# Unidad 02: ELEMENTOS BÁSICOS DE PROGRAMACIÓN C++

## 2.1 Introducción

Los programas C++ están escritos en un lenguaje de programación de alto nivel, utilizando letras, números y otros símbolos que se encuentran localizados en el teclado de su computadora, bien directamente pulsando teclas, bien por combinación de teclas u otros artificios de escritura. Las computadoras, como ya conoce el lector, realmente entienden y ejecutan un lenguaje de bajo nivel denominado lenguaje o código máquina (un conjunto de números binarios).

Por consiguiente, se requiere que antes de que un programa se pueda utilizar, se debe someter a diferentes transformaciones en un proceso denominado **traducción**.

Una vez que se ha escrito el código del programa debe proceder a su ejecución. Las etapas prácticas dependerán del compilador o del entorno de programación que utilice; pero, en cualquier forma, serán muy similares a los siguientes:

1. Utilizar un editor de texto para escribir el programa fuente y guardarlo en un archivo llamado archivo fuente o código fuente (en C++ tiene un nombre y una extensión: .cpp).
2. Compilar el código fuente. El compilador traduce el código fuente al lenguaje interno de la máquina y lo convierte en un archivo denominado código objeto. En C++, el archivo resultante tiene el mismo nombre que el fuente y extensión .obj o bien .o.
3. Si la etapa anterior tiene éxito, el código objeto se enlaza con las bibliotecas de C++ mediante el enlazador (linker). Una biblioteca de C++ contiene el código objeto (máquina) de una colección de funciones (rutinas de las computadoras), que realizan tareas tales como visualizar información en pantalla, calcular la raíz cuadrada de un número, etc. El enlazado o montaje de los códigos objeto se combina también con el código de arranque estándar para producir una versión ejecutable de su programa. El archivo que contiene el producto final de la ejecución del programa se denomina archivo programa o código ejecutable (nombre igual que el fuente y extensión .exe).

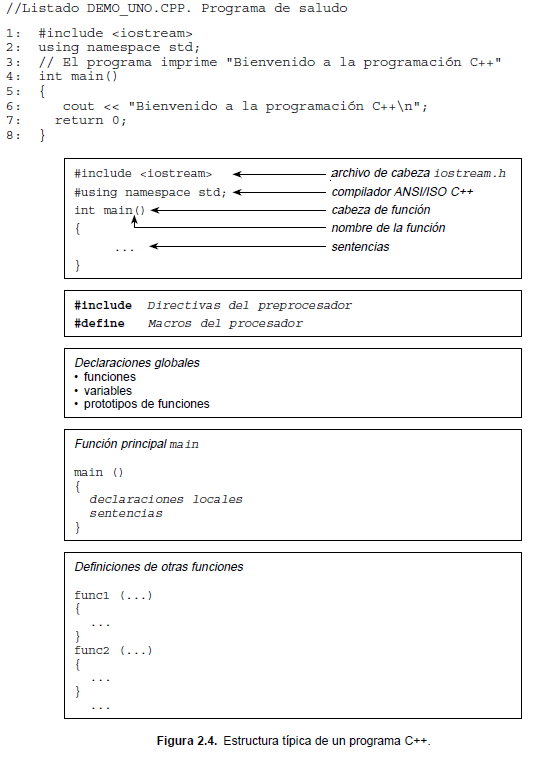
## 2.2 Estructura general de un programa en C++

Un programa en C++ se compone de una o más funciones. Una de las funciones debe ser obligatoriamente **main**. Una función en C++ es un grupo de instrucciones que realizan una o más acciones. Asimismo, un programa contendrá una serie de directivas #include que permitirán incluir en el mismo archivo de cabecera que a su vez constarán de funciones y datos predefinidos en ellos. De un modo más explícito, un programa C++ puede incluir:

* directivas de preprocesador con #include, using...
* declaraciones globales;
* la función main();
* funciones definidas por el usuario;
* comentarios del programa (utilizados en su totalidad):
* sentencias.

La estructura típica completa de un programa C++ se muestra en la Figura 2.4. Un ejemplo de un programa sencillo en C++.

(Diapositiva 2)



(Diapositiva 3)

La directiva #include de la primera línea es necesaria para que el programa tenga salida. Se refiere a un archivo externo denominado iostream.h en el que se proporciona la información relativa al objeto cout. Obsérvese que los ángulos < y > no son parte del nombre del archivo; se utilizan para indicar que el archivo es un archivo de la biblioteca estándar C++.

La segunda línea es la directiva using que permite incluir el espacio de nombre (namespace) std y actuar con el flujo de salida cout.

La tercera línea es un comentario, identificado por dobles barras inclinadas (//). Los comentarios se incluyen en programas que proporcionan explicaciones a los lectores de los mismos. Son ignorados por el compilador.

La cuarta línea contiene la cabecera de la función main(), obligatoria en cada programa C++. Indica el comienzo del programa y requiere los paréntesis () a continuación de main().

La quinta y octava línea contienen sólo las llaves { y } que encierran el cuerpo de la función main() y son necesarias en todos los programas C++.

La sexta línea contiene la sentencia **cout << "Bienvenido a la programación C++\n";** que indica al sistema que envía el mensaje "Bienvenido a la programación en C++\n" al objeto cout. Este objeto es el flujo estándar de salida que normalmente representa la pantalla de presentación de la computadora (el nombre cout es una abreviatura de «console output» —salida a consolas).

La salida será

**Bienvenido a la programación C++**

El símbolo '\n' es el símbolo de nueva línea. Poniendo este símbolo al final de la cadena entre comillas, indica al sistema que comience una nueva línea después de imprimir los caracteres precedentes, terminando, por consiguiente, la línea actual.

La séptima línea contiene la sentencia return 0. Esta sentencia termina la ejecución del programa y devuelve el control al sistema operativo de la computadora. El número 0 se utiliza para señalar que el programa ha terminado correctamente (con éxito).

La sentencia de salida de la sexta línea incluye diversos símbolos típicos de C++. El símbolo << se denomina operador de salida u operador de inserción. Inserta el mensaje en el flujo de salida. El símbolo \n incluido al final del mensaje significa carácter de nueva línea. Siempre que aparece en un mensaje de salida, se termina la línea actual de salida y comienza una nueva línea.

Obsérvese el punto y coma (;) al final de la sexta y séptima línea. C++ requiere que cada sentencia termine con un punto y coma. No es necesario que esté al final de una línea. Se pueden poner varias sentencias en la misma línea y se puede hacer que una sentencia se extienda sobre varias líneas.

## 2.3 Directivas del Preprocesador

El preprocesador es un programa que procesa un archivo fuente antes de que tenga lugar la compilación principal. En la práctica, el preprocesador prepara el código fuente para que el compilador pueda realizar correctamente su compilación. Permite incluir en el código otros archivos (denominados cabecera), definir macros, eliminar los comentarios, etc. No se requiere hacer nada especial para invocar a este procesador, ya que funciona automáticamente cuando se cumpla el programa.

**Directivas:** La directiva #include indica al procesador que inserte otro archivo en su archivo fuente. En efecto, la directiva #include es reemplazada por el contenido del archivo indicado a continuación. En la práctica, usar una directiva #include para insertar otro archivo en su archivo fuente es similar a la tarea de «pegar» un bloque de texto en un documento con un procesador de textos. El archivo de texto que se incluye en #include y en otras directivas se denomina **archivo de** **cabecera**.

**Archivos de cabecera**: Los archivos tales como iostream se denominan archivos de inclusión (debido a que se incluyen en otros archivos) o archivos de cabecera (ya que se incluyen al principio de un archivo).

(Diapositiva 4)

**La directiva using:** La directiva indica que todas las sentencias del programa que vienen a continuación están dentro del espacio de nombres std. Diversos componentes de programa, tales como cout, están declarados dentro de este espacio de nombres.

(Diapositiva 5)

## 2.4 Función main()

Es la función principal y que es el primer punto de entrada al programa. Cuando se ejecuta el programa, se invoca, en primer lugar, a la función main(). Cualquiera otra función se llama directa e indirectamente desde main. La función main() tiene la siguiente estructura: **nombre, cabecera, cuerpo, definición y terminación de la función**. El cuerpo de la función main() es un conjunto de sentencias que se ejecutan cuando se ejecuta main y termina con una última sentencia.

(Diapositiva 6)

En C++, una sentencia representa una instrucción completa a la computadora. Cada sentencia se debe terminar con un punto y coma, por lo que debe recordar no omitir los caracteres punto y coma cuando escriba sus programas.

En general, una función C++ se **activa** o llama por otra función y la cabecera de la función describe la interfaz entre una función y la función que la llama (función llamadora o invocadora). La parte que precede al nombre de la función se denomina **tipo de retorno** de la función, que describe el flujo de información desde una función a la función invocadora. La parte dentro de los paréntesis, a continuación del nombre, se denomina **lista de argumentos o lista de parámetros**, y describe el flujo de información de la función, desde la función llamadora a la función llamada.

En el caso de main() no se consideran argumentos, ya que main, normalmente, no se llama desde otras partes de su programa. En general, main es llamada por el código inicial que el compilador añade a su programa para conectar el programa y el sistema operativo (Unix, Linux, Windows XP, Windows Vista, etc.), y por ello actúa como la interfaz entre main y dicho sistema operativo. Una función C++ invocada por otra función puede devolver un valor a la función que le llama. Este valor se denomina valor de retorno. En el caso de main() puede devolver un valor de tipo entero cuando se indica con la palabra reservada int.

## 2.5 Funciones definidas por el usuario

Un programa C++ es una colección de funciones. Todos los programas se construyen a partir de una o más funciones que se integran para crear una aplicación. Todas las funciones contienen una o más sentencias C++ y se crean generalmente para realizar una única tarea, tales como imprimir la pantalla, escribir un archivo o cambiar el color de la pantalla. Se pueden declarar y ejecutar un número de funciones casi ilimitado en un programa C++.

Las funciones definidas por el usuario se invocan por su nombre y los parámetros opcionales que puedan tener. Después de que la función se ejecuta, el código asociado con la función se ejecuta y, a continuación, se retorna a la función **llamadora**.

Todas las funciones tienen nombre y una lista de valores que reciben. Se puede asignar cualquier nombre a su función, pero normalmente se procura que dicho nombre describa el propósito de la función. En C++, las funciones requieren una **declaración** o **prototipo** en el programa.

(Diapositiva 7)

Una **declaración** de función indica al compilador el nombre de la función que se está invocando en el programa. Si la función no se define, el compilador informa de un error. La palabra reservada void significa que la función no devuelve un valor.

C++ proporciona también funciones predefinidas que se denominan funciones de **biblioteca**. Las funciones de biblioteca son funciones listas para ejecutar que vienen con el lenguaje C++. Requieren la inclusión del archivo de cabecera estándar, tal como STDIO.H, MATH.H, etc. Existen centenares de funciones definidas en diversos archivos de cabecera.

## 2.6 Comentarios

Un comentario es cualquier información que se añade a su archivo fuente para proporcionar información de cualquier tipo. El compilador ignora los comentarios, no realiza ninguna tarea concreta. El uso de comentarios es totalmente opcional, aunque dicho uso es muy recomendable. /\*….\*/

## 2.7 Los Elementos de un programa C++

Un programa C++ consta de uno o más archivos. Un archivo es traducido en diferentes fases. La primera fase es el preprocesado, que realiza la inclusión de archivos y la sustitución de macros. El preprocesador se controla por directivas introducidas por líneas que contienen # como primer carácter. El resultado del preprocesado es una secuencia de tokens.

## 2.7.1 Tokens

Existen cinco clases de tokens: identificadores, palabras reservadas, literales, operadores y otros separadores.

## 2.7.2 Identificadores

Un identificador es una secuencia de caracteres, letras, dígitos y subrayados (\_). El primer carácter debe ser una letra (puede ser un subrayado) en ANSI/ISO (++). Las letras mayúsculas y minúsculas son diferentes a efectos del identificador. Reglas básicas:

1. Secuencia de letras o dígitos; el primer carácter puede ser una letra o un subrayado (compiladores de Borland, entre otros).
2. Los identificadores son sensibles a las mayúsculas y minúsculas.
3. Los identificadores pueden tener cualquier longitud, pero sólo son significativos los 32 primeros (ése es el caso de Borland y Microsoft).
4. Los identificadores no pueden ser palabras reservadas, tales como if, switch o else.

## 2.7.3 Palabras reservadas

Una palabra reservada tal como **void** es una característica del lenguaje C++ asociada con algún significado especial. Una palabra reservada no se puede utilizar como nombre de identificador, objeto o función.

Los siguientes identificadores están reservados para utilizarlos como palabras reservadas, y no se deben emplear para otros propósitos.

(Diapositiva 8)

## 2.7.4 Signos de puntuación

Todas las sentencias deben terminar con un punto y coma. Otros signos de puntuación son

(Diapositiva 9 y 10)

## 2.7.5 Archivos de cabecera

Un archivo de cabecera es un archivo especial que contiene las declaraciones de objetos y funciones de la biblioteca. Para utilizar objetos y funciones almacenadas en una biblioteca, un programa debe utilizar la directiva #include para insertar el archivo de cabecera correspondiente.

## 2.7.6 ANSI C++

El estándar ANSI C++ ha cambiado el convenio (notación) de los archivos de cabecera. Es posible utilizar sólo los nombres de las bibliotecas sin el sufijo .h; es decir, se puede usar iostream, cmath, cassert y cstdlib en lugar de iostream.h, math.h, assert.h y stdlib.h, respectivamente. En ANSI C++ se puede utilizar indistintamente.

## 2.8 Tipos de datos en C++

C++ no soporta un gran número de tipos de datos predefinidos, pero tiene la capacidad para crear sus propios tipos de datos. Todos los tipos de datos simples o básicos de C++ son, esencialmente, números. Los tres tipos de datos básicos son: **enteros; números de coma flotante (reales) y caracteres**.

(Diapositiva 11)

## 2.8.1 Enteros

Probablemente el tipo de dato más familiar es el entero, o tipo int. Los enteros son adecuados para aplicaciones que trabajen con datos numéricos. Los tipos enteros se almacenan internamente en 2 bytes (o 16 bits) de memoria.

## 2.8.2 Coma flotante

Los tipos de datos de coma (punto) flotante representan números reales que contienen una coma (un punto) decimal, tal como 3.14159, o números muy grandes, tales como 1.85\*1015.

## 2.8.3 Carácter

Un carácter es cualquier elemento de un conjunto de caracteres predefinidos o alfabeto. La mayoría de las computadoras utilizan el conjunto de caracteres ASCII

Declaración de variables: La forma más simple de una declaración de variable en C++ es poner primero el tipo de dato y, a continuación, el nombre de la variable. Si se desea dar un valor inicial a la variable, éste se pone a continuación. **<tipo de dato> <nombre de variable> = <valor inicial>**

## 2.9 Tipo de dato BOOL

El tipo bool3 se suele utilizar para indicar si ha ocurrido o no un suceso. También para efectuar comparaciones. Así, por ejemplo, si una variable puerta representa el hecho de que una puerta esté abierta o cerrada, puede tomar el valor falso cuando está cerrada y verdadero cuando está abierta. Otros casos pueden ser: si una bombilla está encendida o apagada, si hay clase o no hay clase a una determinada hora, etc.

(Diapositiva 12)

## 2.10 Constantes

Las constantes se pueden declarar con la palabra reservada const y se les asigna un valor en el momento de la declaración; este valor no se puede modificar durante el programa y cualquier intento de alterar el valor de un identificador definido con el calificador const producirá un mensaje de error del calificador. Las constantes no cambian durante la ejecución del programa. En C++ existen cuatro tipos de constantes: **constantes literales, constantes definidas, constantes enumeradas y constantes declaradas**.

Las constantes **literales** son las más usuales; toman valores tales como 45.32564,222 o bien "Introduzca sus datos" que se escriben directamente en el texto del programa. Las constantes **definidas** son identificadores que se asocian con valores literales constantes y que toman determinados nombres. Las constantes **declaradas** son como variables: sus valores se almacenan en memoria, pero no se pueden modificar. Las constantes **enumeradas** permiten asociar un identificador, tal como Color, con una secuencia de otros nombres, tales como Azul, Verde, Rojo y Amarillo.

## 2.11 Variables

En C++ una variable es una posición con nombre en memoria donde se almacena un valor de un cierto tipo de dato y puede ser modificado. Las variables pueden almacenar todo tipo de datos: cadenas, números y estructuras. Una constante, por el contrario, es una variable cuyo valor **no puede** ser modificado.

Una variable típicamente tiene un nombre (un identificador) que describe su propósito. Toda variable utilizada en un programa debe ser declarada previamente. La definición en C++ puede situarse en cualquier parte del programa (en C ANSI debe declararse, sin embargo, al principio del bloque, antes de toda sentencia ejecutable). Una definición reserva un espacio de almacenamiento en memoria. El procedimiento para definir (crear) una variable es escribir el tipo de dato, el identificador o nombre de la variable y, en ocasiones, el valor inicial que tomará. Por ejemplo, **char Respuesta;**

Significa que se reserva espacio en memoria para Respuesta, en este caso, un carácter ocupa un byte.

El nombre de una variable ha de ser un identificador válido. Es frecuente, en la actualidad, utilizar subrayados en los nombres, bien al principio o en su interior, con objeto de obtener mayor legibilidad y una correspondencia mayor con el elemento del mundo real que representa.

* **Declaración:** Una declaración de una variable es una sentencia que proporciona información de la variable al compilador C++.
* **Utilización:** es posible declarar una variable en el punto donde se vaya a utilizar. Esta propiedad se utiliza mucho en el diseño de bucles.
* **Iniciación:** Cuando se declara una variable, ésta no tiene ningún valor asociado; eso significa que, si se utiliza esa variable sin ningún valor inicial, el programa podría funcionar incorrectamente. Aunque no es obligatoria la inicialización de variables, sí es recomendable.
* **Definición:** Una declaración introduce un nombre de un objeto o de una variable (tal como c\_var) y asocia un tipo con la variable/objeto tal como int. Una definición es una declaración que asigna simultáneamente memoria al objeto/variable.

(Diapositiva 13)

## 2.12 Duración de una variable

Dependiendo del lugar donde se definan las variables de C++, éstas se pueden utilizar en la totalidad del programa dentro de una función o pueden existir sólo temporalmente dentro de un bloque de una función. La zona de un programa en la que una variable está activa se denomina, normalmente, ámbito o alcance.

## 2.12.1 Variables locales

Son aquéllas definidas en el interior de una función y son visibles sólo en esa función específica. Las reglas por las que se rigen las variables locales son:

1. En el interior de una función, una variable local no puede ser modificada por ninguna sentencia externa a la función.
2. Los nombres de las variables locales no han de ser únicos. Dos, tres o más funciones, por ejemplo, pueden definir variables de nombre Interruptor. Cada variable es distinta y pertenece a la función en que está declarada.
3. Las variables locales de las funciones no existen en memoria hasta que se ejecuta la función. Esta propiedad permite ahorrar memoria, ya que permite que varias funciones compartan la misma memoria para sus variables locales (pero no a la vez).

(Diapositiva 14)

## 2.12.2 Variables globales

Son variables que se declaran fuera de la función y por defecto (omisión) son visibles a cualquier función incluyendo main().

Todas las variables **locales** desaparecen cuando termina su bloque. Una variable **global** es visible desde el punto en que se define hasta el final del programa.

## 2.12.3 Variables dinámicas y de objeto

Tienen características que en algunos casos son similares tanto a variables **locales** como a **globales**. Al igual que una variable local, una variable dinámica se crea y libera durante la ejecución del programa. La diferencia entre una variable local y una variable dinámica es que la variable dinámica se crea tras su petición (en vez de automáticamente, como las variables locales), es decir, a su voluntad, y se libera cuando ya no se necesita. Al igual que una variable global, se pueden crear variables dinámicas que son accesibles desde múltiples funciones.

## 2.13 Entradas y Salidas

C++ no define directamente ninguna sentencia para realizar la entrada y salida de datos. La biblioteca de E/S proporciona un conjunto amplio de facilidades. La biblioteca iostream contiene cuatro objetos de E/S: **cin, cout, cerr y clog**, que permiten la entrada y salida por consola.

Para utilizar esta biblioteca, su programa debe incluir las siguientes instrucciones al principio del archivo que contiene su programa:

(Diapositiva 15)

**iostream**, para manejar la entrada y salida, tiene dos tipos denominados istream y ostream, que representan **flujos** de entrada y salida, respectivamente. Un flujo (stream) es una secuencia de caracteres preparados para leer o escribir en un dispositivo de E/S de cualquier clase.

En general, el sistema asocia cada uno de estos objetos con la ventana en la cual se ejecuta el programa. De este modo, cuando se leen datos de cin, los datos se leen de la ventana en la que se está ejecutando el programa y cuando escribimos con cout, cerr o clog, la salida se escribe en la misma ventana.

* **Entrada (cin):** El archivo iostream define cin como un objeto que representa el flujo. Para la salida, el operador << inserta caracteres en el flujo de salida. Para la entrada, cin utiliza el operador >> para extraer caracteres del flujo de entrada y por ello se denomina operador de extracción. Al igual que cout, es un objeto inteligente. Convierte la entrada, una serie de caracteres escritos desde el teclado, en un formato aceptable a la variable donde se almacena la información.
* **Salida (cout):** El operador de inserción de flujo, <<, inserta los datos a la derecha del mismo (operando derecho) en el objeto a la izquierda del operador (cout, operando izquierdo). Los valores de variables, así como cadenas de texto, se pueden sacar (visualizar) a la pantalla; también puede sacarse a la salida una combinación de variables.

## 2.14 Espacios de nombres

Un espacio de nombres (namespaces) es una región declarativa con nombre opcional. El nombre de un espacio de nombres se puede utilizar para acceder a entidades declaradas en ese espacio de nombre; es decir, los miembros del espacio de nombre. En esencia, son conjuntos de variables, de funciones, de claves y de subespacios de nombre, miembros que siguen unas reglas de visibilidad. El espacio de nombres es una característica de C++ introducida en las últimas versiones, diseñada para simplificar la escritura de programas.

Un programa C++ puede dividirse en diferentes espacios de nombres. Un espacio de nombre es una parte del programa en el cual se recuerdan ciertos componentes y fuera de ese espacio son desconocidos o no son reconocidos.

(Diapositiva 16)