CAPÍTULO 4
SENTENCIAS DE CONTROL ITERATIVAS [1]

Las sentencias de control iterativas permiten ejecutar un bloque de código un número determinado de veces. Estas implementan lo que se conoce como bucles o lazos. Un lazo en C, suele estar determinado los siguientes elementos:

- el bloque de código a ejecutarse repetidamente (cuerpo del lazo);
- el número de veces que ese bloque se ejecutará.

El número de veces que se ejecuta un lazo comúnmente está definido por el **cumplimiento de una condición**. Si dicha **condición se cumple**, el código dentro del lazo se sigue ejecutando; si no se cumple, el flujo del programa continua con la ejecución de la sentencia inmediatamente posterior a la del lazo ("sale del lazo"). Como práctica común, esta **condición** se representa mediante una **expresión** formada por una **variable de control**, a la que se le suele llamar "contador". Así, esta variable de control es clave para determinar el funcionamiento del lazo, y las operaciones que deben considerarse sobre ella son las siguientes:

- declaración e inicialización de la variable de control;
- evaluación de la variable de control (verificación del cumplimiento de la **condición**);
- actualización de la variable de control.

Cabe notar que la actualización de la variable de control es necesaria para que en algún momento la condición deje de cumplirse y el flujo del programa pueda "salir del lazo".

En ANSI C, se tienen 3 tipos de implementaciones de lazo, mediante las sentencias `for`, `while`, y `do-while`.

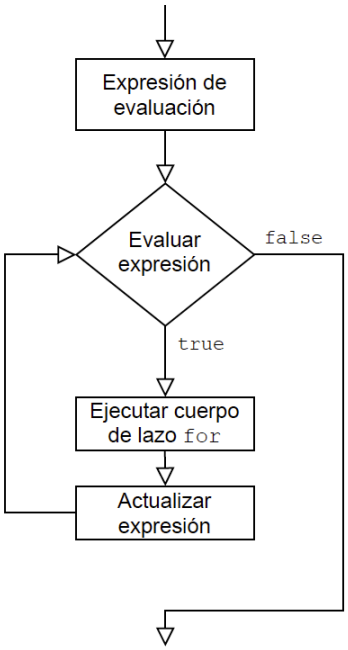
1) La sentencia `for`

La sintaxis de la sentencia `for` es la siguiente:

```
for(sentenciaInicializacion; expresionEvaluacion; sentenciaActualizacion){
            //sentencias dentro del cuerpo del lazo;
}
```

El lazo `for` funciona de la siguiente manera:

- La sentencia de inicialización se ejecuta solamente una vez. Se inicializa la variable de control.
- Luego, se evalúa la expresión de evaluación. Si dicha expresión se evalúa como `false`, el lazo `for` termina.
- Si la expresión se evalúa como `true`, las sentencias en el cuerpo del lazo se ejecutan, y la sentencia de actualización actualiza el valor de la variable de control.
- Se vuelve a evaluar la expresión de evaluación.
- El proceso continúa hasta que la expresión de evaluación se evalúe como `false`, en cuyo caso el lazo termina.



A continuación, se ofrece un ejemplo de programa donde se implementa un lazo **`for`**:

```
// Imprimir números del 1 al 5
#include <stdio.h>

int main() {
  int i;

  for (i = 1; i <= 5; i=i+1){
    printf("%d ", i);
  }
  return 0;
}
```

Los pasos seguidos en este ejemplo son los siguientes:

- En el programa anterior, en el lazo **`for`**, la variable **`i`** se inicializa con 1.
- Luego, la expresión **`i<=5`** se evalúa. Dado que **`1<=5`** es **`true`**, se ejecuta el cuerpo del lazo **`for`**. Esto imprime el valor 1 de **`i`** en pantalla.
- La sentencia de actualización se ejecuta (**`++i`**), cambiando el valor de **`i`** a 2.
- Nuevamente, la expresión **`i<=5`** se evalúa como true pues **`2<=5`**. Así, el cuerpo del lazo se ejecuta de nuevo, imprimiendo el valor 2 de **`i`** en pantalla.
- La sentencia de actualización se vuelve a ejecutar, cambiando el valor de **`i`** a 3.
- Todo este proceso se repite sucesivamente hasta que el valor de **`i`** es 6, cuando la expresión **`6<=5`** se evalúa como false, con lo que el lazo **`for`** termina.

2) La sentencia `while`

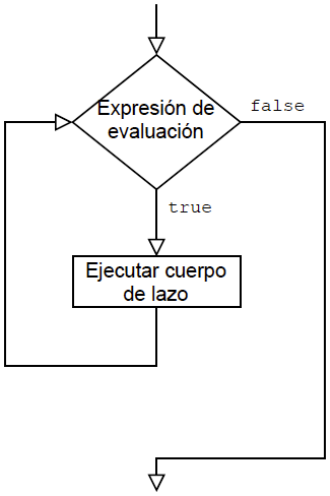
La sintaxis del lazo **`while`** es la siguiente:

```
while(expresionEvaluacion){
      //El cuerpo del lazo
}
```

El lazo **`while`** funciona de la siguiente manera:

- El lazo **`while`** evalúa la expresión `expresionEvaluacion` dentro de los paréntesis.
- Si `expresionEvaluacion` se evalúa como **`true`**, las sentencias en el cuerpo del lazo se ejecutan. Luego, la expresión se vuelve a evaluar.

- Si `expresionEvaluacion` se evalúa como `false`, el lazo termina.



A continuación, se ofrece un ejemplo de programa donde se implementa un lazo `while`:

```
// Imprimir números del 1 al 4

#include <stdio.h>
int main() {
  int i = 1;

  while (i <= 5) {
    printf("%d\n", i);
    i=i+1;
  }

  return 0;
}
```

Los pasos seguidos en este ejemplo son los siguientes:

- La variable `i` se inicializa a 1.
- Cuando `i=1`, la expresión de evaluación `i<=5` se evalúa como `true`. Así, el cuerpo del lazo se ejecuta y se imprime 1 en la pantalla. También en el cuerpo del lazo se incrementa el valor de `i`a 2.
- Con `i=2`, la expresión `i<=5` se vuelve a evaluar como `true`, con lo que el cuerpo del lazo se vuelve a ejecutar, imprimiendo en pantalla el valor 2, e incrementando el valor de `i`a 3.
- Este proceso continúa hasta que `i=6`, en cuyo caso la expresión `i<=5` se evalúa como `false`, y el lazo termina.

Cabe destacar que, a diferencia de la sentencia `for`, la sentencia `while` requiere que la variable de control se actualice en el cuerpo del lazo.

3) La sentencia do-while

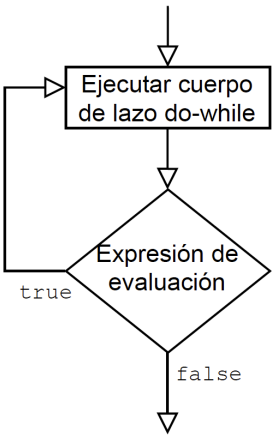
La sentencia `do-while` es similar a la sentencia `while`. La diferencia es que el **cuerpo** del lazo correspondiente se **ejecuta al menos una vez**. Recién luego de esa primera ejecución es que se evalúa la expresión de evaluación.

La sintaxis del lazo `do-while` es la siguiente:

```
do{
        //El cuerpo del lazo
}while(expresionEvaluacion);
```

El lazo **do-while** funciona de la siguiente manera:

- El cuerpo del lazo **do-while** se ejecuta una vez. Luego, se evalúa la expresión **expresionEvaluacion**
- Si la expresión se evalúa como **true**, el cuerpo del lazo se ejecuta nuevamente y la expresión es evaluada nuevamente.
- Este proceso continúa hasta que **expresionEvaluacion** se evalúe como **false**.
- Si la expresión **expresionEvaluacion** se evalúa como **false**, el lazo termina.



A continuación, se ofrece un ejemplo de programa donde se implementa un lazo **do-while**:

```
// Program to sumar numeros hasta que el usuario ingrese 0

#include <stdio.h>
int main() {
  double numero, suma = 0;

  // the body of the loop is executed at least once
  do {
    printf("Ingrese un numero: ");
    scanf("%lf", &numero);
    suma = suma + numero;
  }
  while(numero != 0.0);

  printf("Suma = %.2lf",suma);

  return 0;
}
```

Los pasos seguidos en este ejemplo son los siguientes:

- El lazo **do-while** pide al usuario ingresar un número.
- El lazo **do-while** se ejecuta mientras el usuario no ingrese el número 0.
- El lazo **do-while** se ejecuta por lo menos una vez, es decir la primera vez se ejecuta sin haber antes evaluado la condición **numero != 0.0**
- La condición se empieza a evaluar luego de la primera iteración.
- Si el número que ingresa el usuario es distinto de 0, éste se suma a la variable suma.
- El proceso se repite hasta que el usuario ingrese 0.
- Finalmente, el programa imprime el valor de **suma**.

Bibliografía

[1] Luis Joyanes Aguilar and Ignacio Zahonero Martínez, *Programación en C, Metodología, algoritmos y estructuras de datos*, Segunda Edición. Mc Graw Hill.