ÍNDICE

3 Análisis y Diseño: Algoritmos, Diagramas de Flujo y Pseudocódigo

En el modelo en cascada la fase de análisis y de diseño son importantes porque consisten en analizar el problema y elaborar un bosquejo de la solución, respectivamente.

En estas fases aun no se determina un lenguaje de programación, es decir, el desarrollo de estas fases no se requiere determinar en que lenguaje se va a programar, por el contrario el programador puede seleccionar un lenguaje en específico para continuar con la siguiente fase "codificación".

Análisis

En esta primera etapa se debe de determinar ***qué es*** los que el programa deberá de hacer. Se deben de plantear y contestar las siguientes preguntas:

- ¿Cuáles son los datos (la información) de entrada? (**Entrada**)
- ¿Cuáles son los resultados esperados? (**Salida**)

La información de **entrada** es la que el usuario de forma directa o indirecta proporcionará al programa, mientras que los **resultados esperados** (**salida**) son los que se obtendrán después de ejecutar uno o más procesos dentro del programa.

Diseño

Una vez que la etapa de análisis se haya finalizado se prosigue con la etapa de diseño. El objetivo de la etapa de diseño es determinar ***cómo*** debe de hacerse el proceso que permitiría resolver un problema planteado. Para describir el cómo debe de desarrollarse generalmente se describen con el uso de **algoritmos** (sección 2.2) que representan el **procedimiento** (serie de pasos finitos y con una secuencia determinada).

- ¿Cómo se efectúa el proceso? (**Procedimiento**)

Para desarrollar la fase de diseño se hace uso de dos mecanismos:

- **Diagramas de flujo**.
- **Pseudocódigo**.

Diagrama de Flujo

Un **diagrama de flujo** representa un algoritmo de manera gráfica haciendo uso de diagramas que utilizan símbolos (cajas) de manera estandarizada [5] y que permiten representar los pasos de un algoritmo considerando sus tres características. La **secuenciación** se determina mediante flechas que unen los símbolos (cajas) e indica la **línea de flujo** de ejecución del algoritmo.

Para ver los distintos tipos de símbolos y su significado, que son utilizados en el diseño de diagramas de flujos consulte la Tabla 1 en la sección 2.5.2 Diseño.

Pseudocódigo

El **pseudocódigo** es un lenguaje cercano al lenguaje de programación de alto nivel, tiene como objetivo describir la ejecución del programa utilizando un lenguaje con una sintaxis y semántica definida y estandarizada.

En el pseudocódigo no se consideran aspectos específicos a un lenguaje de programación, es decir, es ***independiente del lenguaje de programación***. Al elaborar un pseudocódigo bien estructurado permite fácilmente obtener la traducción a un lenguaje seleccionado como puede ser: C, Java, .Net. etc.

Permite concentrarse en la lógica y estructura del control y no en el desarrollo de un lenguaje en específico. En programas grandes es conveniente tener un bosquejo del programa en diagrama de flujo, pero es necesario tener el pseudocódigo completo.

Es posible utilizar palabras en un idioma particular, sin embargo es necesario recalcar que existen **palabras reservadas** (aquellas que son utilizados para definir el pseudocódigo y no pueden ser utilizadas por el programador para evitar conflictos de nombrado) que son marcadas en negrita o subrayadas.

Existen reglas que permiten hacer legible el pseudocódigo, las más importante es respetar el ***identado*** o el espaciado entre el inicio de cada operación. Es importante contemplar el identado para tener **legible** el pseudocódigo.

Estructura General

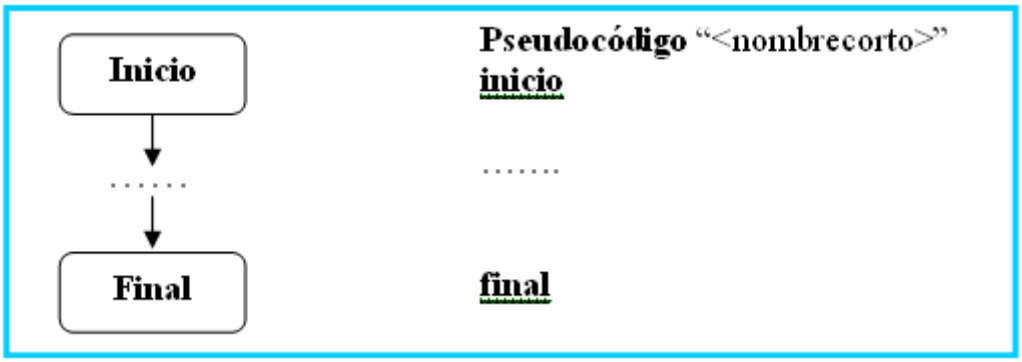


Figura 16.- Estructura General – Diagrama de Flujo y Pseudocódigo.

3.1 Sentencias Simples

En esta sección se describen las tres sentencias principales y que serán utilizadas con mucha frecuencia en el desarrollo de programas.

3.1.1 Asignación

La asignación se utiliza para:

1. Asignar un valor inicial a una variable.
2. Asignar el resultado de una operación o varias operaciones a una variable.
3. Reasignación de un valor en la variable previamente utilizada.

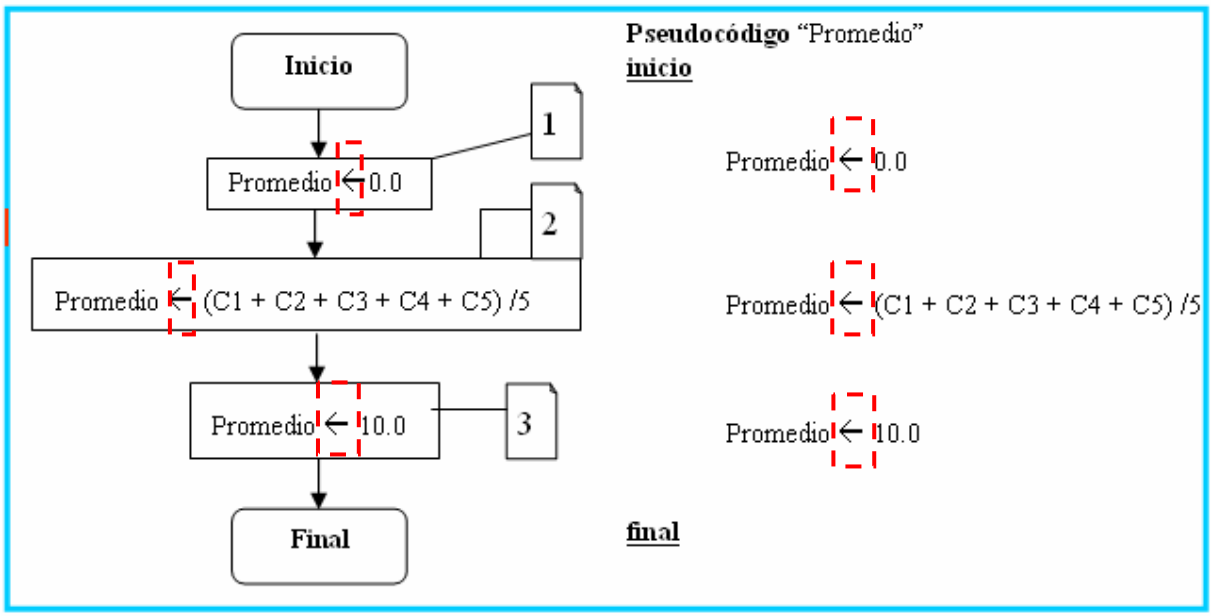


Figura 17.- Representación de la operación de asignación en Diagrama de Flujo y Pseudocódigo.

3.1.2 Entrada y Salida

Generalmente la operación de **entrada** permite tener una interacción con el usuario, con esta operación es posible leer valores y representarían los ***datos de entrada*** identificados en el algoritmo. Sin embargo la operación leer puede ser utilizada para obtener información de un dispositivo, de una fuente de información, etc.

La operación de **salida** esta relacionada generalmente en el desplegado de la información para que el usuario determine los ***resultados esperados*** o la ***salida*** del programa. Sin embargo es posible direccional la salida a otros dispositivos como la impresora, o bien a una fuente de información como lo es un archivo.

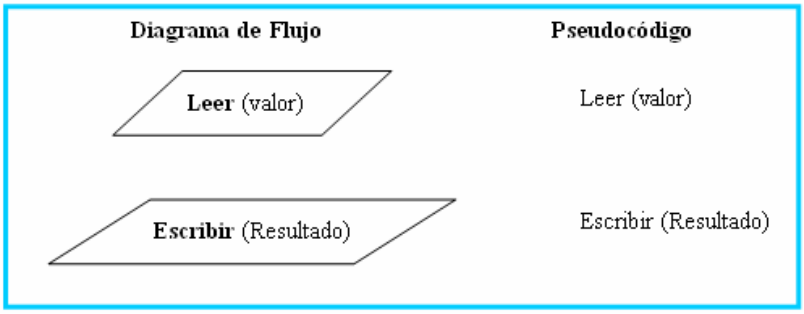


Figura 18.- Operación lectura (entrada) y escritura (salida) en Diagrama de Flujo y Pseudocódigo.

Ejemplo "**Conversión de grados Celsius a Fahrenheit**"

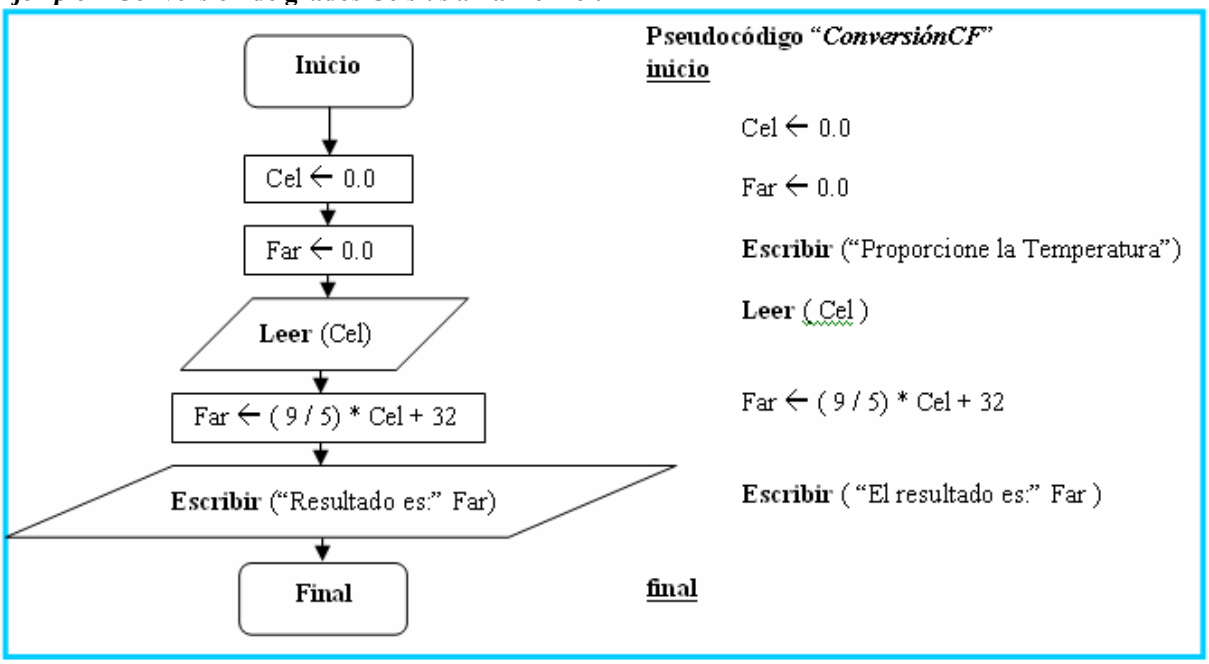


Figura 19.- Ejemplo resuelto en Diagrama de Flujo y Pseudocódigo.

Ejercicio "**Promedio de 5 calificaciones**"

Dadas 5 calificaciones determinar el promedio de las 5.

Ejercicio "**Conversión de kilómetros a centímetros**"

Dada una medida en kilómetros efectuar la conversión a centímetros mostrando el resultado

Ejercicio "**Conversión de metros a pulgadas**"

Dada una medida en metros efectuar la conversión a pulgadas

Ejercicio "Calcular masa molecular del agua"

Dada los pesos atómicos del H (1 g) y del O (16 g) determinar la Masa molecular del Agua (H_20).

Ejercicio "Calcular masa molecular de la glucosa"

¿Qué modificaciones se requiere efectuar en el ejercicio anterior para calcular la masa molecular de la glucosa ($C_6H_{12}O_6$)?

3.2 Construcción de Expresiones

En esta sección se describen el manejo de los operadores aritméticos, relacionales y lógicos, estos últimos serán analizados haciendo uso de la tabla de verdad para cada operador. Al finalizar esta sección se analizaran una serie de ejercicios que permiten la combinación entre los distintos operadores para generar un resultado y que posteriormente serán utilizados en el desarrollo de programas.

3.2.1 Operadores Aritméticos

Para el desarrollo de programas se utilizaran con frecuencia los operadores aritméticos para evaluar formulas, generar resultados, etc. En la siguiente tabla se muestran los operadores aritméticos.

Nombre	Operador	Ejemplo
Multiplicación	*	5*6*0.5 → 15.0
División	/	100.0/2.0 → 50.0
Modulo (enteros)	mod	12 mod 3 → 0
Suma	+	5 + 10.5 → 15.5
Resta	-	5 – 10.5 → 5.5
Agrupación	( )	5*(6+1) → 35

Tabla 2.- Tabla de operadores aritméticos

Tanto como la asociatividad y precedencia de los operadores dentro de las expresiones aritméticas son factores importantes, ya que es posible que la expresión no se represente adecuadamente generando resultados distintos a los esperados. Es necesario utilizar paréntesis cuando la expresión involucra más de dos operadores aritméticos distintos y puedan generar confusión.

Operador	Asociatividad	Ejemplo
+, - (unitario)	Izq -Der	5 *-6 → 5*(-6)
*, /, mod	Izq -Der	4 mod 2 / 5 * 6 → ((4 mod 2) /5)*6
+, -	Izq -Der	5+-6-+15 → (5 + (-6)) - (15)

Tabla 3.- Tabla de precedencia y asociatividad [5].

Ejemplos "Expresiones Aritméticas"

1. 5*6*6*6
2. 7 * 10 – 5 mod 3 * 4 + 9
3. -5 + -3 * 8 mod 2 + -5 * 6
4. 8 mod 3 mod 2 * 3

3.2.2 Operadores Lógicos

Los **operadores lógicos** son utilizados para evaluar expresiones lógicas y generar como resultado un valor único: **Verdadero** (true) o **Falso** (false). Estos operadores lógicos permiten determinar el **flujo de control del programa,** es por ello la importancia de su estudio.

Nombre	Operador	Ejemplo
Conjunción (and, /\)	Y	V Y F → F
Disyunción (or, \/)	O	F O V → V
Negación (not, ¬)	NO	NO F → V

Tabla 4.- Tabla de operadores lógicos

3.2.3 Tabla de Verdad

El resultado que se genere al utilizar los operadores lógicos son en base a la definición de su correspondiente **tabla de verdad**, en una tabla de verdad se colocan los posibles valores a tomar y se evalúan con el operador generando un resultado.

Tabla de Verdad				
Valor	**Valor**	**Conjunción**	**Disyunción**	**Negación**
p	**q**	**p Y q**	**p O q**	**NO p**
V	V	V	V	F
V	F	F	V	F
F	V	F	V	V
F	F	F	F	V

3.2.4 Operadores Relacionales

Este tipo de operadores permiten evaluar expresiones que efectúan comparaciones entre ellos, generalmente se efectúan comparaciones numéricas, pero el operador de igualdad (y el de diferencia) permiten la comparación de otro tipo de operando.

Al evaluar una expresión con operadores relacionales el resultado generado un valor único: **Verdadero** (true) o **Falso** (false).

Operadores Relacionales		
Descripción	**Pseudocódigo**	**Ejemplo**
Menor que	<	5 < 3 → F
Mayor que	>	5 > 3 → V
Igual	=	6 = 10 → F
Menor igual que	≤	5 ≤ 5 → V
Mayor igual que	≥	5 ≥ 6 → F
No igual (distinto)	≠	5 ≠ 10 → 10

Es posible tener combinación de operadores aritméticos, relacionales y lógicos dentro de una expresión, el resultado que se genere es Verdadero o Falso. Este tipo de expresiones 'combinadas' tienen son de uso frecuente en el desarrollo de programas, ya que en base al resultado obtenido determinan el flujo de control del programa.

Ejemplos "Expresiones de operadores aritméticos, relacionales y lógicos"

1. 3 > 2 + 1 → F
2. 4 ≠ (2*2) → F
3. NO (4 ≠ 4) → V
4. 3 ≤ (3*2) →V
5. 3 > 2 Y 2 > 1→V
6. 3 > 2 Y 1 > 1→F
7. 3 > 2 O 1 > 1→V
8. 3 ≤ 3 Y 2 = 2 →
9. 3 ≤ 3 O 2 = 2 →
10. 3 ≤ 3 Y 2 ≠ 2 →
11. 3 ≤ 3 Y 2 ≠ 2 Y 3 = 4 →
12. (3 ≤ 3 Y 2 ≠ 2) O 3 = 3 →
13. (3 ≤ 3 Y 2 ≠ 2) O 3 = 4 →
14. 3 - 2 > 3 + 2 →
15. 2 mod 2 = 0 →
11. 4 / 2 = 2 Y 3 – 2 = 1 →
12. 5*2+3 ≠ 2*5+3 →
13. 14 mod 2 = 0 →
14. 21 mod 3 = 0 →
15. 15 mod 2 ≠ 0 →
16. 21 mod 2 =1 →
17. 4 mod 2 = 0 Y 4 mod 3 ≠ 0 →
18. 5 * 2 mod 3 = 0 Y 5 * 3 mod 2 = 1

3.3 Estructuras de Control

Las **estructuras de control** controlan el flujo de ejecución de un programa (o función), permiten combinar instrucciones individuales en una simple unidad lógica con un punto de entrada y un punto de salida [5].

Las instrucciones básicas y comúnmente utilizadas en la programación se condensan en cuatro grupos:

- Instrucciones de entrada y salida. (de transferencia de información y datos).
- Instrucciones aritméticas y lógicas.
- Instrucciones selectivas.
- Instrucciones repetitivas.

La clasificación de las estructuras de control se define en tres estructuras básicas mismas que permiten ***controlar el flujo de la ejecución***:

- Secuenciación: Ejecución sucesiva de pasos, tareas, acciones o instrucciones.
- Selección: Dependiendo de una condición dada, se determina si se efectúa una serie de acciones o se efectúa otra serie distinta.
- Iteración: Repetición de una serie de instrucciones u operaciones, mientras se cumpla una condición proporcionada.

En base a las estructuras básicas, el **pseudocódigo** representa una forma de expresar algoritmos de manera 'natural' utilizando un leguaje específico y hace uso de las tres estructuras básicas de la programación estructurada.

3.3.1 Secuenciación

La **secuenciación** se refiere a la ejecución continua de instrucciones. Por ejemplo, con el uso de diagramas de flujo para las instrucciones A, B y C, se describen de la siguiente manera:

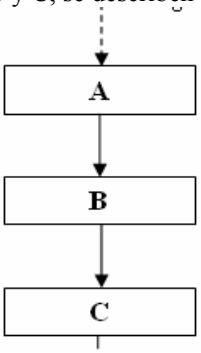


Figura 20.- Representación de secuencia con el uso de diagramas de flujo.

3.3.2 Selección condicional simple, doble y múltiple

La **selección condicional** es representada mediante una sentencia en la cual contiene una expresión que representa la condición de la sentencia de selección. Si la expresión se cumple (es ***verdadera***) se desarrollan una sección de instrucciones, en caso contrario se ejecuta otra serie de instrucciones.

Selección condicional simple

La sentencia condicional simple considera un sólo caso, se verifica la condición en caso de que se cumpla, es decir que sea verdadera la condición, se desarrolla una serie de instrucciones, en caso contrario no es considerado.

En **algoritmo** se representa de la siguiente forma:
1) Evaluar **Si** condición **entonces** ir a paso 2
2) Realizar la operación A

La representación en un **Diagrama de Flujo** se muestra en del lado izquierdo del esquema, y del lado derecho se tiene la traducción del Diagrama de Flujo a **Pseudocódigo**.

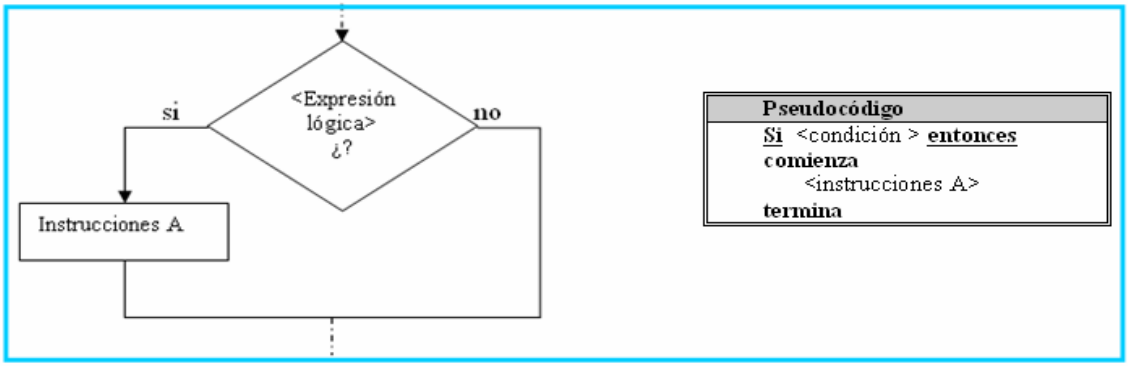


Figura 21.- Representación de una selección condicional simple.

Ejemplo "Valor positivo"

Determinar si un valor introducido por el usuario es positivo.

Algoritmo

Entrada: Valor numérico introducido por el usuario (Valor).

Salida: Mensaje indicando si el Valor es positivo.

Procedimiento:

1) Escribir mensaje solicitando al usuario que teclee un número
2) Leer el valor tecleado (Valor)
3) Evaluar **Si** Valor > 0 **entonces** ir a paso 4
4) Escribir mensaje indicando que el valor es positivo.

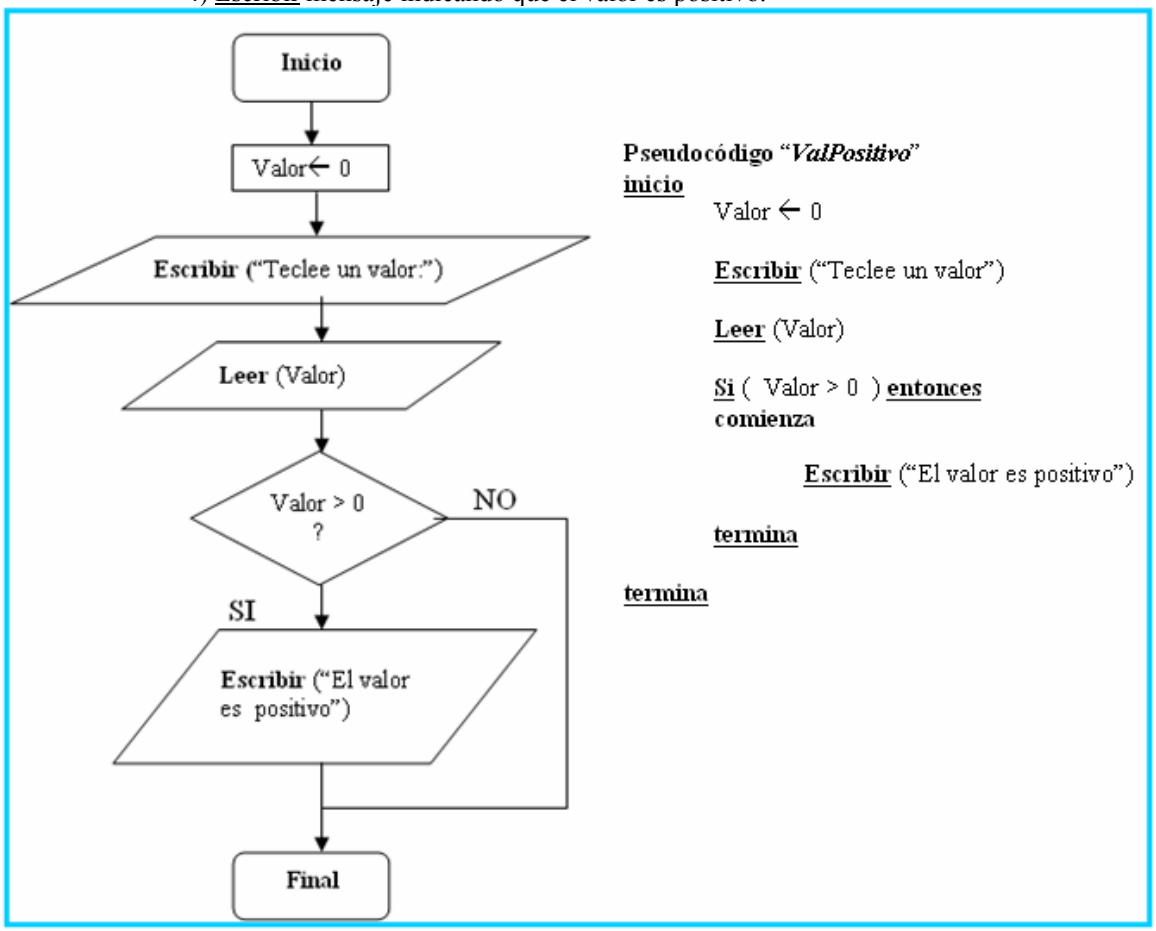


Figura 22.- Ejemplo Diagrama de Flujo y Pseudocódigo.

Ejercicio "Valor positivo, negativo o cero"

De acuerdo al siguiente algoritmo desarrolle el respectivo diagrama de flujo, y posteriormente el pseudocódigo.

Algoritmo

Entrada: Valor numérico introducido por el usuario (Valor).

Salida: Mensaje indicando si el Valor es positivo, negativo o igual a cero.

Procedimiento:

1) Escribir mensaje solicitando al usuario que teclee un número
2) Leer el valor tecleado (Valor)
3) Evaluar **Si** Valor > 0 **entonces** ir a paso 4
4) Escribir mensaje indicando que el valor es positivo.
5) Evaluar **Si** Valor < 0 **entonces** ir a paso 6
6) Escribir mensaje indicando que el valor es negativo.
7) Evaluar **Si** Valor = 0 **entonces** ir a paso 8
8) Escribir mensaje indicando que el valor es igual a cero.

Selección condicional doble

La selección condicional doble a diferencia de la simple, considera el caso alterno. Al verificar la condición y en caso de que se cumpla (sea verdadera) se efectúa una serie de instrucciones, caso contrario de que no se cumpla (sea falsa) se desarrollan otra serie de instrucciones. En esta estructura depende de la condición para determinar el desarrollo de una serie de instrucciones u otra serie.

En **algoritmo** se representa de la siguiente forma:

1) Evaluar **Si** condición **entonces** ir a paso 2 **en otro caso** ir a paso 3
2) Realizar la operación A
3) Realizar la operación B

La representación en un **Diagrama de Flujo** se muestra en del lado izquierdo del esquema, y del lado derecho se tiene la traducción del Diagrama de Flujo a **Pseudocódigo**.

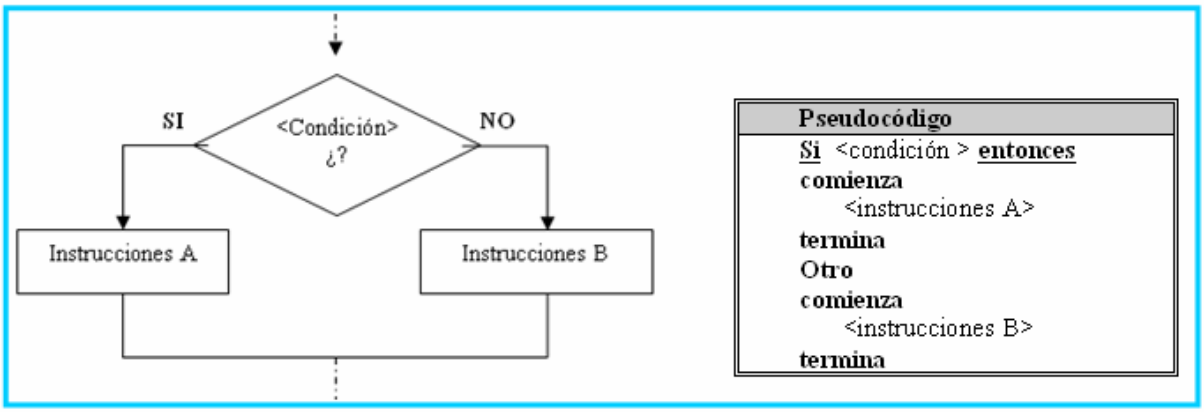


Figura 23.- Representación de una selección condicional doble.

Ejemplo "Calcular el valor absoluto de un número entero"

Calcular el valor absoluto de un número entero (incluye al cero).

Algoritmo

Entrada: Valor numérico introducido por el usuario (Valor).
Salida: Mensaje indicando el valor absoluto del Valor.
Procedimiento:

1) Escribir mensaje solicitando al usuario que teclee un número
2) Leer el valor tecleado (Valor)
3) Evaluar **Si** Valor < 0 **entonces** ir a paso 4 **en otro caso** ir a paso 5
4) Evaluar Valor * (-1) y reasignar a ValorAbsoluto.
5) Asignar Valor a ValorAbsoluto.
6) Escribir mensaje indicando el valor absoluto del Valor que es ValorAbsoluto.

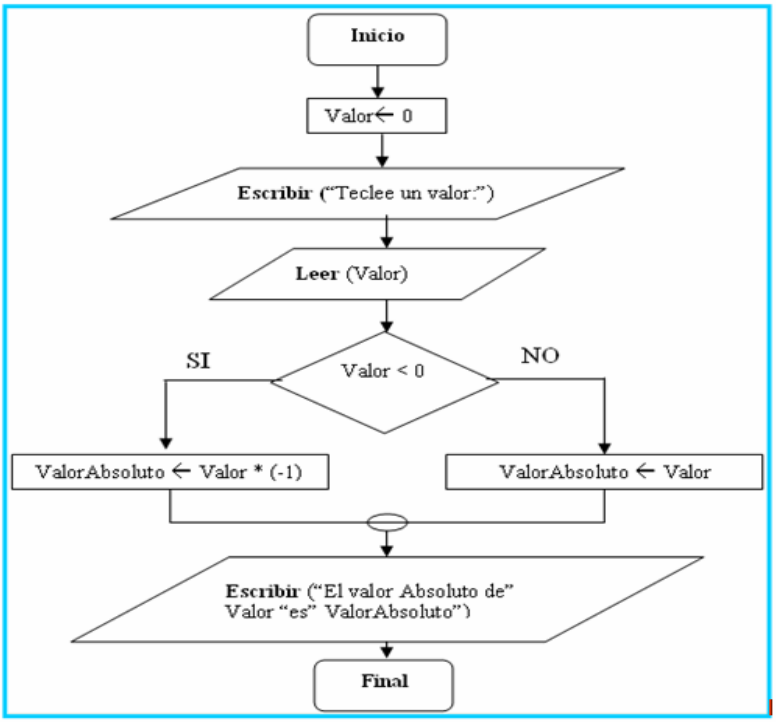


Figura 25.- Ejemplo Diagrama de Flujo y Pseudocódigo.

Pseudocódigo "***ValPosNegOCero***"

```
inicio
        Valor ← 0
        Escribir ("Teclee un valor")
        Leer (Valor)

        Si ( Valor < 0 ) entonces
        comienza
                ValorAbsoluto ← Valor * (-1)
        termina
        Otro
        comienza
                ValorAbsoluto ← Valor
        termina
        Escribir ("El valor absoluto de" Valor "es" ValorAbsoluto)
termina
```

Ejercicio "**Número positivo o negativo**"

El usuario ingresa un número (distinto de cero), el programa debe determinar si dicho número es positivo o negativo e indicarlo con un mensaje. Desarrolle el respectivo Diagrama de Flujo y Pseudocódigo

Ejemplo "Valor positivo, negativo o cero con condicional doble"

Determinar si un valor introducido por el usuario es positivo, negativo o igual a cero, utilizando la estructura condicional doble.

Algoritmo

Entrada: Valor numérico introducido por el usuario (Valor).

Salida: Mensaje indicando si el Valor es positivo, negativo o igual a cero.

Procedimiento:

1) Escribir mensaje solicitando al usuario que teclee un número
2) Leer el valor tecleado (Valor)
3) Evaluar **Si** Valor > 0 **entonces** ir a paso 4 **en otro caso** ir a paso 5
4) Escribir mensaje indicando que el valor es positivo.
5) Evaluar **Si** Valor < 0 **entonces** ir a paso 6 **en otro caso** ir a paso 7
6) Escribir mensaje indicando que el valor es negativo.
8) Escribir mensaje indicando que el valor es igual a cero.

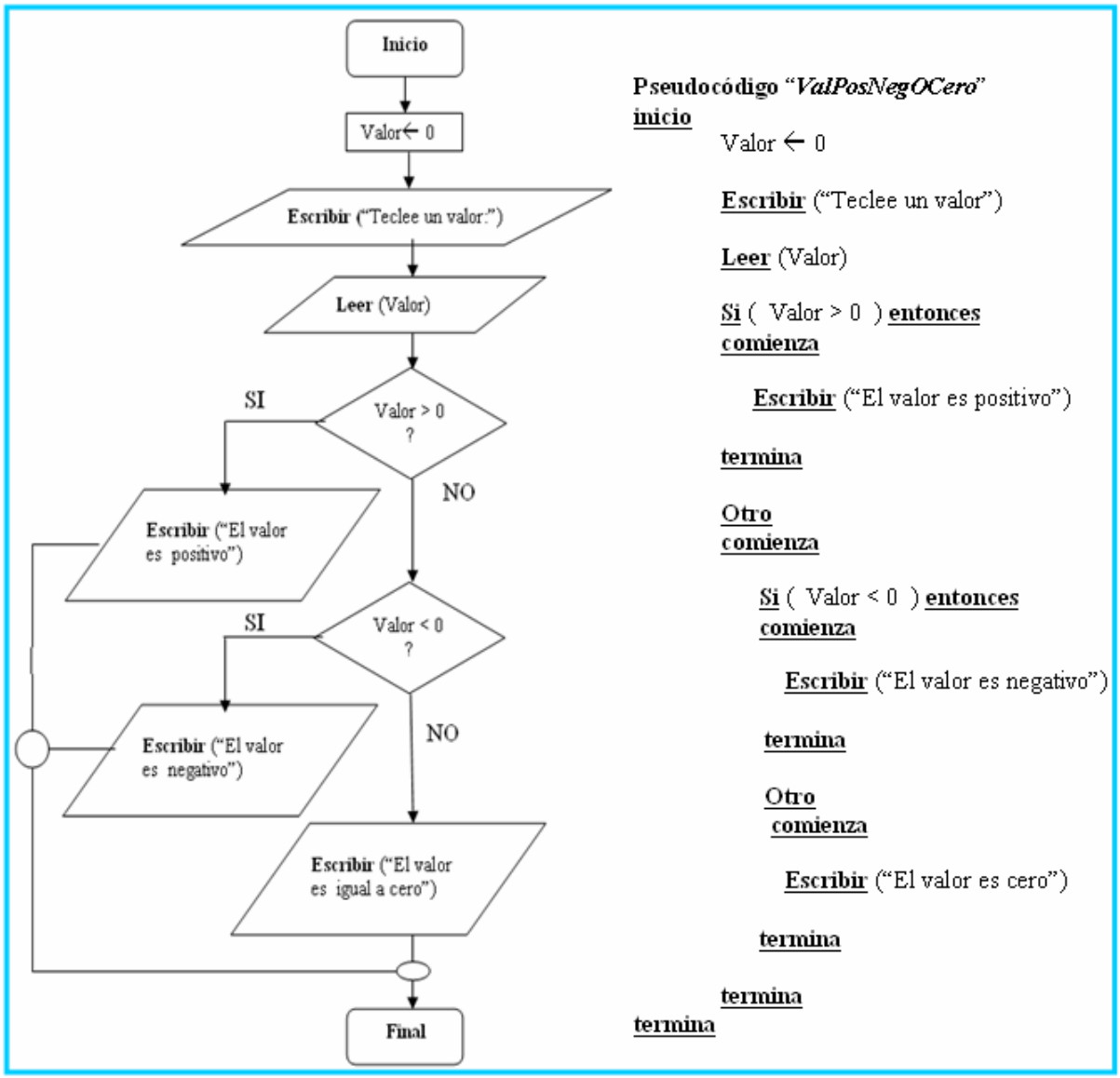


Figura 25.- Ejemplo Diagrama de Flujo y Pseudocódigo.

Ejercicios de la estructura de control selección condicional

Desarrolle el Diagrama de Flujo y Pseudocódigo para los siguientes ejercicios.

a) "**Igualdad de dos números**"

El usuario ingresa dos números enteros, el programa debe determinar enviando un mensaje indicando si dichos números son iguales.

b) **"Parida de un número** "

El usuario ingresa un número entero positivo, el programa determina si dicho número es par o impar.

Sugerencia: Utilizar la estructura condicional para determinar si es divisibles, es decir si el residuo es igual a cero entonces es divisible. Ejemplo: al efectuar 400 mod 2 = 0 entonces es par, pero 13 mod 2 = 1 entonces es impar.

c) "**Divisibilidad de un número**"

El usuario ingresa dos números entero positivo, el programa debe determina si el primer número es divisible por el segundo, en cualquier caso se envía un mensaje. ***Sugerencia***: Puede basarse en el ejercicio anterior.

d) "**Aprobado o no aprobado**"

Al ingresar el usuario una calificación entre 0 y 10, el programa determina si dicha calificación es aprobatoria o no enviando un mensaje al usuario.

e) "**Calificación con Nota**"

Al ingresar el usuario una calificación entre 0 y 10, el programa determina la nota de la calificación en base al esquema de evaluación de curso (ver plan de estudios). Ejemplo: calificación 9.2 → nota MB.

f) "**Raíces de una ecuación cuadrática**"

La siguiente formula permite calcular las raíces de una ecuación cuadrática.

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Con la condición de que $a \neq 0$.

Desarrolle un programa que solicite al usuario los valores de a, b y c, Que valide primero que $a \neq 0$, en otro caso calcular el discriminante para obtener el resultado de las dos raíces.

Algoritmo

Entrada: Valores **a, b** y **c**.

Salida: Mensaje indicando si existieron conflictos o en su defecto el resultado de las dos raíces.

Procedimiento:

1) Escribir "Teclee el valor de a"
2) Leer valor **a**
3) Escribir "Teclee el valor de b"
4) Leer valor **b**
5) Escribir "Teclee el valor de c"
6) Leer valor **c**
7) Evaluar **Si a** = 0. **entonces** ir paso 8 **otro caso** ir paso 9
8) Escribir "Error en la división" y pasar al paso 15.
9) Calcular el discriminante , es decir evaluar b*b – 4* a*c y asignárselo a **discrim**
10) Evaluar **Si discrim** <0 **entonces** ir paso 11 **otro caso** ir paso 12
11) Escribir "El resultado genera raíces imaginarias" y pasar al paso 15.
12) Calcular la primera raíz, evaluar la expresión (–b + **raiz** (**discrim**))/ (2*a) y asignar a **x1**.
13) Calcular la segunda raíz, evaluar la expresión (–b - **raiz** (**discrim**))/ (2*a) y asignar a **x2**.
14) Escribir "La raíces son: **x1** y **x2**"
15) Escribir mensaje de despedida.

Notas:

La función **raiz** () permite calcular la raíz cuadrada de un valor.

Efectué **solo** el pseudocódigo.

3.3.3 Iteración condicional (repetición)

Existen problemas en los cuales se requiere efectuar una serie de instrucciones una cierta cantidad de veces, es decir, se requiere de repetir dichas instrucciones. Para repetir una cierta cantidad de veces un bloque de instrucciones se utilizan las **estructuras de control iterativas o repetitivas**.

Las tres estructuras de control con las cuales se trabajarán son:

- **Mientras**
- **Hacer …. Mientras.**
- **Para … hasta … hacer**

Cada una de estas estructuras presenta características que permiten efectuar iteraciones de acuerdo al diseño de la solución planteada para un problema específico.

Un **ciclo** (bucle) representa una sección del programa que repite una serie de instrucciones (o una sola instrucción), por un número determinado de veces de acuerdo a la validación de una **condición**. La condición es determinada por una **expresión lógica, relacional o booleana**, que permite controlar la secuencia de repetición de la estructura.

Las dos partes principales de un ciclo, son:

-**Cuerpo del ciclo**: representa el conjunto de instrucciones que se repetirán de acuerdo a una condición. Es posible considerar dentro del cuerpo del ciclo otro tipo de estructuras, como las de selección condicional o de iteración).

-**Iteración**: representa cada una de las repeticiones del cuerpo del ciclo.

De acuerdo a la **condición** de la estructura condicional, se determina el número de iteraciones que se efectuaran sobre el cuerpo del ciclo. Regularmente este tipo de estructuras van acompañadas de un **contador,** él cual incrementa o decrementa cada vez que se desarrollan las instrucciones del cuerpo del ciclo.

Reflexión de las estructuras de control de iteración condicional

Las estructuras de control iterativas son utilizadas cuando "se desea ejecutar una serie de instrucciones un numero determinado de veces".

Por ejemplo si se desea sumar '**5**' veces un valor dado '**10**', se tendría que efectuar 5 veces la suma del mismo valor. En el siguiente análisis y diseño se desarrolla con la estructura de control secuencial, en donde se hace uso de la variable **resultado** (una variable tiene la capacidad de almacenar un valor y dicho valor puede variar o cambiar).

Algoritmo
Entrada: No hay elementos de entrada
Salida: Mensaje indicando el resultado de las operaciones efectuadas.
Procedimiento:
1) Asignar a **resultado** el valor de 0.
2) Evaluar la operación de sumar **resultado** y 10 y reasignar a la variable **resultado** otra vez.
3) Evaluar la operación de sumar **resultado** y 10 y reasignar a la variable **resultado** otra vez.
4) Evaluar la operación de sumar **resultado** y 10 y reasignar a la variable **resultado** otra vez.
5) Evaluar la operación de sumar **resultado** y 10 y reasignar a la variable **resultado** otra vez.
6) Evaluar la operación de sumar **resultado** y 10 y reasignar a la variable **resultado** otra vez.
7) Escribir el resultado del contenido del **resultado** (que es el valor de 50).

Diagrama de Secuencia

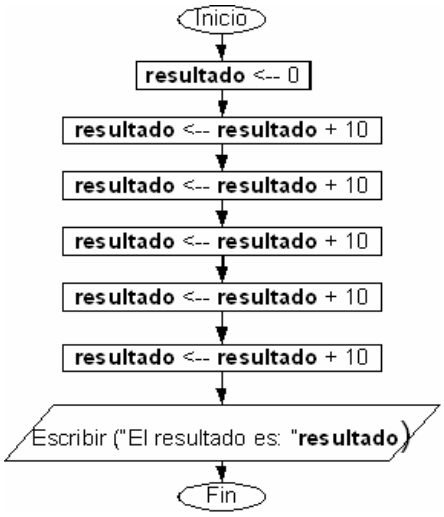


Figura 26.- Diagrama de flujo del ejemplo de sumar 5 veces el valor de 10.

En el ejemplo anterior se deseaba efectuar una misma instrucción un número determinado de veces, en este caso sólo 5 veces. ¿Qué pasaría si se desea desarrollar la misma operación de suma 100 veces o 1000 veces?, se tendrían que desarrollar ¡1000 instrucciones similares!, es claro que esto resultaría muy engorroso para el programador: teclear la misma instrucción un número determinado de veces.

Es aquí en donde se desea hacer uso de un mecanismo que permita desarrollar un conjunto de instrucciones un número determinado de veces, este es el enfoque de las estructuras de control iterativa.

Al desarrollar el mismo algoritmo del ejemplo anterior se obtendría lo siguiente, se consideran los mismos elementos de entrada y salida:

Procedimiento:
1) Asignar a **resultado** el valor de 0.
2) Evaluar la operación de sumar **resultado** y 10 y reasignar a la variable **resultado** otra vez.
3) Repetir el paso 2 hasta que sean 5 veces.
4) Escribir el resultado del contenido del **resultado** (que es el valor de 50).

En el algoritmo anterior, se entiende adecuadamente, pero ¿Cómo se determinará que ya van 5 veces? La respuesta es 'contándolas' o bien utilizando un contador que indique cuantas iteraciones sobre las instrucciones se han desarrollado.

El siguiente algoritmo contempla el uso de un contador con el cual se determinará el número de veces que se han ejecutado dichas instrucciones.

Procedimiento:
1) Asignar a **resultado** el valor de 0.
2) Asignar a **contador** el valor de 1.
3) Evaluar mientras el **contador** sea menor a 5 hacer paso 4 al 5, en caso contrario paso 6.
4) Evaluar la operación de sumar **resultado** y 10 y reasignar a la variable **resultado** otra vez.
5) Incrementar en uno el **contador**.
6) Escribir el resultado del contenido del **resultado** (que es el valor de 50).

Observación: es claro que se esta utilizando un contador y en el paso 3, se presenta un estructura de control con una ***condición*** que valida que el contador no se sobrepase del rango. En este ejemplo el contador empieza con un valor igual a 1, y se efectúa 5 veces la misma operación.

En algunos problemas es posible que el contador empiece con un valor cero o bien con un valor grande. De igual manera es posible que se desarrolle un incremento o decremento de l contador, es por esto que es necesario tener cuidado con la manipulación del contador de acuerdo a la condición del ciclo iterativo.

Diagrama de Secuencia

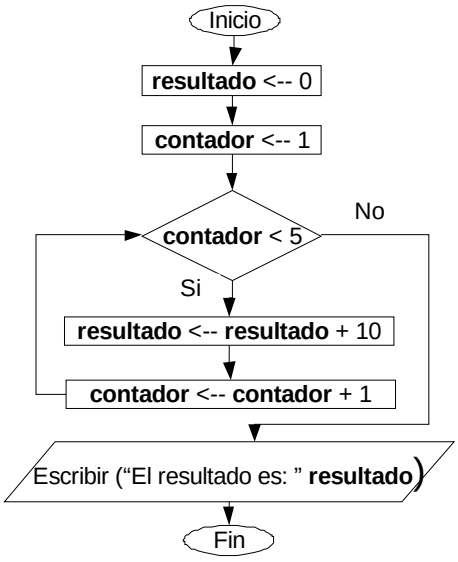


Figura 27.- Diagrama de flujo del ejemplo de sumar 5 veces el valor de 10, utilizando una estructura iterativa.

En el ejemplo anterior se utilizó la sentencia de control iterativa llamada ***mientras***, la cual se detallará en la siguiente sección. El problema del ejemplo permite desarrollar la multiplicación de dos valores, en este caso se suma 5 veces el valor de 10 lo que da como resultado 50, es decir el resultado de multiplicar el valor de 5 por 10.

3.3.3.1 Sentencia Mientras

La sentencia del cuerpo del ciclo se repite **Mientras** que la condición (de la expresión) sea validad o verdadera. Cuando se evalúa la condición y resulta invalidad o falsa, se terminar y se sale del ciclo. Después de salir del ciclo se ejecuta la siguiente instrucción fuera del ciclo **Mientras** [5].

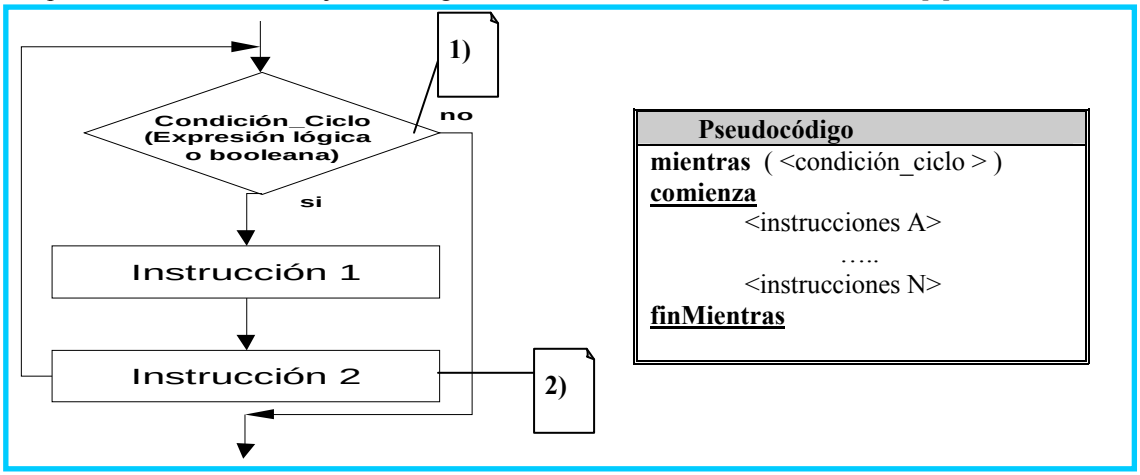


Figura 28.- Representación en Diagrama de Flujo y Pseudocódigo de la estructura de control iterativa **Mientras**

Notas:

1) "Si la condición es falsa (aun desde el inicio), no se efectúan las intrusiones del cuerpo del ciclo. Se continua con la siguiente instrucción del programa".

2) Una vez finalizada la ultima instrucción del cuerpo del ciclo, el flujo continua para verificar nuevamente si se cumple la condición.

La estructura de control **Mientras**, es posible que se efectúe:

-***Ninguna iteración***: en caso de que la condición de la iteración no se cumpla por completo.

-***Una sola iteración:*** considerando que se efectúa la primera iteración y en la siguiente ya no se cumpla la condición.

-***Por un tiempo indeterminado (ciclo infinito)***: esto ocurre cuando siempre se cumple la condición y no se especifica como finalizar las iteraciones.

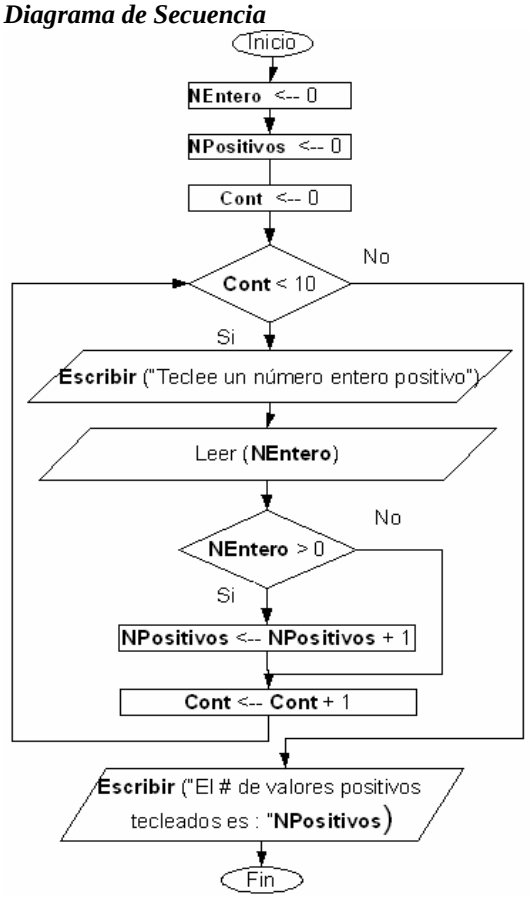
Ejemplo "Número de Positivos Tecleados"

El usuario tecleara 10 números positivos y el programa deberá de ir contando la cantidad de números positivos tecleados.

Entrada: Una serie de números enteros (**NEntero**).
Salida: Mensaje indicando el números de valores positivos tecleados por el usuario (**NPositivos**).
Procedimiento:

1) Inicializar el valor **NEntero** en cero.
2) Inicializar el valor **NPositivos** en cero.
3) Inicializar el contador (**Cont**) en cero.
4) Evaluar **mientras** contador sea menor a 10 **hacer** del paso 5 a 9 en caso contrario pasar al paso 10.
5) Escribir mensaje indicándole al usuario que teclee un número entero
6) Leer el valor tecleado por el usuario y reasignárselo a **NEntero**.
7) Evaluar **si NEntero** es mayor a cero, **entonces** pasar al paso 8, caso contrario al paso 9.
8) Incrementar en uno el contenido de la variable **NPositivos**.
9) Incrementar en uno el contenido de la variable **Cont**.
10) Regresar al paso 4.
11) Escribir el número de valores positivos contenido en la variable **NEntero**.
12) Escribir mensaje de despedida.

Diagrama de Secuencia



Pseudocódigo "ContPositivos"

comienza
NEntero ← 0
NPositivos ← 0
Cont ← 0

mientras (Cont < 0)
comienza

Escribir ("Teclee un número entero positivo")

Leer (NEntero)

Si (NEntero > 0) **entonces**
comienza

NPositivos ← Npositivos + 1

termina

Cont ← Cont +1

termina

Escribir ("El número de valores positivos tecleados
es:" NPositivos)

termina

Figura 29.- Diagrama de flujo con el uso del mientras.

Ejemplo "Diez Números Siguientes"

Leer un número entero y escribir en pantalla los siguientes 10 enteros.

Entrada: Número entero (**NumEntero**).
Salida: Los siguientes 10 números enteros.
Procedimiento:

13) Escribir mensaje indicándole al usuario que teclee un número entero
14) Leer **NumEntero**, es decir, el valor inicial.
15) Asignar al valor de **Fin** el resultado de la suma del valor inicial más 10.
16) Evaluar **si NumEntero** es menor al valor de **Fin**, **entonces** ir al paso 5, **en otro caso** ir al paso 8.
17) Incrementar en uno el valor inicial (**NumEntero**).
18) Escribir el contenido de **NumEntero**.
19) Regresar el paso 4.
20) Escribir mensaje de despedida.

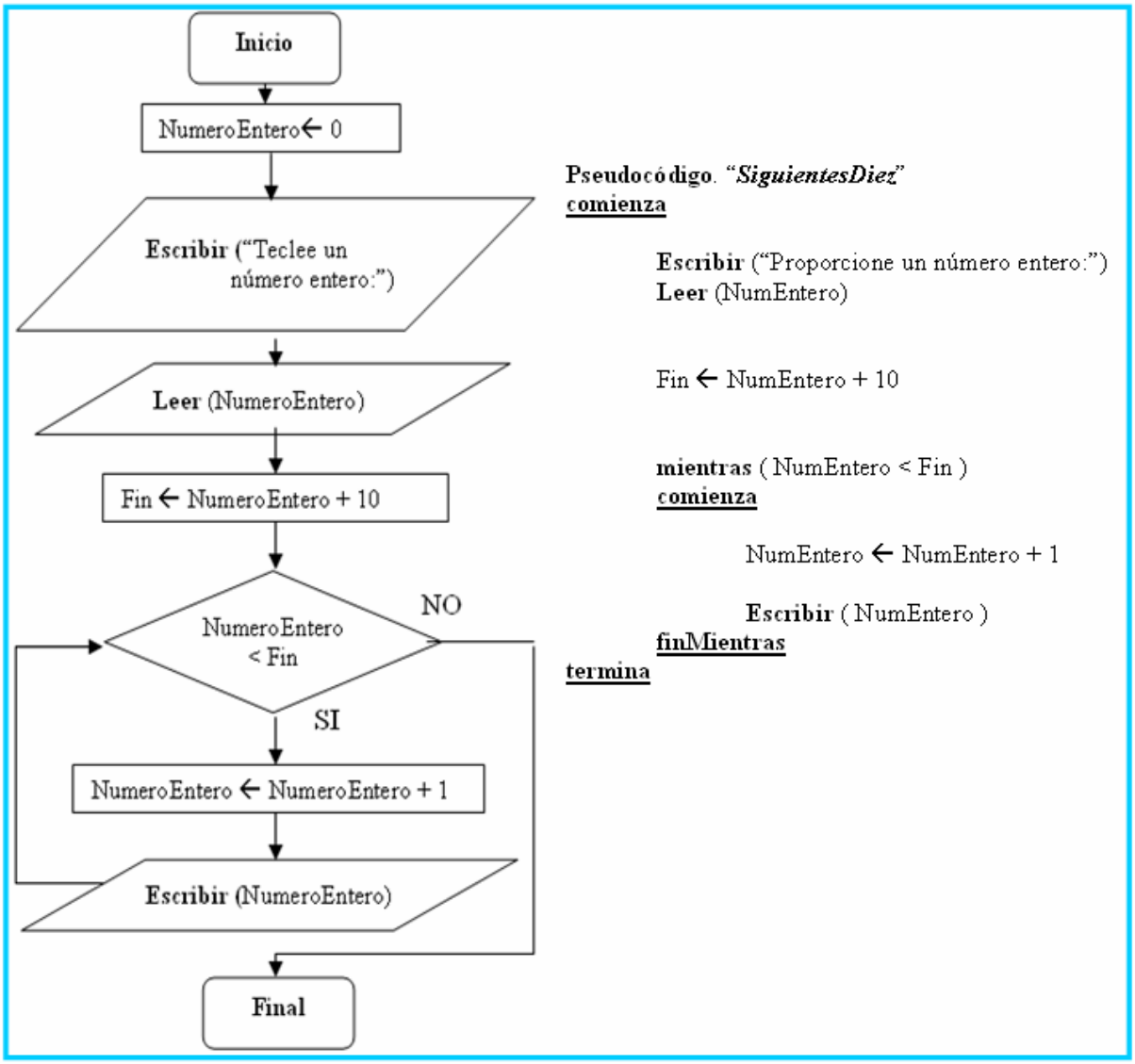


Figura 30.- Ejemplo diez números siguientes, representación del Diagrama de Flujo y Pseudocódigo.

Ejercicios de la estructura de control iteración condicional – Mientras

Desarrolle el Diagrama de Flujo y Pseudocódigo para los siguientes ejercicios.

a) "**Calificación promedio**"

El usuario ingresará 5 números reales (punto flotante) y el programa debe de ser capaz de leer esas cinco calificaciones sumándolas. Posteriormente se calcula el promedio y se determina si aprobó (calificación mayor o igual a 6.0) o no aprobó (calificación menor a 6.0).

b) "**Cantidad de números ingresados**"

El usuario ingresara cierta cantidad de números enteros hasta que teclee el valor de -1, en ese momento el programa debe de indicar con un mensaje cuantos números el usuario ingresó. El programa debe de ir contando cada uno de los datos ingresados, considerando la condición hasta que se haya tecleado el valor de -1.

c) "**Potencia**"

El usuario ingresara un valor entero positivo (mayor a cero) que representará la base, posteriormente ingresara un valor entero positivo (mayor a cero) que representara el exponente. El programa obtendrá la potencia multiplicando tantas veces la base por si misma como el valor del exponente.

Algoritmo

Entrada: Dos número entero positivos (**Base** y **Exponente**).
Salida: El resultado de elevar la base a un exponente dado, es decir, la potencia.
Procedimiento:

1) Asignar a **Resultado** el valor de 1 y a **Contador** el valor de 0.
2) Escribir mensaje indicándole al usuario que teclee un número entero positivo
3) Leer **Base**
4) Escribir mensaje indicándole al usuario que teclee un número entero positivo
5) Leer **Exponente**
6) Evaluar **si Contador** es menor al valor de **Exponente**, **entonces** ir al paso 7, **en otro caso** ir al paso 10.
7) Evaluar la expresión **Resultado** * **Base** y asignar a **Resultado**
8) Incrementar en uno el **Contador**.
9) Regresar el paso 6.
10) Escribir el resultado de la potencia, es decir, **Resultado**.
11) Escribir mensaje de despedida.

3.3.3.2 Sentencia Hacer … Mientras

La sentencia ***hacer… mientras*** presenta una similitud con el ciclo condicional – ***mientras***, con la diferencia que **se ejecuta siempre al menos una vez**. La condición del ciclo ***mientras*** se evalúa al inicio del ciclo, por otro lado la condición de ***hacer… mientras*** se evalúa al final. Es útil cuando se requiere que se desarrolle al menos una vez el cuerpo del ciclo.

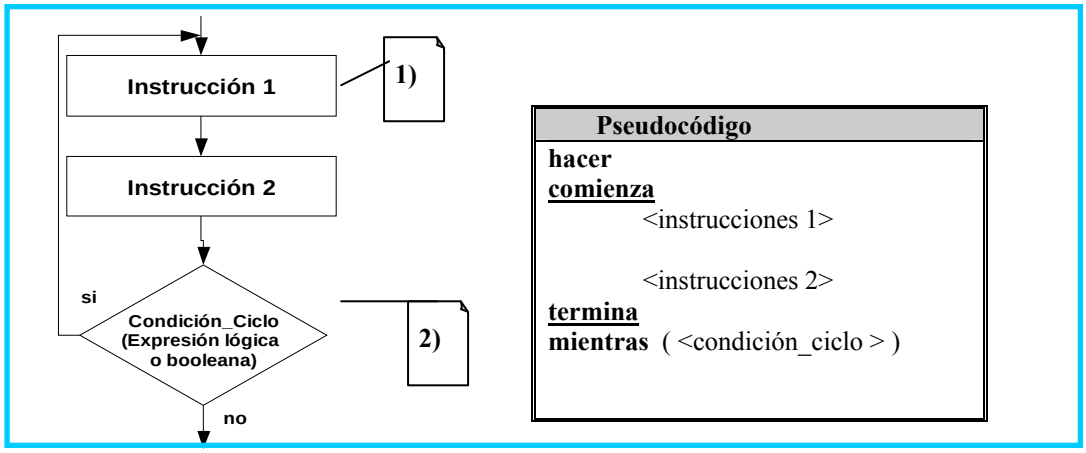


Figura 31.- Diagrama de Flujo y Pseudocódigo de la estructura de control iterativa **Hacer… Mientras**

Notas:

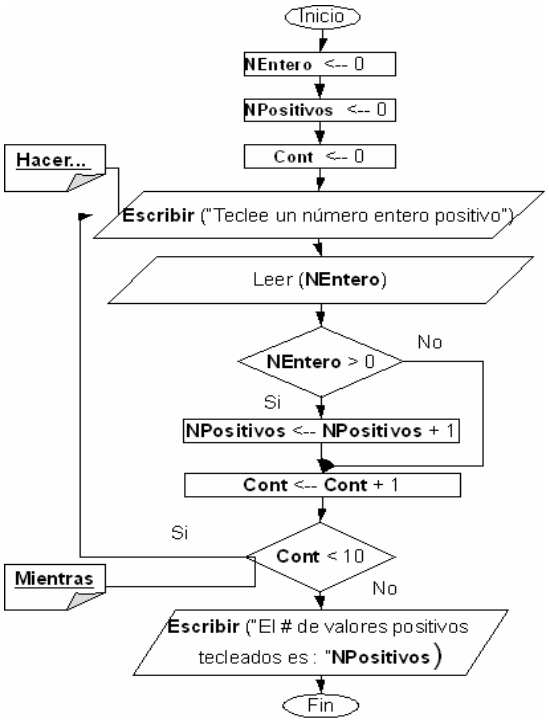
1) Cada una de las instrucciones que antepone a la condición del ciclo se efectúa al menos una vez.

2) Al evaluar la condición de ciclo y da como resultado **falso** (no se cumple la condición) se sale del ciclo y continua con la siguiente instrucción después del ciclo, en caso de que el resultado sea **verdadero** (se cumple la condición) se continua con la ejecución del cuerpo del ciclo una vez más.

Ejemplo "Número de Positivos Tecleados – Versión *Hacer... Mientras*"

El usuario tecleara 10 números positivos y el programa deberá de ir contando la cantidad de números positivos tecleados.

Diagrama de Flujo



Pseudocódigo "*ContPositivosHM*"

```
comienza
        NEntero ← 0
        NPositivos ← 0
        Cont ← 0

        hacer
        comienza
            Escribir ("Teclee un número entero positivo")
            Leer ( NEntero)

            Si ( NEntero > 0 ) entonces
            comienza
                NPositivos ← Npositivos + 1
            termina

            Cont ← Cont +1
        termina
        mientras ( Cont < 0 )

        Escribir ("El número de valores positivos tecleados es:" NPositivos)
Termina
```

Figura 31.- Diagrama de flujo con el uso de *Hacer… Mientras*.

Ejemplo "Multiplicación de 5 números enteros"

Elaborar el diseño de programa que multiplique los 5 primeros números enteros (1, 2, 3, 4 y 5), al final se muestre el resultado obtenido.

Análisis:

-Se requiere de un contador para determinar el número de iteraciones.

-El contador permitirá determinar la "***condición de paro***" del ciclo, que indica hasta que momento se deja de efectuar el ciclo de la iteración.

Entrada: Ninguna entrada por parte del usuario.
Salida: Resultado de multiplicar los 5 primeros números enteros.
Procedimiento: "El procedimiento se representará mediante un diagrama de flujo y el Pseudocódigo".

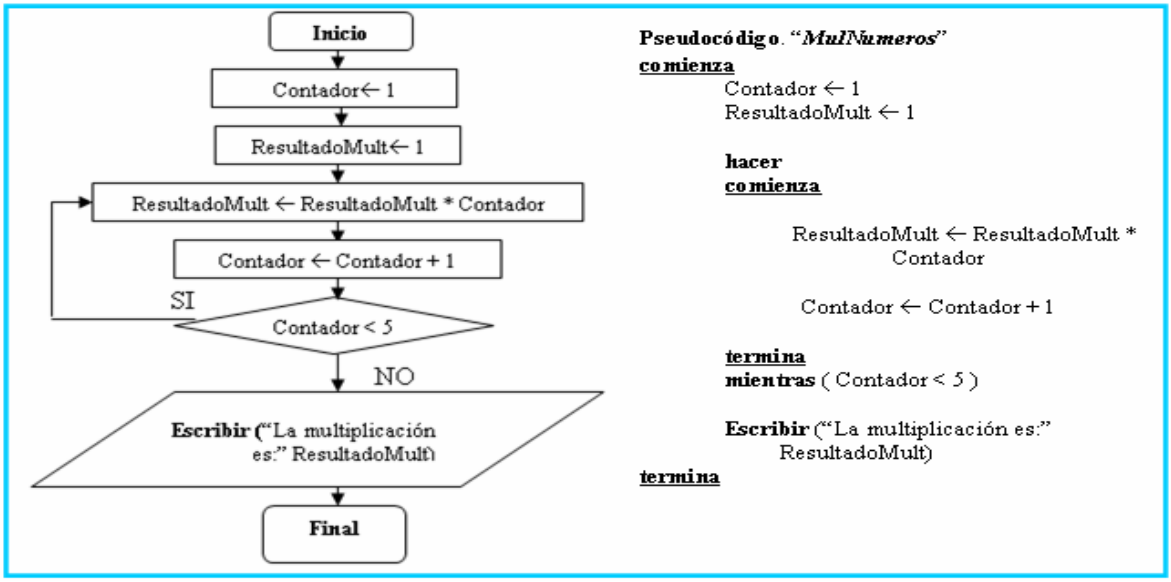


Figura 32.- Diagrama de flujo y Pseudocódigo del ejemplo de multiplicación de números.

Ejemplo "Promedio de N Calificaciones"

El usuario ingresara una cierta cantidad de calificaciones (números reales positivos) hasta que teclee un número negativo. Cada calificación que es ingresada por el usuario deberá de irse sumando, para al final se calcule el promedio y se determine si aprobó (calificación mayor o igual a 6.0) o no aprobó (menor a 6.0).

Análisis

-No se sabe la cantidad de números a ingresar. La '**condición de paro**' del ciclo, que indica hasta que momento se deja de recibir números, es cuando el usuario teclee un número negativo (< 0).
-El cuerpo del ciclo debe de efectuar la suma de los valores ingresados.
-Se requiere de un contador para determinar cuantas calificaciones se han ingresado y calcular el promedio.
-Al finalizar el ciclo se debe de calcular el promedio y utilizar una estructura condicional.

Entrada: Calificaciones que representan números reales (**CalificaN**).
Salida: Promedio de las calificaciones y un mensaje indicando si aprobó o no.
Procedimiento: "El procedimiento se representará mediante un diagrama de flujo y el Pseudocódigo".

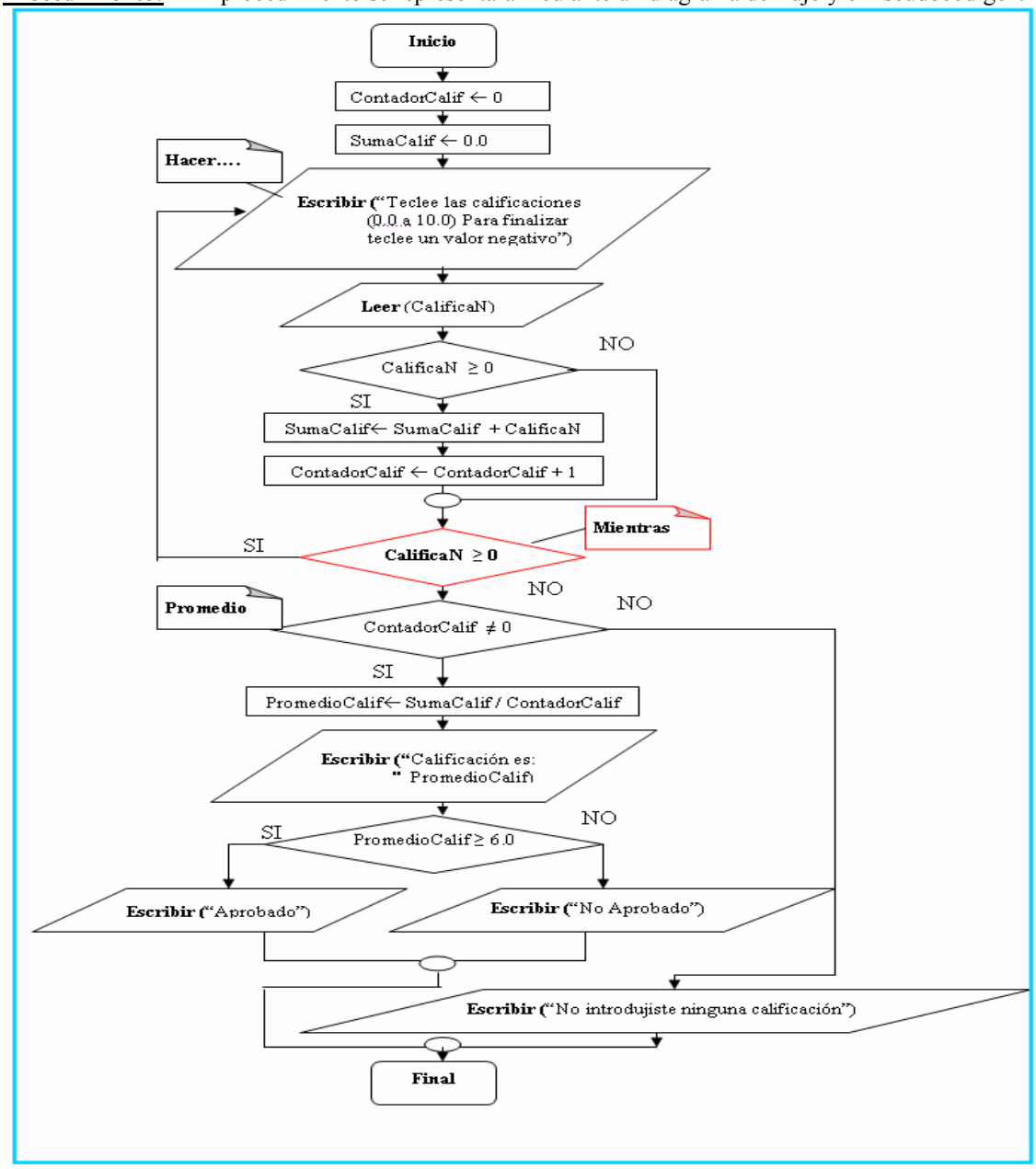


Figura 33.- Ejemplo promedio de N calificaciones representación del Diagrama de Flujo.

Pseudocódigo. "***Promedio de N Calificaciones***"
comienza

ContadorCalif ← 0
SumaCalif ← 0.0
PromedioCalif ← 0.0

hacer
comienza
Escribir ("Teclee las calificaciones (0.0 a 10.0) Para finalizar teclee un valor negativo:")
Leer (CalificaN)

Si (CalificaN ≥ 0) **entonces**
comienza

SumaCalif← SumaCalif + CalificaN
ContadorCalif ← ContadorCalif + 1

t**ermina**
termina
mientras (CalificaN ≥ 0)

Si (ContadorCalif ≠ 0) **entonces**
comienza

PromedioCalif← SumaCalif / ContadorCalif

Escribir ("Calificación es: **"** PromedioCalif)

Si (PromedioCalif ≥ 6.0) **entonces**
comienza

Escribir ("Aprobado")

termina
Otro
comienza

Escribir ("No Aprobado")

termina

termina
Otro
comienza

Escribir ("No introdujiste ninguna calificación")

termina

termina

Ejercicios de la estructura de control iteración condicional – Hacer … Mientras

Desarrolle el Diagrama de Flujo y Pseudocódigo para los siguientes ejercicios.

a) "**Multiplicación de 10 números enteros (versión 2)**"

El usuario ingresará 10 números enteros y el programa debe leer uno a uno de los números y efectuar la multiplicación. Al final se debe de mostrar el resultado obtenido. ***Sugerencia:*** Utilice como base el ejemplo "multiplicación de 5 números enteros".

b) "**Contador de Números Pares**"

El usuario ingresara cierta cantidad de números enteros pares hasta que se teclee un número impar, en ese momento el programa debe de indicar con un mensaje cuantos números pares se teclearon. El programa debe de ir contando cada uno de los números ingresados siempre y cuando sea par. Ejemplo:

Teclee un valor: 6
Teclee un valor: 10
Teclee un valor: 3
Usted tecleo 2 valores pares.

c) "**Suma de Números Impares**"

El usuario ingresara cierta cantidad de ***números enteros positivos*** hasta que se teclee ***el cero o un valor negativo***, al finalizar se debe de indicar la suma de todos aquellos números impares que el usuario ingreso. El programa debe de ir sumando SÓLO los números impares ingresados. Ejemplo:

Teclee un valor: 9
Teclee un valor: 10
Teclee un valor: 3
Teclee un valor: 0
La suma de los números impares tecleados es: 11

3.3.3.3 Sentencia Para … hasta con … hacer

Esta ultima estructura de control de iteración se verá más a detalle en el capitulo 4, en donde se incluirá la sintaxis en un lenguaje de programación seleccionado.

3.3.4 *Ejercicios de Unidad 3*

Los siguientes ejercicios permiten ejercitar aspectos prácticos de la unidad 3.

I. Algoritmos.

Realice el análisis (elementos de entrada y salida) y diseño (procedimiento) de cada uno de los siguientes problemas.

1. "***Preparación de una palomitas en horno de microondas***"
   Desarrolle un algoritmo que indique el proceso que se lleva al preparar unas deliciosas palomitas con el uso de un horno de microondas.
2. "***Cambiar un foco***"
   Desarrolle un algoritmo que describa el proceso para cambiar un foco que se encuentra en el techo de una casa de tamaño regular.

II. Expresiones Aritméticas, Lógica y Relacionales

Dadas las siguientes expresiones, analice y responda si la expresión es verdadera (V) o falsa (F), si es necesario haga uso de la tabla de verdad.

Expresiones lógicas

1. V **Y** F **Y** V →
2. (V **O** V) **Y** (F **Y** F) →
3. **NO** (F **Y** V) →
4. **NO** (**NO** (V)) →
5. F **O** F **O** F **O** F **O** F **O** V →
6. V **Y** V **Y** V **Y** V **Y** F →
7. (F **Y** V) **Y** (V **O** F) →
8. NO (V **Y** V) →

Expresiones aritméticas, lógicas y relacionales.

1. $4 > 2 + 2$ →
2. $4 \neq (2 + 2)$ →
3. **NO** ($10 \neq 10$) →
4. $3 \leq (3*1)$ →
5. $3 < (3*1)$ →
6. $3 > 2$ **Y** $1 > 1$ →
7. $3 > 2$ **O** $1 > 1$ →
8. $3 \leq 3$ **Y** $2 = 2$ →
9. $3 \leq 3$ **O** $2 = 2$ →
10. $3 \leq 3$ **Y** $2 \neq 2$ →
11. $3 \leq 3$ **Y** $2 \neq 2$ **Y** $3 = 4$ →
12. $(3 \leq 3$ **Y** $2 \neq 2)$ **O** $3 = 3$ →
13. $(3 \leq 3$ **Y** $2 \neq 2)$ **O** $3 = 4$ →
14. $200 \bmod 2 = 0$ →
15. $203 \bmod 3 = 0$ →
16. $8/ 2 = 4$ **Y** $5 - 2 = 3$ →
17. $2*2+1 \neq 0*5+5$ →
18. $21 \bmod 2 \neq 0$ →
19. $21 \bmod 2 = 1$ →
20. $20 \bmod 2 = 0$ **Y** $13 \bmod 3 \neq 0$ →
21. $3*2 \bmod 3 = 0$ **Y** $5*5 \bmod 2 = 1$ →

III. Algoritmos, Diagramas de Flujos y Pseudocódigo

Desarrolle el análisis y diseño de cada uno de los siguientes problemas.

1. "***Conversión de kilómetros a metros y a centímetros***"
   Diseñe un programa que dada una cantidad en kilómetros, sea convertida en metros y en centímetros. Desarrolle algoritmo, diagrama de flujo y pseudocódigo. Ejemplo: 1.250 Km. equivale a 1,250 metros y a 125000 centímetros.

2. "***Mayor de dos números***"
Diseñe un programa que dado dos números enteros se determine el mayor de los dos. Desarrolle algoritmo, diagrama de flujo y pseudocódigo.

3. "***Convertir horas minutos y segundos a sólo segundos***"
Diseñe un programa que efectué la conversión de horas con minutos a segundos. Desarrolle pseudocódigo. Ejemplo: 2 horas y 35 minutos y 5 segundos son 9305 segundo.

4. "***Convertir segundos a horas, minutos y segundos***"
Diseñe un programa que efectué la conversión de segundos a horas y minutos. Desarrolle pseudocódigo. Ejemplo: 10000 segundos equivale a 2 horas y 46 minutos (y 40 segundos).

5. "***Mayor de tres números***"
Diseñe un programa que dado tres números enteros se determine el mayor de los tres. Desarrolle diagrama de flujo y pseudocódigo.

Mientras

6. "***Suma de 10 valores***"
Diseñe un programa que lea 10 números enteros y que los vaya sumando, al finalizar despliegue la suma total de los números ingresados. Desarrolle diagrama de flujo y pseudocódigo. (¿Qué haría si en lugar de leer 10 números fuesen 100? ¿En qué partes del diseño –diagrama de flujo y pseudocódigo- modificaría?).

7. "***Solo Enteros positivos***"
Diseñe un programa que lea tantos números enteros como sea posible hasta que el usuario teclee un valor negativo o cero. Desarrolle diagrama de flujo y pseudocódigo. (¿Se requiere el uso de un contador? Es decir, ¿Se necesita un contador en el condicional?).

Hacer … Mientras

8. "***Imprimir Contador***"
Diseñe un programa que utilice un contador (inicializado en 0) para imprimir su contenido cada vez que el usuario teclea la letra S. Desarrolle Diagrama de Flujo y Pseudocódigo. Ejemplo:

```
0
Desea continuar (S/N): S
1
Desea continuar (S/N): S
2
Desea continuar (S/N): S
3
Desea continuar (S/N): N
Hasta pronto!!!
```

Sugerencia: El contador no es considerado dentro de la condición, sin embargo la letra que tecleo el usuario 'S' o 'N', es importante para condición.

9. "***Suma de 10 valores Hacer Mientras***"
Diseñe un programa similar al ejercicio 6, considerando el uso de la sentencia ***Hacer... Mientras***. Desarrolle diagrama de flujo y pseudocódigo. (¿Qué modificaciones se tendría que efectuar?. En este ejercicio, ¿Cuál estructura de control iterativa (condicional) es adecuada utilizar: ***Mientras*** o ***Hacer... Mientras***?).

10. "***Solo Enteros positivos***"
Diseñe un programa similar al ejercicio 7, considerando el uso de la sentencia ***Hacer... Mientras***. Desarrolle diagrama de flujo y pseudocódigo. (¿Qué modificaciones se tendría que efectuar?. En este ejercicio, ¿Cuál estructura de control iterativa (condicional) es adecuada utilizar: ***Mientras*** o ***Hacer... Mientras***?).