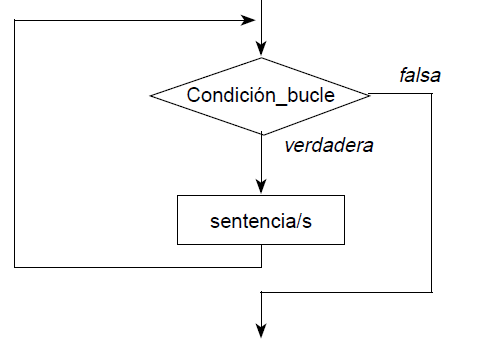
# Unidad 05: ESTRUCTURAS DE CONTROL REPETITIVAS

## 5.1 Sentencia WHILE

**Un bucle** es cualquier construcción de programa que repite una sentencia o secuencia de sentencias un número de veces. La sentencia (o grupo de sentencias) que se repiten en un bloque se denomina **cuerpo** del bucle y cada repetición del cuerpo del bucle se llama **iteración** del bucle. Las dos principales cuestiones de diseño en la construcción del bucle son: ¿Cuál es el cuerpo del bucle? ¿Cuántas veces se iterará el cuerpo del bucle?

Un bucle while tiene una *condición* del bucle (una expresión lógica) que controla la secuencia de repetición. La posición de esta condición del bucle es delante del cuerpo del bucle y significa que un bucle while es un bucle *pretest* de modo que cuando se ejecuta el mismo, se evalúa la condición *antes* de que se ejecute el cuerpo del bucle. La Figura 5.1 representa el diagrama del bucle while.

El diagrama de la Figura 5.1 indica que la ejecución de la sentencia o sentencias expresadas se repite *mientras* la condición del bucle permanece verdadera y termina cuando se hace falsa. También indica el diagrama anterior que la condición del bucle se evalúa antes de que se ejecute el cuerpo del bucle y, por consiguiente, si esta condición es inicialmente falsa, el cuerpo del bucle no se ejecutará. En otras palabras, el cuerpo de un bucle while se ejecutará *cero* o *más veces*.



El cuerpo del bucle se repite **mientras** que la expresión lógica (condición del bucle) sea verdadera. Cuando se evalúa la expresión lógica y resulta falsa, se termina y se sale del bucle y se ejecuta la siguiente sentencia del programa después del cuerpo de la sentencia while.

**Operaciones de incremento**: C++ ofrece los operadores de incremento (++) y decremento (––) que soporta una sintaxis abreviada para añadir (incrementar) o restar (decrementar) 1 al valor de una variable.

**Diseño eficiente de bucles:** Una cosa es analizar la operación de un bucle y otra diseñar eficientemente sus propios bucles. Los principios a considerar son: primero, analizar los requisitos de un nuevo bucle con el objetivo de determinar su inicialización, prueba (condición) y actualización de la variable de control del bucle; segundo, desarrollar *patrones estructurales* de los bucles que se utilizan frecuentemente.

**Con cero interacciones:** demás esta aclarar que la sentencia debe cumplirse (No falsa) para que entre al bucle.

**Bucles controlados por centinela**: Un medio para manejar esta situación es instruir al usuario a introducir un único dato definido y especificado denominado *valor centinela* como último dato. La condición del bucle comprueba cada dato y termina cuando se lee el valor centinela. El valor centinela se debe seleccionar con mucho cuidado y debe ser un valor que no pueda producirse como dato. En realidad, el centinela es un valor que sirve para terminar el proceso del bucle.

**Bucles controlados por banderas:** las variables tipo bool se utilizan con frecuencia como *indicadores* o *banderas de estado* para controlar la ejecución de un bucle. El valor del indicador se inicializa (normalmente a falso "false")antes de la entrada al bucle y se redefine (normalmente a verdadero "true") cuando un suceso específico ocurre dentro del bucle. Un *bucle controlado por bandera* o *indicador* se ejecuta hasta que se produce el suceso anticipado y se cambia el valor del indicador.

## 5.2 Repetición: bucle FOR

El bucle for de C++ es superior a los bucles for de otros lenguajes de programación tales como BASIC, Pascal y C, ya que ofrece más control sobre la inicialización e incrementación de las variables de control del bucle. Además del bucle for, C++ proporciona otros dos tipos de bucles, for y do. El bucle for que se estudia en esta sección es el más adecuado para implementar *bucles controlados por contador,* que son bucles en los que un conjunto de sentencias se ejecuta una vez por cada valor de un rango especificado, de acuerdo al algoritmo:

por cada valor de una variable\_contador de un rango específico:

ejecutar sentencias

La sentencia **for** (bucle **for**) es un método para ejecutar un bloque de sentencias un número fijo de veces. El bucle for se diferencia del bucle while en que las operaciones de control del bucle se sitúan en un solo sitio: la cabecera de la sentencia.

El bucle for contiene las cuatro partes siguientes:

* *Parte de inicialización*, que inicializa las variables de control del bucle. Se pueden utilizar variables de control del bucle simples o múltiples.
* *Parte de iteración,* que contiene una expresión lógica que hace que el bucle realice las iteraciones de las sentencias, mientras que la expresión sea verdadera.
* *Parte de incremento*, que incrementa o decrementa la variable o variables de control del bucle.
* *Sentencias*, acciones o sentencias que se ejecutarán por cada iteración del bucle.

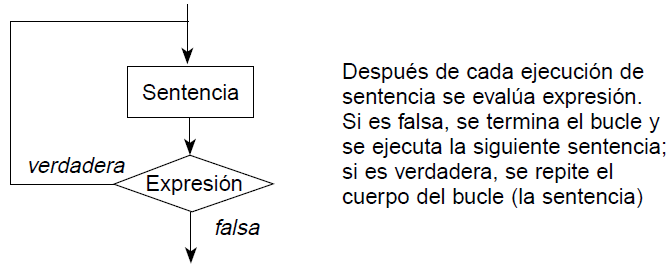
**Diferentes usos de bucles FOR**: ANSI C++ Estándar requiere que las variables sean declaradas en la expresión de inicialización de un bucle for sean locales al bloque del bucle for. De igual modo permite que:

* El valor de la variable de control se puede modificar en valores diferentes de 1.
* Se pueden utilizar más de una variable de control.

## 5.3 Repetición: el bucle DO WHILE

La sentencia **do-while** se utiliza para especificar un bucle condicional que se ejecuta al menos una vez. Esta situación se suele dar en algunas circunstancias en las que se ha de tener la seguridad de que una determinada acción se ejecutará una o varias veces, pero al menos una vez.

La construcción **do** comienza ejecutando sentencia. Se evalúa a continuación expresión. Si expresión es verdadera, entonces se repite la ejecución de sentencia. Este proceso continúa hasta que expresión es falsa. La semántica del bucle do se representa gráficamente en la Figura 5.3.

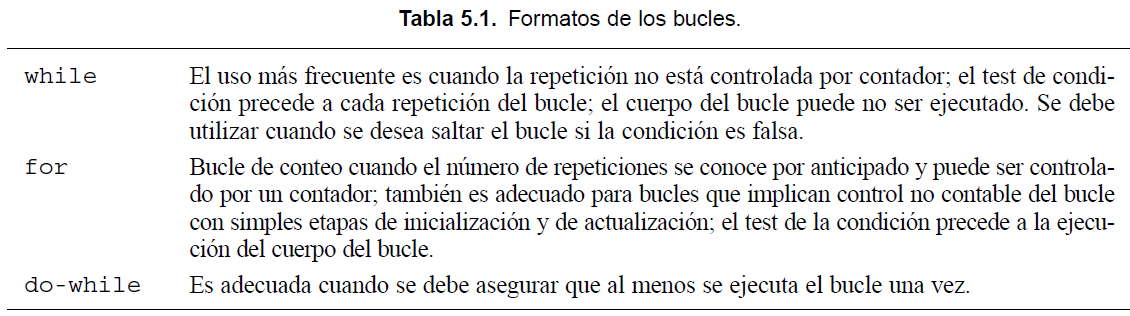


**Diferencias entre DO WHILE y WHILE**: Una sentencia do-while es similar a una sentencia while, excepto que el cuerpo del bucle se ejecuta siempre al menos una vez.

## 5.4 Comprobación de bucles: WHILE, FOR y DO WHILE

El bucle while se repite *mientras* su condición de repetición del bucle es verdadera; el bucle for se utiliza normalmente cuando el conteo esté implicado, o bien el control del bucle for, en donde el número de iteraciones requeridas se puede determinar al principio de la ejecución del bucle, o simplemente cuando existe una necesidad de seguir el número de veces que un suceso particular tiene lugar. El bucle do-while se ejecuta de un modo similar a while excepto que las sentencias del cuerpo del bucle se ejecutan siempre al menos una vez.

La Tabla 5.1 describe cuándo se usa cada uno de los tres bucles. En C++, el bucle for es el más frecuentemente utilizado de los tres. Es relativamente fácil reescribir un bucle do-while como un bucle while, insertando una asignación inicial de la variable condicional. Sin embargo, no todos los bucles while se pueden expresar de modo adecuado como bucles do-while, ya que un bucle do-while se ejecutará siempre al menos una vez y el bucle while puede no ejecutarse. Por esta razón, un bucle while suele preferirse a un bucle do-while, a menos que esté claro que se debe ejecutar una iteración como mínimo.



## 5.5 Diseño de bubles

El diseño de un bucle requiere tres partes:

1. El cuerpo del bucle.
2. Las sentencias de inicialización.
3. Las condiciones para la terminación del bucle.

**Bucles para suma de productos**: es aconsejable para estos casos el FOR

**Fin de un bucle:** Existen cuatro métodos utilizados normalmente para terminar un bucle de entrada. Estos cuatro métodos son:

1. **Lista encabezada por tamaño**. Si su programa puede determinar el tamaño de una lista de entrada por anticipado, bien preguntando al usuario o por algún otro método, se puede utilizar un bucle «repetir n veces» para leer la entrada exactamente n veces, en donde n es el tamaño de la lista.
2. **Preguntar antes de la iteración.** El segundo método para la terminación de un bucle de entrada es preguntar, simplemente, al usuario, después de cada iteración del bucle, si el bucle debe ser o no iterado de nuevo.
3. **Lista terminada con un valor centinela**. El método más práctico y eficiente para terminar un bucle que lee una lista de valores del teclado es mediante un valor centinela. Un **valor centinela** es aquél que es totalmente distinto de todos los valores
4. **Agotamiento de la entrada.** Éste es el método usual en la lectura de archivos, que suele utilizar una marca al final de archivo, eof.

Un bucle **controlado por contador** es cualquier bucle que determina el número de iteraciones antes de que el bucle comience y, a continuación, repite (itera) el cuerpo del bucle esas iteraciones. La técnica de la lista encabezada por tamaño es un ejemplo de un bucle controlado por contador.

La técnica de **preguntar antes de iterar** se puede utilizar para bucles distintos de los bucles de entrada, pero el uso más común de esta técnica es para procesar la entrada.

La técnica del valor centinela es una técnica conocida también como **salida con una condición bandera** o **señalizadora**. Una variable que cambia su valor para indicar que algún suceso o evento ha tenido lugar, se denomina normalmente **bandera** o **indicador**. En el ejemplo anterior de suma de números, la variable bandera es número de modo que cuando toma un valor negativo significa que indica que la lista de entrada ha terminado.

**Sentencias para alterar el flujo de control**:

* break (ruptura). La sentencia break transfiere el control del programa al final del bucle (o en el caso de sentencia swith de selección, la sentencia más interna que lo encierra).
* continue (continuar). La sentencia continue consta de la palabra reservada continue seguida por un punto y coma. Cuando se ejecuta, la sentencia continue termina la iteración actual del cuerpo del bucle actual más cercano que lo circunda.
* goto (ir\_a). C++, como otros lenguajes de programación, incluye la sentencia goto (ir\_a), pero nunca se debe utilizar. Sólo en casos muy extremos se debe recurrir a ella. Un caso extremo puede ser, salir de varios bucles anidados.

## 5.6 Bucles anidados

Es posible *anidar* bucles. Los bucles anidados constan de un bucle externo con uno o más bucles internos. Cada vez que se repite el bucle externo, los bucles internos se repiten, se reevalúan los componentes de control y se ejecutan todas las iteraciones requeridas.