Estructuras de control

Los siguientes conceptos fueron extraídos del libro Introducción a la programación | Carlos E. Cimino
Desde codo a codo recomendamos su canal **https://www.youtube.com/c/CharlyCimino**

Estructuras secuenciales

Como primera medida, debés entender que las instrucciones se ejecutan de manera natural una tras otra, en el orden en que fueron escritas. Este flujo se denomina secuencial y es el más sencillo de todos. La máquina interpretará y ejecutará paso a paso las líneas, desde la primera hasta la última. Observá el siguiente diagrama de flujo:

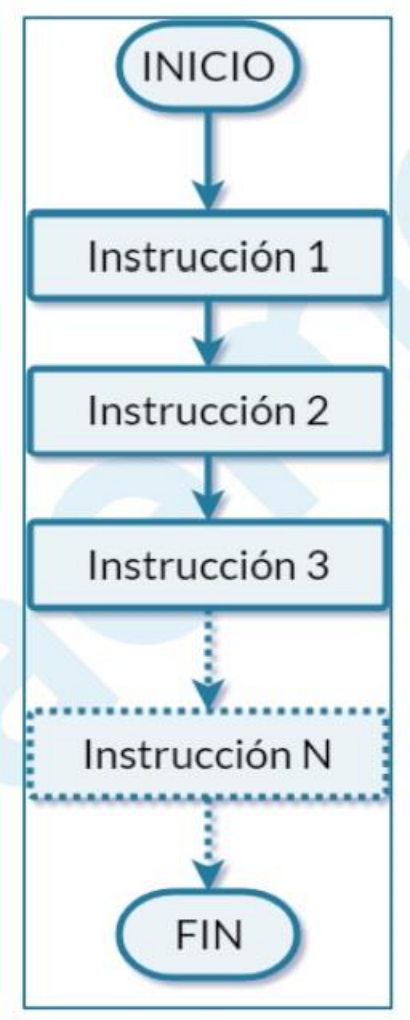
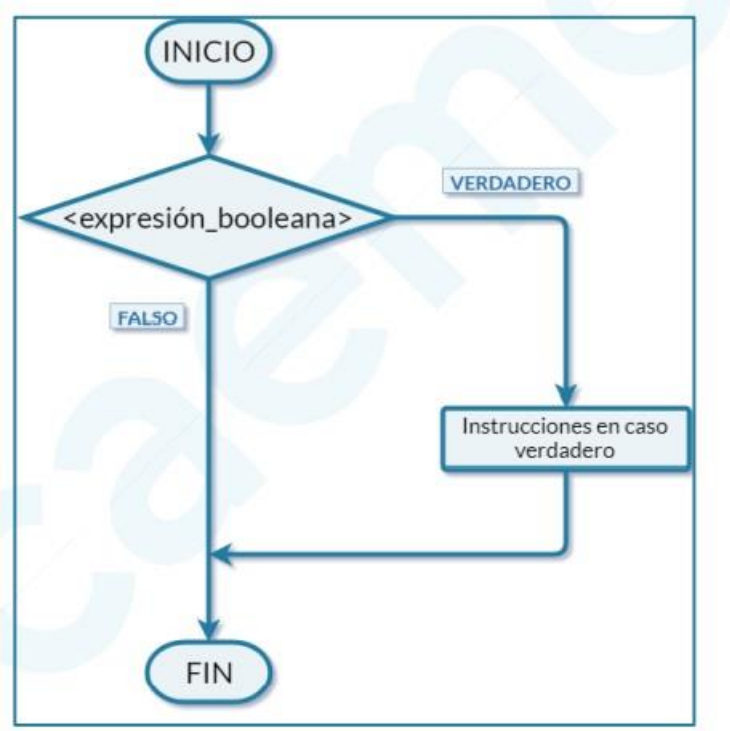


Ilustración 11: Flujo secuencial.

Flujo de selección simple

La primera y más sencilla manera de alterar el flujo secuencial por defecto de cualquier programa estructurado es utilizar una estructura de selección simple. Observá el siguiente diagrama:



En cierto momento, definido por el programador, la computadora evalúa una condición, es decir, una expresión booleana, que se representa mediante un rombo. Si la expresión devuelve un resultado VERDADERO, la computadora ejecuta las instrucciones dentro de un bloque especial que luego retorna al flujo original, de lo contrario, el flujo continúa normalmente. La sintaxis en pseudocódigo es la siguiente:

```
Si (expresion booleana) Entonces
        instrucciones
FinSi
```

Si: se tipea literalmente, indica que vamos a iniciar un bloque de selección.

•expresion booleana: es la condición que la computadora evaluará. El hecho de estar entre paréntesis es opcional, aunque te recomiendo que lo hagas para mejorar la legibilidad y porque muchos lenguajes formales requieren que la condición esté obligatoriamente encerrada entre paréntesis.

• Entonces se tipea literalmente, indica que a continuación serán listadas las instrucciones que se ejecutarán en caso de que la condición sea VERDADERO.

• Instrucciones: son las instrucciones que se ejecutarán. Puede ser solo una, o varias, siguiendo los conceptos vistos hasta aquí.

• FinSi se tipea literalmente, indica que ha terminado el bloque de selección. Un programa sencillo que facilita la comprensión podría ser el siguiente: pedirle al usuario que ingrese su edad. En caso de que tenga menos de 18 años, mostrarle un mensaje que diga "No podés ingresar". Sea cual fuere la edad, el programa termina diciendo "Suerte". El código sería el siguiente:

El código sería el siguiente:

```
1  Algoritmo seleccion_simple
2      Definir edad Como Entero;
3      Escribir "Ingresá tu edad:";
4      Leer edad;
5      Si (edad < 18) Entonces
6          Escribir "No podés ingresar"; // Se escribe según la edad
7      FinSi
8      Escribir "Suerte"; // Se escribe siempre
9  FinAlgoritmo
```

Código 19: Algoritmo que valida un pase según la edad.

Si probás el programa anterior con un valor igual o mayor a 18, por ejemplo, 30, la computadora evaluará la condición como FALSO, por lo que las instrucciones que están dentro del bloque Si...FinSi no serán ejecutadas. La ejecución continúa justo después del bloque Si...FinSi, encontrándose con la escritura de la palabra "Suerte" y dando por finalizada la ejecución. Si probás el programa anterior con un valor menor a 18, por ejemplo, 12, la computadora evaluará la condición como VERDADERO, por lo que las instrucciones que están dentro del bloque Si...FinSi serán ejecutadas. La ejecución continúa justo después del bloque Si...FinSi, encontrándose con la escritura de la palabra "Suerte" y dando por finalizada la ejecución. Es importante que notes que la palabra "Suerte" se escribe siempre, ya que al estar fuera del bloque Si...FinSi, su ejecución no se encuentra condicionada. El diagrama de flujo que genera PSeInt es el siguiente:

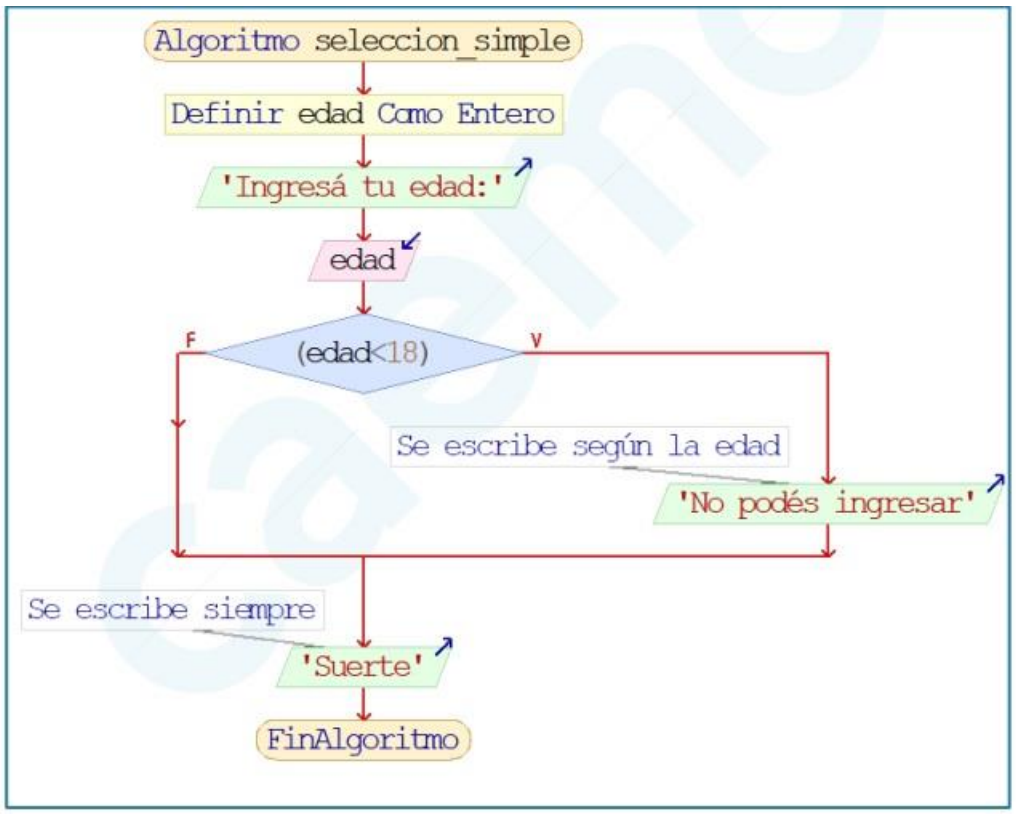


Ilustración 17: Algoritmo que valida un pase según la edad.

Una buena práctica que mejora la legibilidad del programa es darle sangría a las instrucciones que están dentro de un bloque Si...FinSi. Tal acción permite un código ordenado que facilita mucho la depuración a medida que el mismo se torna más complejo. Para dejar una sangría en una instrucción, basta con poner el cursor justo delante del primer carácter y tocar la tecla tabulador (TAB). A este proceso se lo conoce como "indentar", aunque el término no es una palabra reconocida del español, sino un anglicismo de la palabra inglesa "indentation".

Flujo de selección doble

La estructura anterior permite realizar una serie de instrucciones en caso de que una condición sea VERDADERO, pero no especifica nada en caso de que sea FALSO. De hecho, si la condición resulta FALSO, el programa continúa su ejecución como si el bloque Si...FinSi no existiera. Para contemplar y hacer una serie de instrucciones específicas en caso de que una condición resulte FALSO, sin dejar de lado lo que se hacía cuando resultaba VERDADERO, es necesario utilizar una estructura de selección doble, según el siguiente diagrama

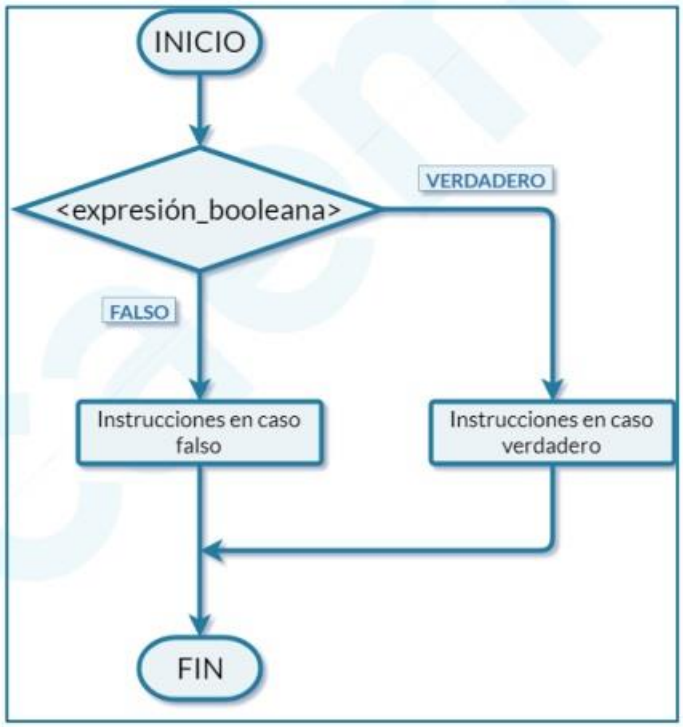


Ilustración 18: Flujo de selección doble.

Te invito a que te tomes el trabajo de comparar el diagrama de flujo de selección doble con el de selección simple. La única diferencia es que ahora sí podemos tomar acción en caso de que la condición resulte FALSO. La sintaxis en pseudocódigo es la siguiente:

Si (expresion booleana)
    Instrucciones caso verdadero
Sino
    Instrucciones caso falso
FinSi •

Si: se tipea literalmente, indica que vamos a iniciar un bloque de selección.

•expresion booleana: es la condición que la computadora evaluará. El hecho de estar entre paréntesis es opcional, aunque te recomiendo que lo hagas para mejorar la legibilidad y porque muchos lenguajes formales requieren que la condición esté obligatoriamente encerrada entre paréntesis.

• Entonces se tipea literalmente, indica que a continuación serán listadas las instrucciones que se ejecutarán en caso de que la condición sea VERDADERO.

•instrucciones caso verdadero: son las instrucciones que se ejecutarán, si es que la condición resulta VERDADERO. Puede ser solo una, o varias, siguiendo los conceptos vistos hasta aquí.

• Sino se tipea literalmente, indica que a continuación serán listadas las instrucciones que se ejecutarán en caso de que la condición sea FALSO.

• instrucciones caso falso : son las instrucciones que se ejecutarán, si es que la condición resulta FALSO. Puede ser solo una, o varias, siguiendo los conceptos vistos hasta aquí.

• FinSi se tipea literalmente, indica que ha terminado el bloque de selección.

Continuando con el programa sencillo de ejemplo anterior, vamos a pedirle al usuario que ingrese su edad. En caso de que tenga menos de 18 años, mostrarle un mensaje que diga "No podés ingresar", en caso contrario, dirá "Te damos la bienvenida, ¡pasá!". Sea cual fuere la edad, el programa termina diciendo "Suerte". El código sería el siguiente:

El código sería el siguiente:

```
1   Algoritmo seleccion_doble
2       Definir edad Como Entero;
3       Escribir "Ingresá tu edad:";
4       Leer edad;
5       Si (edad < 18) Entonces
6           Escribir "No podés ingresar"; // Si es VERDADERO
7       Sino
8           Escribir "Te damos la bienvenida, ¡pasá!";
9       FinSi
10      Escribir "Suerte"; // Se escribe siempre
11  FinAlgoritmo
```

Código 20: Algoritmo que valida o no un pase según la edad.

El diagrama de flujo que genera PSeInt es el siguiente:

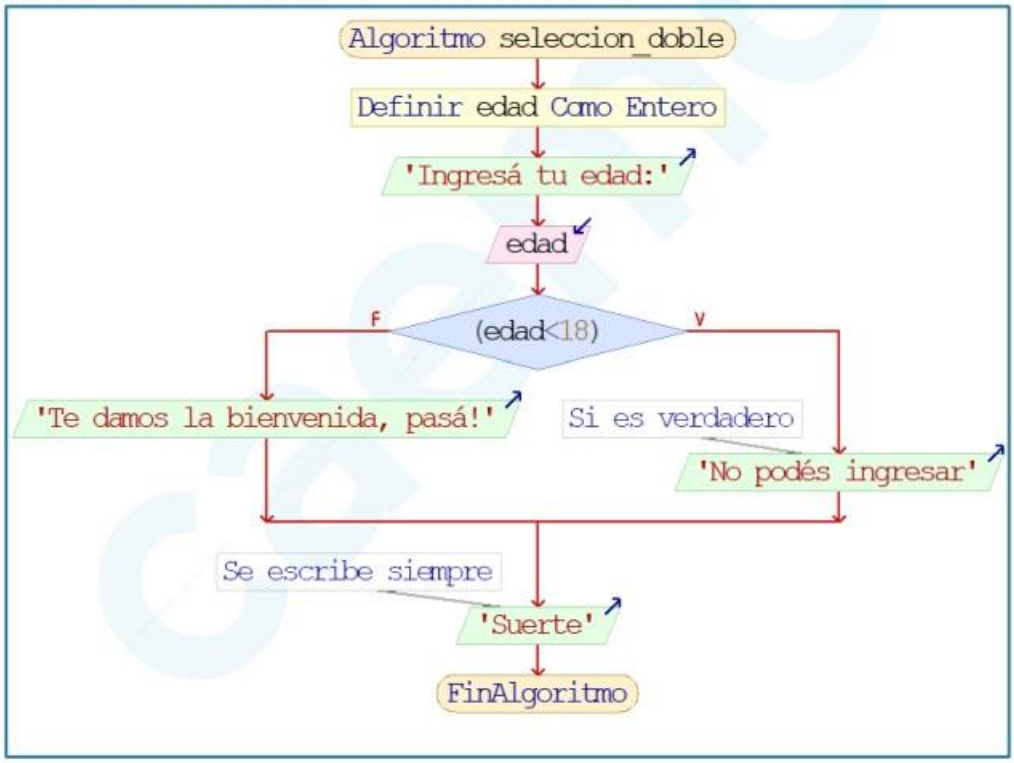


Ilustración 19: Algoritmo que valida o no un pase según la edad.

Anidamiento de estructuras de selección

Para comprender este tema, la idea es realizar el siguiente algoritmo: el usuario ingresa un número y la computadora indica si se trata de un número positivo, negativo o el cero.

Habrás visto que la máquina solo puede resolver expresiones booleanas, las cuales solos tienen dos resultados posibles: VERDADERO o FALSO. Como nuestro programa evidentemente contempla tres posibles acciones, debemos hacer uso de anidamiento de estructuras de selección, obteniendo un flujo como el siguiente: Un ejemplo puede ser el siguiente: si el número ingresado es mayor que 0, se trata sin discusión de un número positivo, por lo tanto, suponiendo que alojo el número en una variable llamada num, la expresión num > 0 resultaría VERDADERO. En caso FALSO, no puedo asegurar que se trate de un número negativo, pues pudo haber sido el 0. En ese caso, se tiene que volver a emplear una estructura de selección, cuya condición puede ser num < 0. Si la expresión anterior resulta VERDADERO, puedo asegurar que se trata de un número negativo, en caso contrario, puedo asegurar que se trata del 0. Una imagen vale más que mil palabras. Observá cómo queda el diagrama de flujo

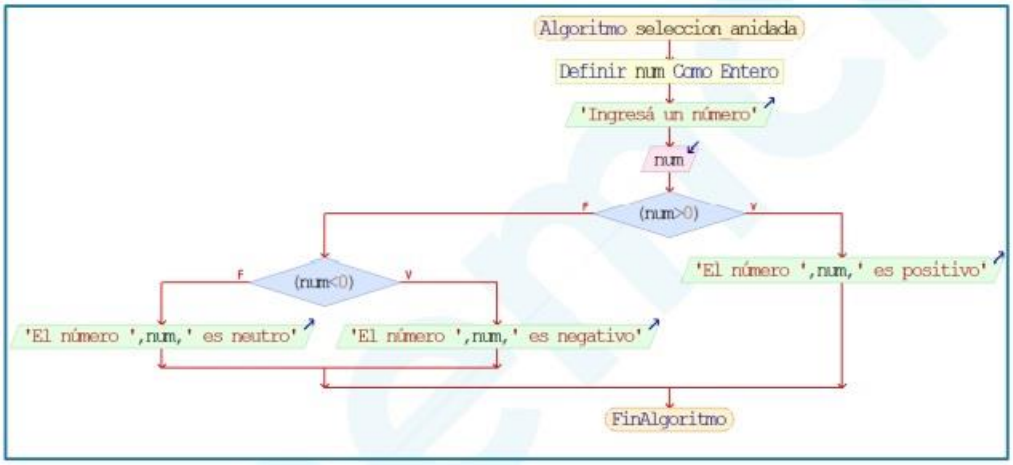


Ilustración 21: Algoritmo que reconoce el signo de un número.

Traduciendo el diagrama a código, queda lo siguiente:

```
1   Algoritmo seleccion_anidada
2       Definir num Como Entero;
3       Escribir "Ingresá un número";
4       Leer num;
5       Si (num > 0) Entonces
6           Escribir "El número " , num , " es positivo";
7       SiNo
8           Si (num < 0) Entonces
9               Escribir "El número " , num , " es negativo";
10          SiNo
11              Escribir "El número " , num , " es neutro";
12          FinSi
13      FinSi
14  FinAlgoritmo
```

Código 21: Algoritmo que reconoce el signo de un número.

Flujo de repetición

Tanto el flujo secuencial como el flujo de selección tienen en común que las instrucciones siempre siguen un curso hacia adelante, es decir, una vez que una instrucción fue ejecutada, jamás volverá a ejecutarse, a no ser, claro está, que el programador la repita más adelante. Hay muchas situaciones donde se requiere repetir una o más instrucciones un número definido o no definido de veces, pero queriendo evitar que las instrucciones pertinentes sean repetidas en el código. Las estructuras de repetición nos permiten solucionar esto. Se denominan ciclos, bucles o loops. Un ciclo puede tener una o más repeticiones, las cuales normalmente se denominan vueltas o iteraciones. Hay tres componentes que deben ser correctamente analizados para poder efectuar ciclos:

• Inicialización: Cómo será el valor inicial de la(s) variable(s) en la condición.

• Condición: Qué tiene que ocurrir para que el ciclo continúe su ejecución. Debe tener al menos una variable involucrada, de lo contrario, el resultado sería constante.

• Actualización: La(s) variable(s) en la condición debe(n) cambiar su valor por cada iteración para que el resultado de la condición no sea siempre constante.

Flujo de repetición Mientras…FinMientras

Una estructura de repetición nos permite especificar que un programa debe repetir una o más acciones mientras cierta condición valga VERDADERO. Cuando la condición pase a valer FALSO, el programa continuará con la ejecución de las demás sentencias debajo de la estructura de control de repetición. Supongamos que queremos escribirle al usuario la frase "Hola, mucho gusto." unas cinco veces. Es cierto que una posibilidad es usar cinco instrucciones Escribir, una debajo de otra con un resultado satisfactorio. Pero es evidente que esto no es muy práctico. Peor aún, imaginá que olvidaste poner un punto al final de la oración. ¿Te imaginás cambiándolo en las cinco instrucciones? ¿Y si hubiese querido mostrar diez mil oraciones? Aquí es cuando una estructura de repetición nos salva. Observá el siguiente diagrama:

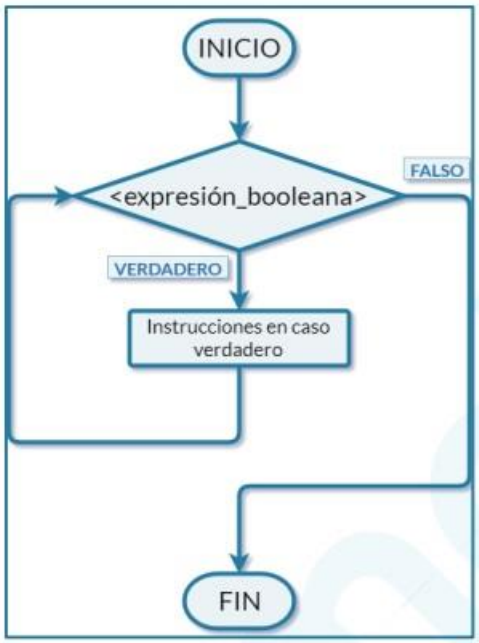


Ilustración 26: Flujo de repetición.

Volviendo al ejemplo de mostrarle al usuario la frase "Debo aprender ciclos." cinco veces, el código puede ser el siguiente:

```
1  Algoritmo salidas_repetidas
2      Definir veces Como Entero;
3      veces = 0; // La inicialización
4      Mientras (veces < 5) Hacer // La condición
5          Escribir "Debo aprender ciclos.";
6          veces = veces + 1; // La actualización
7      FinMientras
8      Escribir "Fin.";
9  FinAlgoritmo
```

Código 34: Algoritmo que muestra cinco veces una cadena.

Resultando en el siguiente diagrama de flujo:

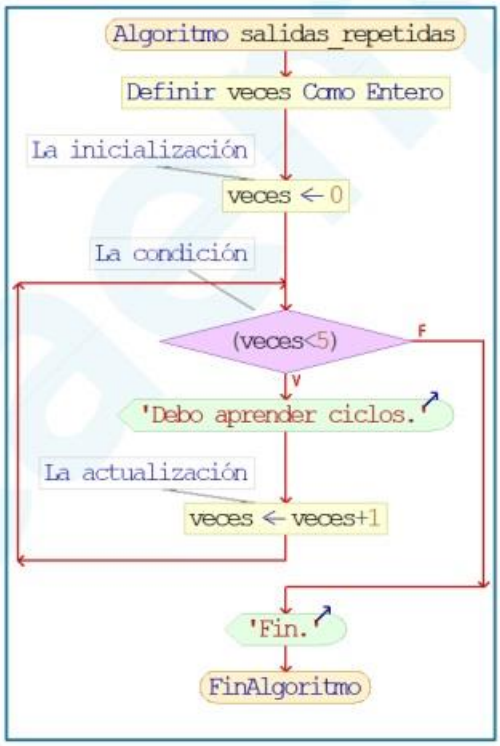


Ilustración 27: Algoritmo que muestra cinco veces una cadena.

Control de ciclos por contador

En los casos donde se necesita la ejecución de un número definido de iteraciones, se recurre a un ciclo controlado por contador. Una variable entera oficia de contadora de repeticiones. Comenzará con un valor inicial. Cada iteración actualizará el valor de la variable contadora. El ciclo seguirá iterando mientras la variable contadora supere o no (según el caso) a cierto valor

```
1  Algoritmo salidas_repetidas_numeradas
2       Definir veces Como Entero;
3       veces = 0; // La inicialización
4       Mientras (veces < 5) Hacer // La condición
5            Escribir "Iteración " , veces , ": "Debo aprender ciclos.";
6            veces = veces + 1; // La actualización
7       FinMientras
8       Escribir "Fin.";
9  FinAlgoritmo
```

Código 35: Algoritmo que muestra cinco veces una cadena contando hacia adelante.

En el ejemplo anterior ya se ha usado una variable contadora, la cual llamamos veces. Para ver explícitamente cómo veces se trata de un contador, podemos hacer que en cada iteración se muestre el valor de veces, junto a la cadena "Hola, mucho gusto.". De esta manera, lograrás ver los resultados en pantalla numerados: Como verás, ahora se muestra la frase precedida por el número de iteración correspondiente. Como veces se inicializó en 0, es normal ver que las iteraciones mostradas vayan de 0 a 4. Lo importante es que el objetivo se ha cumplido: mostrar la cadena 5 veces. Hay, desde ya, infinitas maneras de realizar el mismo ejercicio. ¿Qué tal si hacemos la recorrida en sentido contrario? Esta vez, la variable veces se inicializa en 5. El ciclo va a ejecutarse mientras veces sea mayor que 0. La actualización esta vez consiste en decrementar veces en una unidad cada vez que se itera. El código quedaría así:

```
1  Algoritmo salidas_repetidas_numeradas_reversa
2       Definir veces Como Entero;
3       veces = 5; // La inicialización
4       Mientras (veces >= 1) Hacer // La condición
5            Escribir "Iteración " , veces , ": Debo aprender ciclos.";
6            veces = veces - 1; // La actualización
7       FinMientras
8       Escribir "Fin.";
9  FinAlgoritmo
```

Código 36: Algoritmo que muestra cinco veces una cadena contando hacia atrás.

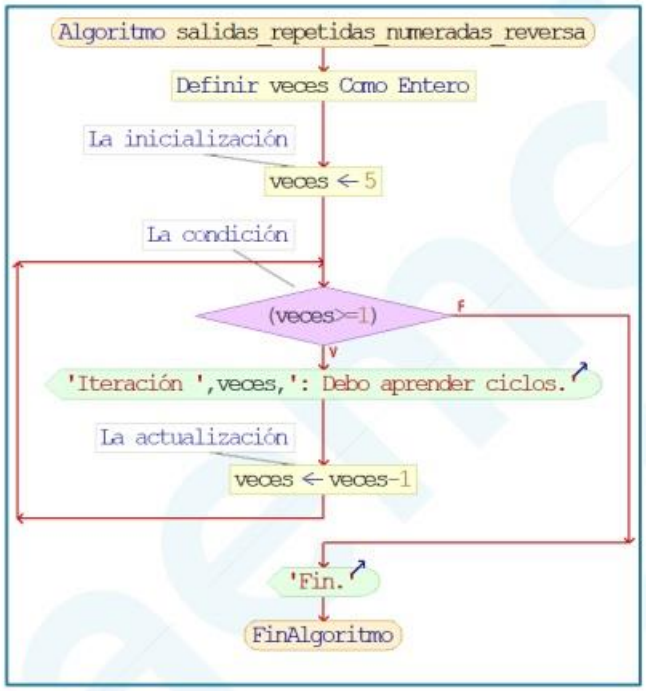


Ilustración 28: Algoritmo que muestra cinco veces una cadena contando hacia atrás.