aprenderaprogramar.com

Instrucción SalirMientras para el control del flujo de programas. Pseudocódigo y diagramas de flujo. (CU00180A)

Sección: Cursos

Categoría: Curso Bases de la programación Nivel I

Fecha revisión: 2024

Autor: Mario R. Rancel

Resumen: Entrega nº 79 del Curso Bases de la programación Nivel I

24

INSTRUCCIÓN SALIRMIENTRAS

La instrucción *SalirMientras* se usa para provocar una salida forzada de un bucle *Mientras ... Hacer.* El flujo del programa salta a la instrucción inmediatamente posterior al *Repetir* que marca la terminación del bucle. En el caso de bucles anidados *SalirMientras* afecta únicamente al bucle que está procesándose cuando se llega a esta instrucción de salida. Su uso antes o después de un bucle *Mientras ... Hacer* no tiene ningún efecto al carecer de sentido.

SalirMientras puede encontrarse en cualquier punto intermedio entre el *Mientras* y *Repetir*. Normalmente irá después de evaluar una situación que evoluciona a medida que lo hace el bucle, y que es la desencadenante de la salida.

Ejemplo de uso de SalirMientras.

Vamos a ver lo que sería una situación de bucle anidado. Para ello nos valdremos del mismo algoritmo que utilizamos como ejemplo en la instrucción *Finalizar* con ciertas variantes. En aquel caso se pedía al usuario un número entre 100 y 200 y si el número introducido no era válido se repetía la petición. Tras 20 intentos se mostraba un mensaje y el programa se cerraba. La variante consistirá en que el usuario entra a un menú que le da opción a salir, a proceder a la entrada de datos o a proceder al proceso de datos. Si se producen los 20 intentos nulos, tras el mensaje de advertencia no se le cerrará el programa, sino que se le devuelve al menú de forma que tiene él que decidir si sale del programa o lo vuelve a intentar (obviamente disponiendo nuevamente de 20 intentos). La salida del programa se hará siempre a través del menú.

Organización previa de variables:

Variable	**Descripción - Uso**
E	Señalero que almacena la decisión del usuario con la clave numérica 1 = Entrada de datos 2 = Proceso de datos 3 = Salir del programa
i	Contador de la cantidad de veces que el usuario introduce un dato no válido
Numero	Almacena el número proporcionado por el usuario

Nota: En este ejemplo se solicita un único dato. Obviamente podrían haber sido varios datos como *Numero, Nombre, Edad, etc*. o un array de datos como *Dato(1), Dato(2), Dato(3), etc.* Igualmente en el apartado de proceso de datos únicamente hemos contemplado lo que sería la estructura del programa, sin darle contenido. Para esta opción se han supuesto dos posibilidades:

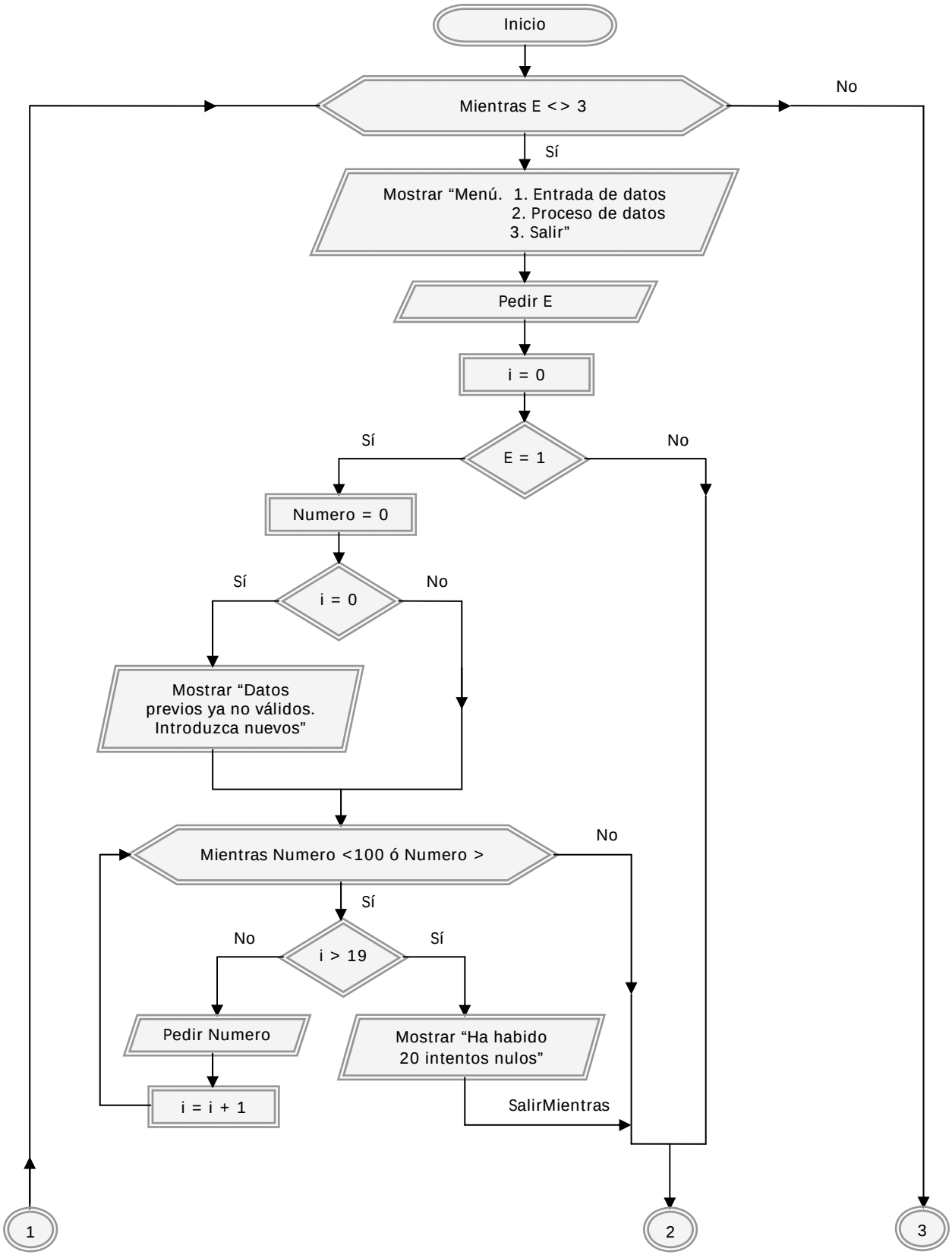
a) El usuario trata de acceder al proceso de datos y existen datos válidos.

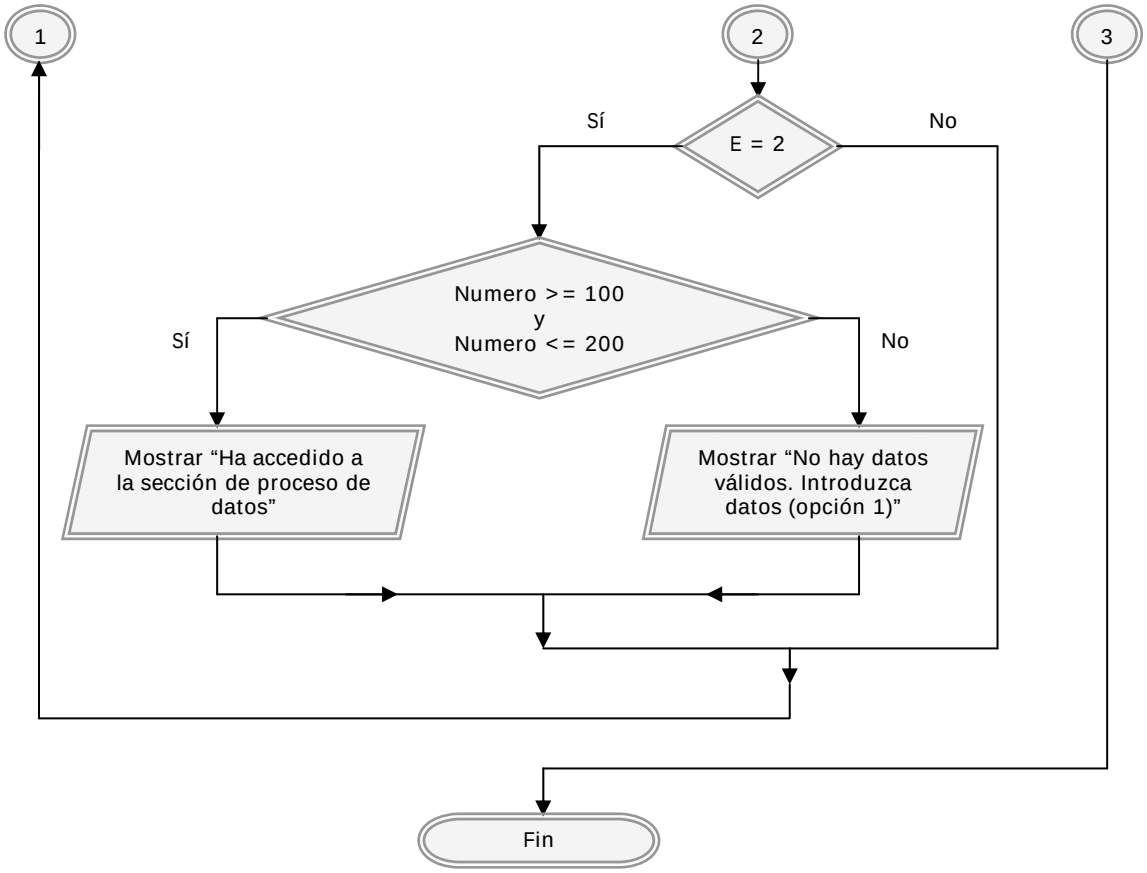
b) El usuario trata de acceder al proceso de datos y no existen datos válidos. En este caso, se muestra un mensaje de advertencia y se vuelve al menú.

Pseudocódigo:

```
1. Inicio [Ejemplo de SalirMientras aprenderaprogramar.com]
    2. Mientras E <> 3 Hacer
        2.1 Mostrar "Menú. 1. Entrada de datos
                           2. Proceso de datos
                           3. Salir del programa"
        2.2 Pedir E [Almacena la elección del usuario]
        2.3 i = 0
        2.4 Si E = 1 Entonces
            2.4.1 Numero = 0
            2.4.2 Si i = 0 Entonces
                      Mostrar "Datos previos ya no válidos. Introduzca nuevos datos"
                  FinSi
            2.4.3 Mientras Numero < 100 ó Numero > 200 Hacer
                      Si i > 19 Entonces
                          Mostrar "Se han realizado 20 intentos nulos.
                                   Consulte el manual del programa"
                          SalirMientras
                      FinSi
                      Mostrar "Por favor, introduzca un número comprendido entre
                               100 y 200"
                      Pedir Numero
                      i = i + 1
                  Repetir
            FinSi
        2.5 Si E = 2 Entonces
            2.5.1 Si Numero >= 100 y Numero <= 200 Entonces
        Mostrar "Ha accedido a la sección de proceso de datos"
                  SiNo
                      Mostrar "No hay datos válidos. Introdúzcalos a través de la
                               opción 1 del menú"
                  FinSi
            FinSi
        Repetir
3. Fin
```

Diagrama de flujo:





Comentarios: Las posibilidades para el diseño de un algoritmo de este tipo son varias. Analicemos la configuración que se ha propuesto. En primer lugar, observamos que la totalidad del programa está subordinado a la condición *Mientras E <> 3*. Ya tenemos herramientas con las que podemos evitar esta circunstancia si así lo deseáramos.

Respecto a la instrucción *SalirMientras*, ésta se ejecuta tras 20 entradas de dato no válidas (no cumplen los criterios pedidos) y da lugar a la salida del bucle que se está ejecutando en ese momento. Y aquí puede haber un matiz lingüístico porque alguien podría decir “Es que el bucle externo también se está ejecutando en ese momento”. Y así es. El caso es que cuando la ejecución de un programa transcurre por un bucle es habitual decir “se está procesando el bucle nº 17” o “el control lo tiene el bucle 17” en alusión a que ningún otro proceso más amplio continua hasta que se termine ese bucle. Más allá de matices lingüísticos, lo que hay que tener claro es que un *SalirMientras* (así como un *SalirDesde* para el caso *Desde ... Siguiente*) afecta únicamente al *Mientras / Repetir* más próximo dentro del cual está englobado. Ni siquiera podemos poner dos *SalirMientras* uno detrás de otro esperando un “doble salto”, ya que antes de que se pueda leer el segundo el control habrá sido ya transferido a otro punto del programa.

Otra cuestión a comentar es la inversión de condicionantes que se observa en el programa. Por un lado leemos *Mientras Numero < 100 ó Numero > 200* ... y por otro *Si Numero >= 100 y Numero <= 200.*

Cuando se hacen inversiones de este tipo conviene tener en cuenta un par de cosas. La primera, si realizar esa inversión va a facilitar o a complicar la lectura y entendimiento del programa. La segunda, si el tratamiento dado a los extremos de intervalos es el adecuado, en particular en relación a que el inverso de un signo ”mayor”, no es el “menor”, sino el “menor o igual”. Y el inverso de un “mayor o igual”, no es el “menor o igual”, sino el “menor”.

Expresión	Inversa
A > B	A <= B
A < B	A >= B
A >= B	A < B
A <= B	A > B
A = B	A > B ó A < B
A <> B	- - -

En el pseudocódigo de ejemplo la condición *Mientras Numero < 100 ó Numero > 200* significa que se aceptan los valores 100 y 200 ya que no se cumple 100 < 100 ni 200 > 200. En la opción de procesar datos se procede a verificar si existe un dato válido precisamente con un *Si Numero >= 100 y Numero <= 200.* De este modo, si el usuario ha proporcionado el número 100 como dato, se accede a la sección de proceso de datos. Si se hubiera escrito *Si Numero > 100 y Numero < 200* ambos extremos hubieran quedado excluidos. Todo esto entra dentro de "lo obvio", pero conviene tenerlo en cuenta porque es motivo habitual de errores en los programas.

Otro aspecto a tener en cuenta es algo que venimos ya viendo en distintos programas, y no es otra cuestión que la necesidad de control del valor inicial de las variables en general y para la entrada en los bucles en particular. Analicemos en concreto las líneas *2.3 i = 0* y 2.4.1 *Numero = 0* ¿Por qué están puestas ahí y no en otro punto del programa? La respuesta hay que buscarla en la reflexión sobre cómo deben funcionar dichas variables.

Numero = 0 está situado después de verificar que el usuario ha decidido realizar una entrada de datos (*E = 1*). Si no lo hiciéramos así y el valor almacenado en *Numero* derivado de un ciclo anterior fuera por ejemplo 150, no sería posible volver a realizar la petición de datos al no poder entrar en el bucle *Mientras Numero < 100 ó Numero > 200.*

i = 0 debe realizarse antes de entrar en el bucle de petición de datos para que cada vez que se acceda a él haya *20* intentos disponibles. En caso contrario los intentos disponibles serían *20 – i,* siendo *i* el valor que hubiera quedado almacenado en el ciclo anterior.

En este algoritmo *i = 0* puede estar antes del *Si E = 1* o inmediatamente después, pero siempre antes de la evaluación *Si i = 0 ...* de la línea *2.4.2.*

Como venimos haciendo con todas las instrucciones que suponen una modificación directa del flujo de programas recomendaremos:

- Evitar el uso de *SalirMientras* siempre que sea posible.
- Recordar lo expuesto en la introducción a la modificación directa del flujo de programas.

Próxima entrega: CU00181A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=28&Itemid=59