declaran dentro de un método, y solo puede ser accedido dentro del método en el cual se declaró.

Cuando declaramos variables, si no las inicializamos, el valor que almacena no puede ser determinado, ya que solo se está reservando un espacio en la memoria que puede que ya contenga algún tipo de dato (basura), por ello, para evitar esto, lo recomendable es inicializar nuestras variables con un valor por defecto, y evitamos este tipo de errores.

Cuando inicializamos una variable podemos hacerlo de dos formas:

Tipo identificador = valor_por_defecto;

Tipo identificador (valor_por_defecto);

Por defecto la salida básica de un programa en C++ es la pantalla, y se hace de la siguiente forma: cout << “Texto a mostrar”;

El dispositivo estándar de entrada usualmente es el teclado, y se hace de la siguiente forma*: int edad; cin >> age;* se inicializa una variable y luego se le asigna el valor ingresado por el teclado.

Usando funciones podemos estructurar nuestros programas de una manera más modular, accesando a todo el potencial que la programación estructurada de C++ nos puede ofrecer. Una función es un grupo de sentencias que se ejecutan cuando la función es llamada desde algún punto del programa. Tiene la forma:

Tipo Nombre (parametro1, parametro2, ...){sentencias de la función}

Además podemos declarar funciones con el tipo “void”, usada normalmente para cuando la función no retorna ningún tipo de dato.

Un arreglo es un tipo de estructura en el cual una serie de datos de un mismo tipo, son almacenados en ubicaciones contiguas de memoria, que pueden ser referenciadas individualmente por un índice a un único identificador.

Cuando declaramos un arreglo lo hacemos de la siguiente forma:

Tipo nombre [elementos];

Donde el tipo es el tipo de datos que vamos a almacenar, nombre es el identificador del arreglo, y los elementos son la cantidad de elementos que vamos a introducir en el arreglo. Para accesar los valores del arreglo, usamos el identificador del arreglo y un índice con la posición del valor a accesar, de la siguiente forma: identificador[indice]; donde el índice va desde la posición cero hasta n-1, y n el tamaño del arreglo.

Un puntero es semejante a una variable, guarda un valor, pero en este caso los punteros solo guardan referencias a direcciones de memoria de otros valores.

El operador & es usado para asignar valores a los punteros, y el operador * para retornar el valor de la referencia del puntero.

Para declarar un puntero, lo hacemos de la siguiente forma:

Tipo Identificador1

Tipo *Identificador2;

*Identificador2 = &identificador1;

La memoria dinámica, normalmente usada para cuando necesitamos una cantidad variable de memoria que solo se puede determinar en tiempo de ejecución.