

Este documento forma parte de una unidad didáctica que está en desarrollo en la Universidad Estatal a Distancia de Costa Rica.
Su contenido se encuentra bajo la ley de Propiedad Intelectual y cualquier mención a este debe ser indicado como obra en proceso.

Unidad didáctica 3071

Contenidos

.....	8
Capítulo III. Resolución de algoritmos con diagramas de flujo y pseudocódigo ...	9
Introducción	9
Algoritmos	12
Definición y características	12
Etapas de un algoritmo	13
Partes de un algoritmo	14
Instrucciones básicas	17
Ejemplos de algoritmos	19
Ejemplo 1	19
Ejemplo 2	19
Ejemplo 3	20
Ejemplo 4	20
Representaciones de algoritmos	21
Diagramas de flujo	21
Simbología de los diagramas de flujo (programa DFD)	24
Pseudocódigos	25
Estructura	27
Instrucciones básicas y cierre de instrucciones	28
Variables	36
Nombre	36
Tipo de dato	37
Numéricos: <i>enteros</i>	37
Numéricos: reales	38
Caracter y cadena	38
Booleanos (Lógicos)	39
Declaración de variables	39
Ejemplos de variables	42
Ejercicio 1	42
Ejercicio 2	44
Ejercicio 3	45
Constantes	46
Estructuras de decisión	49
Estructura Si (If)	49
Características	49
Representación	50

Condiciones	50
Ejemplos	53
Estructura Si-Sino (If-Else)	71
Estructura Si anidados	76
Estructura Según (Switch - case)	84
Estructuras de repetición (Ciclos, Bucles o Loops)	91
Estructura Mientras (While)	91
<i>Contadores</i>	93
<i>Acumuladores</i>	97
Estructura Repetir– Hasta (Do – While)	107
<i>Azar (x)</i>	113
<i>Random (x)</i>	113
<i>Sin Saltar</i>	113
Estructura Para (For)	119
Resumen	131
Ejercicios de autoevaluación	134
1.8 Respuestas a los ejercicios de autoevaluación	136
Glosario de términos	154

Gráficos

Gráfico 1 Estructura de un algoritmo	12
Gráfico 2 Etapas de un algoritmo	13
Gráfico 3 Algoritmo donde se suman dos valores específicos	14
Gráfico 4 Algoritmo donde se suman dos valores desconocidos	14
Gráfico 5 Partes de un algoritmo	15
Gráfico 6 Algoritmo donde se suman dos valores desconocidos	18
Gráfico 7 Comparativa de Algoritmo y Diagrama de flujo para sumar dos números	22
Gráfico 8 Comparativa de Algoritmo y Diagrama de flujo para convertir colones a dólares ...	23
Gráfico 9 Comparativa de Algoritmo y Diagrama de flujo para enviar un saludo	24

Índice de ejemplos

Ejemplo 1 Algoritmo con mensaje de sed	53
Ejemplo 2 Algoritmo para dividir dos números	55
Ejemplo 3 Mayor de tres números	58

Ejemplo 4 Calculadora con las cuatro operaciones básicas.....	62
Ejemplo 5 Calcular el salario de un trabajador.....	67
Ejemplo 6 Determinar el mayor y menor de cinco números	70
Ejemplo 7 Recorrido en bus o caminando.	72
Ejemplo 8 División y determinar si el cociente es múltiplo de tres y de cinco	74
Ejemplo 9 Menor de tres números.	77
Ejemplo 10 Venta de entradas al cine.....	77
Ejemplo 11 Cálculos de perímetros y áreas de figuras planas	80
Ejemplo 12 Cálculo del índice de masa corporal (IMC)	81
Ejemplo 13 Día laboral y fin de semana	85
Ejemplo 14 Nombre de colores y características de estos	87
Ejemplo 15 Venta de entradas al cine con Switch	88
Ejemplo 16 Saludar mientras el usuario digite 'S'	92
Ejemplo 17 Contar la cantidad de veces que se digitó la letra "a"	94
Ejemplo 18 Factorial de un número.....	98
Ejemplo 19 Suma de recibos telefónicos y promedio de pago	101
Ejemplo 20 Elevar un número a un exponente con el ciclo while	104
Ejemplo 21 Calcular el perímetro y área de un rectángulo	108
Ejemplo 22 Determinar mayúsculas y minúsculas de las vocales.....	110
Ejemplo 23 Adivinar un número aleatorio	112
Ejemplo 24 Adivinar un número aleatorio con intentos restringidos	116
Ejemplo 25 Tabla de multiplicar del 1.....	120
Ejemplo 26 Tabla de multiplicar del 5.....	122
Ejemplo 27 Mostrar los números pares de 0 a 100.....	125
Ejemplo 28 Sumar los números pares de un rango dado por el usuario	126
Ejemplo 29 Determinar si un número es primo	130

Índice de figuras

Figura 1 Ejemplo de una división entre dos números empleando el pseudocódigo de PSeInt	26
Figura 2 Pseudocódigo que imprime en pantalla el mensaje "Hola"	28

Figura 3 Diagrama de flujo de datos que imprime en pantalla el mensaje “Hola”	28
Figura 4 Pseudocódigo que muestra el resultado de una suma, con un mensaje textual que acompaña el resultado	29
Figura 5 Diagrama de flujo de datos que muestra el resultado de una suma donde se muestra directamente el resultado (sin mensaje textual)	30
Figura 6 Pseudocódigo que muestra únicamente el resultado de la suma	30
Figura 7 Pseudocódigo que muestra los valores sumados y el resultado de la suma	31
Figura 8 Diagrama de flujo de datos que muestra los valores sumados y el resultado de la suma	31
Figura 9 Pseudocódigo que solicita un nombre y emite el mensaje de “Hola” al nombre digitado por teclado	32
Figura 10 Diagrama de flujo que solicita un nombre y emite el mensaje de “Hola” al nombre digitado por teclado	33
Figura 11 Pseudocódigo que envía el mensaje “Hola mundo”	34
Figura 12 Pseudocódigo con condición “Si” para beber agua	53
Figura 13 Diagrama de flujo con condición “Si” para beber agua	54
Figura 14 Pseudocódigo con división de números y verificación de múltiplos de 3 y 5	56
Figura 15 Diagrama de flujo con división y verificación de múltiplos de 3 y 5	57
Figura 16 Pseudocódigo para determinar el mayor de tres números	59
Figura 17 Diagrama de flujo para determinar el mayor de tres números.....	60
Figura 18 Pseudocódigo para la calculadora con cuatro operaciones básicas	64
Figura 19 Diagrama de flujo para una calculadora	66
Figura 20 Pseudocódigo para calcular el salario de un trabajador	69
Figura 21 Pseudocódigo para determinar el mayor y menor de cinco números	70
Figura 22 Diagrama de flujo para un recorrido en bus o caminando	73
Figura 23 Diagrama de flujo para la división de números con Si-Sino	76
Figura 24 Pseudocódigo para determinar el menor de tres números con “Si” anidados.....	77
Figura 25 Pseudocódigo para vender entradas al cine con “Si” anidados.....	78
Figura 26 Pseudocódigo para calcular áreas y perímetros con “Si” anidados.....	80
Figura 27 Diagrama de flujo para determinar el índice de masa corporal	83
Figura 28 Pseudocódigo para determinar el día laboral o fin de semana con Switch	85

Figura 29 Pseudocódigo para determinar colores y características de estos con Switch	87
Figura 30 Pseudocódigo para ventas de entradas de cine con Switch	88
Figura 31 Pseudocódigo para saludar “n” veces.....	92
Figura 32 Pseudocódigo para contar las letras “a” o “A” digitadas	94
Figura 33 Diagrama de flujo para contar las letras “a” o “A” digitadas.....	96
Figura 34 Pseudocódigo para calcular el factorial de un número	98
Figura 35 Diagrama de flujo de datos para calcular el factorial de un número.....	100
Figura 36 Pseudocódigo que suma los pagos de un año de recibos y calcula el promedio mensual	101
Figura 37 Diagrama de flujo que suma los pagos de un año de recibos y calcula el promedio mensual	104
Figura 38 Pseudocódigo para elevar una base a un exponente.....	104
Figura 39 Diagrama de flujo de datos para elevar una base a un exponente	106
Figura 40 Pseudocódigo para calcular el perímetro y área de un rectángulo, realizando validaciones con ciclos	108
Figura 41 Pseudocódigo para contar la cantidad de vocales minúsculas y mayúsculas que se digitaron, por medio del uso de ciclos	110
Figura 42 Pseudocódigo para adivinar un número aleatorio	112
Figura 43 Pantalla de ejecución del programa con la instrucción Sin Saltar	114
Figura 44 Pantalla de ejecución del programa eliminando la instrucción Sin Saltar	115
Figura 45 Pseudocódigo para adivinar un número aleatorio con intentos restringidos	116
Figura 46 Pseudocódigo para mostrar la tabla de multiplicar del 1	120
Figura 47 Salida a pantalla de la tabla de multiplicar del 1	121
Figura 48 Diagrama de flujo de la tabla de multiplicar del 1	121
Figura 49 Pseudocódigo para mostrar la tabla de multiplicar del 1	122
Figura 50 Salida a pantalla de la tabla de multiplicar del 5	123
Figura 51 Diagrama de flujo de la tabla de multiplicar del 5	124
Figura 52 Pseudocódigo para los números pares del 0 al 100 usando el ciclo for.....	125
Figura 53 Salida a pantalla con los números pares del 0 al 100	126
Figura 54 Diagrama de flujo de los números pares del 0 al 100.....	126
Figura 55 Pseudocódigo para sumar los números pares de un rango	127

Figura 55 Diagrama de flujo para sumar los números pares de un rango.....	129
Figura 57 Pseudocódigo para determinar si un número es primo por medio de ciclos	130
Figura 57 Pseudocódigo para cambiar de colones a dólares	137
Figura 58 Pseudocódigo para saludar.....	137
Figura 59 Pseudocódigo y diagrama de flujo para calcular el producto de un número	138
Figura 60 Pseudocódigo para el día de la semana	139
Figura 61 Diagrama de flujo para el día de la semana	140
Figura 62 Diagrama de flujo para el cálculo de salario	142
Figura 63 Pseudocódigo para viajar en bus o caminando	144
Figura 64 Pseudocódigo para calcular el promedio de tres números	145
Figura 65 Diagrama de flujo para calcular el promedio de tres números	145
Figura 66 Diagrama de flujo de datos para determinar el mayor de dos números	146
Figura 67 Pseudocódigo para determinar el mayor de dos números	146
Figura 68 Pseudocódigo para calcular la edad de una persona	147
Figura 69 Pseudocódigo para determinar el día de la semana	148
Figura 70 Pseudocódigo para determinar el día de la semana, validando el día con ciclos	149
Figura 71 Pseudocódigo mostrar con el ciclo “Mientras” los números pares del 0 al 200....	150
Figura 72 Diagrama de flujo mostrar con el ciclo “Mientras” los números pares del 0 al 200	150
Figura 73 Pseudocódigo para determinar un número primo, con el uso de ciclos	151
Figura 74 Pseudocódigo con menú para: suma, tabla de multiplicar, división, exponente y áreas, con ciclos y validaciones	153



ALGORITMOS

Capítulo III. Resolución de algoritmos con diagramas de flujo y pseudocódigo



Objetivo general

Elaborar la secuencia de procesos y operaciones ordenados lógicamente para el análisis, solución y modelado de un problema de programación, a partir de las relaciones, restricciones y sintaxis, por medio del aprendizaje visual y pseudocódigo.



Objetivos específicos

Conocer los diagramas de flujo y el pseudocódigo.

Reconocer los tipos de datos, variables, constantes, estructuras de selección y estructuras de repetición.

Implementar el funcionamiento de las estructuras de control en la creación de algoritmos.

Introducción

En este capítulo se explica el concepto de algoritmo y los métodos para resolverlo. Se inicia con el lenguaje natural en la serie de pasos ordenados para la resolución de un problema y se culmina con representaciones gráficas computacionales como los diagramas de flujo e instrucciones más abstractas como el pseudocódigo, para probar la eficacia de la solución propuesta.

Para iniciar, se parte del reconocimiento de los tipos de datos que establece el contexto de un problema, por ejemplo, si se desea contar la cantidad de camisas o pares de zapatos que tenemos, utilizaríamos cantidades enteras puesto que no es funcional el tener media camisa o medio zapato, puesto que se tiene la camisa entera o los dos zapatos, pero no parte de estos. Por otra parte, si se desean contar los kilogramos de carne o verduras que se compran en la feria del agricultor, necesariamente se considerarán los decimales en esas cantidades porque es difícil que los pesos de esos alimentos sean enteros.

Ahora bien, para estructurar una solución a los problemas planteados se emplean estructuras de control (de selección y de repetición) que se utilizan en el lenguaje cotidiano casi de manera desapercibida y que son necesarias para

discriminar las decisiones que los humanos planteamos. Por ejemplo, ¿cuántos segundos tiene un minuto? Sabemos que 60 segundos completan un minuto y que eso nunca cambiará, para ello se realiza un conteo de uno a sesenta segundos para incrementar al siguiente minuto; este ejemplo tan básico se realiza con una estructura de repetición porque se conoce la cantidad de segundos de un minuto, el siguiente minuto no tendrá ni más ni menos segundos y solo debemos repetir un comportamiento de suma de segundos para incrementar los minutos en un reloj. Referente a las estructuras de decisión imaginemos que hay una condición previa que se debe cumplir, por ejemplo ¿compro naranjas o fresas? Bueno, esto dependerá si tenemos más o menos cantidad de alguna de estas frutas en nuestra casa. Si tenemos suficientes naranjas, pero no tenemos fresas, entonces se comprarán fresas. Este ejemplo que parece bastante simple es importante para emplear una estructura de decisión y sus condiciones previas para determinar qué fruta comprar.

Finalmente, cuando se han desarrollado los pasos y estructuras necesarias para la resolución de un problema, el algoritmo puede tener una representación visual por medio de diagramas o por medio de instrucciones a través de pseudocódigo. Para ayudar a la comprensión de estos temas, se empleará la analogía de una fábrica y sus procesos en los casos de estudio que se plantean.



Guía de lectura

En este tercer capítulo se pretende contextualizar al lector acerca de los primeros conceptos de la resolución de algoritmos, sus partes y especificaciones, sin los cuales es imposible resolver problemas de diversa índole.

¿Qué podríamos considerar un problema? Cualquier cosa que dadas unas entradas y después de recibir un proceso en varios pasos, obtenemos una salida con información que es útil.

El capítulo inicia con la definición y etapas de un algoritmo y diversos ejemplos, posteriormente se explica la representación de estos, posteriormente se explican las variables, constantes, tipos de datos y finalmente se explican las estructuras de control y de iteración que son requeridas para la resolución de un algoritmo.

Al cierre del capítulo podrá observar un resumen, además contará con una sección de ejercicios de autoevaluación para que auto regule los aprendizajes de este capítulo y analice la comprensión de los temas tratados.

Algoritmos

Definición y características

Un algoritmo se define como una serie de pasos para resolver una situación específica. Los algoritmos presentan las siguientes características:

- ✓ Son finitos, es decir, poseen inicio y fin.
- ✓ Tienen un orden lógico y secuencial, es decir, sus instrucciones deben estar ordenadas de forma que el problema se solucione de forma lógica y cada instrucción se ejecuta una después de otra.

Hay que tener claro que el algoritmo se enfoca en especificar los pasos a seguir por el sistema informático y no se centra en lo que el usuario debe realizar.

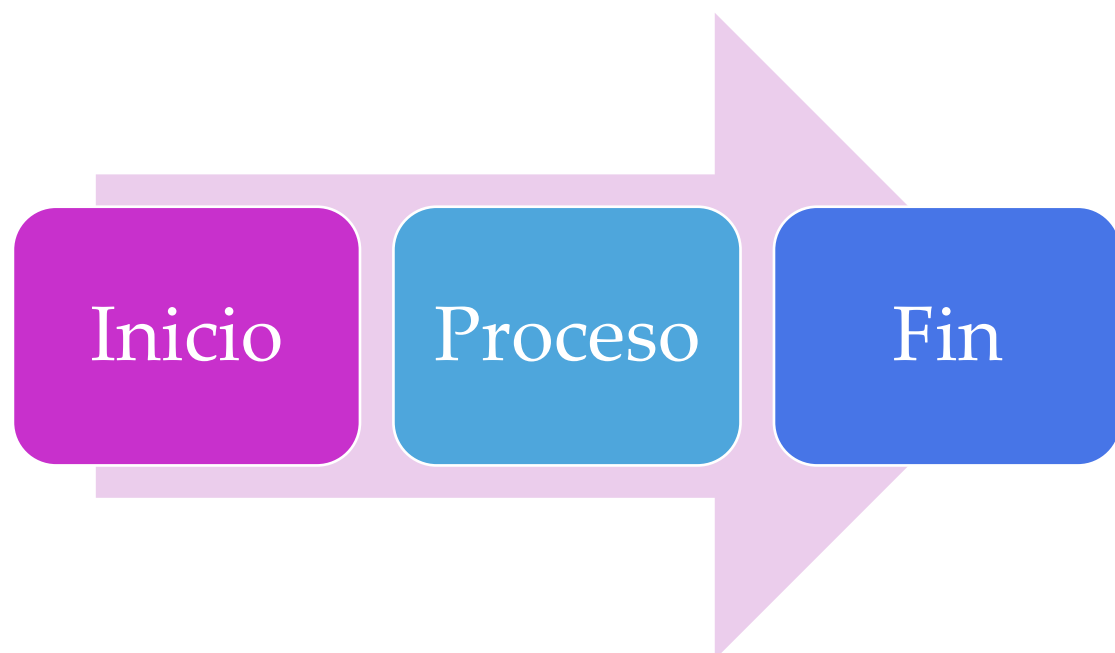


Gráfico 1 Estructura de un algoritmo

Etapas de un algoritmo

Los algoritmos poseen tres etapas en su ejecución: lectura de datos, procesamiento o cálculos y muestra de resultados.



Gráfico 2 Etapas de un algoritmo

Las **entradas** son aquellos valores que tanto el usuario como el contexto del problema pueden aportar. Para ello se emplean las **constantes** y **variables** que se analizarán más adelante en este capítulo.

Los **procesos** son los cálculos, decisiones y lógica de control que debe tener un problema para su resolución, esto se analizará a partir de este capítulo y en los subsiguientes para considerar las estructuras que se deben incorporar, según el contexto del problema a resolver.

Finalmente, las **salidas** son los mensajes del sistema después de procesar la información, por ejemplo: el monto a pagar en salario a un trabajador, el monto total de las compras en un supermercado, el promedio del pago de recibos de agua luz y teléfono de un año específico, entre otros.

En las **entradas** se solicitan los datos al usuario (el usuario puede ser una persona que opera el sistema, una base de datos, un archivo, una página web, entre otros), en nuestro caso tomaremos al usuario como una persona que utiliza el sistema informático, una vez que se solicitan los datos al usuario, se leen para que el algoritmo los tome en consideración antes de pasar a la siguiente etapa.

Al finalizar la lectura de datos, se realiza el procesamiento o cálculo de estos, es en esta etapa donde se hacen diversas operaciones para obtener los resultados que se mostrarán al usuario.

En las **salidas** se muestran los resultados de los cálculos hechos durante el procesamiento de datos y se muestran al usuario de forma ordenada, clara y concisa.

Puede existir el caso de un algoritmo que no cumpla con la etapa de lectura, pero tendría limitantes bastante importantes en cuanto a su rango de resultados. Un algoritmo que no lea los datos de entrada siempre mostrará los mismos resultados, ya que los cálculos no tendrán ningún cambio. Por ejemplo, si tenemos un algoritmo que sume los números 15 y 25 siempre mostrará como resultado 40, pero si hacemos que el algoritmo sume dos números dados por el usuario el resultado variará en función de los números que se ingresen. Observemos en el Gráfico 3 la primera aproximación del algoritmo anterior donde se suman los números 15 y 25, según las etapas descritas.

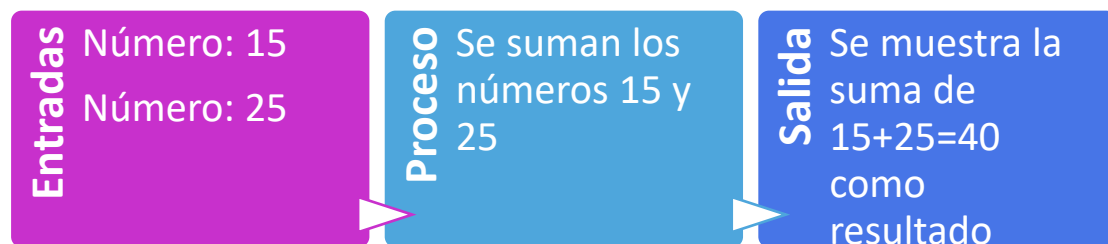


Gráfico 3 Algoritmo donde se suman dos valores específicos

Ahora, observemos en el Gráfico 4 la modificación del mismo algoritmo para sumar dos números dados por el usuario donde desconocemos los valores de ambos sumandos. Para ello utilizaremos la ayuda matemática de las **variables**, donde a “a” y “b” guardarán los números escritos por el usuario y mostrarán la suma de estos. En este caso el resultado varía en función de los números de entrada, por ende, nuestro algoritmo se torna más versátil y no como el anterior que es totalmente estático.

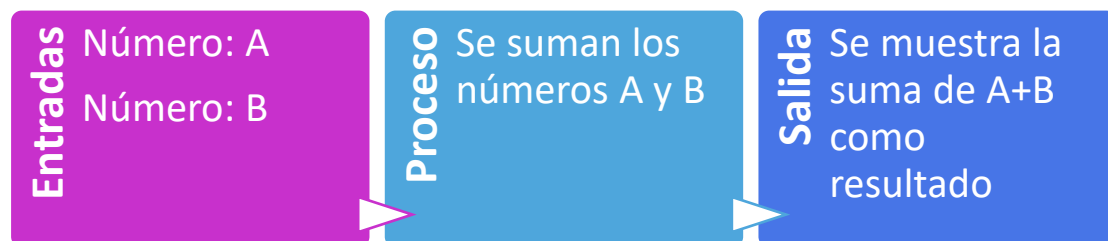


Gráfico 4 Algoritmo donde se suman dos valores desconocidos

Partes de un algoritmo

Los algoritmos poseen las siguientes partes: Inicio, Instrucciones y Fin. Su estructura sería igual a la que se presenta en el Gráfico 5.

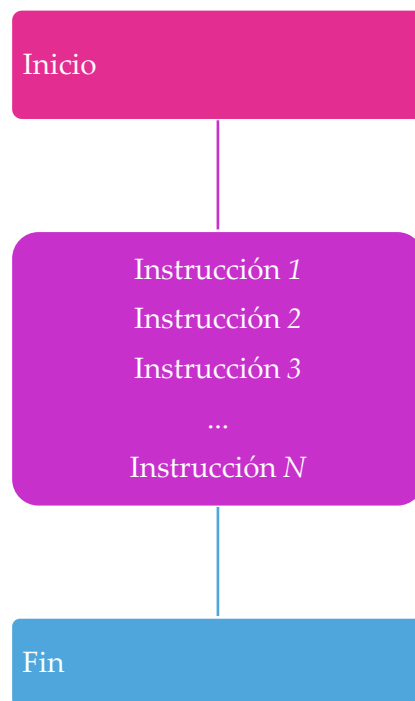


Gráfico 5 Partes de un algoritmo

Como se puede notar el cuerpo de instrucciones se ubica unos espacios más a la derecha de lo que están el Inicio y el Fin, a esto se le denomina **indentación**, más adelante veremos con más detalle este concepto.

En un algoritmo se puede representar cualquier propuesta de solución a diversas situaciones, sin embargo, existen situaciones de vida que por sus características se tornan sumamente complejas de resolver en un algoritmo, es más dado que se empleen algoritmos en situaciones que puedan ser representadas por medio de modelos.

A pesar de lo anterior, implementaremos nuestro primer algoritmo en una situación cotidiana de vida: bañarse. Sabemos que hay muchas formas de bañarse, en cuanto a la variedad de productos a utilizar, si hay que lavarse o no el cabello, si se emplea acondicionador o no, si hay un jabón para la cara y otro para el resto del cuerpo, entre otras características, lo que haremos será simplificar el proceso en cuanto al detalle de sus instrucciones. Una primera propuesta del algoritmo es la siguiente:

Inicio

Quitarse la ropa.

Abrir la cortina o puerta del baño.

Entrar a la ducha.
Cerrar la cortina o puerta del baño.
Abrir la llave del agua.
Mojar todo el cuerpo.
Cerrar la llave del agua.
Enjabonar todo el cuerpo.
Restregar todo el cuerpo.
Abrir la llave del agua.
Enjuagar todo el cuerpo.
Cerrar la llave del agua.

Fin

Importante

En el ejemplo anterior se observa que primero hay que quitarse la ropa para luego bañarse. Este detalle que nos parece tan evidente, muchas veces lo omitimos cuando resolvemos un problema y anteponeamos instrucciones que van antes o después de otras. En los algoritmos, el orden de las instrucciones sí afecta el producto que se obtiene.

Como se mencionó anteriormente, el algoritmo se puede hacer con instrucciones más específicas, como por ejemplo agregar pasos para lavar el cabello con champú o incluir los pasos para secarse y salir de la ducha, pero eso dependerá del grado de detalle que deseemos que tenga nuestra solución. En otros casos ese detalle lo brinda la misma descripción del problema a solucionar.

Sin importar la cantidad de pasos, todos los algoritmos deben ejecutar una instrucción tras otra en orden lógico, esto quiere decir que no deberíamos ejecutar la instrucción “Mojar todo el cuerpo.” antes de “Abrir la llave del agua.” ya que esto sería ilógico.

Se puede dar el caso que el orden de una instrucción no altere el resultado de la solución, por ejemplo, si agregamos la instrucción “Lavar el cabello” la podríamos colocar antes de “Enjabonar todo el cuerpo” o luego de “Restregar todo el cuerpo” y la lógica del algoritmo no cambiaría.

Es importante mencionar que las instrucciones se colocan empleando verbos en infinitivo, es decir, terminados en “ar”, “er”, “ir”. También se debe colocar una única instrucción por línea o renglón, por ejemplo, la instrucción “Abrir la llave del agua y

mojar todo el cuerpo” sería incorrecta, tiene sentido lógico, pero no cumple con lo estipulado para la construcción correcta de un algoritmo.

Instrucciones básicas

Los algoritmos poseen tres instrucciones básicas relacionadas con las etapas descritas anteriormente, estas instrucciones son las siguientes:

- ✓ **Solicitar:** solicitud de datos al usuario (Etapa: Entrada).
- ✓ **Calcular:** cálculo de operaciones (Etapa: Procesamiento).
- ✓ **Mostrar:** muestra se resultados o información al usuario (Etapa: Salida).

Analicemos el siguiente algoritmo que suma dos números suministrados por el usuario.

Inicio

Solicitar el primer número al usuario.

Solicitar el segundo número al usuario.

Calcular la suma del primer número más el segundo número.

Mostrar el resultado de la suma.

Fin

En el Gráfico 6 se observan las partes, etapas e instrucciones básicas del algoritmo.

Inicio

Instrucciones

- **Entrada**

- **Solicitar** el primer número.
- **Solicitar** el segundo número.

- **Proceso**

- **Calcular** la suma del primer número más el segundo número.

- **Salida**

- **Mostrar** el resultado de la suma.

Fin

Gráfico 6 Algoritmo donde se suman dos valores desconocidos

Lo primero que se hace, es solicitar los números al usuario ya que no los podemos sumar sin saber cuáles son los valores, una vez que se solicitaron se suman y finalmente se muestra el resultado de la suma.

Es importante recalcar que todo lo que desee mostrar se debe solicitar o calcular, es decir, no se puede mostrar un dato que no ha sido solicitado al usuario y tampoco se debería mostrar un resultado que no se ha calculado primero. Esto es parte del orden lógico que debe tener un algoritmo y quien lo construya debe analizar e implementar.

Ejemplos de algoritmos

A continuación, se desarrollan algunos ejemplos de algoritmos.

Ejemplo 1

Descripción: elabore un algoritmo que calcule el promedio de tres números.

Inicio

Solicitar el primer número al usuario.

Solicitar el segundo número al usuario.

Solicitar el tercer número al usuario.

Calcular la suma del primer número más el segundo número más el tercer número.

Calcular la división de la suma de los tres números entre la cantidad de números solicitados.

Mostrar el resultado de la división.

Fin

Ejemplo 2

Descripción: elabore un algoritmo que calcule el cuadrado de un número

Inicio

Solicitar un número al usuario.

Calcular la multiplicación del número digitado por el mismo número digitado.

Mostrar el resultado de la multiplicación.

Fin

Ejemplo 3

Descripción: elabore un algoritmo que muestre el saludo “Hola” junto al nombre que digite el usuario.

Inicio

Solicitar el nombre al usuario.

Mostrar Hola y el nombre digitado por el usuario.

Fin

Ejemplo 4

Descripción: elabore un algoritmo que calcule el promedio de un estudiante. Tome en cuenta la solicitud de los siguientes datos:

- ✓ Nombre del estudiante.
- ✓ Sección del estudiante.
- ✓ Número de identificación del estudiante.
- ✓ Nota de la primera tarea. (La nota se otorga de 1 a 10)
- ✓ Nota de la segunda tarea. (La nota se otorga de 1 a 10)
- ✓ Nota de la primera prueba escrita. (La nota se otorga de 1 a 10)
- ✓ Nota de la segunda prueba escrita. (La nota se otorga de 1 a 10)

El cuadro de evaluación de las tareas y pruebas escritas es el siguiente:

- ✓ Primera tarea 1.5%.
- ✓ Segunda tarea 2.0%.
- ✓ Primera prueba escrita 3.0%.
- ✓ Segunda prueba escrita 3.5%.

El reporte del promedio debe tener los siguientes datos:

- ✓ Nombre del estudiante.
- ✓ Sección del estudiante.
- ✓ Número de identificación del estudiante.
- ✓ Promedio de 1 a 10.

Inicio

Solicitar el Nombre del estudiante al usuario.

Solicitar la Sección del estudiante al usuario.

Solicitar el Número de identificación del estudiante al usuario.

Solicitar la Nota de la primera tarea al usuario.

Solicitar la Nota de la segunda tarea al usuario.

Solicitar la Nota de la primera prueba escrita al usuario.

Solicitar la Nota de la segunda prueba escrita al usuario.

Calcular el porcentaje de la primera tarea multiplicando la nota de la primera tarea por 1.5 y dividiendo el resultado entre 10.

Calcular el porcentaje de la segunda tarea multiplicando la nota de la segunda tarea por 2.0 y dividiendo el resultado entre 10.

Calcular el porcentaje de la primera prueba escrita multiplicando la nota de la primera prueba escrita por 3.0 y dividiendo el resultado entre 10.

Calcular el porcentaje de la segunda prueba escrita multiplicando la nota de la segunda prueba escrita por 3.5 y dividiendo el resultado entre 10.

Calcular el promedio final sumando el porcentaje de la primera tarea más el porcentaje de la segunda tarea más el porcentaje de la primera prueba escrita más el porcentaje de la segunda prueba escrita.

Mostrar el Nombre del estudiante.

Mostrar la Sección del estudiante.

Mostrar el Número de identificación del estudiante.

Mostrar el Promedio final del estudiante.

Fin

Representaciones de algoritmos

Los algoritmos pueden representarse de dos formas, por medio de diagramas de flujo o mediante pseudocódigo.

Diagramas de flujo

Los diagramas de flujo son esquemas visuales formados por diversas figuras que representan acciones en el algoritmo, acá ya no se empleará el lenguaje natural, sino una representación específica para las entradas, procesos y salida.

Para ver el paralelismo entre el lenguaje natural, que es el que hemos desarrollado con los ejemplos anteriores, y la representación en diagrama de flujo, volveremos al algoritmo de sumar dos números. El Gráfico 7 muestra las similitudes y diferencias entre ambos.

Algoritmo en lenguaje natural

Diagrama de flujo

Inicio

1. Solicitar el primer número al usuario.
2. Solicitar el segundo número al usuario.
3. Calcular la suma del primer número más el segundo número.
4. Mostrar el resultado de la suma.

Fin

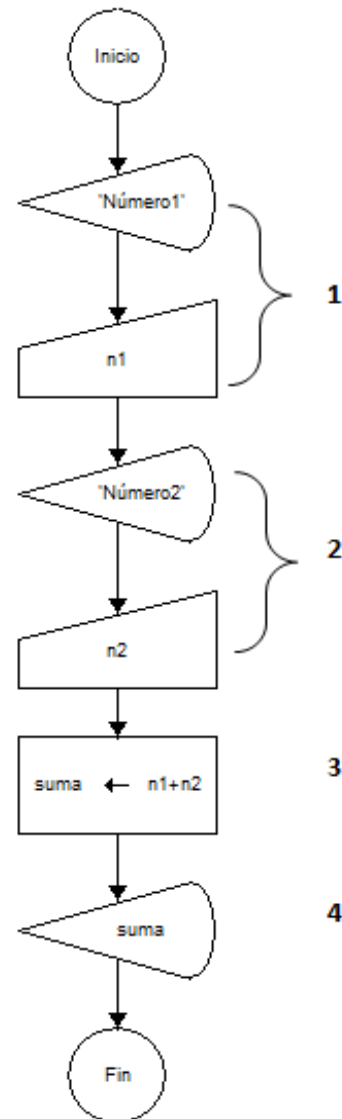


Gráfico 7 Comparativa de Algoritmo y Diagrama de flujo para sumar dos números

Observe que en los pasos 1 y 2 del algoritmo en lenguaje natural tenemos una sola instrucción, pero en el diagrama tenemos dos elementos para cada paso, esto se debe a que el primer elemento corresponde al mensaje en pantalla que recibe el usuario para que digite el número y el segundo elemento es para recibir por teclado, el valor que el usuario digite. En este sentido, el diagrama se vuelve más específico en los pasos.

El Gráfico 8 es un ejemplo de un diagrama de flujo que solicita y lee una cantidad X de colones y muestra el equivalente en dólares, considerando que el tipo de cambio es:

\$1 dólar = ₡500 colones.

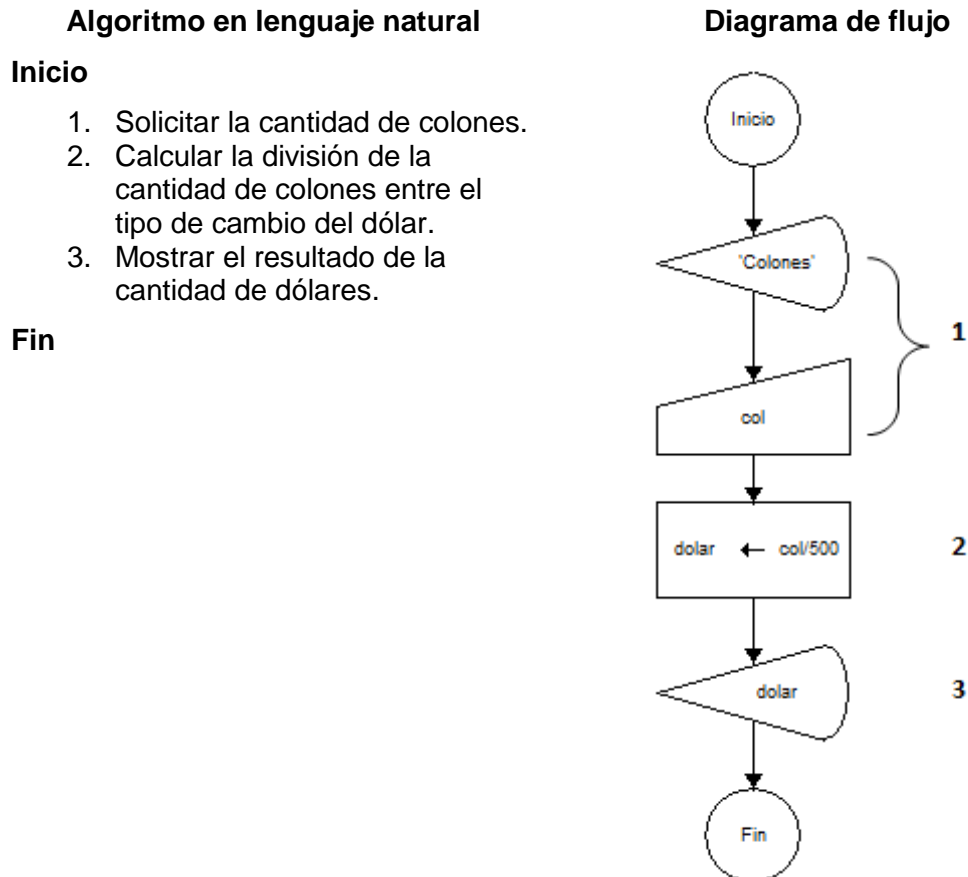


Gráfico 8 Comparativa de Algoritmo y Diagrama de flujo para convertir colones a dólares

Observe que en el paso 1 del algoritmo en lenguaje natural, tenemos una sola instrucción, pero en el diagrama tenemos dos elementos para este paso. Al igual que en el ejemplo anterior, esto se debe a que el primer elemento corresponde al mensaje en pantalla que recibe el usuario para que digite la cantidad de colones y el segundo elemento es para recibir por teclado la cantidad de colones que el usuario posee. Considere también que en el contexto del problema se está indicando que quinientos colones son equivalentes a un dólar, por lo que en el paso 2 del diagrama se realiza la división con ese monto, y por lo tanto este dato no se le solicitó al usuario, sin embargo, esto vuelve menos flexible el cálculo porque, aunque el tipo de cambio se modifica diariamente, acá se considera como un valor sin modificación.

El Gráfico 9 es un ejemplo de un diagrama de flujo que solicita el nombre de una persona y emite un mensaje de saludo: “Hola *nombre*” donde *nombre* corresponde al texto que el usuario haya digitado.

Algoritmo en lenguaje natural

Diagrama de flujo

Inicio

1. Solicitar el nombre al usuario.
2. Mostrar Hola y el nombre digitado por el usuario.

Fin

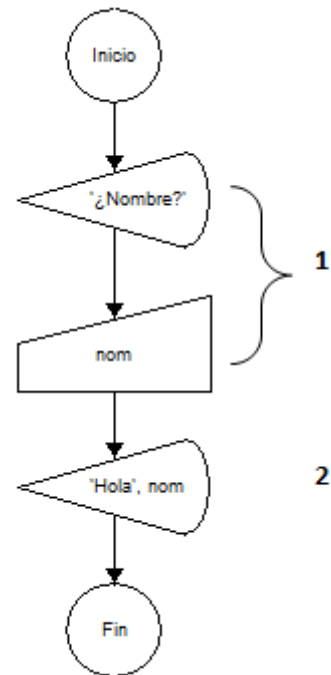


Gráfico 9 Comparativa de Algoritmo y Diagrama de flujo para enviar un saludo

Observe que en el paso 1 del algoritmo en lenguaje natural, tenemos una sola instrucción, pero en el diagrama tenemos dos elementos para este paso. Al igual que en el ejemplo anterior, esto se debe a que el primer elemento corresponde al mensaje en pantalla que recibe el usuario para que digite su nombre y el segundo elemento es para recibir por teclado el nombre de la persona, en este caso para que el diagrama presente mayor claridad, el nombre fue guardado en una variable llamada “nom”. Considere también que en el contexto del problema no se realiza ningún cálculo, solo se emite un mensaje con: “Hola usuario” donde el usuario se sustituye por el nombre: Ana, Juan, María, u otro que se haya digitado. Finalmente, observe que, al no realizarse cálculos en este algoritmo, no fue necesario emplear el rectángulo que representa a los cálculos que se podrían realizar si el problema lo solicitara. Antes de analizar otros diagramas, observe en el siguiente apartado la simbología y usos de cada uno de los símbolos.



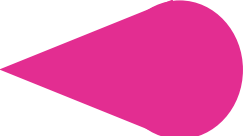




Simbología de los diagramas de flujo (programa DFD)

Antes de continuar con más ejemplos de diagramas de flujo, se debe considerar que para efectos de la asignatura se empleará la simbología del programa de Diagrama de Flujo de Datos (DFD).

TABLA 1

SIMBOLOGÍA DEL DIAGRAMA DE FLUJO DE DATOS

Símbolo	Nombre	Función
---------	--------	---------

	Inicio/Fin	Marca el punto de inicio y fin del diagrama. Dentro del símbolo debe llevar la palabra Inicio o Fin, según sea el caso.
	Asignación	Declara, inicializa y asigna valor a una variable o constante. Esto lo hace mediante asignación de valor o por alguna fórmula.
	Salida	Muestra en pantalla información al usuario. Esto puede ser: únicamente texto, texto con variables o solo variables.
	Lectura	Lee y almacena en una variable lo que el usuario digite, ya sean letras o números.
	Decisión (Si)	Analiza una condición y ejecuta instrucciones dependiendo si la condición es verdadera (True) o falsa (False).
	Ciclo, bucle	Ejecuta las instrucciones que tenga dentro de forma repetida dependiendo de una condición.
	Conector	Se emplea en diagramas hechos en editores de texto o de forma física en papel. Indica que el diagrama se corta en ese punto y continúa más adelante. Al final de la página, se coloca un símbolo con un número adentro. Y al inicio de la siguiente, se coloca el símbolo con el mismo número que el anterior. Si se emplea otro conector se debe utilizar un número diferente.

Pseudocódigos

Por su parte, los pseudocódigos muestran los algoritmos de forma textual, es decir, únicamente mediante palabras que tienen ciertas funcionalidades para el ejemplo que se está brindando.

```
1  Algoritmo sin_titulo
2      //Declaración de variables
3      Definir PrimerNumero Como Real;
4      Definir SegundoNumero Como Real;
5      Definir Resultado Como Real;
6
7      //Inicialización de variables
8      PrimerNumero=0;
9      SegundoNumero=0;
10     Resultado=0;
11
12     //Lectura de datos
13     Escribir "Digite el primer número";
14     Leer PrimerNumero;
15
16     //Cálculo del resultado
17     Mientras SegundoNumero=0 Hacer
18         Escribir "Digite el segundo número";
19         Leer SegundoNumero;
20         Si SegundoNumero=0 entonces
21             Escribir "Error. Debe digitar un número diferente a cero.";
22         FinSi
23     FinMientras
24
25     //Muestra de resultados
26     Resultado=PrimerNumero/SegundoNumero;
27     Escribir "Respuesta: ",Resultado;
28
29  FinAlgoritmo
```

Figura 1 Ejemplo de una división entre dos números empleando el pseudocódigo de PSeInt

Estructura

Todo pseudocódigo tiene una estructura básica, la cual se debe seguir para mantener el orden visual y la lógica de la solución, esta estructura es la siguiente:

Inicio Algoritmo_Nombre

Declaración de variables y constantes

Inicialización de variables y constantes

Bloque de Instrucciones

Fin

Donde:

- ✓ Inicio: es el punto de inicio del pseudocódigo.
- ✓ Declaración de variables y constantes: es donde se declaran todas las variables y constantes que se utilizarán en el pseudocódigo.
- ✓ Inicialización de variables y constantes: es donde se le otorga el primer valor a todas las variables y el valor definitivo a todas las constantes.
- ✓ Bloque de Instrucciones: es aquí donde se desarrollan todos los pasos que ejecutará el pseudocódigo para la resolución de una situación específica.
- ✓ Fin: es el punto de finalización del pseudocódigo.

Un pseudocódigo se puede comparar con una receta de cocina, por ejemplo, los ingredientes son las variables y las constantes, mientras que la preparación (en la receta) son las instrucciones en el pseudocódigo.

Los temas de variables y constantes, además de su declaración e inicialización se estudiarán más adelante.

Instrucciones básicas y cierre de instrucciones

Un pseudocódigo posee dos instrucciones básicas: Escribir y Leer.

Escribir

La instrucción Escribir sirve para mostrar en pantalla información al usuario, esto sirve para solicitar (no leer) datos o para mostrarlos. Esta instrucción puede mostrar solo texto, texto con variables o solo variables.

Para mostrar solo texto se emplea de la siguiente forma: **Escribir ""**; donde, se mostrará en pantalla lo que esté escrito entre las comillas dobles. Por ejemplo, **Escribir "Hola"**; mostraría en pantalla la palabra Hola. Esto se puede observar, empleando la sintaxis de PSeInt en la Figura 2.

```
1  Proceso Saludar
2      Escribir "Hola" //este algoritmo no tiene variables, solo un mensaje a pantalla
3  FinProceso
```

Figura 2 Pseudocódigo que imprime en pantalla el mensaje "Hola"

Esta misma representación, en el diagrama de flujo de datos se puede observar en la Figura 3.

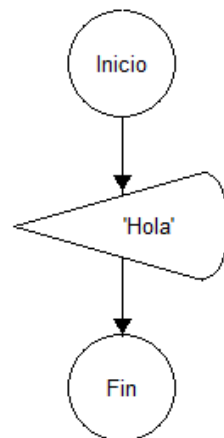


Figura 3 Diagrama de flujo de datos que imprime en pantalla el mensaje "Hola"

Si se desea mostrar texto con variables se debe emplear una coma (,) para separar el texto entre comillas de la variable, esto se haría de la siguiente forma:

Escribir “El resultado es: ”,resultado;

Un ejemplo de este desarrollo en PSeInt, se muestra en la Figura 4.

```
1  Proceso Sumar
2      Definir num, resultado como entero //declaración de variables
3      num=5 //asignación de un valor a la variable num
4      resultado=num+num //guarda en la variable resultado, la suma de 5+5
5      Escribir "El resultado es: ", resultado //imprime en pantalla el resultado
6  FinProceso
```

Figura 4 Pseudocódigo que muestra el resultado de una suma, con un mensaje textual que acompaña el resultado

Por otra parte, esta misma representación en diagrama de flujo de datos, se realizaría como se observa en la Figura 5. Observe que a la derecha del diagrama, se tiene la caja de asignación de valores. La primera gran diferencia entre el pseudocódigo y el diagrama, es que el pseudocódigo declara el tipo de variable y luego asigna un valor a la variable, pero en el diagrama a las variables no se les indica el tipo de dato que almacenan, porque el programa asume que si se asume un número la variable será de tipo entero y si se asignan letras será de tipo caracter. Además, en la caja de asignación nótese que se pueden dar tanto asignaciones de valores, como en las variables “num” y “resultado” y operaciones asignadas a una variable, como “resultado” que guarda la suma de num+num.

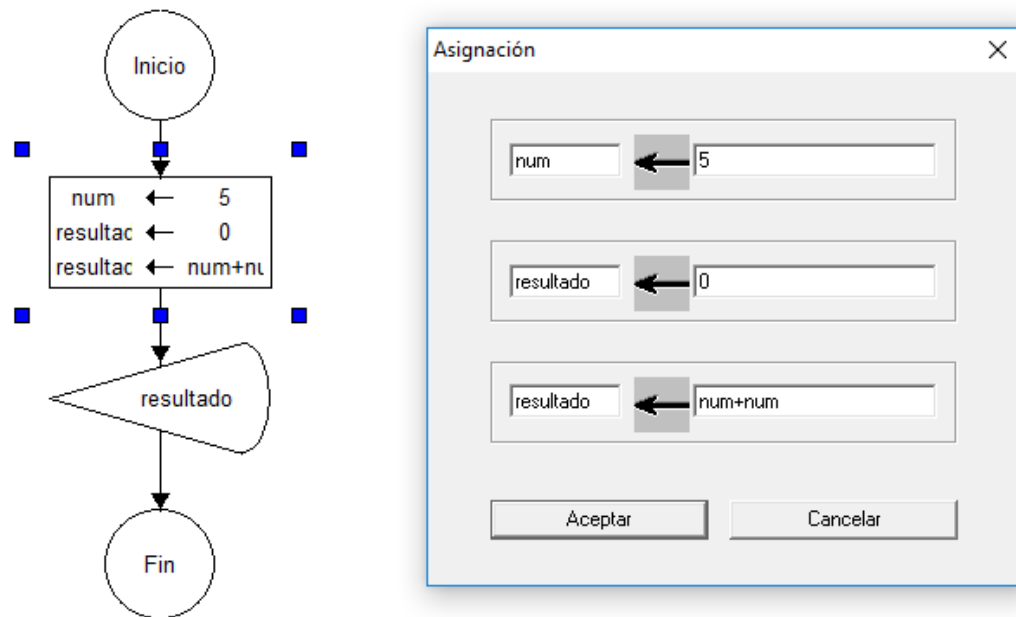


Figura 5 Diagrama de flujo de datos que muestra el resultado de una suma donde se muestra directamente el resultado (sin mensaje textual)

Para el caso del pseudocódigo, nótese que entre el cierre de las comillas dobles y la variable hay una coma, esto sirve para separar el texto de un valor almacenado en una variable, el resultado en pantalla sería: “El resultado es: 10”, esto porque se está sumando dos veces el número 5.

Si la variable “resultado” vale 10 y se desea mostrar en pantalla su valor se haría de esta forma: **Escribir resultado;** lo que se mostraría sería únicamente el número 10. Esto se puede observar en la Figura 6, modificando el ejemplo anterior.

```
1  Proceso Sumar
2      Definir num, resultado como entero //declaración de variables
3      num=5 //asignación de un valor a la variable num
4      resultado=num+num //guarda en la variable resultado, la suma de 5+5
5      Escribir resultado //imprime en pantalla el resultado
6  FinProceso
```

Figura 6 Pseudocódigo que muestra únicamente el resultado de la suma

Si se da el caso que combinar varios textos con variables se debe emplear una coma antes y después de la variable, por ejemplo: **Escribir “El resultado de: ”,num,” más”, num, “ es: “, resultado;** como se muestra en la Figura 7.

```
1  Proceso Sumar
2      Definir num, resultado como entero //declaración de variables
3      num=5 //asignación de un valor a la variable num
4      resultado=num+num //guarda en la variable resultado, la suma de 5+5
5      Escribir "El resultado de ", num, " más ", num, " es: ",resultado
6  FinProceso
```

Figura 7 Pseudocódigo que muestra los valores sumados y el resultado de la suma

El equivalente del pseudocódigo anterior en diagrama de flujo, se muestra en la Figura 8. Observe que, en el ejemplo, la caja de “salida por pantalla” muestra entre comilla simple el mensaje de texto, mientras que, entre comas, se coloca el nombre de las variables para que se muestre el valor de estas.

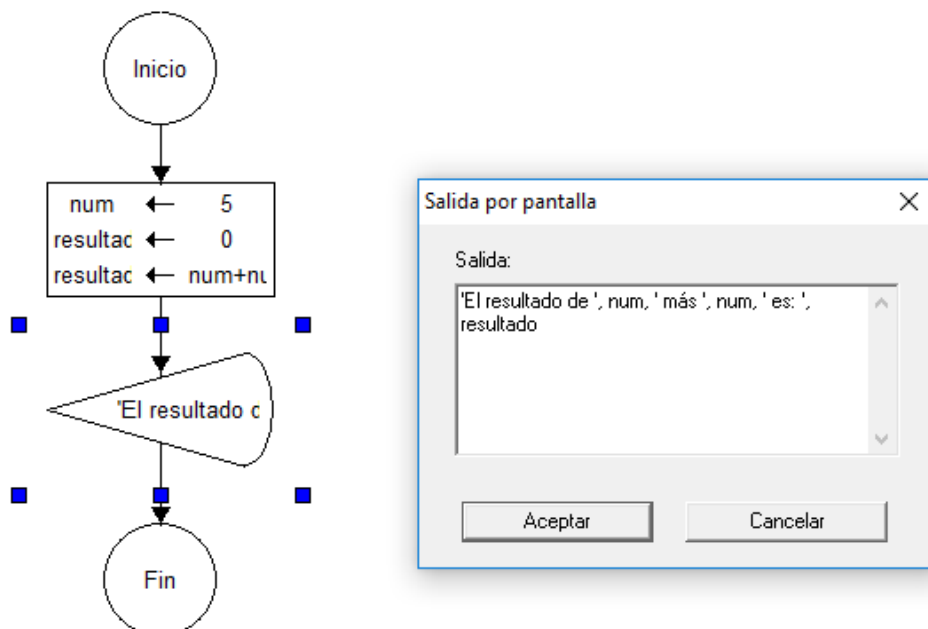


Figura 8 Diagrama de flujo de datos que muestra los valores sumados y el resultado de la suma

Leer

La instrucción Leer sirve para efectuar una lectura de lo que el usuario digite, al ejecutar esta instrucción el sistema lee lo digitado y lo almacena en una variable, por ende, esta instrucción siempre se acompaña de la variable donde el sistema almacenará lo leído. Por ejemplo, **Leer Nombre**; toma lo que el usuario digitó por medio del teclado y lo almacena en la variable **Nombre**, por ende, si el usuario digitó “Ana” el valor de la variable **Nombre** será “Ana”.

Nótese que la instrucción Leer asigna un valor a una variable, pero siempre tomando el dato que digitó el usuario. Si lo que deseamos es asignar un valor o el resultado de una operación a una variable utilizaremos el operador de asignación, ya sea el igual (=) o la flecha (←). Esto en el programa PSeInt, se observaría como la línea 3 de la Figura 9, donde se recibe desde teclado, el nombre que digite el usuario.

```
1  Proceso HolaPersona
2      Definir nombre Como Caracter //esto es una variable donde se almacenan letras
3      Leer nombre
4      Imprimir "Hola", nombre
5  FinProceso
```

Figura 9 Pseudocódigo que solicita un nombre y emite el mensaje de “Hola” al nombre digitado por teclado

En el caso de que este mismo ejercicio se desarrollara con un diagrama de flujo, la Figura 10 ejemplifica cómo se desarrollaría. Observe que, a diferencia de los casos analizados anteriormente, a donde a las variables se les asignaba un número, en el caso del texto a la variable se le coloca como valor inicial unas comillas dobles, esto le indica al programa que lo que se recibirá por teclado es un texto.

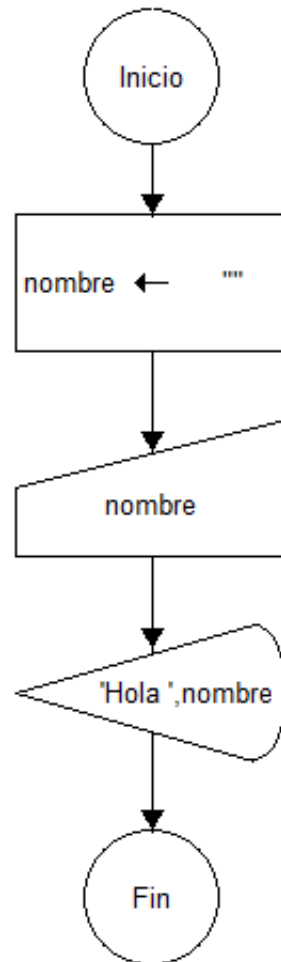


Figura 10 Diagrama de flujo que solicita un nombre y emite el mensaje de “Hola” al nombre digitado por teclado

Cierre de instrucciones

En el pseudocódigo (no así en el diagrama de flujo), la mayoría de las instrucciones tienen un punto y coma (;) al final de su estructura, este símbolo funciona como cierre de la instrucción. Las instrucciones que llevan el punto y coma de cierre son las siguientes: escribir, leer y asignaciones. Por ejemplo:

Escribir “Maleku”;

Leer Nombre;

Nota= 100; //esto es una asignación

Comentarios

Para documentar el pseudocódigo se hacen comentarios en algunas de sus líneas, estos comentarios no son considerados como parte de la solución lógica, pero son fundamentales para el entendimiento del programa para las personas que no lo escribieron. Los comentarios se inician con los símbolos // posterior a ellos se coloca el comentario informativo. Observe en la Figura 11, línea 2 un ejemplo de comentario.

```
1  Proceso saludar
2      Escribir "Hola mundo"; //esto es un comentario
3  FinProceso
```

Figura 11 Pseudocódigo que envía el mensaje “Hola mundo”

Se recomienda el uso de comentarios en las siguientes partes del pseudocódigo:

- En la declaración de cada variable: se debe hacer una descripción breve de lo que almacenará cada variable o un grupo de ellas.
- Bloques de instrucciones que pueden resultar complejas de entender a primera vista y subprocesos en general: se hace una breve descripción de lo que hace esa parte del pseudocódigo.



VARIABLES, CONSTANTES Y TIPOS DE DATOS

Variables

Una variable es un espacio en la memoria de la computadora y se utiliza para almacenar de manera temporal datos que necesite el algoritmo para su correcta ejecución.

Si comparamos un algoritmo con una receta de cocina, las variables son la lista de ingredientes, mientras que la preparación del platillo son las instrucciones del algoritmo. Si no cumplimos con todos los ingredientes, no podremos llevar a cabo la receta como se debe, de igual forma ocurre con las variables, debemos tener las necesarias para que el algoritmo haga su trabajo.

Como se indicó inicialmente, una variable almacena datos; estos datos pueden ser ingresados por el usuario o calculados por el sistema. De manera muy simple, todo lo que el usuario ingrese al sistema debe almacenarse en variables, además, si el sistema hace cálculos y genera nuevos datos, estos deben almacenarse en otras variables.

Las variables poseen las siguientes características: **nombre y tipo de dato**.

Nombre

El nombre de una variable es muy importante ya que será el identificador de referencia en la memoria, es decir, la variable se llamará y se trabajará con su nombre.

El nombre de una variable debe cumplir las siguientes condiciones:

- Iniciar siempre con una letra puede ser mayúscula o minúscula.
- No tener caracteres especiales, por ejemplo: puntos, comas, espacios en blanco, letras tildadas (á, é, í, ó, ú, Á, É, Í, Ó, Ú, entre otras), símbolos (@, %, #, entre otros).
- El único carácter especial admitido es el guion bajo (_).
- Puede tener números, pero no iniciar con números.
- Longitud máxima de 25 caracteres, esto varía de un lenguaje a otro, lo que se recomienda en general es que el nombre no sea muy largo, pero debido a que esa recomendación es subjetiva se fijará el límite en 25.
- Ser descriptivo, el nombre de la variable debe dar una idea del dato que almacenará, por ejemplo, si la variable tendrá el valor del salario de un empleado el nombre podría ser alguno de estos casos: SalarioEmpleado, Salario_Empleado, salarioEmpleado, Salario_Empleado, salarioempleado.
- Ser único, en un mismo algoritmo no puede haber más de una variable con el mismo nombre, es decir, todas las variables deben tener nombres

diferentes. Asimismo, es recomendable que los nombres entre variables no sean similares, para evitar confusiones.

- Omitir preposiciones y artículos tales como: de, del, en, y, la, el, entre otros.
- Una buena práctica de programación es estandarizar el formato de los nombres de las variables, por ejemplo, si la primera letra del nombre es mayúscula, colocar todas las primeras letras de los nombres de la demás variables en mayúscula. Algunos ejemplos de estandarización son los siguientes: SalarioEmpleado, Salario_Empleado, salarioEmpleado, Salario_Empleado.
- Todos los lenguajes tienen palabras reservadas por lo que las variables no pueden tener un nombre igual a una palabra reservada, por ejemplo, si la palabra Suma es reservada por el lenguaje que se utiliza no puede haber una variable con ese nombre. Para saber las palabras reservadas de un lenguaje se debe ir a la sección de ayuda del mismo.

Tipo de dato

Todas las variables deben especificar el tipo de dato que almacenarán. La decisión de qué tipo de dato asignar a una variable depende del análisis lógico que se haga del problema a resolver.

Los diferentes tipos de datos que se analizarán en esta sección son los siguientes: numéricos (enteros y reales), caracter (cadenas) y booleanos.

Numéricos: *enteros*

Las variables definidas con este tipo de dato solo podrán almacenar valores numéricos que no utilicen decimales, por ejemplo:

- 25
- -80
- 202

Algunos ejemplos de situaciones que son datos de tipo entero son: cantidad de automóviles vendidos, cantidad de estudiantes en una tutoría, cantidad de aves de corral en una finca, entre otros. Estos elementos no podrían tener en su cantidad decimales, ya que no se puede vender 2.7 automóviles, o se venden 2 o se venden 3. Igual situación se tiene para los estudiantes y las aves de corral, no se puede contar 30,8 estudiantes o 11,5 aves. Por ende, no sería lógico que una variable que almacene alguno de esos datos sea de tipo real (con decimales).

Numéricos: reales

Las variables definidas con este tipo de dato podrán almacenar valores numéricos con o sin decimales, por ejemplo:

- 25,8
- -20,2
- 9,0

Nótese que en el último ejemplo el valor no tiene parte decimal con peso, pero al ser un tipo de dato real se le asigna una parte decimal, aunque esta sea .0.

Algunos ejemplos de situaciones que son datos de tipo real son: cantidad de kilogramos que pesa una persona, salario de un empleado, monto de impuesto de ventas, intereses de una cuenta de ahorros o de una deuda, entre otros. Estos elementos pueden tener o no una parte decimal.

Caracter y cadena

Las variables definidas con este tipo de dato pueden almacenar un solo caracter o varios caracteres. A la unión de dos o más caracteres se les denomina cadena o cadena de caracteres, tanto los valores caracteres como las cadenas se representan entre comillas. Este tipo de datos contempla no solo letras, sino también símbolos y números, por ejemplo:

- "Damaris"
- "S"
- "@%{(=&&"
- "1980"

A pesar de que una cadena de caracteres tenga solo números no significa que sea un tipo de dato numérico, es decir, a la cadena no se le podrían realizar las diversas operaciones aritméticas que si se pueden hacer con los datos numéricos.

Sin embargo, existen operaciones que se pueden hacer con las cadenas como, por ejemplo; separarlas, buscar elementos en ellas, concatenarlas, entre otros. Esta última operación es conveniente explicarla más a fondo.

Concatenar cadenas es simplemente unir las mediante el signo +. Por ejemplo si concatenamos la palabra "chéga" (significa amigo en lengua cabécar) y la palabra "amigo" lo expresamos de la siguiente forma "chéga"+"amigo" y el resultado sería de esta forma "chégaamigo". Para hacer una cadena con un espacio en blanco en medio

de las dos palabras haríamos esto “chéga” + “ ” + “amigo”, (la cadena resultante sería “chéga amigo”) nótese que lo que se hizo fue agregar un espacio en blanco en la expresión para concatenar. El mismo resultado tendríamos si agregamos el espacio en blanco al final de la palabra “chéga ” o al inicio de la palabra “ amigo”.

Algunos ejemplos de situaciones que son datos de tipo caracter o cadena son: nombres, descripciones, identificaciones de personas o vehículos, respuestas que se basen en letras por ejemplo “S” o “N”, entre otros.

Booleanos (Lógicos)

Las variables definidas con este tipo de dato solo podrán almacenar uno de dos valores opuestos, es decir, verdadero o falso; o en inglés True/False.

Este tipo de dato se emplea para valores que son mutuamente excluyentes, por ejemplo, mayoría de edad, alguien es mayor de edad o no lo es, pero no puede ser ambos en el mismo instante, o la pertenencia o no a algún grupo, como por ejemplo un estudiante puede o no pertenecer al club de deportes de su escuela.

Declaración de variables

La declaración de variables se hace al inicio del algoritmo y sirve para indicarle al sistema las variables que se utilizarán en las instrucciones del algoritmo, todas las variables deben ser declaradas.

La declaración de variables puede hacerse de varias formas, dependiendo del lenguaje en el que se esté trabajando, para nuestro aprendizaje tomaremos el siguiente estándar para la declaración de variables:

Definir Variable Como tipo de dato;

Donde:

- Definir: es una palabra reservada del lenguaje para especificar que lo que viene a continuación es una variable. Esta palabra cambia de un lenguaje a otro o puede que no se use.
- Variable: es el nombre que se le otorga a la variable, recordemos seguir las especificaciones para los nombres.
- Como tipo de dato: es el tipo de dato que tendrá la variable, ya sea: entero, real, caracter, cadena, booleano.
- El punto y coma debe estar al final de casi todas las instrucciones del algoritmo.

Ejemplos

Definir NombreEstudiante Como Caracter; se está declarando una variable de tipo caracter (en este caso el tipo de dato funcionará tanto para caracter como para cadena), cuyo nombre es NombreEstudiante.

Definir Salario Como Real; se está declarando una variable de tipo numérico real (acepta decimales), cuyo nombre es Salario.

Definir CantidadEstudiantes Como Entero; se está declarando una variable de tipo numérico entero (no acepta decimales), cuyo nombre es CantidadEstudiantes.

Definir Mayor Como Logico; se está declarando una variable de tipo booleano (valor verdadero o falso), cuyo nombre es Mayor.

Valor inicial

El valor inicial es el primer valor que se le otorga a una variable, a esto se le denomina inicialización. Esta acción ayuda a limpiar un posible dato que tenga la variable y colocar un valor que se considera válido.

La inicialización es simplemente asignar un valor a una variable, pero al ser el primer valor que se le otorga se le denomina de dicha manera, una variable se inicializa una única vez, antes de ser empleada en el algoritmo. Esta acción se ejecuta mediante el operador de asignación, ya sea el igual (=) o la flecha hacia la izquierda (←).

Ejemplos

NombreEstudiante= "X";

Salario←0;

CantidadEstudiantes=0;

Mayor←Verdadero;

El orden en un algoritmo es primero declarar las variables y luego inicializarlas, esto quedaría de la siguiente forma:

Algoritmo EjemploVariables

Definir NombreEstudiante Como Caracter;

Definir Salario Como Real;

Definir CantidadEstudiantes Como Entero;

Definir Mayor Como Logico;

NombreEstudiante= "X";

Salario=0;

CantidadEstudiantes=0;

Mayor=Verdadero;

FinAlgoritmo

En el ejemplo anterior el Inicio corresponde al principio del algoritmo de igual manera el Fin, en este caso se están haciendo únicamente la declaración e inicialización de variables, faltan las instrucciones del algoritmo. Recordemos que una variable cambiará su valor en algún momento del algoritmo, por lo que el valor inicial se perderá por algún otro valor que necesitemos.

Ejemplos de variables

Ejercicio 1

En la Tabla 2, observe los ejemplos brindados y la explicación de si el nombre de la variable es correcto o incorrecto.

TABLA 2

EJEMPLOS DE NOMBRES DE VARIABLES

Nombre de la variable	Correcto	Incorrecto	Justificación	Nuevo nombre
SalarioEmpleado	X		Emplea un nombre significativo sin espacios	
Nombre Estudiante		X	Tiene un carácter especial, el espacio en blanco.	NombreEstudiante Nombre_Estudiante nombreEstudiante nombreestudiante
Ced	X		Aunque es correcto, podría especificarse mejor.	Cedula
Primer_numero	X		Utiliza guion bajo y el nombre es significativo	
Subtotal2	X		Emplea el 2 hasta el final	
Emplead@		X	No puede emplear el carácter especial: @	Empleado Empleada
P123R		X	La sintaxis es correcta pero no es un nombre significativo para el programa.	Presupuesto
2da nota del examen ordinario		X	Inicia con número, deja espacios en blanco, tiene	NotaOrdinario2 NotaOrdinario_2 notaordinario2 SegundoOrdinario

Este documento forma parte de una unidad didáctica que está en desarrollo en la Universidad Estatal a Distancia de Costa Rica.
 Su contenido se encuentra bajo la ley de Propiedad Intelectual y cualquier mención a este debe ser indicado como obra en proceso.

Nombre de la variable	Correcto	Incorrecto	Justificación	Nuevo nombre
			más de 25 caracteres, tiene elementos gramaticales no recomendados (del)	Segundoordinario segundoordinario Ordinario2 ordinario2
NotaPrimerExamen	X		El nombre es significativo y no contiene espacios	
ImVnt		X	Nombre no descriptivo	ImpuestoVentas
TotaldeAutos		X	Tiene elementos gramaticales no recomendados (de).	TotalAutos Total_Autos totalAutos
Descuento1	X		El nombre es significativo y no contiene espacios	

Ejercicio 2

En la Tabla 3 analice la información que almacena cada variable y observe el tipo de dato que debería asignársele.

TABLA 3
EJEMPLOS DE TIPOS DE DATOS

Descripción	Tipo de dato				
	Entero	Real	Caracter	Cadena	Booleano (Lógico)
Cantidad de estudiantes.	X				
Salario de la gerente general.		X			
Respuesta del usuario (de una sola letra). Si (S) o No (N).			X		
Nombre de un estudiante.				X	
Valor de falso o verdadero para mayoría de edad.					X
Placa costarricense de vehículo particular.				X	
Cédula de identidad costarricense sin espacios ni guiones.				X	
Total de libros vendidos.	X				

Ejercicio 3

En la Tabla 4, se presentan las descripciones de la tabla anterior y se muestra la declaración de variable de cada una de ellas.

TABLA 4
EJEMPLOS DE DECLARACIONES DE VARIABLES

Descripción	Declaración
Cantidad de estudiantes.	Definir CantidadEstudiantes como Entero;
Salario de la gerente general.	Definir SalarioGerenteGeneral como Real;
Respuesta del usuario (de una sola letra). Si (S) o No (N).	Definir RespuestaUsuario como Caracter;
Nombre de un estudiante.	Definir NombreEstudiante como Cadena;
Valor de falso o verdadero para mayoría de edad.	Definir MayorEdad como Lógico;
Placa costarricense de vehículo particular.	Definir PlacaVehiculo como Cadena;
Cédula de identidad costarricense sin espacios ni guiones.	Definir Cedula como Cadena;
Total de libros vendidos.	Definir TotalLibrosVendidos como Entero;

Constantes

Las constantes, al igual que las variables, son espacios en memoria, pero su diferencia es que el valor que almacenan siempre será el mismo, es decir, nunca cambiará **durante la ejecución** el algoritmo.

En cuanto a las características de nombre y tipo, las constantes y variables comparten las mismas normas.

Las constantes deben declararse usando la misma nomenclatura empleada para las variables. En la inicialización es donde se marca la gran diferencia entre las constantes y las variables, en este paso a la constante se le asigna el valor que deseamos que tenga durante todo el algoritmo.

Ejemplo

Algoritmo EjemploConstante

Definir Radio Como Real;

Definir Pi Como Real;

Radio=0;

Pi=3.14;

FinAlgoritmo

En el ejemplo anterior, se declaró una variable de tipo real llamada Radio y una constante de tipo real con nombre Pi, note que a Pi se le asigna, en la inicialización, el valor (3.14) que tendrá durante toda la ejecución del algoritmo.

Algunos otros ejemplos de constantes son descuentos, impuestos, tipos de cambio monetarios, entre otros. Recordemos que una constante no cambiará su valor durante la ejecución del algoritmo, pero si lo puede hacer quien programa el algoritmo antes de que este se ejecute.

Se debe tener presente que las constantes que almacenan porcentajes se deben inicializar con los valores para cálculos aritméticos y no usando el símbolo de porcentaje (%), por ejemplo:

- Descuento del 10%. Descuento=0.10;
- Impuesto del 7%. Impuesto=0.07;

Recordemos que todo lo que digita el usuario o todo lo que se almacena como resultado de algún cálculo será considerado como una variable, por el contrario, un constante **nunca** se le deberá solicitar al usuario ni dependerá de un cálculo.

Ejercicio 1.

En la Tabla 5 se presentan descripciones de variables y constantes, observe a cuál corresponde cada una, según la descripción.

TABLA 5
EJEMPLOS DE CONSTANTES Y VARIABLES

Descripción	Variable	Constante
Nombre de un estudiante.	X	
Descuento del 5%.		X
Impuesto de ventas (13%).		X
Salario mensual.	X	
Tipo de cambio dólar.		X
Respuesta de S o N del usuario.	X	
Costo base de habitación (25000) en colones.		X

AQUÍ DEBE IR LO DEL APP DE VARIABLES



ESTRUCTURAS DE DECISIÓN

Estructuras de decisión

Las estructuras de decisión permiten evaluar una condición dada y dependiendo de ese resultado el flujo de procesamiento del algoritmo toma uno u otro camino. Los resultados de las condiciones para una estructura de decisión siempre serán de valores booleanos, es decir, verdadero (true) o falso (false).

Existen dos tipos de estructuras de decisión, primero están los Si (If) y luego los Según (Switch). El Si puede o no tener una estructura conocida como el Sino, la cual es complementaria al Si.

Estructura Si (If)

La estructura “**Si condición entonces**” es la más simple de las estructuras de decisión, su lógica es bastante sencilla y se analiza de la siguiente forma: Si una condición dada tiene un valor de verdadero, entonces se harán las instrucciones que correspondan en ese caso.

Si pasamos a ejemplos de la vida cotidiana tendremos los siguientes casos:

- **Si suena el teléfono entonces** lo contesto.
- **Si el río está bajo entonces** lo cruzo.
- **Si tengo sed entonces** tomo agua.
- **Si trabajo más de 32 horas semanales entonces** me pagan horas extras.

Como podemos ver, cada “Si” tiene una condición: suena el teléfono, el río está bajo, tengo sed, trabajo más de 32 horas semanales; y también una acción que depende del resultado de esa condición: lo contesto, lo cruzo, tomo agua, me pagan horas extras. El resultado de cada una de las condiciones debe ser verdadero para ejecutar las acciones respectivas, este comportamiento se aplica en cualquier estructura de decisión que trabajemos.

Si analizamos nuestra vida, nos damos cuenta de que toda ella está llena de “Si”, cada decisión que tomamos obedece al resultado booleano de una condición, en muchos casos, y como seres pensantes, tenemos muchas opciones para elegir, pero al final todas esas opciones se reducen a simples “Si”.

Características

Las características de un “Si” son las siguientes:

- Siempre debe tener una condición.

- Puede o no tener un Sino (*Else*), más adelante veremos esta estructura.
- Evalúa datos específicos o rangos de datos.
- Se recomienda su uso cuando los datos a evaluar son en rangos o cuando tenemos de 1 a 3 elementos para elegir, de lo contrario es mejor emplear una estructura conocida como Según (*Switch*).

Representación

De forma general, un “Si” se representa de la siguiente manera:

Si (Condición) entonces

Instrucción

Instrucción...

FinSi

Donde:

- Si: es el inicio de la estructura de decisión y de su encabezado.
- (Condición): es la condición que puede tener un valor verdadero o falso.
- Entonces: es la palabra que cierra el encabezado de la estructura de decisión.
- Instrucción: es una instrucción que se ejecutará en caso de que la condición tenga un resultado verdadero.
- FinSi: es el cierre de la estructura de decisión.

Condiciones

Toda estructura “Si” debe tener una condición, esta se compone de los siguientes elementos: una variable o un valor, un operador relacional(comparativo) y una variable o un valor. Es decir, **Si** (*Variable/Valor Operador Variable/Valor*) **entonces**...

Las condiciones pueden ser de dos tipos: **simples o compuestas**.

Una **condición simple** posee un único grupo de elementos, por ende, tiene la siguiente forma:

- (Variable/Valor Operador Variable/Valor)

Por ejemplo:

Si (*salario < 100000*) **entonces**

Escribir “No paga impuestos”

FinSi

Mientras que una **condición compuesta** tiene dos o más grupos de elementos, pero con la característica de que entre cada grupo hay un operador lógico, y se representa de la siguiente manera:

- (Variable/Valor Operador Variable/Valor) Operador Lógico (Variable/Valor Operador Variable/Valor)

Por ejemplo:

Si (*salario* >150000) Y (*salario* <300000) **entonces**

Escribir “Paga 5% de impuestos”

FinSi

Analicemos los siguientes ejemplos de condiciones simples:

- Si (Variable = 7) entonces...
- Si (Variable <= 18) entonces...
- Si (Variable1 >= Variable2) entonces...
- Si (Variable != “Gabriela”) entonces...
- Si (Variable = Verdadero) entonces...
- Si (1 = 0) entonces...

Notemos que, en la mayoría de los casos, siempre hay una variable al inicio, esto puede sustituirse por un valor, pero lo normal es que usemos una variable. Al ver el último caso podemos notar que son dos valores, por ende, esta condición siempre dará como resultado un Falso (ya que 1 no es igual a 0), mientras que en las condiciones donde hay una o más variables los resultados de Falso o Verdadero cambiarán de acuerdo al valor que tengan las variables. Una condición, siempre llevará en medio de los valores o variables un operador comparativo.

Veamos ahora ejemplos de condiciones compuestas:

- Si ((Variable = 7) O (Variable = 42)) entonces...
- Si ((Variable >= 18) Y (Variable < 65)) entonces...
- Si ((Var1 >= Var2) Y (Var1 >= Var3) Y (Var >= Var4)) entonces...
- Si ((Variable != “Gabriela”) O (Variable != “Xinia”)) entonces...
- Si ((Var1 = Verdadero) Y ((Var2 != 0) O (Var2 >10))) entonces...
- Si ((1 = 0) Y (25 >= 80)) entonces...

Las condiciones compuestas tienen la misma mecánica de solución que las simples, a pesar de que sean un poco más complejas.

Para el análisis de ambos tipos de condiciones debemos tomar en cuenta las reglas de precedencia que estudiamos en el capítulo 2.

Finalmente, es importante aclarar que las condiciones pueden contener expresiones matemáticas, la forma de resolver este tipo de condiciones es empleando las reglas de precedencia matemática. Por ejemplo:

- Si $((Variable = 7) \text{ O } (Variable * 5 = 42))$ entonces...
- Si $(Variable = 7 + 3)$ entonces...
- Si $(Variable \text{ Mod } 3 = 0)$ entonces...

En el primer caso, se resuelve la expresión $Variable = 7$ y, en el mismo paso, la multiplicación $Variable * 5$, posteriormente se resuelve el operador relacional $(=)$ entre el resultado de la multiplicación y el número 42. Finalmente, con los dos resultados booleanos obtenidos, se resuelve el operador lógico O.

En el segundo caso, se efectúa primero la suma de $7 + 3$ y luego la comparación de ese resultado con $Variable$.

En el tercer caso, se opera primero $Variable \text{ Mod } 3$, y luego la comparación con 3.

Ejemplos

A continuación, presentamos algunos ejemplos de algoritmos que utilizan estructuras de decisión con condiciones simples y compuestas.

Ejemplo 1 Algoritmo con mensaje de sed

El algoritmo consulta al usuario si tiene sed y dependiendo de la respuesta ingresada le aconseja tomar agua o no le aconseja nada.

Pseudocódigo

Algoritmo Sed

Definir Sed Como Cadena;

Sed= "S";

Escribir "¿Tiene sed? S / N";

Leer Sed;

Si (Sed="S") entonces

 Escribir "Tome agua.";

FinSi

FinAlgoritmo

El ejemplo anterior desarrollado en PSeInt por medio de un pseudocódigo sería como se muestra en la Figura 12.

```
1  Proceso TengoSed
2      Definir sed Como cadena //declaración de la variable
3      sed="s";                //inicializa la variable en "s"
4      Escribir "¿Tiene sed (s/n)?"; //mensaje al usuario
5      Leer sed                //lee la respuesta del usuario
6      Si sed="S" Entonces     //consulta si el usuario digitó una "S"
7          Escribir "Tome agua" //si el usuario digitó una "S", envía el mensaje
8      Fin Si
9  FinProceso
```

Figura 12 Pseudocódigo con condición "Si" para beber agua

En el ejemplo anterior, se declaró una variable llamada Sed y se inicializó con la letra "S", luego se muestra por pantalla la pregunta "¿Tiene sed? (S / N)" y se lee lo que digitó el usuario. Posterior a la lectura, se coloca un Si que evalúa el valor de la variable Sed y si esta tiene un valor de "S" entonces se ejecuta la instrucción Escribir "Tome agua."

En caso de que el usuario digitara un valor diferente a “S” el resultado de la condición Sed=“S” sería falso, por lo que la instrucción Escribir no se ejecutaría, esto porque el flujo de procesamiento no entraría al Si y saltaría al FinSi y seguidamente al FinAlgoritmo.

En el caso que este ejemplo se desarrollara en un diagrama de flujo, la Figura 13 muestra cómo se desarrollaría.

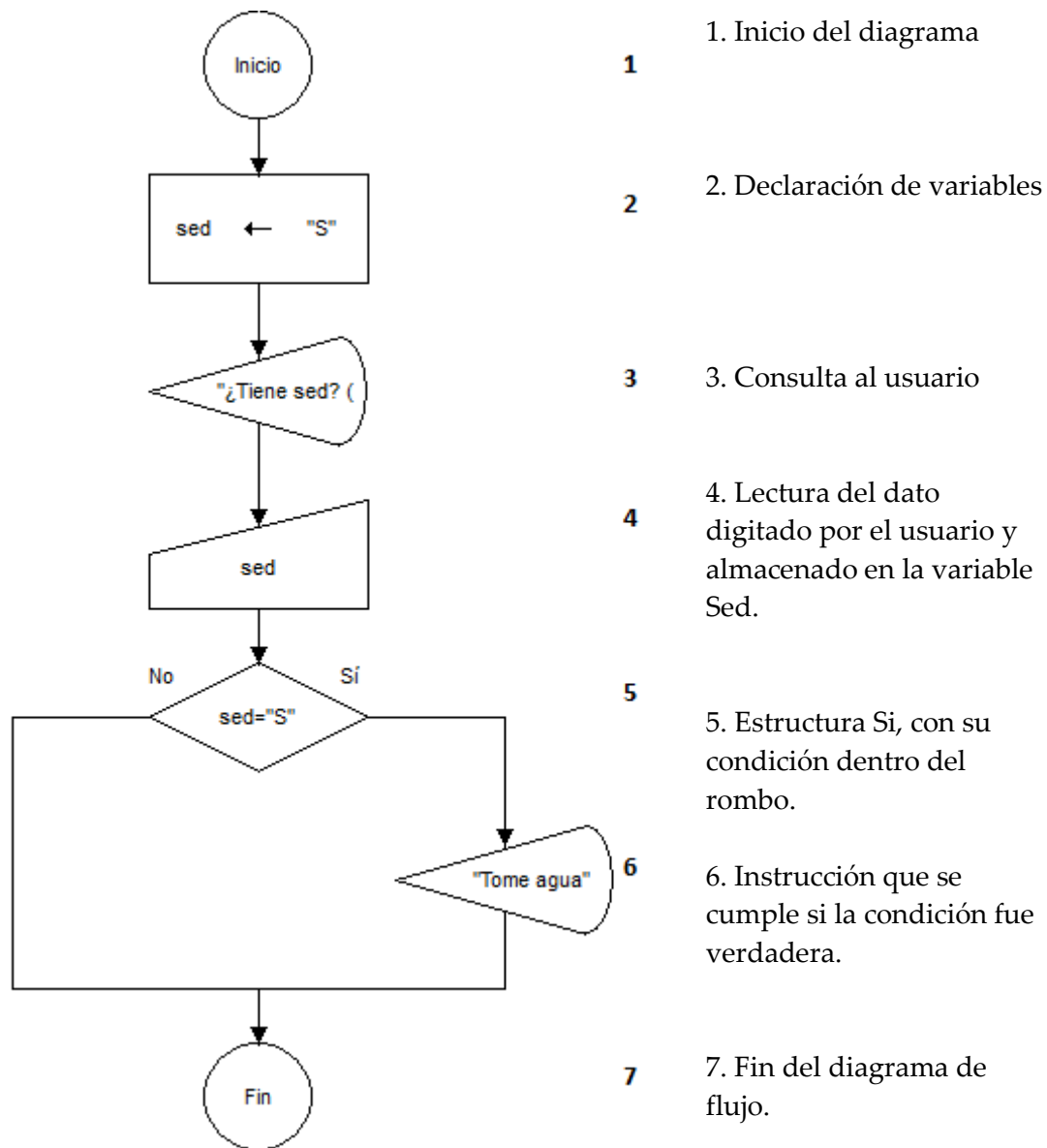


Figura 13 Diagrama de flujo con condición “Si” para beber agua

Ejemplo 2 Algoritmo para dividir dos números.

El algoritmo divide dos números, pero primero valida que el segundo número (el divisor) no sea cero, luego hace la división dependiendo del resultado del "Si" anterior. Finalmente, toma el resultado y determina si el número obtenido es múltiplo de 3 y de 5 a la vez; si lo es; muestra un mensaje indicándolo, de lo contrario no muestra nada. Considere que, si el divisor es cero, debe enviar un mensaje al usuario indicando que no se puede hacer la división.

Pseudocódigo

Algoritmo Division

```
Definir Dividendo, Divisor, Cociente Como Entero;
Dividendo=0;
Divisor=0;
Cociente=0;
Escribir "Digite el dividendo.";
Leer Dividendo;
Escribir "Digite el divisor.";
Leer Divisor;
Si Divisor=0 entonces
    Escribir "Error, el divisor NO puede ser cero. NO se puede realizar la división";
FinSi
Si Divisor!=0 entonces
    Cociente= Dividendo/Divisor;
    Escribir "Cociente es igual a: ",Cociente;
FinSi
Si ((Cociente Mod 3=0) Y (Cociente Mod 5=0) Y (Divisor!=0)) Entonces
    Escribir "El cociente ",Cociente," es múltiplo de 3 y 5 a la vez.";
FinSi
```

FinAlgoritmo

En el ejemplo anterior, se declararon tres variables (Dividendo, Divisor y Cociente) y cada una se inicializó con el valor cero (0), luego se muestra por pantalla la solicitud al usuario para que digite un valor para el dividendo y se lee lo que digitó el usuario; lo mismo se hace con la variable Divisor. Posterior a la última lectura, se coloca un Si que evalúa el valor de la variable Divisor, y si esta tiene un valor de cero (0)

entonces se ejecuta la instrucción Escribir "Error, el divisor NO puede ser cero. NO se puede realizar la división";

Luego del Si anterior, se hace otro que evalúa si el Divisor tiene un número diferente a cero (0), si lo tiene se efectúa la división y se muestra el resultado. Después del último Si, hay otro que analiza si la variable Cociente (que es el resultado de la división) es divisible entre 3 y entre 5 y si el Divisor es diferente a cero (0), si estas tres condiciones se cumplen se muestra el mensaje Escribir "El cociente ",Cociente," es múltiplo de 3 y 5 a la vez.";

Debemos notar que en este algoritmo la clave es la variable Divisor, ya que es la que se analiza para determinar si hacemos o no la división. Esto tiene que ver con el enunciado propuesto y con un postulado matemático universalmente aceptado, el cual dicta que la división entre cero no existe, por ende, el algoritmo debe contemplar esta posibilidad y validar que en caso de que el divisor digitado por el usuario sea cero, no se haga la división.

Para el ejemplo anterior, el pseudocódigo en PSeInt se observa en la Figura 14.

```
1  Proceso DivisionMultiplos
2  Definir dividendo, divisor, cociente Como Entero //declaración de variables
3  //inicialización de los valores de las variables //
4  dividendo =0;
5  divisor=0;
6  cociente =0;
7
8  Escribir "Digite el dividendo"; //mensaje solicitando el dividendo
9  Leer dividendo; //lee el valor desde teclado
10 Escribir "Digite el divisor"; //mensaje solicitando el divisor
11 Leer divisor; //lee el valor desde teclado
12 Si divisor=0 entonces //Consulta si el divisor es cero
13     Escribir "Error, el divisor NO puede ser cero. NO se puede realizar la división";
14 FinSi
15 Si Divisor!=0 entonces //realiza la operación porque el divisor es diferente a cero
16     cociente= dividendo/divisor;
17     Escribir "Cociente es igual a: ",cociente;
18 FinSi
19 //Si se dan las condiciones de los cálculos para determinar si el número es múltiplo de 3 y 5
20 Si ((cociente Mod 3=0) Y (cociente Mod 5=0) Y (divisor!=0)) Entonces
21     Escribir "El cociente ",cociente," es múltiplo de 3 y 5 a la vez."; //envía mensaje
22 FinSi
23 FinProceso
```

Figura 14 Pseudocódigo con división de números y verificación de múltiplos de 3 y 5

Para el mismo ejemplo, la Figura 15 muestra el diagrama de flujo.

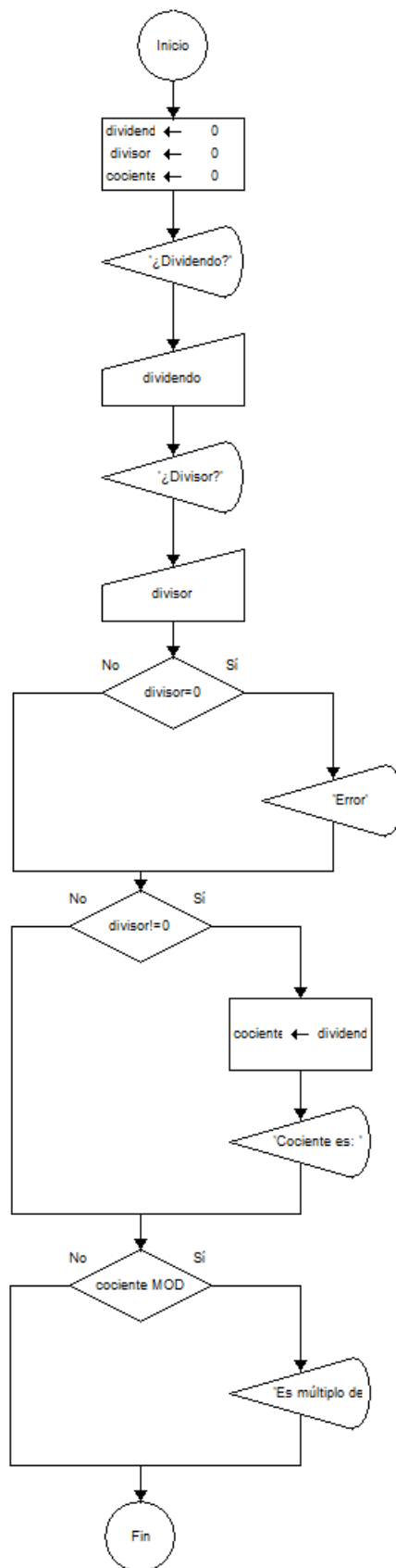


Figura 15 Diagrama de flujo con división y verificación de múltiplos de 3 y 5

Ejemplo 3 Mayor de tres números

El algoritmo recibe por teclado tres números y determina cuál es el mayor entre ellos. Asuma que todos los números introducidos por el usuario son diferentes y no poseen decimales.

Pseudocódigo

Algoritmo MayorTresNumeros

Definir N1, N2, N3 Como Entero;

N1=0;

N2=0;

N3=0;

Escribir "Escriba el primer número";

Leer N1;

Escribir "Escriba el segundo número";

Leer N2;

Escribir "Escriba el tercer número";

Leer N3;

Si $N1 > N2$ Y $N1 > N3$ entonces

Escribir "El ", N1, " es el mayor";

FinSi

Si $N2 > N1$ Y $N2 > N3$ entonces

Escribir "El ", N2, " es el mayor";

FinSi

Si $N3 > N1$ Y $N3 > N2$ entonces

Escribir "El ", N3, " es el mayor";

FinSi

FinAlgoritmo

Para el ejemplo anterior, la Figura 16 muestra cómo se realizaría el pseudocódigo en PSeInt.

```
1  Proceso MayorTresNumeros
2      Definir N1, N2, N3 Como Entero;
3      N1=0;
4      N2=0;
5      N3=0;
6      Escribir "Escriba el primer número";
7      Leer N1;
8      Escribir "Escriba el segundo número";
9      Leer N2;
10     Escribir "Escriba el tercer número";
11     Leer N3;
12     Si N1>N2 Y N1>N3 entonces //consulta si el primer número es mayor que el segundo y tercero
13         Escribir "El ", N1, " es el mayor";
14     FinSi
15     Si N2>N1 Y N2>N3 entonces //consulta si el segundo número es mayor que el primero y tercero
16         Escribir "El ", N2, " es el mayor";
17     FinSi
18     Si N3>N1 Y N3>N2 entonces //consulta si el tercer número es mayor que el primero y segundo
19         Escribir "El ", N3, " es el mayor";
20     FinSi
21 FinProceso
```

Figura 16 Pseudocódigo para determinar el mayor de tres números

En el caso del diagrama de flujo de datos, la Figura 17 muestra el mismo ejemplo, pero con una variación. En el paso 2 se muestra un solo mensaje solicitando los números y en el paso 3 se observa que los números son ingresados ordenadamente por teclado, asignando a n1, n2 y n3, los valores del número 1, número 2 y número 3.

En los casos de las decisiones del “Si” que corresponden a los pasos 4, 6 y 8 que se muestran en los rombos, se realizan las mismas condiciones que el pseudocódigo de la Figura 16 en los pasos 12, 15 y 18, con la única diferencia que en el diagrama de flujo se emplea el operador en inglés “AND” mientras que en el pseudocódigo en PSeInt este se escribe en español “Y”.

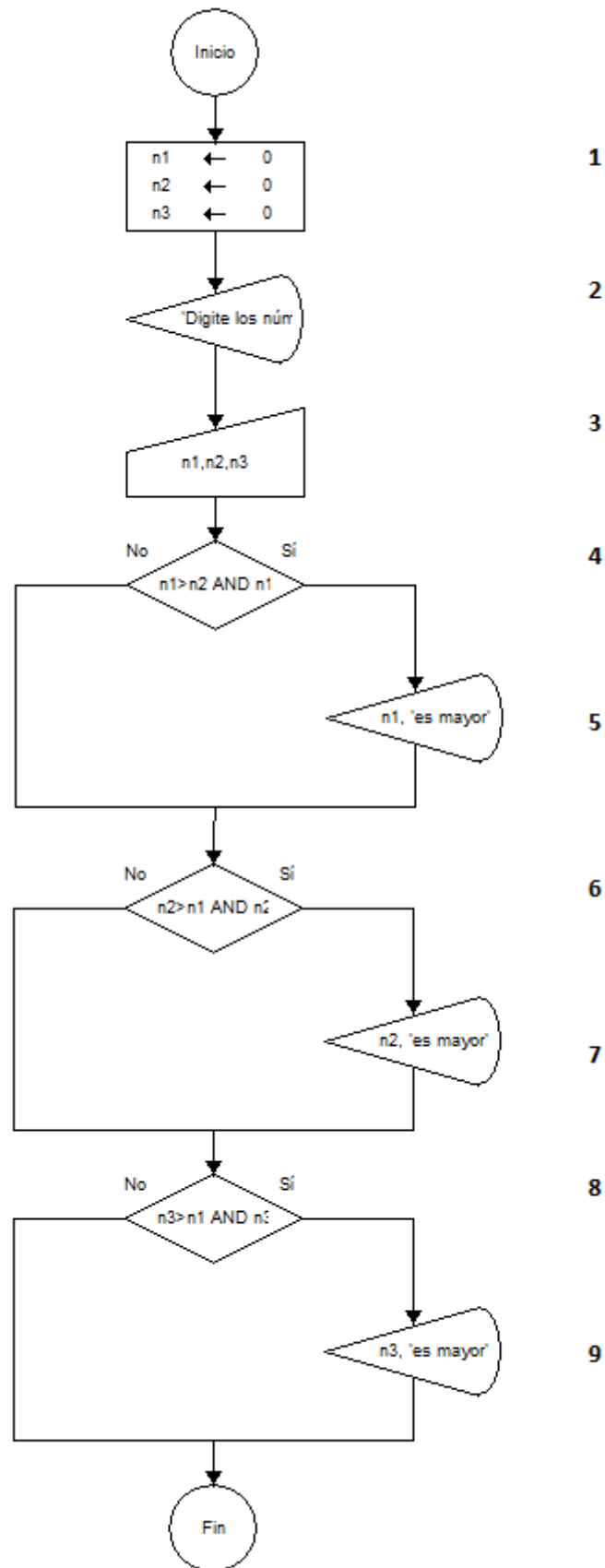


Figura 17 Diagrama de flujo para determinar el mayor de tres números

Este documento forma parte de una unidad didáctica que está en desarrollo en la Universidad Estatal a Distancia de Costa Rica.
Su contenido se encuentra bajo la ley de Propiedad Intelectual y cualquier mención a este debe ser indicado como obra en proceso.

Ejemplo 4 Calculadora con las cuatro operaciones básicas.

El algoritmo recibe por teclado una de cuatro opciones: "S" para hacer una suma, "R" para hacer una resta, "M" para hacer una multiplicación y "D" para realizar una división. Posterior a eso, se realiza la operación solicitada por el usuario. Se debe validar que en el caso de la división el divisor sea diferente de cero.

Pseudocódigo

Algoritmo Calculadora

Definir n1, n2, resultado Como Real;

Definir opc Como Carácter;

Escribir "Bienvenido a la calculadora. S= suma, R=resta, D=división,
M=multiplicación";

Leer opc;

Si opc="S" entonces

 Escribir "Escriba el primer número";

 Leer n1;

 Escribir "Escriba el segundo número";

 Leer n2;

 resultado=n1+n2;

 Escribir "La suma de ",n1, " + ", n2, " es: ", resultado

FinSi

Si opc="R" entonces

 Escribir "Escriba el primer número";

 Leer n1;

 Escribir "Escriba el segundo número";

 Leer n2;

 resultado=n1-n2;

 Escribir "La resta de ",n1, " - ", n2, " es: ", resultado

FinSi

Si opc="M" entonces

 Escribir "Escriba el primer número";

 Leer n1;

 Escribir "Escriba el segundo número";

 Leer n2;

resultado=n1*n2;

Escribir "La multiplicación de ",n1, " * ", n2, " es: ", resultado

FinSi

Si opc="D" entonces

Escribir "Escriba el primer número";

Leer n1;

Escribir "Escriba el segundo número";

Leer n2;

Si n2 != 0 entonces

resultado=n1/n2;

Escribir "La división de ",n1, " / ", n2, " es: ", resultado;

FinSi

Si n2 == 0 entonces

Escribir "Error, la división entre cero no existe";

FinSi

FinSi

FinAlgoritmo

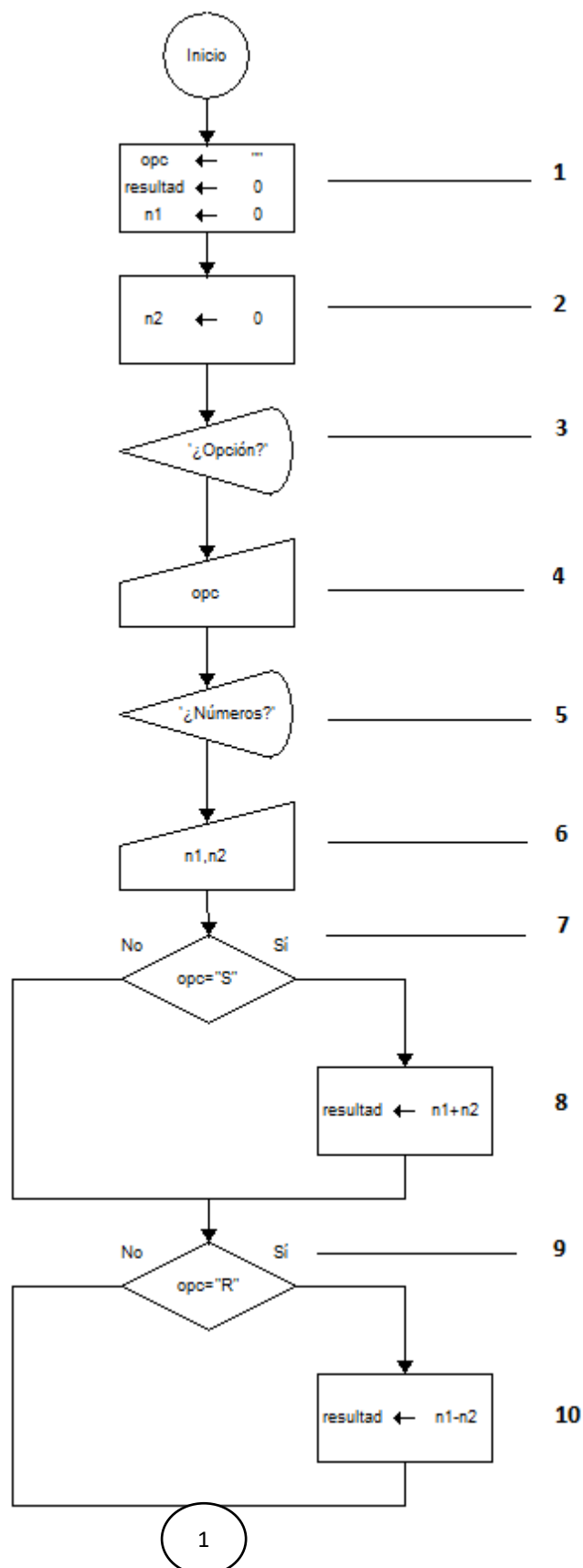
Para el ejemplo anterior, la Figura 18 muestra cómo se realizaría el pseudocódigo en PSeInt, pero con una variación. Cuando se tienen instrucciones repetitivas en todos los casos, como por ejemplo solicitar y leer ambos números, podemos reducir la cantidad de código si solicitamos antes de la operación estos datos y cuando se ingresa a los "Si" únicamente se realizará la operación y se mostrará el mensaje con el resultado final. A esto se le denomina hacer el código más **eficiente**, que quiere decir que con menos líneas de código, resolvemos el mismo problema. El algoritmo que se presentó anteriormente para este caso es **eficaz** porque resuelve el problema, pero utiliza muchas más líneas de código, lo que consume más recursos del computador.

```
1  Proceso Calculadora
2  Definir n1, n2, resultado Como Real;
3  Definir opc Como Carácter;
4  Escribir "Bienvenido a la calculadora. S= suma, R=resta, D=división, M=multiplicación";
5  Leer opc;
6  Escribir "Escriba el primer número";
7  Leer n1;
8  Escribir "Escriba el segundo número";
9  Leer n2;
10 Si opc="S" entonces
11     resultado=n1+n2;
12     Escribir "La suma de ",n1, " + ", n2, " es: ", resultado
13 FinSi
14 Si opc="R" entonces
15     resultado=n1-n2;
16     Escribir "La resta de ",n1, " - ", n2, " es: ", resultado
17 FinSi
18 Si opc="M" entonces
19     resultado=n1*n2;
20     Escribir "La multiplicación de ",n1, " * ", n2, " es: ", resultado
21 FinSi
22 Si opc="D" entonces
23     Si n2 != 0 entonces
24         resultado=n1/n2;
25         Escribir "La suma de ",n1, " + ", n2, " es: ", resultado;
26     FinSi
27     Si n2 == 0 entonces
28         Escribir "Error, la división entre cero no existe";
29     FinSi
30 FinSi
31 FinProceso
```

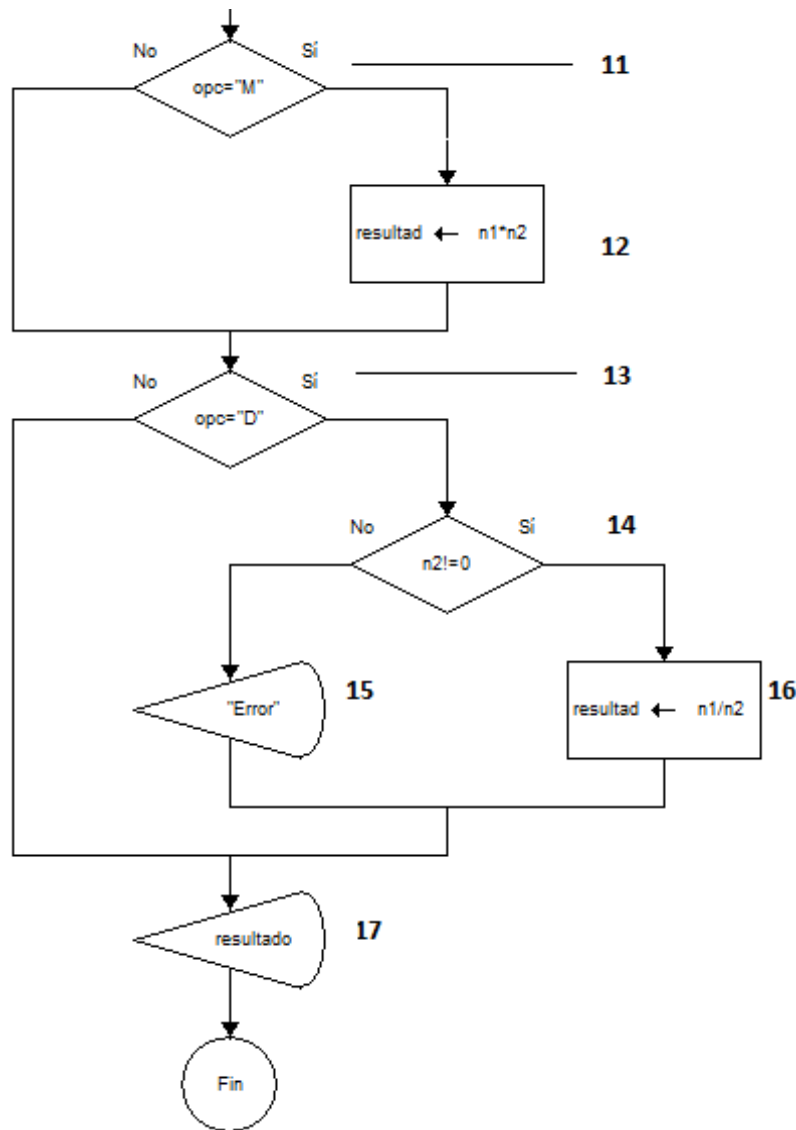
Figura 18 Pseudocódigo para la calculadora con cuatro operaciones básicas

En el caso del diagrama de flujo de datos, la Figura 19 muestra el mismo ejemplo, pero con algunas variaciones. Por ejemplo, al igual que en el pseudocódigo, se agrupó en las instrucciones 5 y 6 el mensaje de solicitud y lectura de los números para no tener que solicitarla siempre en las cuatro operaciones, esto hizo ahorrarse muchas líneas de código. Adicionalmente, otro cambio fue que en cada caso de la calculadora no se emitió un mensaje del cálculo, sino que se dejó hasta el final en el paso 17. Esto se puede hacer porque el contexto del problema indica que solo se realiza una operación (la que elija el usuario) pero si se tuviera que recibir dos números y mostrar

todas las operaciones para esos datos, se requerirían más variables para los resultados de cada operación y los cuatro mensajes independientes.



- 1 y 2. inicialización de las variables a usar en el problema. No se requiere de cuatro variables para el cálculo porque solo se muestra un cálculo (el que elija el usuario).
3. Consulta la opción del menú
4. Elige la operación
- 5 y 6. Sección más eficiente para no repetir en cada cálculo la solicitud de números (paso 5) y la lectura de ambos (paso 6)
7. Consulta si el usuario eligió la suma
8. Asigna al resultado la suma de ambos números
9. Consulta si el usuario eligió la resta
10. Asigna a resultado la resta de los números



11. Consulta si el usuario eligió la multiplicación
12. Asigna al resultado la multiplicación de ambos números
13. Consulta si el usuario eligió la división
14. Consulta si el divisor es diferente de cero
15. Si es igual a cero, envía un mensaje de error
16. Si es diferente de cero, realiza la división y la asigna a resultado
17. Solo se envía un mensaje general con el resultado obtenido

Figura 19 Diagrama de flujo para una calculadora

Ejemplo 5 Calcular el salario de un trabajador

Suponga que usted es el contador de una empresa y le solicitan generar la planilla para cada trabajador, donde considere las siguientes condiciones:

- Debe digitar el nombre completo del empleado
- El salario por hora es: ₡2000
- La hora extra: es el reconocimiento de 1,5 sobre el salario por hora.

Si un obrero trabajó 40 horas, se calculan las horas normales, a partir de 40 horas se debe considerar como horas extras la diferencia que haya laborado y calcularlas, por ejemplo, si alguien trabajó 43 horas se calculan 40 horas normales y 3 horas como extras.

El programa debe consultar si se deben considerar horas dobles (por trabajar en un feriado de pago obligatorio), si se introduce cero (0) entonces no hay que reconocerlas y si se introduce un número distinto de cero, se calcula como un pago doble.

Además de lo anterior, se deben hacer las deducciones de ley:

- el 8% de la CCSS
- el 13% del impuesto sobre la renta
- ₡10000 de póliza de vida básica

Posterior a los cálculos el programa debe mostrar: el salario bruto, todas las deducciones y el salario neto (salario bruto – deducciones).

Pseudocódigo

Algoritmo Salario

Definir SalBruto, SalNeto, ValHora, Horas, HoraExtr Como Real;

Definir HoraDobl, Deduc, DeduCCSS, DeduRenta, CCSS, Renta como Real;

Definir SalHora, Poliza Como Entero;

Definir NombrePer como Cadena;

ValHora =2000; //constante con el valor de cada hora

CCSS=0.08; //constante con el porcentaje deducción de la CCSS

Poliza=10000; //constante con la deducción de la póliza de vida

Renta = 0.13; //constante con el porcentaje deducción de renta

Escribir 'Escriba el nombre del empleado';

Leer NombrePer;

Escribir 'Escriba las horas trabajadas por el empleado'

Leer Horas;

Escribir 'Escriba el número de horas dobles del trabajador (0=no)';

Leer HoraDobl

//si NO trabajó horas extras, no se calculan (trabajó un máximo de 40 horas)

Si Horas<=40 Entonces

 //Cálculo del salario bruto: horas trabajadas + horas dobles

 SalBruto= (Horas*ValHora) + (HoraDobl*(2*ValHora));

Fin Si

//si Sí trabajó horas extras, es decir más de las 40 horas semanales

Si Horas>40 Entonces

 //entonces se calculan cuántas horas extras trabajó

 HoraExtr=Horas-40;

 //Cálculo del salario bruto: horas trabajadas + extras + dobles.

 //Se calcula 40*2000 porque ya se conoce que trabajó las 40 horas

 SalBruto= (40 * ValHora) + (HoraExtr*(1.5*ValHora)) +
(HoraDobl*(2*ValHora));

Fin Si

//Ahora se calculan las deducciones

DeduCCSS=SalBruto*CCSS; // deducción de la CCSS

DeduRenta=SalBruto*Renta; //deducción de la Renta

Deduc=DeduCCSS+DeduRenta+Poliza; //suma todas las deducciones

SalNeto=SalBruto-Deduc; //calcula el salario neto

Escribir 'El empleado ', NombrePer;, ' tiene un salario bruto de: ', SalBruto, ' la deducción de la CCSS es: ', DeduCCSS, ' la deducción de Renta es: ', DeduRenta, ' el total de deducciones fue: ', Deduc, ' el salario neto es: ', SalNeto;FinAlgoritmo

Para el ejemplo anterior, la Figura 20 muestra cómo se realizaría el pseudocódigo en PSeInt.

```
Proceso CalculoSalario
  Definir SalBruto, SalNeto, ValHora, Horas, HoraExtr Como Real;
  Definir HoraDobl, Deduc, DeduCCSS, DeduRenta, CCSS, Renta como Real;
  Definir SalHora, Poliza Como Entero;
  Definir NombrePer como Cadena;
  ValHora =2000; //constante con el valor de cada hora
  CCSS=0.08;      //constante con el porcentaje deducción de la CCSS
  Poliza=10000;   //constante con la deducción de la póliza de vida
  Renta = 0.13;   //constante con el porcentaje deducción de renta
  Escribir 'Escriba el nombre del empleado';
  Leer NombrePer;
  Escribir 'Escriba las horas trabajadas por el empleado'
  Leer Horas;
  Escribir 'Escriba el número de horas dobles del trabajador (0=no)';
  Leer HoraDobl
  //si NO trabajó horas extras, no se calculan (trabajó un máximo de 40 horas)
  Si Horas<=40 Entonces
    //Cálculo del salario bruto: horas trabajadas + horas dobles
    SalBruto= (Horas*ValHora) + (HoraDobl*(2*ValHora));
  Fin Si
  //si SÍ trabajó horas extras, es decir más de las 40 horas semanales
  Si Horas>40 Entonces
    //entonces se calculan cuántas horas extras trabajó
    HoraExtr=Horas-40;
    //Cálculo del salario bruto: horas trabajadas + extras + dobles.
    //Se calcula 40*2000 porque ya se conoce que trabajó las 40 horas
    SalBruto= (40 * ValHora) +(HoraExtr*(1.5*ValHora)) + (HoraDobl*(2*ValHora));
  Fin Si
  //Ahora se calculan las deducciones
  DeduCCSS=SalBruto*CCSS;      // deducción de la CCSS
  DeduRenta=SalBruto*Renta;    //deducción de la Renta
  Deduc=DeduCCSS+DeduRenta+Poliza; //suma todas las deducciones
  SalNeto=SalBruto-Deduc;      //calcula el salario neto
  Escribir 'El empleado ', NombrePer;
  Escribir ' tiene un salario bruto de: ',SalBruto;
  Escribir ' la deducción de la CCSS es: ',DeduCCSS;
  Escribir ' la deducción de Renta es: ', DeduRenta;
  Escribir ' el total de deducciones fue: ', Deduc;
  Escribir ' el salario neto es: ', SalNeto;
FinProceso
```

Figura 20 Pseudocódigo para calcular el salario de un trabajador

Ejemplo 6 Determinar el mayor y menor de cinco números

Elabore un pseudocódigo donde reciba cinco números por teclado y determine cuál es el número mayor y cuál es el menor. Asuma que todos los números son diferentes.

```
1  Proceso MayorMenor
2  Definir num1,num2,num3,num4, num5, men, may como entero;
3      Leer num1, num2, num3, num4, num5;
4      may=num1;
5      men=num1;
6      //Condiciones para determinar el mayor
7      Si num2>may Entonces
8          may=num2;
9      Fin Si
10     Si num3>may Entonces
11         may=num3;
12     Fin Si
13     Si num4>may Entonces
14         may=num4;
15     Fin Si
16     Si num5>may Entonces
17         may=num5;
18     Fin Si
19     Escribir "Mayor: ", may;
20     //Condiciones para determinar el menor
21     Si num2<men Entonces
22         men=num2;
23     Fin Si
24     Si num3<men Entonces
25         men=num3;
26     Fin Si
27     Si num4<men Entonces
28         men=num4;
29     Fin Si
30     Si num5<men Entonces
31         men=num5;
32     Fin Si
33     Escribir "Menor: ", men;
34 FinProceso
```

Figura 21 Pseudocódigo para determinar el mayor y menor de cinco números

Como se observa en la figura 21, la línea 3 define como enteros a todas las variables que se van a emplear: los cinco números, el menor y el mayor.

Las líneas 4 y 5 son muy importantes, pues asume que el primer número es tanto el mayor como el menor, este “truco” sirve para iniciar la comparación entre los valores y realizar las asignaciones de mayor o menor, dependiendo de si encuentra un valor o menor o mayor digitado en el primer valor ¿se podría hacer asumiendo que el valor

de la mitad o el último es el menor y mayor? Sí se podría, pero este orden es especialmente útil cuando hagamos lo mismo con los vectores y matrices que corresponden al siguiente capítulo. Veámoslo como la forma más ordenada de trabajar.

Observe que de las líneas de la 7 a la 19 el pseudocódigo pregunta número por número, si lo que digitó el usuario es mayor al valor del primer número, si alguna de estas condiciones se cumple entonces asigna a la variable mayor (may) el nuevo valor.

Las líneas de la 21 a la 33 el pseudocódigo pregunta número por número, si lo que digitó el usuario es menor al valor del primer número, si alguna de estas condiciones se cumple entonces asigna a la variable menor (men) el nuevo valor.

Estructura Si-Sino (If-Else)

La estructura “Si-Sino” funciona con la misma lógica del “Si”, su diferencia radica en su estructura, ya que cuenta con el “Sino”. En cuanto a su lógica, uso de condiciones y demás elementos, ambas estructuras son iguales.

Si pasamos a ejemplos de la vida cotidiana tendremos los siguientes casos:

- **Si** *llego a tiempo a la parada* **entonces** tomo el bus; **sino** camino.
- **Si** *el río está bajo* **entonces** lo cruzo; **sino** paso por el puente.
- **Si** *es verano* **entonces** viajo a la playa, **sino** viajo a la montaña.

Al analizar los ejemplos, podemos apreciar que, al igual que el Si, el Si-Sino posee una condición (llego a tiempo a la parada, el río está bajo, es verano), pero a diferencia de este, el Si-Sino tiene dos acciones. Un “Si-Sino” tiene dos acciones, y siempre ejecutará una de las dos, nunca las dos al mismo tiempo.

En el primer ejemplo, si se cumple la condición se ejecuta la acción *tomo el bus*, pero si no se cumple se realiza la acción del Sino, es decir, *camino*. Para el segundo caso, las acciones son *lo cruzo* (el río) en caso de que se cumpla la condición dada, de lo contrario *paso por el puente*, la cual sería la acción del Sino. El mismo análisis se hace en el tercer ejemplo, ya sea que se viaje a la playa; si se cumple la condición; o a la montaña, si no se cumple.

De forma general, un Si-Sino se representa de la siguiente manera:

Si (Condición) entonces

Instrucción

Instrucción

Instrucción...

Sino

Instrucción

Instrucción

Instrucción...

FinSi

Ejemplo 7 Recorrido en bus o caminando.

El algoritmo consulta al usuario si llegó a tiempo para tomar el bus, si llegó a tiempo emite un mensaje de tomar el bus y sino llegó a tiempo, se envía el mensaje de que debe caminar.

Si desarrollamos el ejemplo del bus a pseudocódigo tendremos lo siguiente:

Algoritmo Bus_Camino

Definir ATiempo Como Cadena;

ATiempo = "S";

Escribir "¿Llegó a tiempo a la parada? S / N";

Leer ATiempo;

Si (ATiempo ="S") entonces

Escribir "Tome el bus a su casa.";

Sino

Escribir "Camine a su casa.";

FinSi

FinAlgoritmo

En el ejemplo anterior se declaró una variable llamada ATiempo y se inicializó con la letra "S", luego se muestra por pantalla la pregunta "¿Llegó a tiempo a la parada? S/N" y se lee lo que digitó el usuario. Posterior a la lectura, se coloca un Si-Sino que evalúa el valor de la variable ATiempo y si esta tiene un valor de "S" entonces se ejecuta la instrucción Escribir "Tome el bus a su casa."; pero si la condición tiene un valor de falso, se ejecuta la acción Escribir "Camine a su casa.";

Si el usuario digita cualquier otra letra que no sea S (en mayúscula) el flujo de información se irá por el Sino. Ahora, limitar al usuario a digitar solo letras en mayúscula o minúscula es poco amigable, por lo que modificaremos la condición simple del Si-Sino por una compuesta, por ende, la línea Si (ATiempo ="S") entonces

quedará de la siguiente forma: Si (ATiempo ="S") O (ATiempo ="s") entonces. Nótese, que lo que agregamos fue un operador lógico con otra condición, por lo que si el usuario digita una "S" (en mayúscula) o una "s" (en minúscula), la condición tendrá un valor de verdadero, brindándole un poco más de lógica y amigabilidad el algoritmo.

En diagrama de flujo, el ejemplo anterior se representa en la Figura 22.

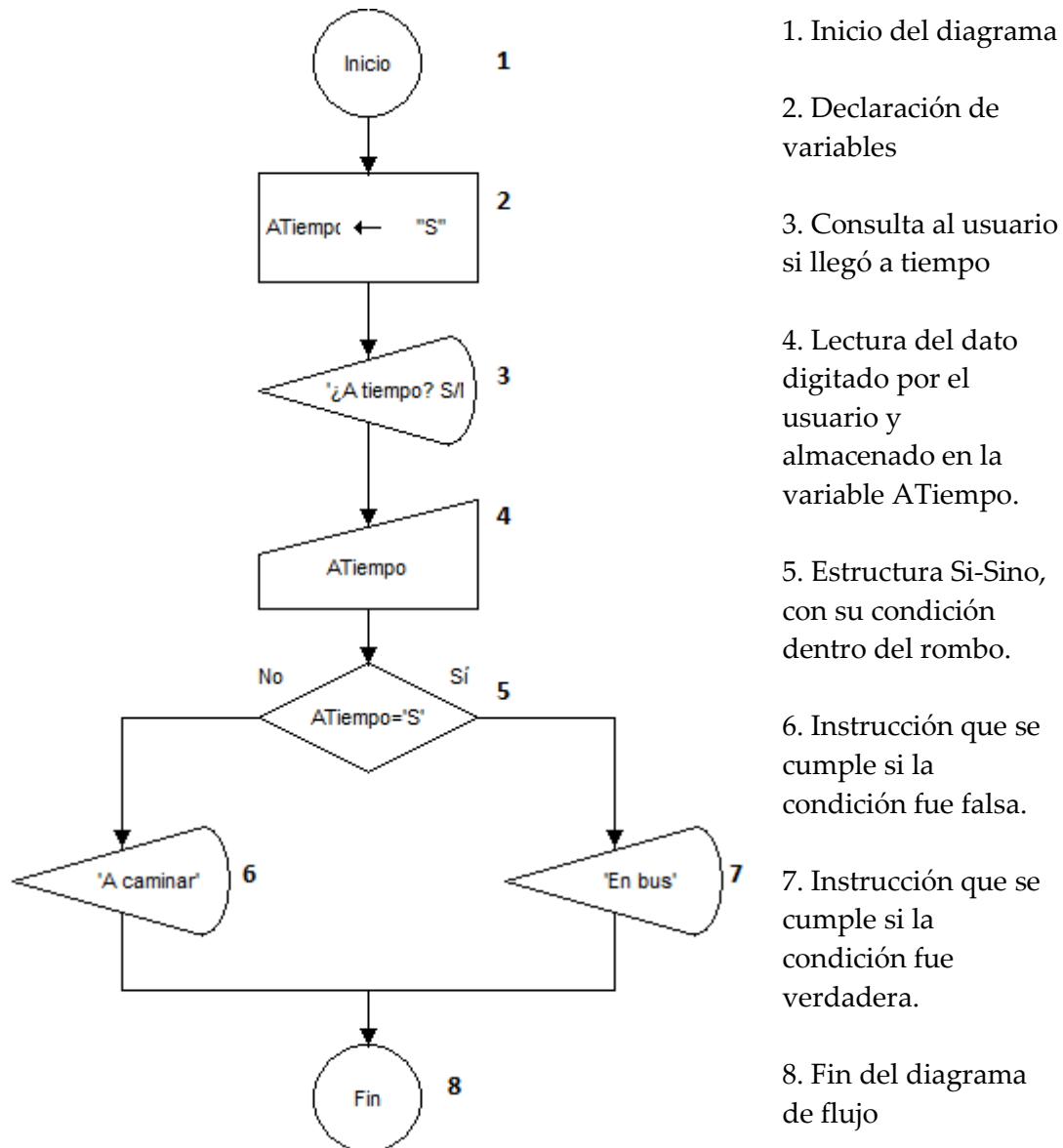


Figura 22 Diagrama de flujo para un recorrido en bus o caminando

Ejemplo 8 División y determinar si el cociente es múltiplo de tres y de cinco

Ahora trabajaremos el algoritmo que divide dos números, recordemos que se debe validar que el divisor sea diferente de cero y que analizaremos el cociente obtenido para determinar si el cociente es múltiplo de tres y múltiplo de cinco.

Algoritmo Division

Definir Dividendo, Divisor, Cociente Como Entero;

Dividendo=0;

Divisor=0;

Cociente=0;

Escribir "Digite el dividendo.";

Leer Dividendo;

Escribir "Digite el divisor.";

Leer Divisor;

Si Divisor=0 entonces

Escribir "Error, el divisor NO puede ser cero.";

Sino

Cociente= Dividendo/Divisor;

Escribir "Cociente es igual a: ",Cociente;

FinSi

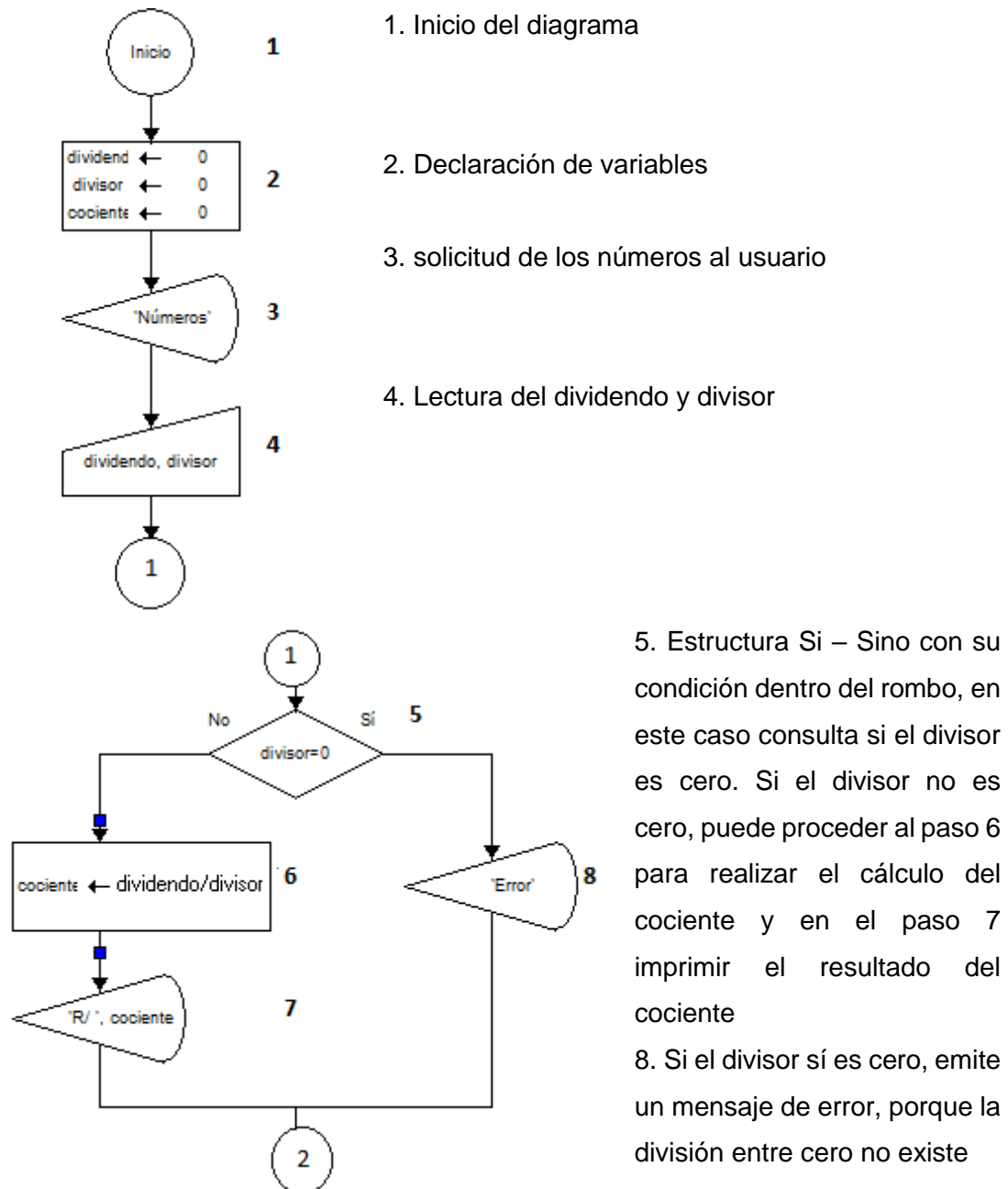
Si ((Cociente Mod 3=0) Y (Cociente Mod 5=0) Y (Divisor!=0)) Entonces

Escribir "El cociente ", Cociente," es múltiplo de 3 y 5 a la vez.";

FinSi

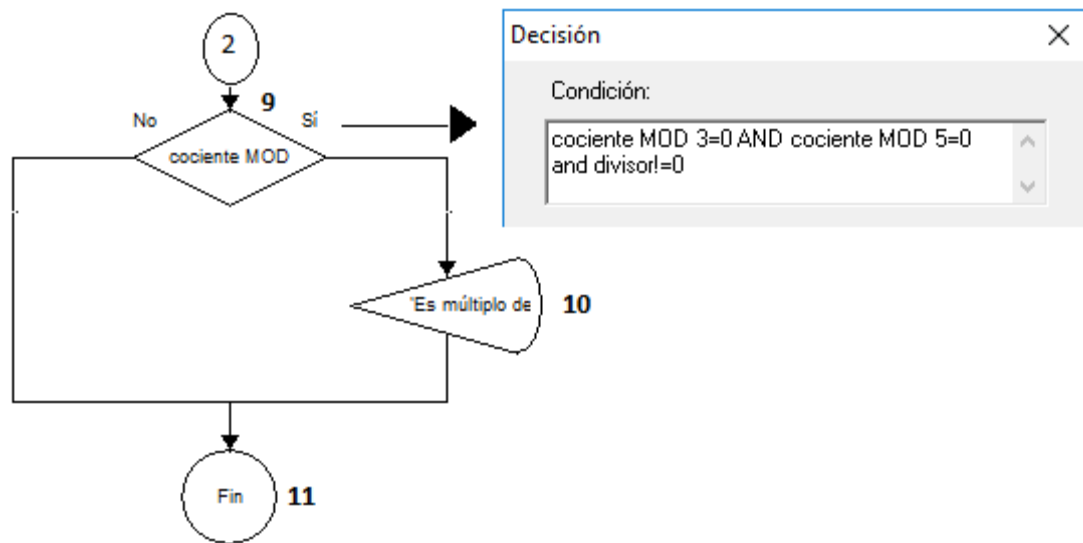
FinAlgoritmo

Al analizar el ejemplo anterior podemos observar que la estructura Si-Sino evalúa si el divisor es igual a cero, si esto ocurre se muestra el mensaje de error, pero sino se realiza la división. Además, al comparar este algoritmo con el de la estructura Si simple se denota que la única diferencia está en el uso del Si-Sino, en cuanto a variables, solicitud y lectura de datos y la última estructura Si, ambos pseudocódigos son iguales. El ejemplo anterior se representa en diagrama de flujo de la forma que se muestra en la Figura 23.



Continúa...

Continuación...



9. Después se evalúa si los cocientes son múltiplos de 3 y de 5. Si lo son entonces se continúa con el paso 10 para indicar que sí son múltiplos. Observe que para que ello se dé, el divisor debe ser distinto de cero.

11. Fin del diagrama.

Figura 23 Diagrama de flujo para la división de números con Si-Sino

Estructura Si anidados

Como se indicó anteriormente, la estructura “Si” permite evaluar una o varias condiciones y determinar si se cumplen o no.

En el caso de los “Si” anidados, estas condiciones podrían tener varios valores que al evaluarlos nos podrían brindar diferentes posibilidades de respuesta. Podríamos verlo como una especie de “menú” donde sino se cumple una condición evaluamos la siguiente y la siguiente hasta que encontremos lo que el usuario desea realizar, en un contexto con opciones finitas.

Es importante considerar, que entre más se “aniden” los “Si” podría ser más compleja su implementación, sobre todo si existen más de tres niveles del “Sino”, por lo que se recomienda emplearlos en contextos donde no hayan más de cuatro niveles, dado que siempre podremos emplear los “Si” – “Sino”, sin necesidad de anidarlos.

La estructura para su uso sería la siguiente:

Si (Condición) entonces

Instrucción...

Sino

Si (*Condición*) **entonces**

Instrucción...

Sino

Instrucción...

FinSi

FinSi

Observe que por cada “Si” utilizado, existe un “FinSi” que cierre el bloque de instrucciones. A continuación, analizaremos el ejemplo del menor de tres números, pero anidando las condiciones.

Ejemplo 9 Menor de tres números.

El pseudocódigo recibe por teclado tres números y determina cuál es el menor entre ellos. Asuma que todos los números introducidos por el usuario son diferentes y no poseen decimales.

```
Proceso MenorTresNumeros
    Definir N1, N2, N3 Como Entero;
    N1=0;
    N2=0;
    N3=0;
    Escribir "Escriba tres números diferentes";
    Leer N1, N2, N3;
    Si (N1<N2) Y (N1<N3) Entonces //valora si el primero es el menor
        Escribir "El ", N1, " es el menor";
    Sino
        Si (N2<N1) Y (N2<N3) Entonces //valora si el segundo es el menor
            Escribir "El ", N2, " es el menor";
        Sino //sino se cumplen las condiciones anteriores, entonces el tercero es el menor
            Escribir "El ", N3, " es el menor";
        Fin Si
    Fin Si
FinProceso
```

Figura 24 Pseudocódigo para determinar el menor de tres números con “Si” anidados

Observe que en la Figura 24, las condiciones siempre mantienen la estructura del “Si” y que el nivel de “anidamiento” entre las condiciones permite que el programa sea bastante legible, dado que solo podían existir tres escenarios: o el primer número era mayor, o el segundo era mayor y si se descartaban las dos opciones anteriores, entonces el tercero sería el mayor.

Ejemplo 10 Venta de entradas al cine.

El pseudocódigo recibe por teclado una opción, donde se establece si la persona es menor o mayor de edad. Si es menor de edad solo pueden vendersele entradas a la

sala regular, si es mayor de edad se pueden elegir entradas a la sala regular o a la VIP. El precio de la sala regular es de ₡2500 y el de la VIP de ₡5500

```
1  Proceso VentaEntradasCine
2      Definir tipoCliente, TipoSala Como Caracter;
3      Definir salaReg, salaVIP, cantEntradas, tot Como Entero;
4      salaReg=2500;
5      salaVIP=5500;
6      Escribir 'Tipo de cliente (m=menor edad, M=Mayor Edad) '
7      Leer tipoCliente;
8      Si tipoCliente='m' Entonces
9          Escribir '¿Cuántas entradas desea?'
10         Leer cantEntradas;
11         tot=cantEntradas*salaReg;
12         Escribir 'El total a pagar es: ', tot;
13     Sino //Se asume que es mayor de edad porque ya se evaluó
14         //en la condición anterior a los menores de edad
15         Escribir '¿Tipo de sala a la que desea asistir?';
16         Leer TipoSala;
17         Si TipoSala='R' Entonces
18             Escribir '¿Cuántas entradas desea?'
19             Leer cantEntradas;
20             tot=cantEntradas*salaReg;
21             Escribir 'El total a pagar es: ', tot;
22         Sino //Sino es a sala regular, entonces es la sala VIP
23             Escribir '¿Cuántas entradas desea?'
24             Leer cantEntradas;
25             tot=cantEntradas*salaVIP;
26             Escribir 'El total a pagar es: ', tot;
27         Fin Si
28     Fin Si
29 FinProceso
```

Figura 25 Pseudocódigo para vender entradas al cine con “Si” anidados

Observe en la Figura 25 que existen dos anidamientos. El primero es para determinar si el cliente es menor o mayor de edad, esto se determina en la línea 8 y sino es así la única otra opción es la línea 13 donde se asume que es mayor de edad, dado que el contexto del problema no plantea otra situación, también observe que en el caso de este primer “anidamiento” de instrucciones la línea 28 hace el cierre de estas.

Ahora, si el usuario es menor de edad note que se realizan las instrucciones de la 9 a la 12 porque a los niños solo se les puede vender entradas de sala regular.

En contraposición a los niños, observe que en el caso de los adultos existe a partir de la línea 15 y hasta la 27 otros “Si” para determinar si el cliente desea entradas de una sala regular (línea 17) o sino entradas a la sala VIP (línea 22) y los cálculos se basan en la escogencia de la sala. Considere que a partir de la línea 22 se asume que las entradas son VIP porque el contexto del problema solo presenta esas dos opciones,

Este documento forma parte de una unidad didáctica que está en desarrollo en la Universidad Estatal a Distancia de Costa Rica. Su contenido se encuentra bajo la ley de Propiedad Intelectual y cualquier mención a este debe ser indicado como obra en proceso.

sin embargo, si se agregara un tercer tipo de sala como 3D o IMAX, se tendrían que crear más “Si” anidados para evaluar todos los escenarios y realizar los cálculos de la venta de entradas.

Ejemplo 11 Cálculos de perímetros y áreas de figuras planas

El pseudocódigo recibe por teclado una opción donde el usuario elige entre cuatro figuras geométricas: triángulo, cuadrado, rectángulo y rombo, para calcular su área y perímetro.

```
1  Proceso PerimetroArea
2  Definir opc Como Caracter;
3  Definir A, P como Real; //Área y Perímetro de la figura elegida
4  Definir L1,L2,L3,B,H Como Entero; //variables del triángulo, cuadrado y rectángulo
5  Definir d, D como Entero; //diagonales del rombo
6  Escribir 'Elija una opción: C=cuadrado, R=rombo, T=triángulo, r=rectángulo';
7  Leer opc;
8  Si opc='c' Entonces //cuadrado
9      Escribir 'Digite el lado';
10     Leer L1;
11     P=4*L1;
12     A=L1*L1; //el lado al cuadrado se representa L1*L1
13 Sino
14     Si opc='t' Entonces //triángulo
15         Escribir 'Digite los 3 lados, la base y altura';
16         Leer L1,L2,L3,B,H;
17         P=L1+L2+L3;
18         A=(B*H)/2;
19     Sino
20         Si opc='R' Entonces //rombo
21             Escribir 'Digite el lado, la diagonal menor y mayor';
22             Leer L1,d,D;
23             P=4*L1;
24             A=(d*D)/2;
25         Sino //rectángulo
26             Escribir 'Digite la base y altura';
27             Leer B,H;
28             P=(2*B) + (2*H);
29             A=B*H;
30         Fin Si
31     Fin Si
32 Fin Si
33 Escribir 'El perímetro de la figura es: ',P, ' y el área: ',A;
34 FinProceso
```

Figura 26 Pseudocódigo para calcular áreas y perímetros con “Si” anidados

Observe en la Figura 26 que existen tres anidamientos (línea 13, línea 19 y línea 25). El primer “Si” de la línea 8 es para determinar si el usuario eligió la figura del cuadrado y si es así realiza las solicitudes de datos y cálculos que abarcan las instrucciones de la 9 a la 12. Algo muy importante de resaltar es que en un pseudocódigo o diagrama las **fórmulas matemáticas hay que descomponerlas a su forma más simple de expresión**. Entonces, en el caso del cuadrado que área es: a^2 (área al cuadrado) la fórmula debe ser: $a*a$, si la fórmula tiene exponente, se deberá multiplicar la cantidad de veces que indica el exponente, por ejemplo: a^3 (“a” al cubo) es: $a*a*a$.

Para el caso del triángulo, se realizan las instrucciones de la línea 15 a la 18. Para el rombo, las instrucciones abarcan de la 21 a la 24 y finalmente como ya no existen más opciones, se asume que la última posible opción es la del rectángulo, haciendo así las instrucciones de la 26 a la 29. Más adelante veremos que inclusive en la última opción

debemos validar la opción registrada por el usuario, para ello emplearemos los ciclos, los cuales están abordados en el siguiente apartado de este capítulo.

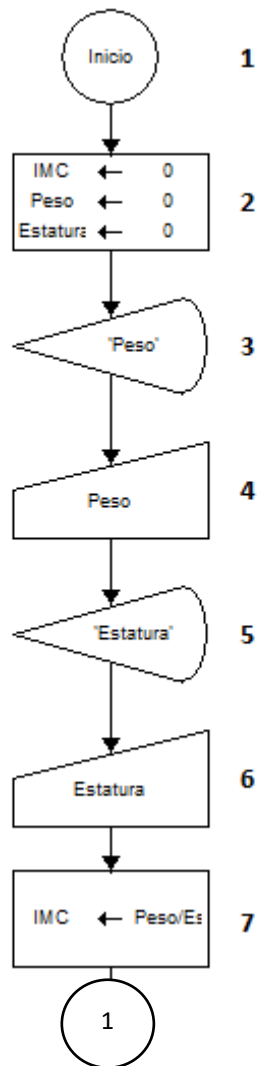
Nótese que solo en la línea 33 se muestra el resultado del perímetro y área, esto permite el ahorro de tres líneas de código para hacer el programa más **eficiente**.

Ejemplo 12 Cálculo del índice de masa corporal (IMC)

El diagrama de flujo recibe por teclado el peso (en kilos) y la estatura (en centímetros) y calcula el IMC, el cual se logra al dividir el peso entre la estatura e indique por medio de un mensaje, el estado de salud, según la tabla 6.

TABLA 6
ÍNDICE DE MASA CORPORAL

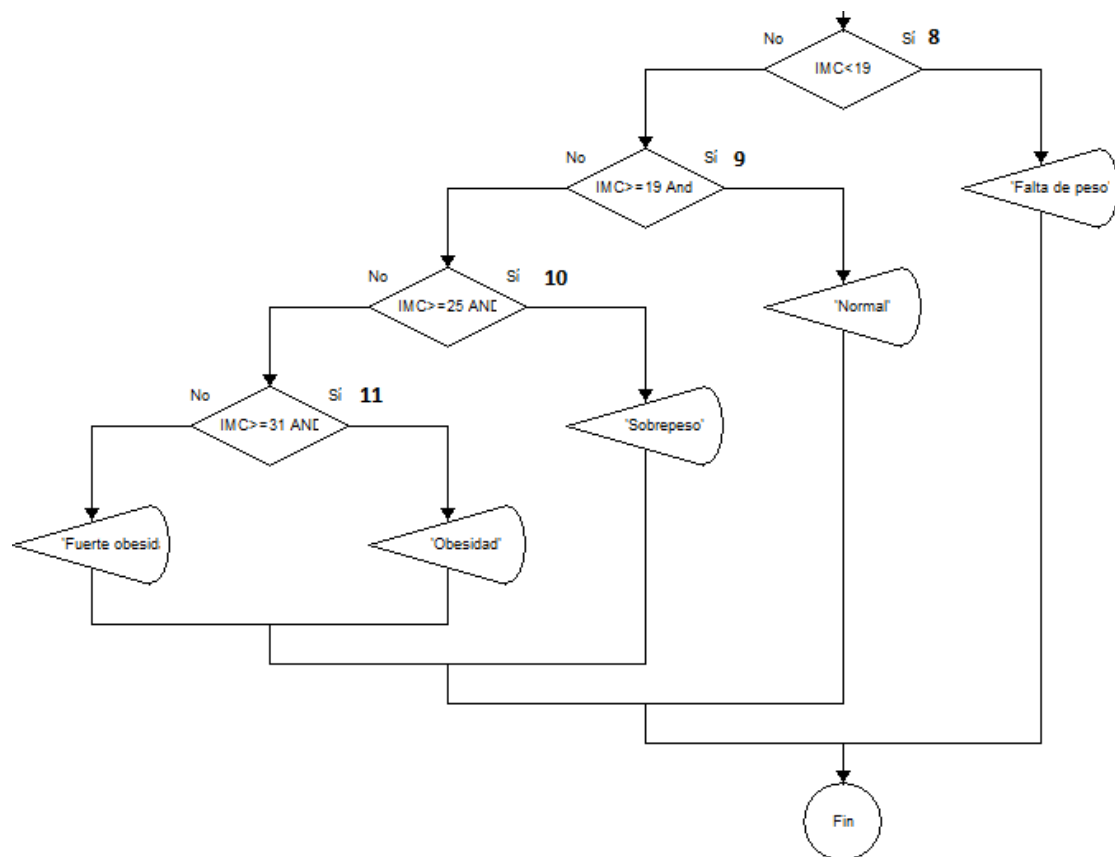
Mensaje	IMC
Falta de peso	por debajo de 19
Normal	19-24
Sobrepeso	25-30
Obesidad	31-40
Fuerte obesidad	mayor de 40



1. Inicio del algoritmo.
2. Inicialización de las variables que se requieren en el problema: la estatura, el peso y el IMC.
3. Mensaje al usuario para que indique el peso.
4. Lee desde teclado, el peso en kilogramos digitado por el usuario.
5. Mensaje al usuario para que digite la estatura en centímetros.
6. Lee desde teclado, la estatura en centímetros digitado por el usuario.
7. Cálculo del índice de masa corporal, según la fórmula:
 $IMC \leftarrow \text{Peso} / \text{Estatura}$

Continúa...

Continuación...



Decisión

×

Condición:

IMC < 19

Decisión

×

Condición:

IMC >= 19 And IMC <= 24

Decisión

×

Condición:

IMC >= 25 AND IMC <= 30

Decisión

×

Condición:

IMC >= 31 AND IMC <= 40

8. Si el IMC es menor a 19, envía el mensaje de Falta de peso.

9. Si el IMC está entre 19 y 24, envía el mensaje de Normal.

10. Si el IMC está entre 25 y 30, envía el mensaje de Normal.

11. Si el IMC está entre 31 y 40, envía el mensaje de Obesidad, pero si es mayor a 41, envía el mensaje de Fuerte obesidad.

Figura 27 Diagrama de flujo para determinar el índice de masa corporal

Estructura Según (Switch - case)

La instrucción Switch se utiliza para seleccionar una de entre múltiples alternativas, es como un menú donde el usuario puede elegir una de varias posibles opciones. Tiene un comportamiento similar a la instrucción "Si", pero con una sintaxis diferente. La diferencia más significativa entre ambas estructuras es que mientras el "Si" valora una condición (simple o compuesta) el Switch compara un valor introducido por el usuario.

El Switch es especialmente útil cuando la selección se basa en el valor de una variable de un tipo simple, la variable que se evaluará deberá ser: entero, carácter o boolean; no se recomienda emplear las cadenas o los reales. La estructura es la siguiente:

Switch (*variable*)

<case1>: Instrucciones...

<case2>: Instrucciones...

<case3>: Instrucciones...

Default: Instrucciones...

End Switch

De la estructura anterior es importante anotar:

- Switch: indica que a partir de ese punto, se emplea una estructura de selección múltiple (como una especie de menú).
- Variable: es una variable de tipo entero, carácter o boolean, que le indicará al "menú" cuál caso debe ejecutar. Cuando finalizan las instrucciones, el programa sigue hasta el End Switch y continúa con el resto de las sentencias.
- <case1>: son los casos u opciones del "menú" que se escogerán de acuerdo con el valor que tiene el selector.
- Instrucciones: son la secuencia de acciones que se seguirán (solicitar valores, realizar cálculos, tomar decisiones a partir de condiciones, usar ciclos, etc) según la opción elegida.
- Default: si el usuario no digita ninguna de las opciones del "menú" (por ejemplo, elige 5) se realizarán las instrucciones que indica esta sección.
- End Switch: es el fin de la estructura.

Es importante anotar que, en el caso del pseudocódigo, la estructura en PSeInt se denomina "Según", pero para diagramas de flujo de datos en el DFD no hay una

simbología que la represente, lo más similar sería el “Si”. Para el caso de PSeInt, la sintaxis sería la siguiente:

Según (variable) hacer:

<valor1>: Instrucciones...

<valor2>: Instrucciones...

De otro modo: Instrucciones...

Fin Según

Ejemplo 13 Día laboral y fin de semana

El pseudocódigo recibe por teclado un número del 1 al 7 (día de la semana) si el día se encuentra entre 1 a 5 (lunes a viernes) se emite un mensaje de que es un día laboral, sino si se encuentra entre 6 y 7 se indica que es fin de semana. Si el número no está entre 1 y 7, envía un mensaje de error.

```
1  Proceso Semana
2      Definir dia como entero;
3      Escribir 'Escriba un número del 1 al 7';
4      Leer dia
5      Según dia Hacer
6          1,2,3,4,5:
7              Escribir 'Es un día laboral entre semana';
8          6,7:
9              Escribir 'Es fin de semana';
10         De otro modo:
11             Escribir 'Error, el número no está entre 1 y 7';
12     FinSegun
13 FinProceso
```

Figura 28 Pseudocódigo para determinar el día laboral o fin de semana con Switch

Observe que en la figura 26, la línea 6 se interpreta como un conjunto de condiciones concatenadas por medio de la coma, como un “o”. Si el usuario digita 1 ó 2 ó 3 ó 4 ó 5 entonces el mensaje es que es un día laboral, porque corresponde a un día de lunes a viernes. Si esta instrucción se desarrollara en un “Si”, la estructura sería la siguiente:

Si (dia =1) o (dia =2) o (dia =3) o (dia =4) o (dia =5) entonces

Otra forma de representarlo por medio de un “Si”, sería:

Si (dia >=1) Y (dia <=5) entonces

Nótese que, en este caso, la estructura Switch simplifica la cantidad de instrucciones que se realizarían si se empleara un “Si”, aunque ambas resuelven el problema. Ahora, esto se puede hacer porque únicamente se puede elegir una opción, pero si tuviéramos que hacer un intervalo de valores o condiciones, la estructura Switch no se podría utilizar y se deberá emplear la estructura “Si”.

La línea 8 responde a la misma lógica anterior, si el usuario digita 6 ó 7, entonces se envía un mensaje de que es fin de semana.

La línea 10 permite validar que el número ingresado por el usuario esté en un rango entre 1 y 7, porque si no, no ejecutará ninguna acción y enviará un mensaje de error. Esta instrucción siempre se debe colocar, no es optativa.

Finalmente, la instrucción de la línea 12 cierra el rango de instrucciones de esta estructura.

Ejemplo 14 Nombre de colores y características de estos

El pseudocódigo recibe por teclado una letra ('a', 'b', 'v', 'r', 'n', 'm') y brinda el nombre del color y una característica de este. Sino elige ninguna de las letras, se emite un mensaje donde se indique que no está el color seleccionado.

```
1 Proceso Colores
2   Definir letra como caracter;
3   Escribir 'Escriba una letra y descubre tu color: ';
4   Escribir 'A, B, V, R, N, M';
5   Leer letra
6   Segun letra Hacer
7       'A', 'a': Escribir 'Azul: autoridad y confianza';
8       'B', 'b': Escribir 'Blanco: pureza y bondad';
9       'V', 'v': Escribir 'Verde: ecología y salud';
10      'R', 'r': Escribir 'Rojo: amor y pasión';
11      'N', 'n': Escribir 'Negro: elegancia';
12      'M', 'm': Escribir 'Morado: realeza y lujo';
13      De otro modo:
14          Escribir 'Lo siento, no está ese color';
15  FinSegun
16 FinProceso
```

Figura 29 Pseudocódigo para determinar colores y características de estos con Switch

Observe en la figura 29, varias situaciones:

- En la línea 2 se declara la variable letra, de tipo de carácter, la cual es empleada en la línea 5 para leer de teclado, la opción digitada por el usuario y se utiliza en la línea 6 para que el Switch parta de esta elección para determinar cuál color mostrar.
- La línea 7 indica que si el usuario escribió una letra “A” mayúscula o una letra “a” minúscula se entiende como una **validación** donde ambas letras representan el color “Azul”. Lo mismo sucede con las siguientes líneas de la 8 a la 12, tanto la letra minúscula como mayúscula representan un color.
- En la línea 13 se realiza otra **validación** donde, es emitido un mensaje de error (línea 14) donde se indica que el color no se encuentra, sino corresponde a ninguna de las letras anteriores.
- La línea 15 cierra la estructura del Según.
- La línea 16 cierra el pseudocódigo del algoritmo.

Ejemplo 15 Venta de entradas al cine con Switch

El pseudocódigo recibe por teclado una opción, donde se establece si la persona es menor o mayor de edad. Si es menor de edad solo pueden vendersele entradas a la sala regular, si es mayor de edad se pueden elegir entradas a la sala regular o a la VIP. El precio de la sala regular es de ₡2500 y el de la VIP de ₡5500.

```
1  Proceso VentaEntradasCine
2      Definir tipoCliente, TipoSala Como Caracter;
3      Definir salaReg, salaVIP, cantEntradas, tot Como Entero;
4      salaReg=2500;
5      salaVIP=5500;
6      Escribir 'Tipo de cliente (m=menor, M=Mayor)';
7      Leer tipoCliente;
8      Segun tipoCliente Hacer
9          'm':
10             Escribir '¿Cuántas entradas desea?';
11             Leer cantEntradas;
12             tot=cantEntradas*salaReg;
13             Escribir 'El total a pagar es: ', tot;
14          'M':
15             Escribir '¿Tipo de sala que desea?';
16             Leer TipoSala;
17             Segun TipoSala Hacer
18                 'R','r':
19                     Escribir '¿Cuántas entradas?';
20                     Leer cantEntradas;
21                     tot=cantEntradas*salaReg;
22                     Escribir 'El total a pagar es: ', tot;
23                 'V','v':
24                     Escribir '¿Cuántas entradas?';
25                     Leer cantEntradas;
26                     tot=cantEntradas*salaVIP;
27                     Escribir 'El total a pagar es: ', tot;
28             De Otro Modo:
29                 Escribir 'Error, la sala no existe';
30             Fin Segun
31          De Otro Modo:
32              Escribir 'Error, el tipo de cliente no existe';
33      Fin Segun
34  FinProceso
```

Figura 30 Pseudocódigo para ventas de entradas de cine con Switch

Analicemos la solución:

- En la línea 2 se declaran las variables del tipo de sala y cliente para asignar a cada variable, la elección del usuario. A diferencia del ejemplo de los colores, en este pseudocódigo no podemos validar para el **tipo de cliente** la “m” mayúscula y minúscula, puesto que la “M” mayúscula se emplea para la venta de entradas a los mayores de edad (línea 14) y la “m” minúscula para los menores de edad (línea 9).
- En el caso de la venta de entradas a los adultos, se pueden vender entradas de sala regular (línea 18) como de sala VIP (línea 23) y en estos casos sí se puede validar que la opción digitada sea mayúscula o minúscula puesto que tanto la “R” como la “V” no se emplean en otra parte del problema en mayúscula o minúscula.
- En el caso de la declaración de variables de la línea 3, todas son de tipo entero puesto que el contexto del problema no indica que los montos a cobrar tengan decimales, además que la venta de entradas siempre se expresará por cantidades enteras, no hay una venta 2,5 entradas, o son 2 o 3 los asientos de la sala regular o la sala VIP que se venden para una película. Así mismo, como los precios de las salas no tienen decimales, el total de la venta (tot) no requiere que se declare como Real.
- En las líneas 4 y 5 los valores de las salas regulares (salaReg) y de las salas VIP (salaVIP) se asignan desde el principio para luego utilizar estas variables en los cálculos. Esto con la finalidad de que, si cambian a futuro los valores, no se deba ir a la línea 12, línea 21 y línea 26 a modificar este dato. Esto permite que el pseudocódigo sea más eficiente y utilizar correctamente las constantes del problema.



ESTRUCTURAS DE REPETICIÓN

Estructuras de repetición (Ciclos, Bucles o Loops)

Las estructuras de repetición tienen como fin repetir las instrucciones que tienen dentro de si mismas las veces que sea necesario según una condición dada. A estas estructuras se les llama ciclos, bucles o loops (en inglés).

Los ciclos poseen las siguientes características:

- Tienen inicio y fin (son finitos).
- Poseen una condición que determina la cantidad de veces que se repiten.
- Deben contener al menos una instrucción.

Existen diferentes tipos de ciclos, cada uno con características que los hacen únicos, pero todos comparten el principio de funcionamiento, repetir una o varias instrucciones. Los bucles que estudiaremos serán los siguientes:

- Mientras. (While)
- Repetir-Hasta. (Repeat-Until)
- Para. (For)

Estructura Mientras (While)

El ciclo Mientras (While) tiene la condición al inicio de su estructura, la misma es de la siguiente forma:

Mientras (Condición) Hacer

Instrucción...

Instrucción...

Instrucción...

Fin Mientras

El ciclo tiene una condición que controla la secuencia de repetición de las instrucciones. Esta condición se evalúa al inicio del bucle y por ello, primero se valida la condición y si la condición es verdadera, entonces se ejecutan las instrucciones que están dentro de éste.

Las instrucciones del bucle se ejecutan mientras se cumpla la condición de control. Si la condición es falsa, entonces las instrucciones no se ejecutan.

Es muy importante considerar que para que el ciclo tenga un fin, **la condición debe volverse a consultar dentro del ciclo** y antes de que finalice el ciclo porque si la condición se evalúa fuera del ciclo, el programa siempre se ejecutará y no se podrá

detener, a esto se le denomina que el programa se **“encicló”** y es un error común cuando resolvemos un algoritmo cuando no prestamos atención a este detalle.

En este ciclo, no se sabe cuántas veces se realizarán las instrucciones porque depende de un cálculo que se puede ejecutar “n” veces o de que un usuario desee hacer una operación las veces que desee, es decir, la cantidad de veces que se ejecutan las instrucciones no está determinada en la sección de la *condición* del ciclo.

Ejemplo 16 Saludar mientras el usuario digite ‘S’

El pseudocódigo recibe por teclado una letra ‘s’ o ‘S’ para enviar un mensaje de saludo al usuario, el programa se termina cuando se digite otra tecla.

```
1  Proceso Saludar
2      Definir nombre como Cadena;
3      Definir seguir Como Caracter;
4      seguir='s';
5      Escribir 'Escriba su nombre';
6      Leer nombre;
7      Mientras seguir='s' o seguir='S' Hacer
8          Escribir 'Hola ', nombre;
9          Escribir '¿Desea continuar? s=sí, n=no';
10         Leer seguir;
11     Fin Mientras
12 FinProceso
```

Figura 31 Pseudocódigo para saludar “n” veces

Analicemos el pseudocódigo de la figura 31:

- En la línea 2 y en la línea 3, se declaran las variables que se utilizarán en el programa.
- En la línea 4 se realiza un paso muy importante, que es la inicialización de la variable que tiene la condición del ciclo. Siempre se debe inicializar con el valor o condición para que ingrese el ciclo, sino se realiza de esta forma, nunca ingresará al ciclo.
- En las siguientes dos líneas, se solicita el nombre de la persona (línea 5) y se lee por teclado la información (línea 6).
- En la línea 7 inicia el ciclo, la condición es que mientras el usuario digite una “s” minúscula o una “S” mayúscula se enviará un mensaje de saludo a la persona.
- Las líneas 9 y 10 son cruciales para que el programa no se “encicle”, dado que consulta si la persona desea continuar con el saludo y lo lee sobre la misma variable que se emplea en la línea 7, antes de que se cierre el ciclo (línea 11).

Un error común es colocar otra variable en la línea 10 o colocarlo después de la línea 11, lo que implica que la condición no variaría y siempre se ejecutarían las instrucciones de las líneas 8, 9 y 10.

- En la línea 11 se cierra la estructura del ciclo.

Antes de continuar con los tipos de ciclos, introduciremos dos conceptos que se emplean frecuentemente con estas estructuras y son los **contadores** y los **acumuladores**.

Contadores

Un contador es una variable, generalmente de tipo entero, que acumula un valor de una serie de números o pasos. Por ejemplo, si deseamos contar cuántos recibos de agua pagamos al año, contaremos los meses desde enero hasta diciembre, por lo que el contador =12.

Otro ejemplo es si deseamos contar los segundos que tiene un minuto, entonces el contador =60. Para emplear un contador se requiere apenas de dos variables:

- La variable donde se guardará el conteo (contador).
- El número que incrementará el contador, en este caso sería el 1.

La estructura de un contador es la siguiente:

- **Contador=Contador+1**

Donde “Contador” irá sumando al valor anterior, una unidad para contar la cantidad de algo, por ejemplo: contar la cantidad de pares de zapatos que tenemos.

Veamos un ejemplo de un algoritmo sencillo para aplicar contadores.

Ejemplo 17 Contar la cantidad de veces que se digitó la letra “a”

El pseudocódigo recibe por teclado cualquier tecla y debe contar únicamente las veces que se digitó la letra “a” o “A”, por medio del uso de ciclos. El algoritmo finaliza cuando se introduce una “n” para finalizar el programa.

```
1  Proceso ContarLetra
2      Definir conta como entero;
3      Definir letra, seguir como caracter;
4      conta=0;
5      seguir='s';
6      Mientras seguir!='n' Hacer
7          Escribir 'Digite una letra';
8          Leer letra;
9          Si letra="a" o letra="A" Entonces
10             conta=conta+1;
11         Fin Si
12         Escribir '¿Desea continuar? n=no, s=sí'
13         Leer seguir
14     Fin Mientras
15     Escribir 'Se digitaron ', conta, ' letras a';
16 FinProceso
```

Figura 32 Pseudocódigo para contar las letras “a” o “A” digitadas

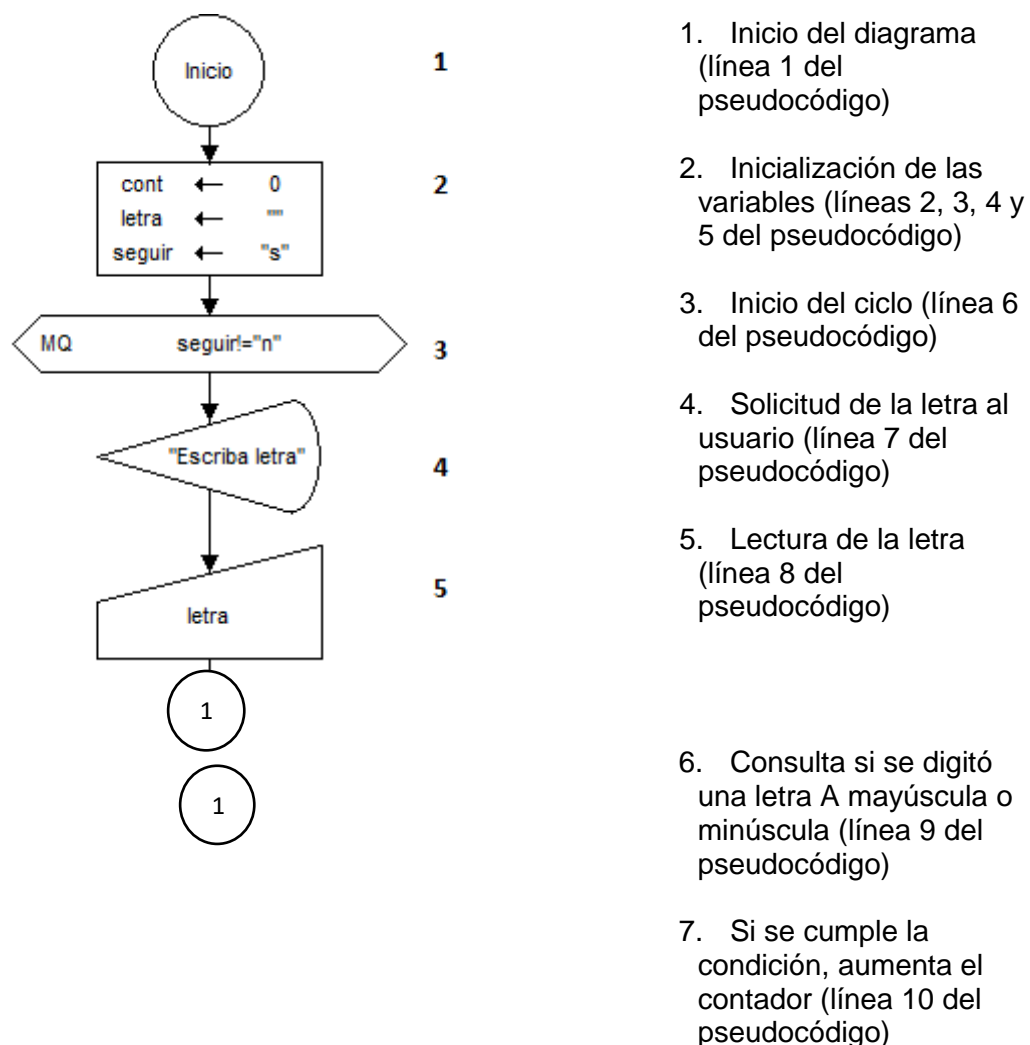
Analicemos el pseudocódigo de la figura 32:

- En la línea 2 y en la línea 3, se declaran las variables que se utilizarán en el programa.
- En la línea 4 se realiza un paso muy importante, que es la **inicialización del contador**, este siempre deberá empezar en **cero** porque si se le asigna uno, sumará al resultado final una “a” demás en el conteo de las digitadas por el usuario.
- En la línea 5, se inicializa la variable que tiene la condición del ciclo. Siempre se debe inicializar con el valor o condición para que ingrese al ciclo, sino se realiza de esta forma, nunca ingresará a las instrucciones.
- En la línea 6 se establece la condición para continuar con el ciclo, la cual es que el usuario digite una letra diferente de “n”, así continuará con las instrucciones de la 7 a la 13.
- En la línea 7 se le envía un mensaje al usuario para que digite una letra y en la línea 8 se lee desde teclado la letra introducida.
- En la línea 9 se establece la condición para saber si la letra se debe sumar, la cual es que se digite una “a” minúscula o una “A” mayúscula, si lo digitado por el usuario corresponde a esto entonces procede la línea 10, donde se suma

una unidad al contador, para llevar la suma de cuántas letras “a” se han digitado.

- Las líneas 12 y 13 son cruciales para que el programa no se “encicle”, dado que consulta si la persona desea continuar con el programa y lo lee sobre la misma variable (línea 13) que se emplea en la línea 6, antes de que se cierre el ciclo (línea 14). Un error común es colocar otra variable en la línea 13 o colocar ambas instrucciones después de la línea 14 (cuando ya no tienen relación con el ciclo), lo que implica que la condición no variaría de la inicialización que se le brindó en la línea 5 y siempre se ejecutarían las instrucciones que estén dentro del ciclo porque la condición nunca varió.
- En la línea 14 se cierra la estructura del ciclo.
- En la línea 15 se envía el mensaje con el conteo de letras “A” que se digitaron (incluye mayúsculas y minúsculas), por medio del uso del contador (conta).

Para el ejemplo anterior, la figura X muestra el diagrama de flujo de datos elaborado en DFD.



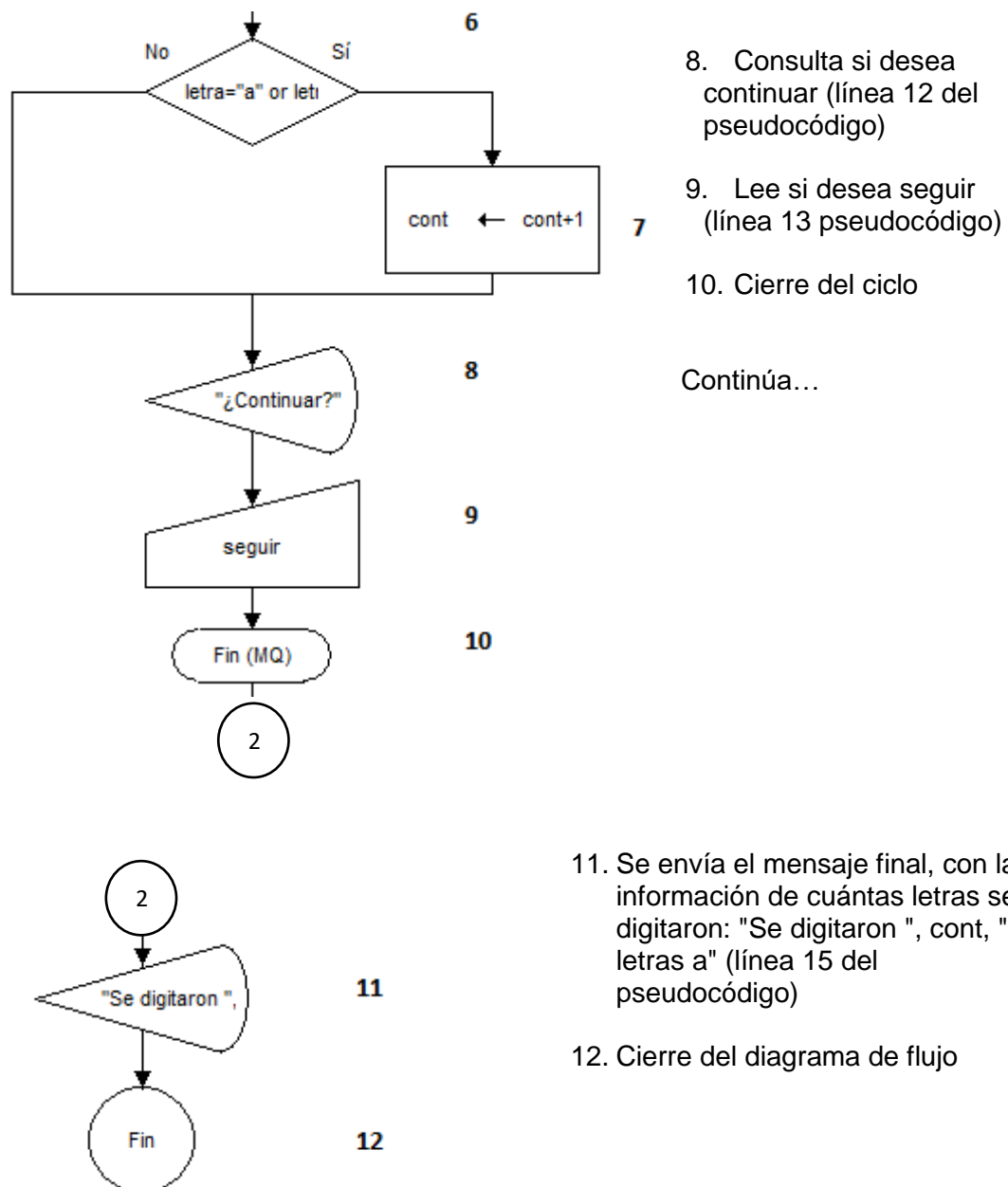


Figura 33 Diagrama de flujo para contar las letras "a" o "A" digitadas

Acumuladores

Un acumulador es una variable, generalmente de tipo entero o real, que acumula la suma de unos valores puede ser después de aplicar una fórmula o después de solicitarlos al usuario.

Un ejemplo de acumulador al emplear una fórmula es cuando deseamos saber el factorial de un número. Por ejemplo, el factorial de 5! Es: $1*2*3*4*5$

Un ejemplo de un acumulador de datos solicitados al usuario, es cuando este digita los montos de los recibos de electricidad de un año.

Para emplear un acumulador se requiere apenas de dos variables:

- La variable donde se guardará el valor (acumulador).
- El valor introducido por el usuario o por la fórmula.

La estructura de un contador es la siguiente:

- **Acumulador=Acumulador+valor**

Donde "Contador" irá sumando al valor anterior, una unidad para contar la cantidad de algo, por ejemplo: contar la cantidad de pares de zapatos que tenemos.

Veamos un par de ejemplos de algoritmos para aplicar acumuladores: calcular un factorial y el total de recibos telefónicos de un año.

Ejemplo 18 Factorial de un número

El pseudocódigo recibe por teclado cualquier número y el programa muestra el factorial de dicho valor. Suponiendo que el usuario digitó el valor “n”, la fórmula del factorial sería: $1*2*3*...*n-1*n$.

```
1  Proceso Factorial
2      Definir num, acum, cont Como Entero;
3      acum=1;
4      cont=1;
5      Escribir 'Digite el número';
6      Leer num;
7      Mientras cont<=num Hacer
8          acum=acum*cont;
9          cont=cont+1;
10     Fin Mientras
11     Escribir 'El factorial de ',num,' es ',acum;
12 FinProceso
```

Figura 34 Pseudocódigo para calcular el factorial de un número

Analicemos el pseudocódigo de la figura 34:

- En la línea 2 se declaran las variables que se utilizarán en el programa, en este caso se requieren un contador (cont) para llevar el conteo del 1 al 5 y un acumulador (acum) que guarde el producto de $1*2*3*4*5$.
- La línea 3 y la línea 4 inicializan las variables. Es importante recalcar que en este caso las variables no pueden iniciar en cero, porque se calcula el producto de números y todo número multiplicado por cero, siempre dará cero.
- En la línea 5, se le envía un mensaje al usuario, solicitando que digite el número al que se le calculará el factorial.
- En la línea 6 se lee el número al que se le calculará el factorial. En el ejemplo de la Tabla 7, se parte de que el número digitado por el usuario es el 5.
- En la línea 7, se establece la condición para ejecutar el ciclo y este es que mientras el contador (que irá creciendo de uno en uno) sea menor o igual al número introducido por el usuario (en este caso el 5) se realizará el cálculo del producto (línea 8) y se incrementará el contador (línea 9). Esto es para lograr el conteo desde 1 hasta 5 que se requiere para la multiplicación de $1*2*3*4*5$ que se almacenará en el acumulador (acum). Observemos la tabla 7 para comprender los valores que se emplean en el ciclo, suponiendo que el usuario digitó el número 5.

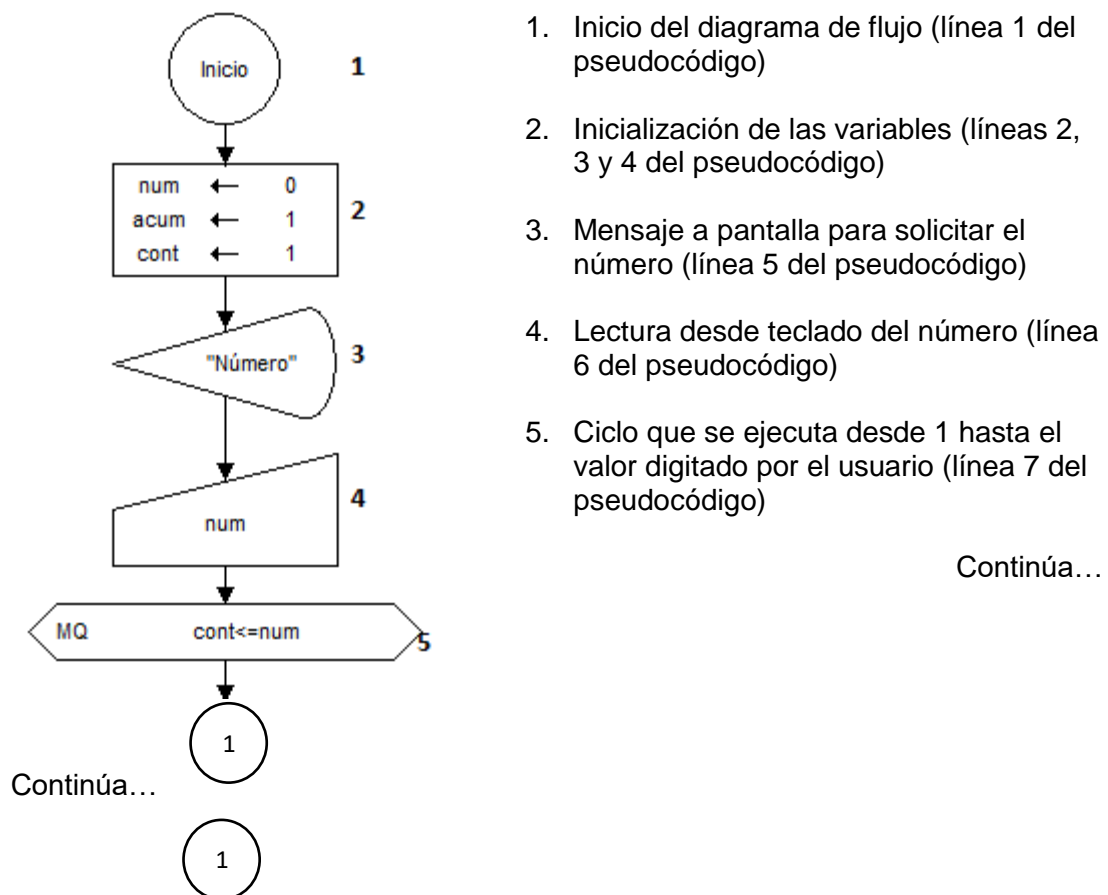
TABLA 7

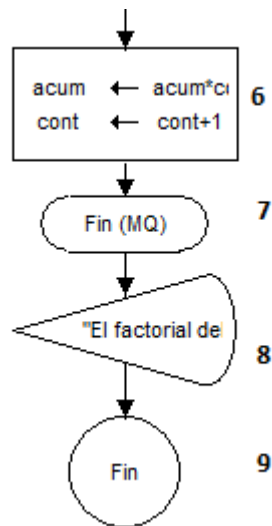
RECORRIDO DEL CICLO PARA EL CÁLCULO DEL FACTORIAL, LÍNEAS 7 A LA 9

Valor inicial:	acum	cont	Mientras cont <= num	acum=acum*cont	cont=cont+1
	1	1	Mientras 1<=5	acum=1*1 acum=1	cont=1+1 cont=2
	1	2	Mientras 2<=5	acum=1*2 acum=2	cont=2+1 cont=3
	2	3	Mientras 3<=5	acum=2*3 acum=6	cont=3+1 cont=4
	6	4	Mientras 4<=5	acum=6*4 acum=24	cont=4+1 cont=5
	24	5	Mientras 5<=5	acum=24*5 acum=120	cont=5+1 cont=6
	120	6	Mientras 6<=5	Ya no ingresa a las instrucciones porque no se cumple la condición de que 6 sea menor a 5	

- En la línea 10 se cierra la estructura del ciclo.
- En la línea 11 se envía el mensaje con el factorial del número. En este caso, partiendo de que el número sea el 5, el resultado del factorial será 120.

El ejemplo anterior en diagrama de flujo de datos, se observa en la Figura 35.





6. Cálculo del factorial e incremento del contador (líneas 8 y 9 del pseudocódigo)
7. Cierre del ciclo (línea 10 del pseudocódigo)
8. Se envía el mensaje con el cálculo del factorial del número: "El factorial del número ", num, " es: ", acum (línea 11 del pseudocódigo)
9. Fin del diagrama de flujo

Figura 35 Diagrama de flujo de datos para calcular el factorial de un número

Ejemplo 19 Suma de recibos telefónicos y promedio de pago

El pseudocódigo recibe por teclado los montos de los 12 recibos telefónicos que pagó el usuario y el programa muestra al final el total de los pagos del año y el promedio de pago de los recibos.

```
1  Proceso RecibosTelef
2      Definir cont Como Entero;
3      Definir acum, num, prom como real;
4      acum=0;
5      cont=1;
6      Mientras cont<=12 Hacer
7          Escribir 'Digite el monto del recibo';
8          Leer num;
9          acum=acum+num;
10         cont=cont+1;
11     Fin Mientras
12     prom=acum/12;
13     Escribir 'Total pagos: ',acum,' promedio: ',prom;
14 FinProceso
```

Figura 36 Pseudocódigo que suma los pagos de un año de recibos y calcula el promedio mensual

Analicemos el pseudocódigo de la figura 36:

- En la línea 2 y en la línea 3 se declaran las variables que se utilizarán en el programa, en este caso se requieren un contador (cont) que servirá para repetir 12 veces el programa (porque un año tiene 12 meses y cada mes tiene un recibo telefónico). En la línea 3 se encuentra el acumulador (acum) donde se almacenará el valor de los 12 recibos telefónicos, el número (num) es el monto que la persona digita en cada recibo y el promedio (prom) es una fórmula que **debe realizarse fuera del ciclo**, para calcular el promedio de pagos de cada mes. Por ejemplo, si el acum = ₡120 000 el promedio de los pagos debe ser de ₡10 000 mensuales.
- La línea 4 y la línea 5 inicializan las variables. En el caso del acumulador (acum) este debe iniciar en cero porque no deben agregarse ningún monto a los recibos telefónicos y en el caso del contador (cont) este debe iniciar en 1 para contar desde el 1 y hasta el 12, los recibos telefónicos.
- En la línea 6 se inicia el ciclo, el cual solicitará doce veces, que se digite el monto del recibo telefónico y se irá acumulando en la variable acum (línea 9). También se deberá incrementar el contador (línea 10) porque sino el programa “se encicla” y nunca se detendría en la petición del monto del recibo. En este

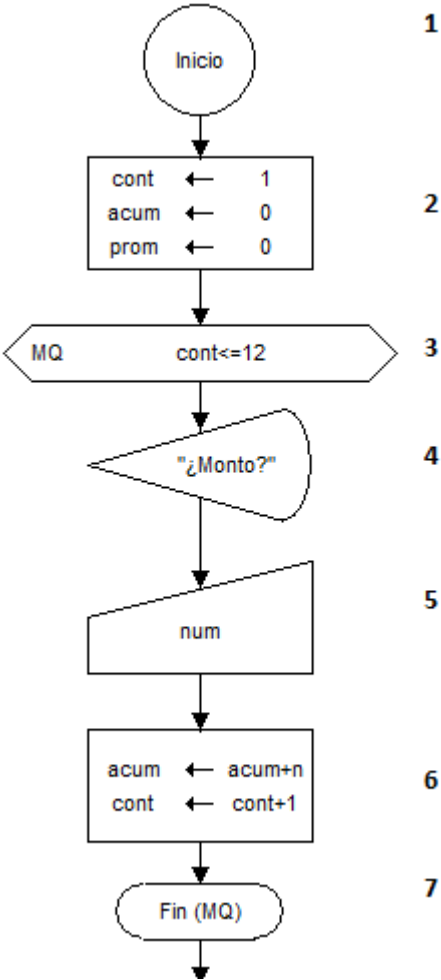
caso, el contador irá desde 1 hasta 12. Observemos la tabla 8 para comprender el proceso de conteo y acumulación de los montos de los recibos

TABLA 8
RECORRIDO DE LAS LÍNEAS DE LA 6 A LA 11 DEL CICLO

Valor inicial:	cont	Mientras cont <=12	num	acum=acum+num	cont=cont+1
	1	Mientras 1<=12	10500	acum=0+10500 acum=10500	cont=1+1 cont=2
	2	Mientras 2<=12	9800	acum=10500+9800 acum=20300	cont=2+1 cont=3
	3	Mientras 3<=12	8625	acum=20300+8625 acum=28925	cont=3+1 cont=4
	4	Mientras 4<=12	11200	acum=28925+11200 acum=40125	cont=4+1 cont=5
	5	Mientras 5<=12	7000	acum=40125+7000 acum=47125	cont=5+1 cont=6
	6	Mientras 6<=12	5500	acum=47125+5500 acum=52625	cont=6+1 cont=7
	7	Mientras 7<=12	8000	acum=52625+8000 acum=60625	cont=7+1 cont=8
	8	Mientras 8<=12	7375	acum=60625+7375 acum=68000	cont=8+1 cont=9
	9	Mientras 9<=12	18000	acum=68000+18000 acum=86000	cont=9+1 cont=10
	10	Mientras 10<=12	20000	acum=86000+20000 acum=106000	cont=10+1 cont=11
	11	Mientras 11<=12	15350	acum=106000+15350 acum=121350	cont=11+1 cont=12
	12	Mientras 12<=12	12900	acum=121350+12900 acum=134250	cont=12+1 cont=13
	13	Mientras 13<=12		Ya no ingresa a las instrucciones porque no se cumple la condición de que 13 sea menor o igual a 12	

- En la línea 11 se cierra la estructura del ciclo.
- En la línea 12 se calcula el promedio de los recibos. El promedio es la sumatoria de los montos dividido entre 12 (porque son 12 recibos telefónicos).
- En la línea 13 se envía el mensaje con el monto de los pagos y el promedio mensual realizado. En este caso, partiendo del ejemplo de la tabla 8, el monto total sería de ₡134250 y el promedio de pagos sería: ₡11187,5.

El diagrama de flujo de datos del ejemplo anterior, se observa en la figura 38.



1. Inicio del diagrama de flujo (línea 1 del pseudocódigo)
2. Inicialización de las variables (líneas 2, 3, 4 y 5 del pseudocódigo)
3. Inicio del ciclo, desde 1 hasta 12 (línea 6 del pseudocódigo)
4. Solicitud del monto del recibo al usuario (línea 7 del pseudocódigo)
5. Lectura desde teclado del monto del recibo telefónico (línea 8 del pseudocódigo)
6. Acumulador de los montos de los recibos e incremento del contador del ciclo (líneas 9 y 10 del pseudocódigo)
7. Cierre del ciclo (línea 11 del pseudocódigo)

Continúa...



8. Calcula el promedio de los pagos mensuales de los recibos (línea 12 del pseudocódigo)
9. Mensaje con el total de los pagos realizados en los 12 meses y el promedio de los pagos mensuales: "Monto de los pagos: ", acum, " promedio de los pagos mensuales: ", prom (línea 13 del pseudocódigo)

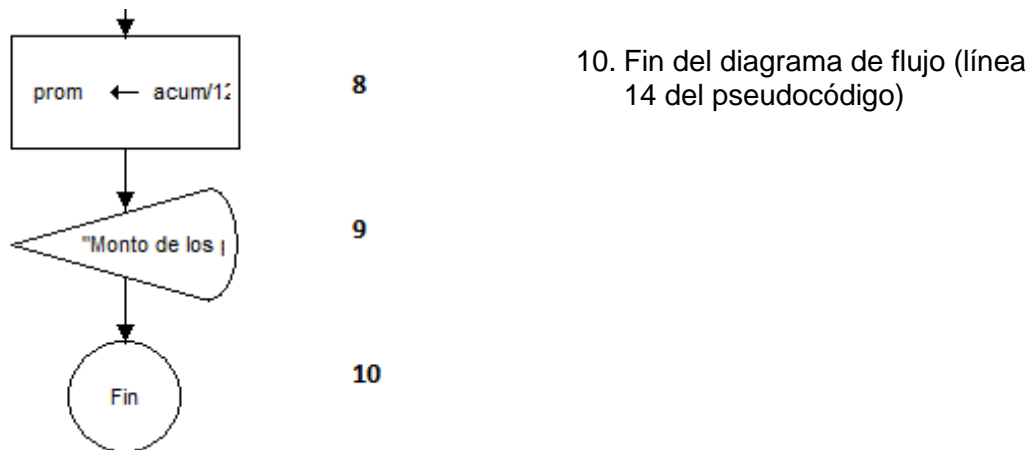


Figura 37 Diagrama de flujo que suma los pagos de un año de recibos y calcula el promedio mensual

Ejemplo 20 Elevar un número a un exponente con el ciclo while

El pseudocódigo recibe por teclado una base y un exponente al que se elevará la base. Al final debe emitir un mensaje con el resultado. Por ejemplo: 4^3 implica que 4 se multiplica por sí mismo, la cantidad de veces que indique el exponente, en este caso: $4*4*4$ y esto se almacena en un **acumulador**.

```

1  Proceso CicloExponente
2      Definir base, expo, contador, tot Como Entero;
3      base=1;
4      expo=1;
5      contador=1;
6      tot=1;
7      Escribir 'Escriba la base y el exponente';
8      Leer base, expo;
9      Mientras contador<=expo Hacer
10         tot=tot*base;
11         contador=contador+1;
12     Fin Mientras
13     Escribir 'El número ',base,' elevado a ',expo,' es ',tot;
14 FinProceso

```

Figura 38 Pseudocódigo para elevar una base a un exponente

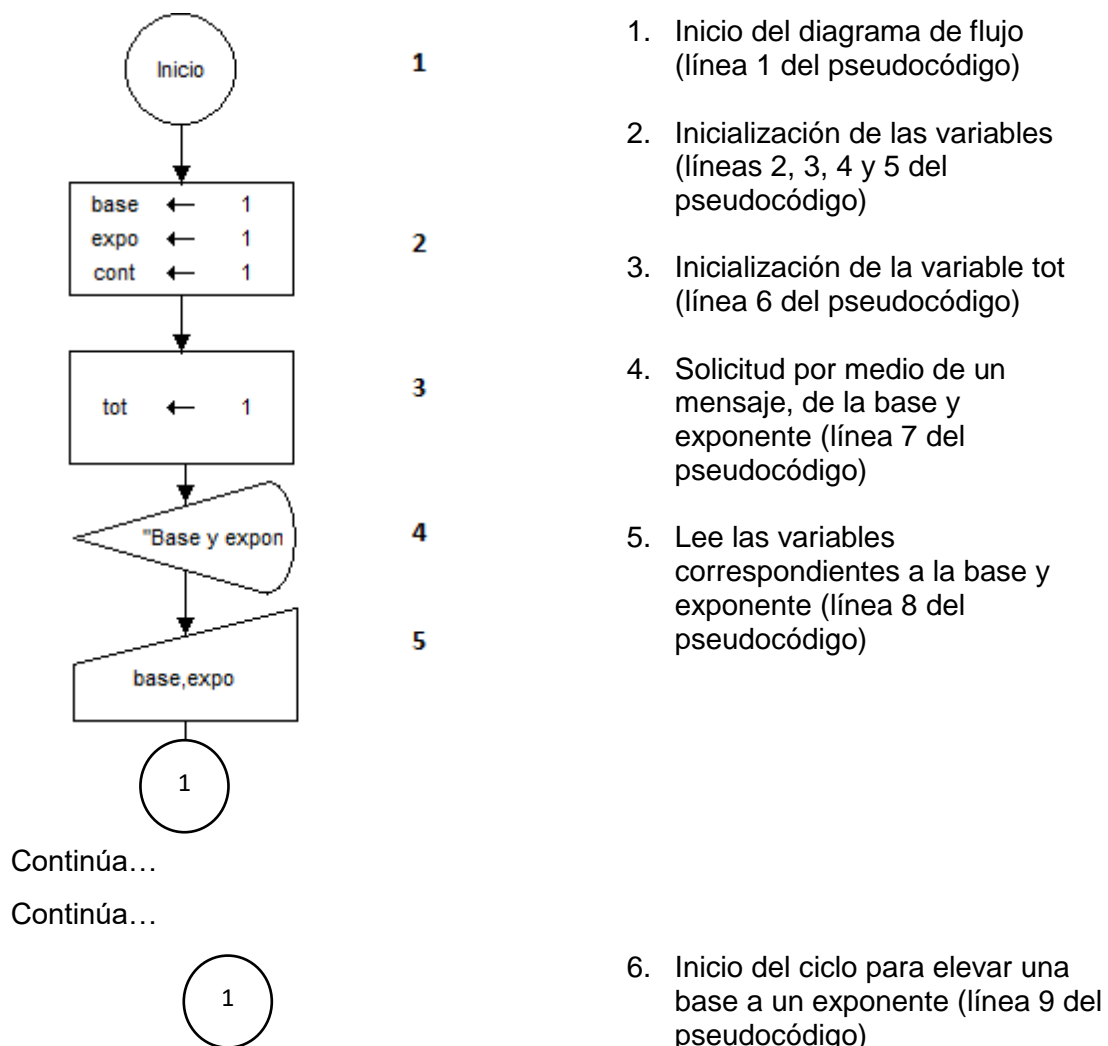
Analicemos el pseudocódigo de la figura 39:

- En la línea 2 y en la línea 3, se declaran las variables que se utilizarán en el programa.
- En la línea 4 se realiza un paso muy importante, que es la inicialización de la variable que tiene la condición del ciclo. Siempre se debe inicializar con el valor

o condición para que ingrese el ciclo, sino se realiza de esta forma, nunca ingresará al ciclo.

- En las siguientes dos líneas, se solicita el nombre de la persona (línea 5) y se lee por teclado la información (línea 6).
- En la línea 7 inicia el ciclo, la condición es que mientras el usuario digite una “s” minúscula o una “S” mayúscula se enviará un mensaje de saludo a la persona.
- Las líneas 9 y 10 son cruciales para que el programa no se “encicle”, dado que consulta si la persona desea continuar con el saludo y lo lee sobre la misma variable que se emplea en la línea 7, antes de que se cierre el ciclo (línea 11). Un error común es colocar otra variable en la línea 10 o colocarlo después de la línea 11, lo que implica que la condición no variaría y siempre se ejecutarían las instrucciones de las líneas 8, 9 y 10.
- En la línea 11 se cierra la estructura del ciclo.

El diagrama de flujo de datos del ejemplo anterior, se observa en la figura 40.



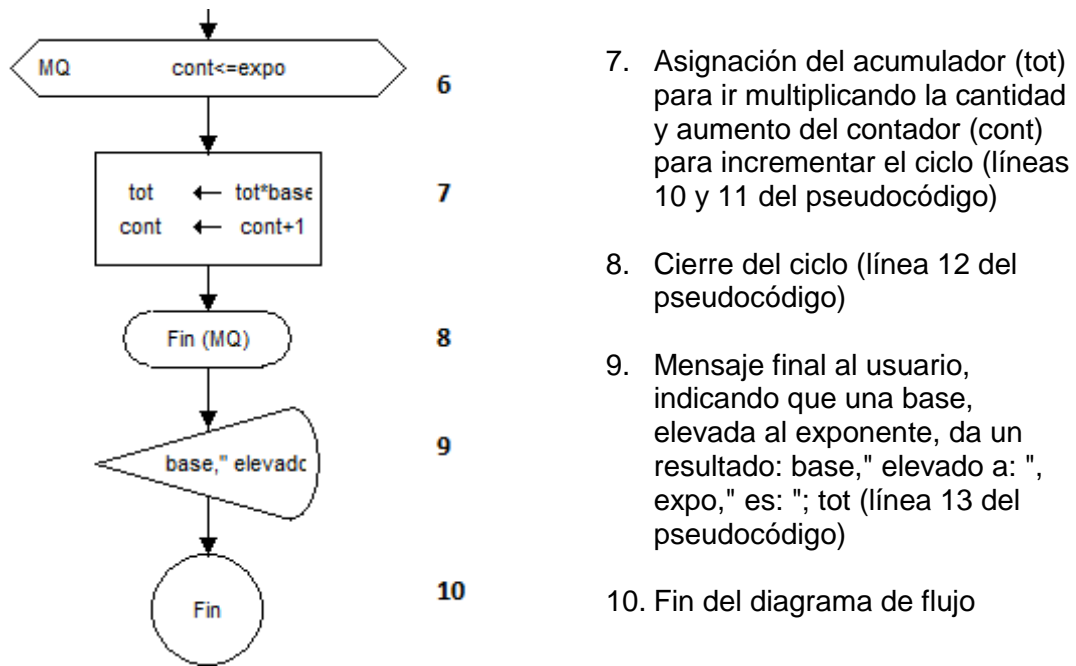


Figura 39 Diagrama de flujo de datos para elevar una base a un exponente

Importante

El ciclo Mientras, se utiliza generalmente cuando la condición de terminar el programa es dada por el usuario, por ejemplo, que digite una "n" para finalizar el bucle. También se puede usar cuando la condición está dada por números como en el caso del factorial o los recibos telefónicos.

Estructura Repetir– Hasta (Do – While)

El ciclo Repetir - Hasta (Do - While) repite una serie de instrucciones al menos una vez, mientras se cumpla la condición del Mientras. La condición la tiene hasta el final del ciclo, por ello, es que aunque esta no se cumpla, las instrucciones siempre se realizarán una vez porque el impedimento de ingresar o no, se evalúa hasta el final. La estructura de este ciclo es la siguiente:

Repetir

Instrucción...

Instrucción...

Instrucción...

Hasta que (condición)

El ciclo tiene una condición que controla la secuencia de repetición pero a partir de la segunda vez que se ejecutan el bloque de instrucciones, puesto que al tener la condición al final, siempre se ejecutará al menos una vez, aunque desde el principio no se cumpliera la condición para realizar las instrucciones.

La condición se evalúa hasta el final del bucle, entonces al finalizar el primer bloque de instrucciones se valida la condición y si la condición es verdadera, entonces se repiten las instrucciones que están dentro de éste. Al igual que en los otros ciclos, las instrucciones del bucle se ejecutan mientras se cumpla la condición de control. Si la condición es falsa, entonces las instrucciones no se ejecutan.

Para finalizar este ciclo, **la condición debe volverse a consultar dentro del bucle** y antes de que finalice este porque si no se enciclará y no se podrá detener.

Al igual que en el ciclo “Mientras” en este ciclo, no se sabe cuántas veces se realizarán las instrucciones, eso sí, se tiene la certeza de que se ejecutarán al menos una vez (la primera ocasión).

El bucle do-while también se le denomina post-prueba, ya que la expresión lógica se comprueba hasta después de que se han ejecutado todas las instrucciones que contiene el ciclo, por lo que se ejecutarán al menos una vez.

Nos podemos preguntar ¿qué tiene de conveniente que se ejecuten al menos una vez las instrucciones en un ciclo? Este ciclo tiene la particularidad de que permite la introducción de datos y **validar que estén en los rangos o valores que deseamos. Este ciclo se emplea por excelencia, para validar la información que digita el usuario.**

Es importante acotar, que esta estructura no está disponible en el diagrama de flujo de datos, por lo que el algoritmo en DFD debe emplear únicamente el ciclo “Mientras” o el bucle “Para”.

A continuación, analizaremos varios ejemplos de la aplicación de este bucle.

Ejemplo 21 Calcular el perímetro y área de un rectángulo

El pseudocódigo recibe por teclado ambos lados del rectángulo y se debe validar por medio del uso de ciclos, que los valores introducidos por teclado sean mayores a cero. Si el usuario introduce un cero, continuará solicitando el valor hasta que digite un número mayor a cero (un número positivo).

```
1  Proceso Rectangulo
2      Definir lado1, lado2, p, a como Entero
3      Escribir 'Digite el primer lado del rectángulo'
4      Repetir
5          Leer lado1
6      Hasta Que lado1>0
7      Escribir 'Digite el segundo lado del rectángulo'
8      Repetir
9          Leer lado2
10     Hasta Que lado2>0
11     p= 2*lado1 + 2*lado2
12     a=lado1*lado2
13     Escribir 'El perímetro del rectángulo es: ', p
14     Escribir 'El área del rectángulo es: ', a
15 FinProceso
```

Figura 40 Pseudocódigo para calcular el perímetro y área de un rectángulo, realizando validaciones con ciclos

Analicemos el ejemplo de la Figura 40:

- En la línea 1 se inicia el pseudocódigo.
- En la línea 2, se declaran las variables (lados del rectángulo) para calcular el perímetro y el área ¿Por qué no se declaran los cuatro lados como variables? Porque hay dos lados cortos (con la misma medida) y dos lados largos (con la misma medida) por lo tanto, con solo que solicitemos el lado largo y el lado corto, podemos calcular el perímetro y área de la figura.
- En las líneas 3 y 7 se envía un mensaje a pantalla solicitando al usuario que digite el lado de la figura.
- En las líneas 4, 5 y 6 se realiza el uso del ciclo “Repetir” para **validar** que si el usuario en la línea 5 digita un cero o un número negativo, la condición de la línea 6 le dice al ciclo que se repita, hasta que el valor introducido en el lado

sea mayor a cero. La misma lógica se emplea en las líneas 8, 9 y 10, pero con el otro lado del rectángulo. En ambos bloques de instrucciones (del 4 al 6 y del 8 al 10) no se continuarán con las subsiguientes líneas de código hasta que se cumpla la condición, por lo tanto, el programa se “encicla” hasta que se digiten números positivos.

- En la línea 11 se realiza el cálculo del perímetro y en la línea 12 el cálculo del área de la figura.
- En las líneas 13 y 14 se imprime la información con los cálculos.
- Fin del pseudocódigo.

Ejemplo 22 Determinar mayúsculas y minúsculas de las vocales

El pseudocódigo recibe por teclado una vocal en mayúscula o minúscula y se debe validar por medio del uso de ciclos, que el programa se ejecute varias veces, hasta que el usuario digite una “n” o una “N” para finalizarlo. Si el usuario no introduce una vocal, se envía un mensaje de error. Se contabilizan cuántas veces se ejecutó el programa y se envía un mensaje final con esa información, además de indicar cuántas minúsculas y cuántas mayúsculas introdujo el usuario.

```
1  Proceso Vocal
2      Definir i, seguir como caracter;
3      Definir cont, contm, contMa Como Entero;
4      cont=0
5      contm=0;
6      contMa=0;
7      Repetir
8          Escribir 'Digite una vocal';
9          Leer i;
10         Segun i Hacer
11             'a' O 'e' O 'i' O 'o' O 'u' :
12                 Escribir 'La vocal está escrita en minúscula';
13                 contm=contm+1;
14             'A' O 'E' O 'I' O 'O' O 'U' :
15                 Escribir 'La vocal está escrita en mayúscula';
16                 contMa=contMa+1;
17         De Otro Modo:
18             Escribir 'Error, no digitó una vocal';
19         Fin Segun
20         Escribir '¿Desea continuar? S=sí, N=no';
21         Leer seguir;
22         cont=cont+1;
23     Hasta Que (seguir="N" O seguir="n")
24     Escribir 'El programa se ejecutó ', cont, 'veces';
25     Escribir 'Se digitaron ', contm, 'vocales minúsculas';
26     Escribir 'Se digitaron ', contMa, 'vocales mayúsculas';
27 FinProceso
```

Figura 41 Pseudocódigo para contar la cantidad de vocales minúsculas y mayúsculas que se digitaron, por medio del uso de ciclos

Analicemos el ejemplo de la Figura 41:

- En la línea 1 se inicia el pseudocódigo.
- En las líneas de la 2 hasta la 6, se declaran las variables y se inicializan los valores de los contadores, se requieren tres: uno para saber cuántas veces se ejecutó el programa (cont) otro para las vocales minúsculas (contm) y otro para las vocales mayúsculas (contMa).
- En la línea 7 inicia el ciclo “Repetir” puesto que todas las instrucciones de la 8 a la 20 se deben ejecutar al menos una vez. Observemos que las líneas 21 y

22 se emplean para consultar si se desea volver a ejecutar el programa. Si esto se hiciera fuera del ciclo (después de la línea 23) el programa nunca se podría detener porque la condición para detenerlo quedó fuera del alcance del *loop*.

- Para solicitar una vocal, se emplea la línea 8 y para leer esta se emplea la instrucción de la línea 9.
- En las líneas de la 10 a la 19 empleamos la estructura “Según” para determinar si se digitaron vocales minúsculas (línea 11) o mayúsculas (línea 14). En el caso que se digitó una vocal minúscula se emplea el contador de la línea 13 para contar cuántas se han introducido por teclado y si se tratara de una vocal mayúscula, se emplea el contador de la línea 16.
- El paso de la línea 22 es muy importante, puesto que antes de salir del ciclo, se debe contar cuántas veces se ha ejecutado el programa.
- En las líneas de la 24 a la 26 se imprime la información con los conteos.
- Fin del pseudocódigo.

Ejemplo 23 Adivinar un número aleatorio

El programa genera un número entre 0 y 10 y el usuario debe adivinar cuál fue el número, se debe solicitar el número “N” veces hasta que la persona acierte. Cuando la persona adivina el número, se genera un mensaje indicándole que acertó y se indica cuántos intentos realizó para adivinarlo.

```
1  Proceso AdivinaNumero
2      Definir num, cont, numUsu Como Entero;
3      num=AZAR(11);
4      cont=0;
5      Escribir 'Adivine el número de 0 a 10';
6      Repetir
7          Leer numUsu;
8          cont=cont+1;
9      Hasta Que numUsu=num
10     Escribir 'Acertó, el número era: ',num, Sin Saltar;
11     Escribir ' le tomó adivinarlo: ',cont,' intentos';
12 FinProceso
```

Figura 42 Pseudocódigo para adivinar un número aleatorio

Analicemos el ejemplo de la Figura 42:

- En la línea 1 se inicia el pseudocódigo.
- En la línea 2 se declaran las tres variables que necesitaremos: el número generado aleatoriamente por el programa (num), el número que el usuario digitará (numUsu) que se comparará con el generado por el programa y el contador (cont) que llevará la cuenta de la cantidad de veces que el usuario digita un número.
- En la línea 3 se asigna a “num” un número aleatorio del 0 al 10 por medio de la función AZAR. En este caso se coloca 11 porque la función siempre requiere ser ajustada, dado que asigna un rango de números de 0 a x-1, lo que quiere decir que si colocamos AZAR (10) solo nos daría un valor del 0 al 9, por lo que si colocamos AZAR (11), nos dará un número de 0 al 10 como lo solicita el enunciado.
- En la línea 4, se inicializa el contador en 0 para que una vez que ingrese al ciclo, lleve la cuenta de los intentos del usuario para adivinar el número.
- En la línea 5 se le envía un mensaje al usuario para que adivina el número del 0 al 10.
- En las líneas del 6 al 9 se ejecuta el ciclo, donde este se repetirá hasta que el número introducido por el usuario (numUsu) sea igual al generado aleatoriamente por el programa (num) (línea 9). Observe que mientras esto se repite la línea 7 leerá desde teclado el número introducido por el usuario y en

la línea 8 se incrementará el contador (línea 8) para llevar la cuenta de los intentos.

- Observe que la condición del ciclo se detendrá hasta que el número generado por el programa (num) sea adivinado por el usuario (numUsu) pero, está permitiendo que se digite cualquier número, ya sea negativo o mayor a 10 puesto que la condición no especificó este aspecto. Por otra parte, los intentos tampoco están restringidos, por lo que continuará ejecutándose “N” veces.
- Una vez que la persona acertó, se finaliza el ciclo y se continúa con las siguientes instrucciones.
- En la línea 10 se envía el mensaje con el número generado por el programa y además se emplea la instrucción “Sin Saltar” que permite que el mensaje de la línea 11 se concatene con la línea 10 y aparezcan como una sola en la pantalla de salida.
- Finalmente, en la línea 11 se envía el mensaje de la cantidad de intentos realizados por la persona para adivinar el número.

Azar (x)

La función Azar se emplea en PSeInt y permite generar un número aleatorio desde 0 hasta x-1. Por ejemplo, Azar (100) genera un número aleatorio del 0 al 99 (pero no incluiría al 100). Para emplear esta función siempre deberemos asignarle a una variable entera, el número generado con el Azar. La estructura es la siguiente:

- **num=Azar(x);**

Donde “x” es el valor máximo del rango de números aleatorios.

Random (x)

La función Random se emplea en DFD y permite generar un número aleatorio desde 0 hasta x. Para emplear esta función siempre deberemos asignarle a una variable entera, el número generado con el Random. La estructura es la siguiente:

- **num=Random(x);**

Donde “x” es el valor máximo del rango de números aleatorios.

Sin Saltar

Esta instrucción permite que no haya un salto de línea entre la línea que se coloca y la siguiente, siempre se coloca al final de la instrucción Escribir y del mensaje y las variables.

La estructura es la siguiente:

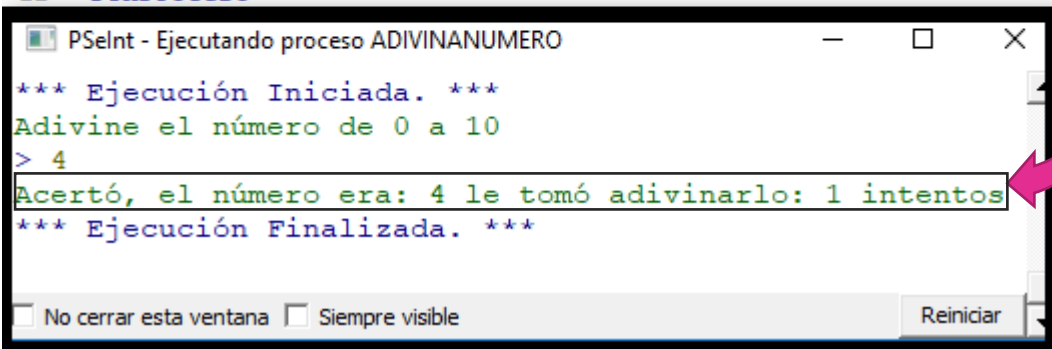
- **Escribir ‘Hola ‘, nom, Sin Saltar;**

Donde:

- Escribir indica que se enviará un mensaje a pantalla
- “Hola ” es el mensaje a pantalla
- nom es la variable que mostrará un valor
- la coma permite separar la instrucción (mensaje o variable) entre el último elemento del mensaje y la instrucción Sin Saltar

Observe en la siguiente figura, el efecto de utilizar la instrucción Sin Saltar, donde en una sola línea, aparece el mensaje “concatenado” de las líneas 10 y 11.

```
1  Proceso AdivinaNumero
2      Definir num, cont,numUsu Como Entero;
3      num=AZAR(11);
4      cont=0;
5      Escribir 'Adivine el número de 0 a 10';
6      Repetir
7          Leer numUsu;
8          cont=cont+1;
9      Hasta Que numUsu=num
10     Escribir 'Acertó, el número era: ',num, Sin Saltar;
11     Escribir ' le tomó adivinarlo: ',cont,' intentos';
12 FinProceso
```



The screenshot shows a window titled "PSeInt - Ejecutando proceso ADIVINANUMERO". The output text is as follows:

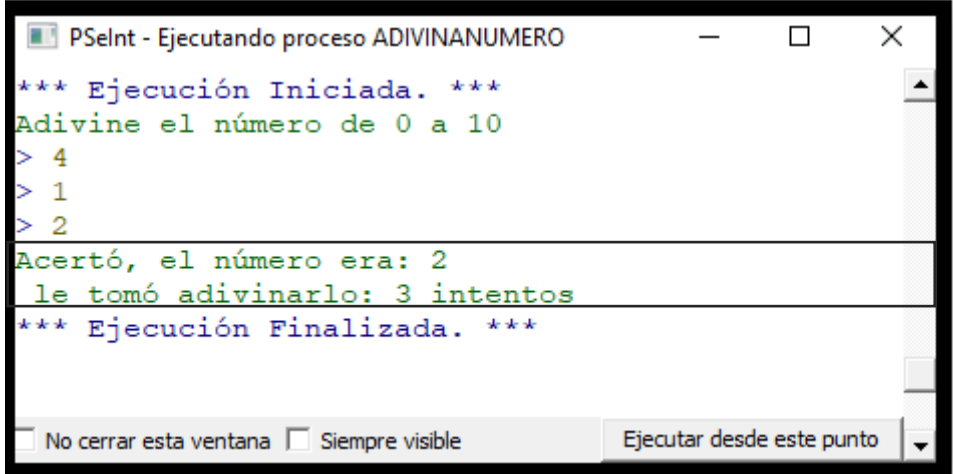
```
*** Ejecución Iniciada. ***
Adivine el número de 0 a 10
> 4
Acertó, el número era: 4 le tomó adivinarlo: 1 intentos
*** Ejecución Finalizada. ***
```

At the bottom of the window, there are two checkboxes: "No cerrar esta ventana" and "Siempre visible", both of which are unchecked. A "Reiniciar" button is located on the right side of the window.

Figura 43 Pantalla de ejecución del programa con la instrucción Sin Saltar

Ahora, veamos en la siguiente figura, la diferencia al eliminar del pseudocódigo anterior la instrucción Sin Saltar. Observe que en la ventana de ejecución las líneas no se concatenarán y el mensaje aparecerá en líneas separadas.

```
1  Proceso AdivinaNumero
2      Definir num, cont,numUsu Como Entero;
3      num=AZAR(11);
4      cont=0;
5      Escribir 'Adivine el número de 0 a 10';
6      Repetir
7          Leer numUsu;
8          cont=cont+1;
9      Hasta Que numUsu=num
10     Escribir 'Acertó, el número era: ',num;
11     Escribir ' le tomó adivinarlo: ',cont,' intentos';
12 FinProceso
```



*** Ejecución Iniciada. ***
Adivine el número de 0 a 10
> 4
> 1
> 2
Acertó, el número era: 2
le tomó adivinarlo: 3 intentos
*** Ejecución Finalizada. ***

☐ No cerrar esta ventana ☐ Siempre visible

Figura 44 Pantalla de ejecución del programa eliminando la instrucción Sin Saltar

Ejemplo 24 Adivinar un número aleatorio con intentos restringidos

El programa genera un número entre 0 y 5 y el usuario debe adivinar cuál fue el número, para lo cual solo tiene tres intentos, sino adivina, se le indica cuál era el número correcto junto con una leyenda de que no acertó. Debe validar que únicamente se introduzcan números entre cero y cinco, si se introducen valores diferentes a estos, se eliminará el intento y se vuelve a solicitar el número, ya sea hasta que acierte o hasta que utilice las tres oportunidades para adivinar. Si la persona adivina el número, se genera un mensaje indicándole que acertó y se indica cuántos intentos realizó para adivinarlo.

```
1  Proceso AdivinaNumero
2      Definir num, cont,numUsu,intent Como Entero;
3      num=Azar(6);
4      cont=0;
5      intent=3
6      Escribir 'Adivine el número de 0 a 5';
7      Mientras intent>0 Y numUsu!=num Hacer
8          Repetir
9              Leer numUsu;
10             cont=cont+1;
11             intent=intent-1;
12             Escribir 'Le restan: ', intent,' intentos';
13         Hasta Que numUsu>=0 Y numUsu<=5
14     Fin Mientras
15     Si numUsu!=num Entonces
16         Escribir 'No adivinó, el número era: ',num;
17     Sino
18         Escribir 'Acertó, el número era: ',num; Sin Saltar;
19         Escribir ' le tomó adivinarlo: ',cont,' intentos';
20     Fin Si
21 FinProceso
```

Figura 45 Pseudocódigo para adivinar un número aleatorio con intentos restringidos
Analicemos el ejemplo de la Figura 45:

- En la línea 1 se inicia el pseudocódigo.
- En la línea 2 se declaran las tres variables que necesitaremos: el número generado aleatoriamente por el programa (num), el número que el usuario digitará (numUsu) que se comparará con el generado por el programa, el contador (cont) que llevará la cuenta de la cantidad de veces que el usuario digita un número y los intentos u oportunidades que tiene la persona para adivinar el número.
- En la línea 3 se asigna a “num” un número aleatorio del 0 al 5 por medio de la función AZAR. En este caso se coloca 6 porque la función siempre requiere ser ajustada, dado que asigna un rango de números de 0 a x-1, lo que quiere

decir que si colocamos AZAR (5) solo nos daría un valor del 0 al 4, por lo que si colocamos AZAR (6), nos dará un número de 0 al 5 como lo solicita el enunciado.

- En la línea 4, se inicializa el contador en 0 para que una vez que ingrese al ciclo, lleve la cuenta de los intentos del usuario para adivinar el número.
- En la línea 5 se colocan la cantidad de intentos permitidos, en este caso serían tres.
- En la línea 6 e le envía un mensaje al usuario para que adivina el número del 0 al 5.
- En la línea 7 se emplea el ciclo Mientras para establecer las dos condiciones que se deben cumplir para continuar con la ejecución de las instrucciones: la primera es tener más de un intento disponible y segundo, que se realice mientras la persona no acierte el número.

Importante

El **ciclo Mientras puede analizar condiciones con variables diferentes**, en este caso se están valorando que se realice un máximo de tres intentos (variable "intent") como la igualdad del número aleatorio (variable "num") con el número digitado por el usuario (variable "numUsu").

El **ciclo Repetir no puede analizar condiciones con variables diferentes**, solo puede analizar las condiciones de una misma variable con diferentes valores, en este caso solo se puede aplicar para validar que el número digitado por el usuario (variable "numUsu") esté entre cero y cinco.

- En las líneas del 8 al 13 se ejecuta el segundo ciclo (esto es un **ciclo anidado**, porque se encuentra dentro del ciclo Mientras), donde este se repetirá hasta que el número introducido por el usuario (numUsu, línea 9) esté entre 0 y 5 (línea 13) e incrementará el contador (línea 10), restará los intentos (línea 11) y emitirá un mensaje con los intentos restantes (línea 12).
- Observe que como el ciclo Repetir está **anidado** dentro del ciclo Mientras, una vez realizado el proceso explicado en el punto 8, el ciclo Mientras consulta si el número digitado por el usuario es igual al número generado por el programa y además consulta si le quedan intentos. Sino adivinó el número, pero le quedan intentos, repite los pasos del 8 al 13. Si adivina el número termina el ciclo Mientras, no ingresa al ciclo Repetir (líneas 8 a la 13) y salta a la siguiente línea del programa (línea 15).
- An la línea 15 se consulta si el usuario acertó el número, esto sucede porque no se tiene certeza si la ejecución del ciclo finalizó porque se acabaron los

intentos o porque la persona adivinó el número. Si la persona sí acertó el número, se envía el mensaje de la línea 16, sino acertó el número y se le acabaron los intentos se envían los mensajes de las líneas 18 y 19.

Estructura Para (For)

El ciclo *for* o ciclo "Para" es una estructura de repetición donde ya se sabe cuántas veces se realizará el bucle, es decir, cuantas veces se ejecutará el conjunto de instrucciones dentro del ciclo. La cantidad de veces está determinada en la sección de decisión del ciclo. La sintaxis del ciclo *for* en pseudocódigo es la siguiente:

```
for (variable=valor_inicial; variable <= valor_final; incremento)
```

```
    Instrucción
```

```
    Instrucción...
```

```
fin del for
```

La primera línea constituye la decisión. Aquí la condición indica que la variable debe tomar valores entre el valor inicial y el valor final para que se ejecute el conjunto de instrucciones del ciclo.

Esta estructura en el programa PSeInt, se expresa de la siguiente forma:

```
Para variable_numerica<-valor_inicial Hasta valor_final Con Paso paso  
Hacer
```

```
    secuencia_de_instrucciones
```

```
Fin Para
```

A diferencia que el ciclo Repetir ... Hasta, el ciclo Para sí tiene un símbolo de representación en DFD. Tanto la representación en PSeInt como en DFD se verán en los siguientes ejemplos.

Ejemplo 25 Tabla de multiplicar del 1

El pseudocódigo muestra la tabla de multiplicar del 1, desde 0 hasta 10.

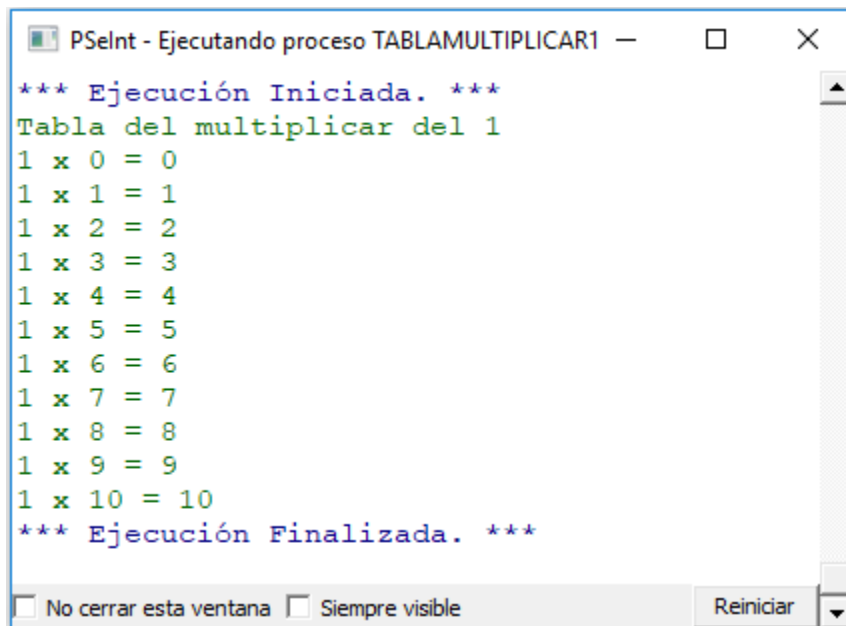
```
1  Proceso TablaMultiplicar1
2      Definir i como entero;
3      Escribir 'Tabla del multiplicar del 1';
4      Para i<-0 Hasta 10 Con Paso 1 Hacer
5          Escribir '1 x ', i, ' = ', i
6      Fin Para
7  FinProceso
```

Figura 46 Pseudocódigo para mostrar la tabla de multiplicar del 1

Analicemos el pseudocódigo de la Figura 46:

- En la línea 1 se inicia el pseudocódigo.
- En la línea 2 se declaran la única variable necesaria y que es la empleada en el ciclo. No se requieren más variables porque en la tabla del uno, cualquier número multiplicado por uno da como resultado el mismo número.
- En la línea 3 se envía el mensaje de que se mostrará la tabla de multiplicar.
- En la línea 4 se inicia el ciclo desde 0 (Para i<-0) y hasta 10 (Hasta 10), de una en una unidad (Con Paso 1 Hacer). Esto quiere decir que el ciclo se ejecutará 10 veces y que el “i” es un **contador** de 1 en 1, iniciará en 0 y terminará en 10 ¿Por qué no se requiere un acumulador para esta tabla de multiplicar? Porque cualquier número multiplicado por 1 da como resultado el mismo número.
- En la línea 5 se muestra el mensaje al usuario, en la línea 6 finaliza el ciclo y en la 7 termina el programa.

A continuación se muestra la salida a pantalla del pseudocódigo anterior:



```
*** Ejecución Iniciada. ***
Tabla del multiplicar del 1
1 x 0 = 0
1 x 1 = 1
1 x 2 = 2
1 x 3 = 3
1 x 4 = 4
1 x 5 = 5
1 x 6 = 6
1 x 7 = 7
1 x 8 = 8
1 x 9 = 9
1 x 10 = 10
*** Ejecución Finalizada. ***
```

At the bottom of the window, there are checkboxes for 'No cerrar esta ventana' and 'Siempre visible', and a 'Reiniciar' button.

Figura 47 Salida a pantalla de la tabla de multiplicar del 1

Para el pseudocódigo anterior, el diagrama de flujo en DFD es el siguiente:

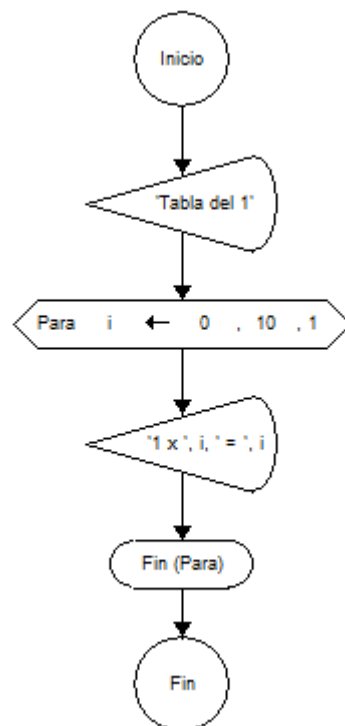


Figura 48 Diagrama de flujo de la tabla de multiplicar del 1

Ejemplo 26 Tabla de multiplicar del 5

El pseudocódigo muestra la tabla de multiplicar del 1, desde 1 hasta 15.

```
1  Proceso TablaMultiplicar5
2      Definir i, num como entero;
3      num=5;
4      Escribir 'Tabla del multiplicar del 5';
5      Para i<-1 Hasta 15 Con Paso 1 Hacer
6          Escribir '5 x ', i, ' = ', i*num
7      Fin Para
8  FinProceso
```

Figura 49 Pseudocódigo para mostrar la tabla de multiplicar del 1

Analicemos el pseudocódigo de la Figura 49:

- En la línea 1 se inicia el pseudocódigo.
- En la línea 2 se declaran el contador del ciclo y el número por el que se multiplicará el contador.
- En la línea 3 se inicia el valor por el que será multiplicado el contador del ciclo, en este caso se le asignará el valor de 5 para mostrar esta tabla.
- En la línea 4 se envía el mensaje de que se mostrará la tabla de multiplicar.
- En la línea 5 se inicia el ciclo desde 1 (Para i<-1) y hasta 15 (Hasta 15), de una en una unidad (Con Paso 1 Hacer). Esto quiere decir que el ciclo se ejecutará 15 veces y que el “i” es un **contador** de 1 en 1, iniciará en 1 y terminará en 15.
- En la línea 5 se muestra el mensaje al usuario, observe que en este caso se necesita multiplicar el contador “i” por el número 5 (num).
- En la línea 6 finaliza el ciclo y en la 7 termina el programa.

A continuación, se muestra la salida a pantalla del pseudocódigo anterior, preste especial atención a las líneas, círculos y flechas para interpretar el pseudocódigo con el mensaje al usuario.

```
1  Proceso TablaMultiplicar5
2      Definir i, num como entero;
3      num=5;
4      Escribir 'Tabla del multiplicar del 5';
5      Para i<-1 Hasta 15 Con Paso 1 Hacer
6          Escribir '5 x ', i, ' = ', i*num
7      Fin Para
8  FinProceso
```

*** Ejecución Iniciada. ***
Tabla del multiplicar del 5
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
5 x 11 = 55
5 x 12 = 60
5 x 13 = 65
5 x 14 = 70
5 x 15 = 75
*** Ejecución Finalizada. ***

☐ No cerrar esta ventana ☐ Siempre visible Reiniciar

Figura 50 Salida a pantalla de la tabla de multiplicar del 5

Para el pseudocódigo anterior, el diagrama de flujo en DFD es el siguiente:

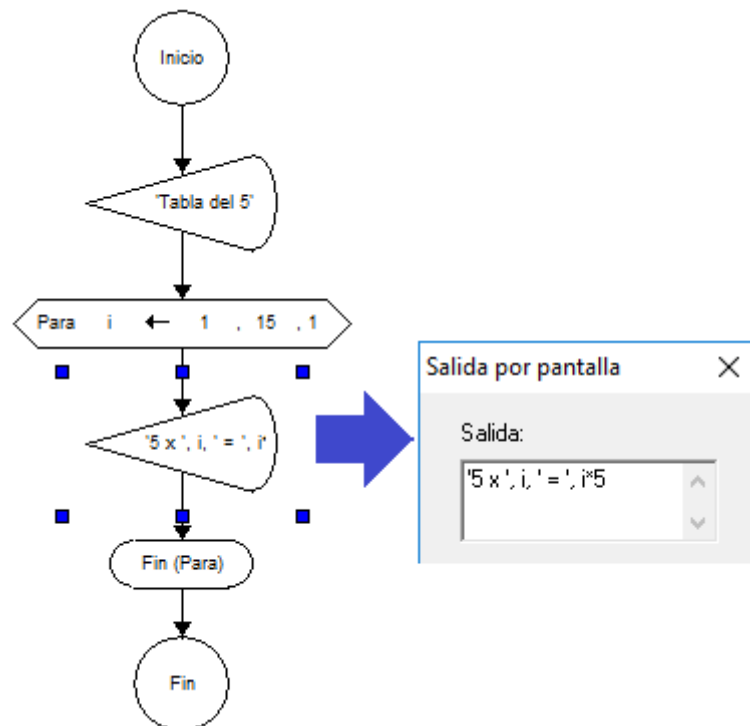


Figura 51 Diagrama de flujo de la tabla de multiplicar del 5

Observe que a diferencia del pseudocódigo, en el diagrama de flujo de datos se está multiplicando el contador por el número 5 sin necesidad de haberlo asignado a un número, tanto la solución mostrada en el diagrama de flujo ($i*5$) como en el pseudocódigo ($i*\text{num}$) es correcto.

Ejemplo 27 Mostrar los números pares de 0 a 100

El pseudocódigo muestra los números pares desde 0 y hasta el 100, de forma que muestre: 2, 4, 6, 8, 10, ..., 96, 98, 100.

```
1  Proceso NumerosPares
2      Definir num Como Entero;
3      Escribir 'Los primeros 100 números pares son: ';
4      Para num<-0 Hasta 100 Con Paso 2 Hacer
5          Escribir num, ', ', Sin Saltar;
6      Fin Para
7  FinProceso
```

Figura 52 Pseudocódigo para los números pares del 0 al 100 usando el ciclo *for*

Analicemos el pseudocódigo de la Figura 52:

- En la línea 1 se inicia el pseudocódigo.
- En la línea 2 se declaran el contador del ciclo.
- En la línea 3 se envía el mensaje de que se mostrarán los números pares.
- En la línea 4 se inicia el ciclo desde 0 (Para num<-0) y hasta 100 (Hasta 100), de dos en dos unidades (Con Paso 2 Hacer). Esto quiere decir que el ciclo se ejecutará 50 veces y que “num” es un **contador** de 2 en 2, iniciará en 0 y terminará en 100.
- En la línea 5 se muestra el mensaje al usuario con el número 2, 4, 6, 8, 10, ...98, 100.
- En la línea 6 finaliza el ciclo y en la 7 termina el programa.

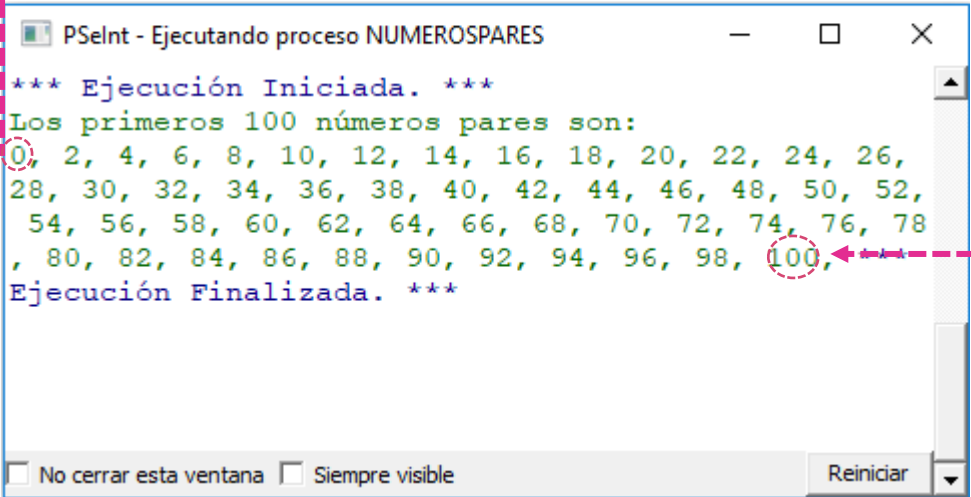
Importante

En el ciclo Para, el contador puede realizar incrementos como queramos configurarlo, supongamos que deseamos mostrar los valores del 0 al 10 pero con diferentes incrementos, veamos varios ejemplos:

- Incremento de **1 en 1**: Para i<-0 Hasta 10 Con Paso 1 Hacer
- Incremento de **2 en 2**: Para i<-0 Hasta 10 Con Paso 2 Hacer
- Incremento de **3 en 3**: Para i<-0 Hasta 10 Con Paso 3 Hacer
- Incremento de **4 en 4**: Para i<-0 Hasta 10 Con Paso 4 Hacer
- Incremento de **5 en 5**: Para i<-0 Hasta 10 Con Paso 5 Hacer
- Incremento de **6 en 6**: Para i<-0 Hasta 10 Con Paso 6 Hacer
- Incremento de **7 en 7**: Para i<-0 Hasta 10 Con Paso 7 Hacer
- Y así sucesivamente, dependiendo de lo que deseemos realizar.

A continuación se muestra la salida a pantalla del pseudocódigo anterior, preste especial atención a las líneas, círculos y flechas para interpretar el pseudocódigo con el mensaje al usuario.

```
1  Proceso NumerosPares
2      Definir num Como Entero;
3      Escribir 'Los primeros 100 números pares son: ';
4      Para num<-0 Hasta 100 Con Paso 2 Hacer
5          Escribir num, ', ', Sin Saltar;
6      Fin Para
7  FinProceso
```



*** Ejecución Iniciada. ***
Los primeros 100 números pares son:
0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26,
28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52,
54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78,
, 80, 82, 84, 86, 88, 90, 92, 94, 96, 98, 100,
Ejecución Finalizada. ***

Figura 53 Salida a pantalla con los números pares del 0 al 100

Para el pseudocódigo anterior, el diagrama de flujo en DFD es el siguiente:

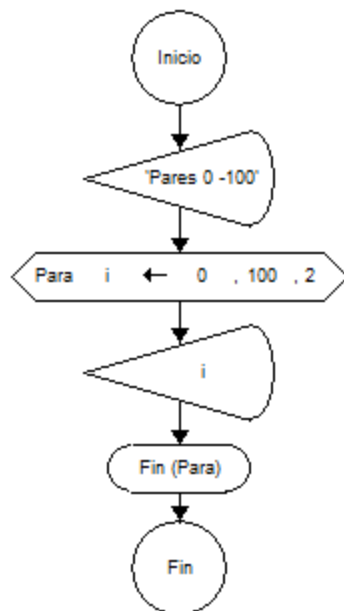


Figura 54 Diagrama de flujo de los números pares del 0 al 100

Ejemplo 28 Sumar los números pares de un rango dado por el usuario

El pseudocódigo recibe por teclado dos valores, uno inicial desde donde parte la serie de números y uno final donde se detiene el ciclo, por lo tanto el segundo valor debe

ser mayor al primero, lo cual debe validarse por medio de un ciclo. Por medio del ciclo *for* debe realizarse el recorrido desde el primer valor hasta el último valor (por ejemplo, del 1 hasta el 15) y determinar si el número es par, si lo es lo suma en un acumulador y contabiliza las sumas realizadas. Al finalizar, se emite un mensaje con la suma de todos los números pares del rango y la cantidad de sumas que se realizaron.

```
1  Proceso SumaPares
2      Definir i,a,b,suma, cont como enteros;
3      cont=0;
4      Repetir
5          Escribir 'Digite el número de inicio';
6          Leer a;
7          Escribir 'Digite el número final';
8          Leer b;
9      Hasta Que b>a
10
11     Para i<-a Hasta b Con Paso 1 Hacer
12         Si i MOD 2 =0 Entonces
13             suma=suma+i;
14             cont=cont+1;
15         Fin Si
16     Fin Para
17     Escribir 'Suma de pares de', a, ' a ', b, 'es: ' suma;
18     Escribir 'Se realizaron ',cont, ' sumas';
19 FinProceso
```

Figura 55 Pseudocódigo para sumar los números pares de un rango

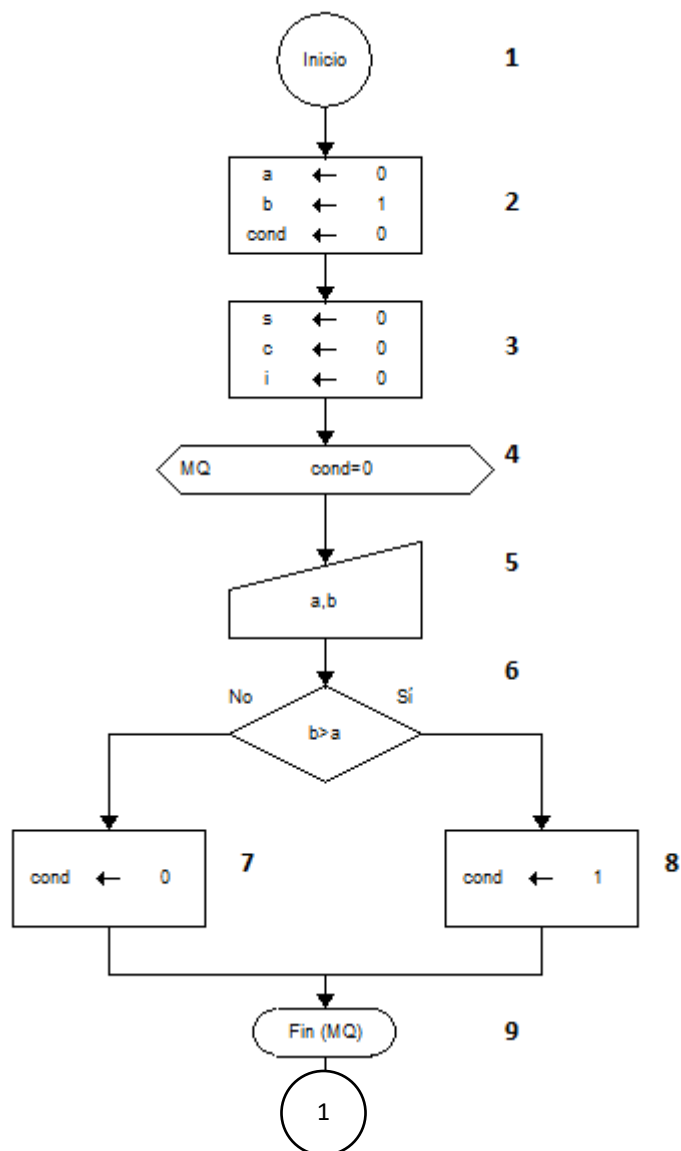
Analicemos el pseudocódigo de la Figura 55:

- En la línea 1 se inicia el pseudocódigo.
- En la línea 2 se declaran las variables a utilizar y en la línea 3 se inicializa el contador en cero.
- En la línea 4 se inicia el ciclo que valida que el número de inicio sea menor al número final (línea 9). Si el primer número (a) es mayor o igual al segundo número (b), el ciclo se repetirá hasta que “b” sea mayor a “a”.
- En las líneas de la 11 a la 16 se realiza el ciclo que hará el recorrido desde “a” hasta “b” de uno en uno los números serán analizados para determinar si su cociente es cero (línea 12), si esto es cierto entonces el número es par y se acumula en la suma (línea 13) y se incrementa el contador de las sumas realizadas (línea 14).
- Una vez realizado el ciclo, se emitirán los mensajes de las líneas 17 y 18.

Para el pseudocódigo anterior, el diagrama de flujo en DFD es el que se muestra en la figura 56. Es importante recalcar que este diagrama no es una copia exacta del

pseudocódigo, porque en DFD no existe la estructura Repetir ... Hasta, por lo que se tuvo que adaptar al bucle Mientras.

Para lo anterior, lo que se consideró es que la condición (cond) iniciara en cero (punto 2) y que el ciclo se ejecutaría (punto 4) mientras que esa condición permaneciera en cero. Entonces, se leen las variables (punto 6) y si "b" es mayor a "a", asigna uno a la condición para finalizar el ciclo, pero por el contrario, si "a" es mayor a "b", entonces la condición sería cero, para obligar al ciclo a repetirse hasta que el segundo número sea mayor al primero.



1. Inicio del algoritmo.

2 y 3. Inicialización de todas las variables.

4. Como el ciclo Repetir...hasta no existe, se realiza la adaptación al ciclo Mientras, para que la cond=0 cuando sea verdadero que el segundo número sea mayor al primero.

5. Se leen los números "a" y "b".

6. Se utiliza la estructura "Si" para determinar si "b" es mayor a "a".

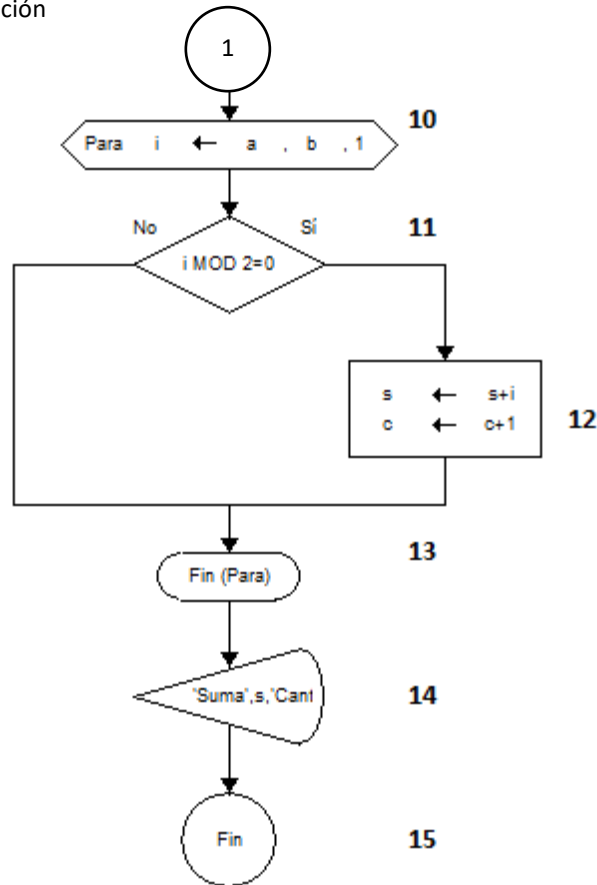
7. Si es cierto, entonces se cambia la condición a 1 para forzar que el ciclo finalice.

8. Pero sino, se mantiene la "cond" en cero para volver a ejecutar el ciclo y volver a pedir los números.

9. Fin del ciclo Mientras.

Continúa...

Continuación



10. Inicio del ciclo *for* que inicia desde “a”, finaliza en “b” e incrementa de uno en uno la unidad.

11. Consulta si el cociente de “i” dividido entre 2 =0. Para entender este recorrido observe la tabla bajo este ejemplo.

12. Si lo anterior es cierto, entonces acumula el número (en “s”) y cuenta la suma (en “c”).

13. Finaliza el ciclo.

14. Envía el mensaje con la suma y el conteo de cuántas sumas se realizaron.

15. Finaliza el algoritmo.

Figura 56 Diagrama de flujo para sumar los números pares de un rango

Supongamos que a=5 y b=10, entonces el recorrido que se realizaría en los puntos del 10 al 13 serían:

TABLA 9
RECORRIDO DEL DIAGRAMA DE FLUJO PARA SUMAR LOS NÚMEROS PARES

Contador i	i MOD 2 = 0	Operaciones: “s” y “c”	Incremento de 1
5	falso	No se realizan cálculos	5+1=6
6	verdadero	S=0+6 C=0+1	6+1=7
7	falso	No se realizan cálculos	7+1=8
8	verdadero	S=6+8 C=1+1	8+1=9
9	falso	No se realizan cálculos	9+1=10
10	verdadero	S=14+10 C=2+1	10+1=11

Ejemplo 29 Determinar si un número es primo

El pseudocódigo recibe por teclado un número y determinar si es primo. Si en algún momento el número es divisible entre otro valor, debe forzarse la finalización del ciclo. Emplee una variable de tipo Boolean (en PSeInt es conocida como Lógica) para determinar si el número es o no primo. Al finalizar, se emite un mensaje con el resultado de la evaluación.

```
1  Proceso NumerosPrimos
2      Definir i, num como Entero;
3      Definir bandera como Logico;
4      bandera=Verdadero;
5      Escribir "Digite el número a valorar:";
6      Leer num;
7      Para i<-2 Hasta num-1 Con Paso 1 Hacer
8          Si num MOD i = 0 Entonces
9              bandera=Falso;
10             i=num-1; //obliga a terminar el ciclo
11         Sino
12             Si Num MOD i = 1 Entonces
13                 bandera=Verdadero;
14             FinSi
15         FinSi
16     FinPara
17     Si bandera=Falso Entonces
18         Escribir "El número ', num,' no es primo";
19     Sino
20         Escribir "El número ', num,' es primo";
21     Fin Si
22 FinProceso
```

Figura 57 Pseudocódigo para determinar si un número es primo por medio de ciclos

Un número primo es aquel mayor a 1 y que es divisible entre 1 y él mismo. Por medio del ciclo “for” debe realizarse el recorrido desde el número 2 hasta el número menos 1 para determinar si es divisible entre otros números.

Analicemos el pseudocódigo de la Figura 57:

- En la línea 1 se inicia el pseudocódigo.
- En las líneas 2 y 3 se declaran las variables a utilizar y en la línea 4 se inicializa la bandera a verdadero, esto ayudará más adelante en el programa a detener (si el número es divisible entre otro) o continuar (si el número no es divisible entre otro) la ejecución del pseudocódigo.
- En la línea 5 se envía un mensaje al usuario solicitando el número.
- En la línea 6 se almacena en la variable “num” el valor digitado por el usuario.
- En la línea 7 analizaremos varios elementos del contexto del problema:

- a. No podemos iniciar i en cero ($i < 0$) porque la división entre cero no existe.
 - b. No podemos iniciar " i " en uno ($i < 1$) porque todos los números son divisibles entre 1, por lo tanto, no se requiere valorar esta división.
 - c. El ciclo se ejecuta hasta el número-1 ($\text{num}-1$) porque si el tope fuera el número mismo, se repite la situación del punto "b", ya que todo número es divisible entre 1 y consigo mismo, por lo tanto, debe quitársele una unidad para el tope de las divisiones. Suponiendo que el usuario digitó el número 5, entonces el ciclo "Para" se ejecutará desde 2 y hasta 4.
 - d. El incremento del ciclo se realiza en una unidad (Con Paso 1 Hacer) porque todos los números entre el 2 y el tope-1 se deben valorar. Si partimos de la suposición de que el número digitado es 5, entonces se valorarán las divisiones de: 2, 3 y 4.
- En la línea 8 se consulta si el número (num) dividido entre el contador del ciclo (i) es igual a cero. Si esto es cierto significa que el número es divisible entre otro número aparte de uno y sí mismo, por lo tanto, ya no es un número primo, por lo que se procede a colocar la bandera en falso (línea 9) para indicarle al programa que ese número no es primo (esta condición se evalúa en la línea 17) y se fuerza al ciclo a llegar al máximo de su contador (línea 10) para que ya no se ejecute más.
 - Por el contrario, si lo anterior no es cierto, se consulta si el número (num) dividido entre el contador del ciclo (i) es igual a uno. Si esto es cierto significa que el número no es divisible entre otro número, por lo tanto la bandera se coloca en verdadero (línea 13) indicando que el número continúa siendo primo y se procede a valorar el siguiente " i ", repitiendo el punto anterior.
 - Una vez finalizada la ejecución del ciclo Para, ya sea porque se forzó a finalizar o porque llegó a la valoración de todos los contadores, se consulta en la línea 17 si la bandera es Falso, si esta condición se da, se emite el mensaje de la línea 18 indicando que el número no es primo, Sino, se emite el mensaje de la línea 20 indicando que el número sí es primo.

Resumen

Tenemos que las variables son los datos indispensables que se necesitan en un problema para poder resolver un algoritmo. La mayoría de los datos se recibirán por teclado ya que se deben solicitar al usuario (variables) pero existen otros que el contexto del algoritmo nos lo brindará y a estos datos donde conocemos el valor

previamente se le conoce como constante (por ejemplo, el valor de Pi). Asimismo, tanto las variables como constantes pueden contener texto largo (cadena), un solo número o letra (carácter), números sin decimales (entero), números con decimales (real) o valoraciones que pueden ser verdaderas o falsas (boolean).

Por otra parte, después de haber estudiado las estructuras de programación, podemos clasificarlas en dos grandes áreas: de decisión y de repetición.

Las estructuras de decisión nos ayudan a discriminar si la condición de un problema es verdadera o falsa y nos permite determinar las siguientes secuencias de acciones, dependiendo de la condición valorada.

Las estructuras de repetición o también conocidas como ciclos o bucles, permiten repetir las instrucciones o estructuras de decisión, según una condición o cantidad de veces que se requiera de la iteración.

En el caso del ciclo Mientras y el Repetir... Hasta, no sabemos cuántas veces se ejecutarán las instrucciones pues generalmente depende de la decisión del usuario; a diferencia del ciclo *for* del que sabemos cuántas veces se repetirán las instrucciones.

Cuando en un ciclo necesitamos saber cuántas veces se realizan las instrucciones, empleamos un contador, que generalmente tendrá la sintaxis $i=i+1$ o bien, $i++$ (ambas significan lo mismo); esto quiere decir que al valor que tenía la variable, le sumamos una unidad.

Por otra parte, cuando necesitamos ir sumando valores, por ejemplo, cuánto gastamos en las diferentes compras en una tienda; usaremos un acumulador, el que tendrá la siguiente sintaxis: $\text{suma}=\text{suma}+\text{compra}$; esto quiere decir que al valor que tenía la variable le sumamos el nuevo valor de un artículo.

Para realizar una síntesis de las semejanzas y diferencias entre los ciclos, observe la siguiente tabla resumen.

TABLA 10
RESUMEN DE CARACTERÍSTICAS DE LOS CICLOS

Criterio	Ciclo Mientras	Ciclo Repetir ... Hasta	Ciclo Para
Condición	Se ejecuta mientras la condición sea verdadera. Si la	Se ejecuta mientras que la condición sea falsa. Si la condición	No usa condiciones

Criterio	Ciclo Mientras	Ciclo Repetir ... Hasta	Ciclo Para
	condición pasa a falso ya no se ejecuta el ciclo.	pasa a verdadera ya no se ejecuta el ciclo.	
Utilidad	Sirve para combinar diferentes condiciones con diversas variables y ejecutarlas mientras las condiciones sean verdaderas	Sirve para validar valores numéricos o alfanuméricos en un rango. No permite la combinación de condiciones con diferentes variables.	Ejecuta todas las condiciones desde un inicio hasta un final conocidos.
Variables	Puede manejar múltiples variables o condiciones para determinar si es verdadera o falsa la condición.	Puede manejar solo una variable o condición en un rango de valores, para determinar si es verdadera o falsa la condición.	Solo maneja un contador.
Incrementos	No es requerido.	No es requerido.	Es requerido.
Cantidad de veces que se ejecuta	Desconocido, porque depende de una condición.	Desconocido, porque depende de una condición.	Conocido, el contexto del problema indica la cantidad de iteraciones a realizar.
Finaliza	Cuando la condición o condiciones son falsas.	Cuando la condición es verdadera.	Cuando se llega al número máximo de iteraciones indicadas en el problema.
Ejecución	Si las condiciones iniciales son falsas, no se ejecuta.	Se ejecuta al menos una vez y después dependerá de las condiciones.	Siempre se ejecuta.
¿Requiere contadores?	No necesariamente, depende del contexto del problema.	No necesariamente, depende del contexto del problema.	Sí. El ciclo se rige con el contador que determina desde dónde inicia y hasta cuándo finaliza.
¿Requiere acumuladores?	No necesariamente, depende del contexto del problema.	No necesariamente, depende del contexto del problema.	No necesariamente, depende del contexto del problema.

Ejercicios de autoevaluación

1. Tipo de dato que almacena números y que no pose decimales:_____.
2. Tipo de dato que almacena números y que pose decimales: _____.
3. Tipo de dato que almacena textos:_____.
4. Tipo de dato que almacena solo una letra:_____.
5. Tipo de dato que guarda solo un valor de verdadero o falso _____.
6. La instrucción básica en un algoritmo para pedirle al usuario que ingrese datos desde teclado es: _____.
7. La instrucción básica en un algoritmo para que realice las ecuaciones necesarias para brindar un resultado es: _____.
8. La instrucción básica en un algoritmo para recibir desde teclado un dato brindado por el usuario es: _____.
9. La instrucción básica en un algoritmo para brindarle al usuario el mensaje final con los cálculos o resultados es: _____.
10. Representación visual de un algoritmo, con elementos gráficos como círculos, rombos y rectángulos: _____.
11. Representación escrita de un algoritmo, con una sintaxis establecida y con palabras reservadas: _____.
12. Elabore un pseudocódigo en PSeInt que realice la conversión de colones a dólares, solicitándole al usuario la cantidad de colones y el tipo de cambio.
13. Elabore un pseudocódigo en PSeInt que solicite el nombre de una persona y envíe un saludo: *Hola, nombre*.
14. Elabore un pseudocódigo en PSeInt y un diagrama de flujo de datos en DFD que solicite dos números y presente el producto de estos.
15. Elabore un pseudocódigo en PSeInt y un diagrama de flujo de datos en DFD que solicite un número del 1 al 7 y envíe un mensaje con el día de la semana. Utilice la instrucción "Si".
16. Elabore un diagrama de flujo de datos en DFD que calcule el salario de un trabajador, considerando todas las condiciones del Ejemplo 5.
17. Elabore un pseudocódigo en PSeInt donde determine que, si una persona llega a tiempo a la estación del bus, envíe un mensaje de que puede irse en bus y sino llega a tiempo, el mensaje debe indicar que debe irse caminando. Utilice la instrucción "Si – Sino".
18. Elabore un pseudocódigo en PSeInt y un diagrama de flujo de datos en DFD, donde se reciban tres números por teclado y se calcule el promedio de estos. Asuma que los números son enteros.
19. Elabore un diagrama de flujo de datos en DFD y un pseudocódigo en PSeInt, donde se reciban dos números por teclado y se determine cuál es el número mayor o si son iguales.

20. Elabore un pseudocódigo en PSeInt, que solicite el año de nacimiento del usuario y calcule la edad. Si tiene entre 0 y menos de 12 años se indica que es un niño, si tiene entre 12 y menos de 18 años se emite un mensaje que es un adolescente y sino emite el mensaje que es un adulto. Valide que el año actual sea mayor al año de nacimiento de la persona.
21. Elabore un pseudocódigo en PSeInt donde se lea el día de semana (de 1 a 7) y se indique a cuál día corresponde.
22. Modifique el pseudocódigo anterior y valide por medio de ciclos, que el número digitado por el usuario esté únicamente entre 1 y 7.
23. Elabore un pseudocódigo en PSeInt y un diagrama de flujo de datos en DFD, empleando el ciclo "While" donde muestre los primeros 100 números pares (esto abarca del 0 al 200).
24. Modifique el pseudocódigo del número primo, para que se ejecute varias veces, hasta que el usuario digite una "n" o una "N".
25. Elabore un pseudocódigo en PSeInt donde al usuario se le ofrezca un menú con las siguientes opciones:
 1. Sumar "n" cantidad de números (se debe acumular la suma de "n" números por medio de ciclos, hasta que se digite un cero).
 2. Tabla de multiplicar (debe mostrar la tabla de multiplicar de un número introducido por el usuario, desde 1 hasta el tope que indique la persona, para ello debe hacer uso de ciclos).
 3. División de dos números (debe validar por medio de ciclos, que el denominador no sea mayor a cero).
 4. Exponente (el usuario digita la base y el exponente y por medio de ciclos, se realiza el cálculo).
 5. Áreas de figuras, esta opción tiene dos posibilidades para elegir:
 - a. Área de un rectángulo en centímetros.
 - b. Área de un rombo en centímetros.

Si la persona no digita ninguna de las cinco opciones, debe emitir un mensaje de error. Emplee todas las estructuras de decisión y repetición para realizar las validaciones y resolver cada una de las partes del algoritmo.

1.8 Respuestas a los ejercicios de autoevaluación

1. Entero.
2. Real.
3. Cadena.
4. Caracter.
5. Booleano.
6. Solicitar.
7. Calcular.
8. Leer.
9. Mostrar.
10. Diagrama de flujo de datos.
11. Pseudocódigo.

12. Pseudocódigo del tipo de cambio:

```
1  Proceso TipoCambio
2      Definir colon, dolar, cambio Como Real
3      Escribir 'Digite la cantidad de colones que desea cambiar';
4      Leer colon;
5      Escribir 'Digite el tipo de cambio';
6      Leer cambio;
7      dolar=colon/cambio;
8      Escribir 'Con ', colon, ' colones, puede comprar: ', dolar, ' dólares';
9  FinProceso
```

Figura 58 Pseudocódigo para cambiar de colones a dólares

13. Pseudocódigo de saludo con el nombre del usuario.

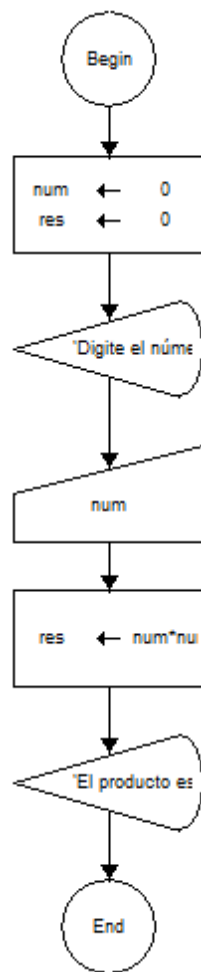
```
1  Proceso HolaPersona
2      Definir nombre Como Cadena
3      Leer nombre
4      Imprimir "Hola", nombre
5  FinProceso
```

Figura 59 Pseudocódigo para saludar

14. Pseudocódigo y diagrama de flujo del producto de dos números.

Diagrama de flujo de datos

Pseudocódigo en PSeInt



```
Proceso Producto
Definir num, result Como Entero
Escribir 'Digite un número';
Leer num;
result=num*num;
Escribir 'El producto es: ', result
FinProceso
```

Figura 60 Pseudocódigo y diagrama de flujo para calcular el producto de un número

15. Pseudocódigo y diagrama de flujo con el día de la semana.

```
Proceso DiaSemana
  Definir día como entero;
  Escribir 'Escriba un número del 1 al 7';
  Leer día;
  Si día=1 Entonces
  ..... Escribir 'Hoy es lunes'
  Fin Si
  Si día=2 Entonces
  ..... Escribir 'Hoy es martes'
  Fin Si
  Si día=3 Entonces
  ..... Escribir 'Hoy es miércoles'
  Fin Si
  Si día=4 Entonces
  ..... Escribir 'Hoy es jueves'
  Fin Si
  Si día=5 Entonces
  ..... Escribir 'Hoy es viernes'
  Fin Si
  Si día=6 Entonces
  ..... Escribir 'Hoy es sábado'
  Fin Si
  Si día=7 Entonces
  ..... Escribir 'Hoy es domingo'
  Fin Si
FinProceso
```

Figura 61 Pseudocódigo para el día de la semana

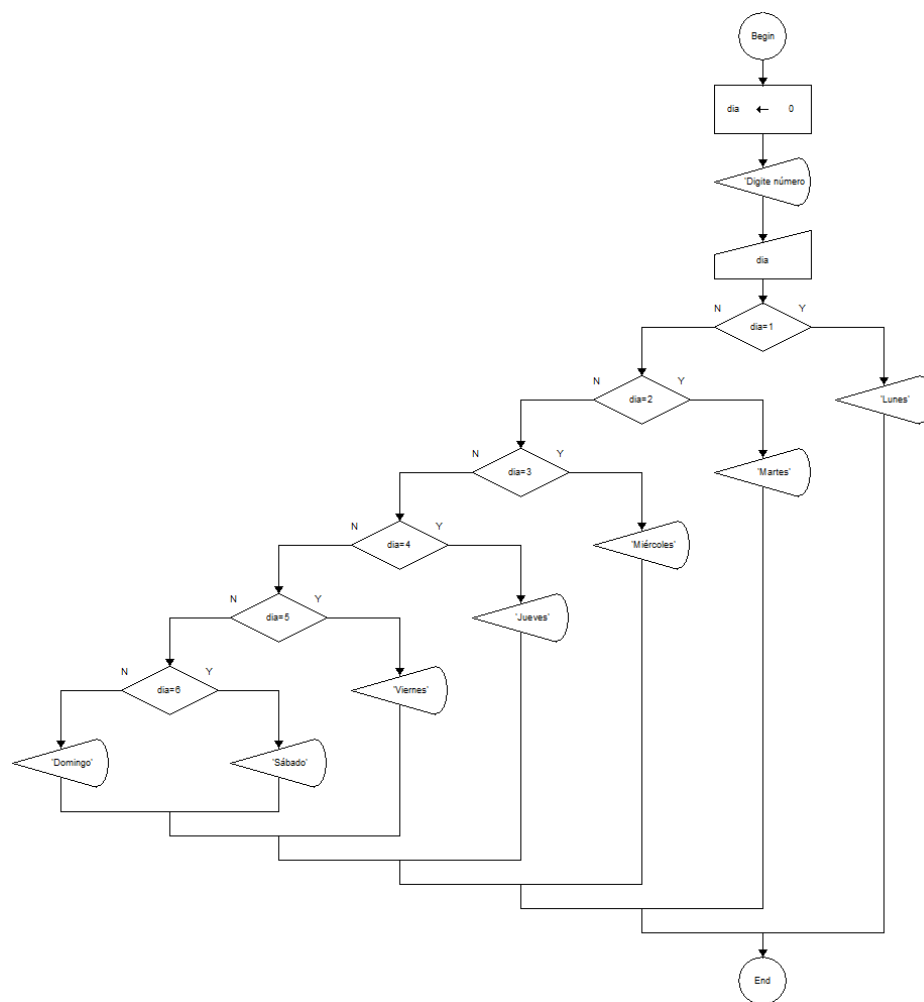
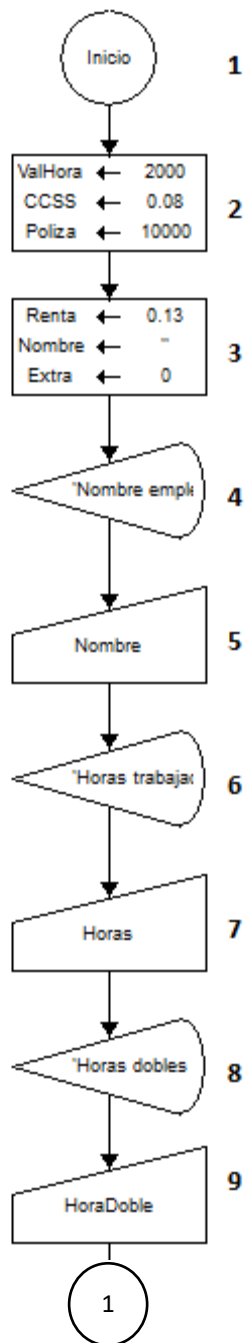


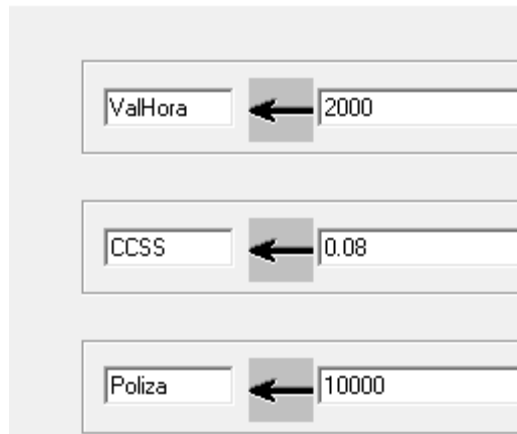
Figura 62 Diagrama de flujo para el día de la semana

16. Diagrama de flujo con el cálculo de salario



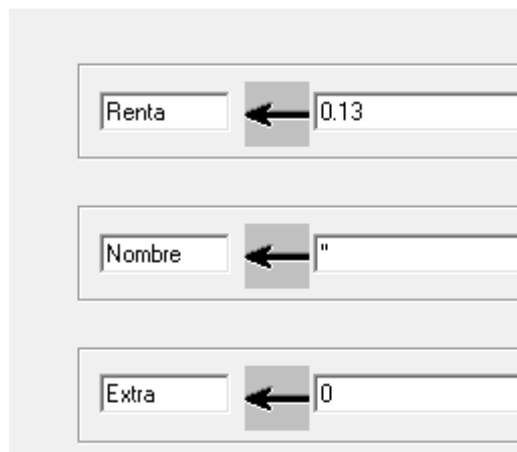
1. Inicio del algoritmo

Asignación



2.

Asignación



3.

4. Mensaje al usuario para solicitar el nombre del empleado

5. Lectura desde teclado del nombre del empleado

6. Mensaje al usuario para solicitar las horas trabajadas

7. Lectura desde teclado de las horas trabajadas

8. Mensaje al usuario para solicitar las horas dobles

9. Lectura desde teclado de las horas dobles

Continúa...

CONTINÚA...

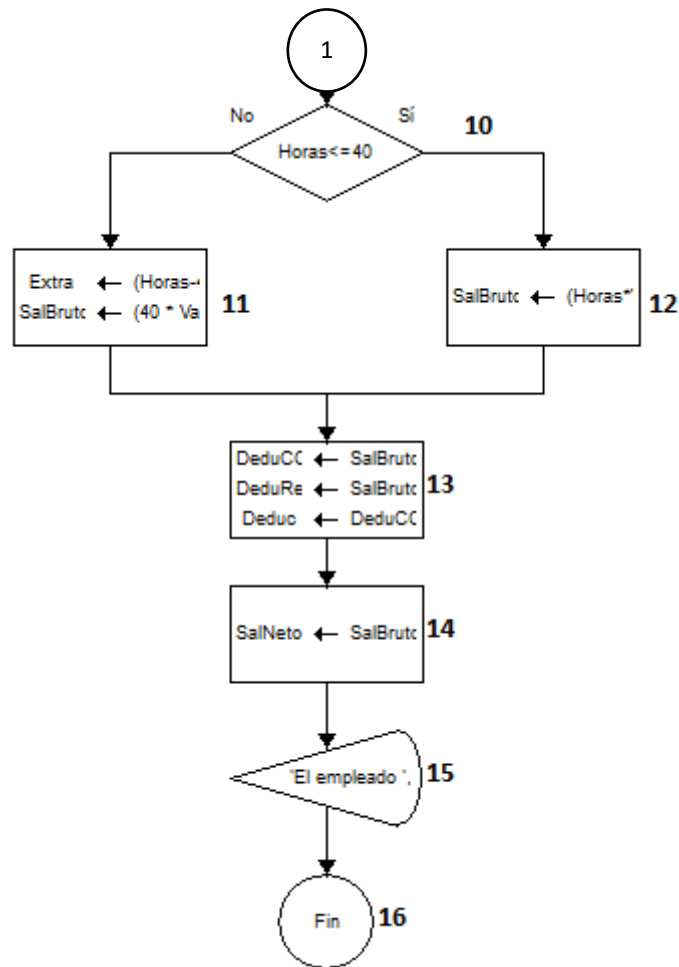


Figura 63 Diagrama de flujo para el cálculo de salario

10. Decisión para determinar si la persona trabajó horas extra. Si trabajó horas extra (más de 40 horas semanales) entonces realiza los cálculos del paso 11.

Asignación

Extra <- (Horas-40) *(1.5*ValHora)

SalBruto <- (40 * ValHora) + Extra +
(HoraDoble* (2*ValHora))

Para comprender la fórmula, haremos una sustitución de valores. Vamos a suponer que un empleado trabajó 42 horas y además trabajó 1 hora doble. Por lo tanto, trabajó 2 horas extra y 1 hora doble. La hora extra se multiplica por 1,5 y por el salario por hora.
Extra= 42-40 * (1,5*2000)
Extra= 2*(3000)
Extra= 6000

El salario bruto serían las 40 horas que trabajó el empleado por el valor de la hora y se le suman las extras y las horas dobles. Es decir:
SalBruto= (40*2000) + 6000 + (1* (2*2000))
SalBruto= 80000+6000 +4000
SalBruto= 90000

Sino trabajó horas extra, es decir, trabajó menos de 40 horas en la semana, se realizan los cálculos del punto 12.

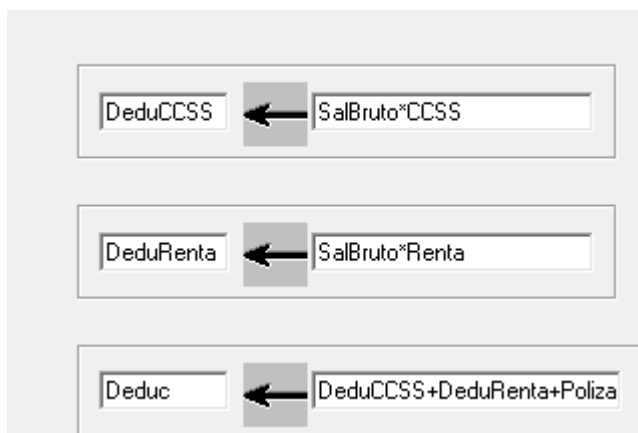
Asignación

$\text{SalBruto} \leftarrow (\text{Horas} * \text{ValHora}) + (\text{HoraDoble} * (2 * \text{ValHora}))$

Para comprender la fórmula, haremos una sustitución de valores. Vamos a suponer que un empleado trabajó 30 horas y además trabajó 6 horas dobles. Como trabajó solo 30 horas, no se calculan las extras. El salario bruto serían las 30 horas que trabajó el empleado por el valor de la hora y se le suman las horas dobles. Es decir:
 $\text{SalBruto} = (30 * 2000) + (6 * (2 * 2000))$
 $\text{SalBruto} = 60000 + 24000$
 $\text{SalBruto} = 84000$

Una vez calculado el salario bruto, se procede a calcular las deducciones de la CCSS, Renta y descontar la póliza, para ello se realizan los cálculos del paso 13. Nótese que el total de las deducciones (Deduc) es la suma del monto de la Caja, de la Renta y de la póliza, si este paso se invirtiera daría un error, puesto que primero se requieren calcular las deducciones para luego sumarlas.

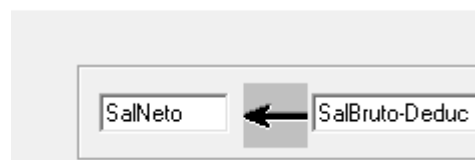
Asignación



Para calcular el salario neto (SalNeto), se le resta al salario bruto (SalBruto) las deducciones (Deduc). Finalmente, en el paso 15 se emite un mensaje con la información del salario del empleado, empleando todas las variables con los montos.

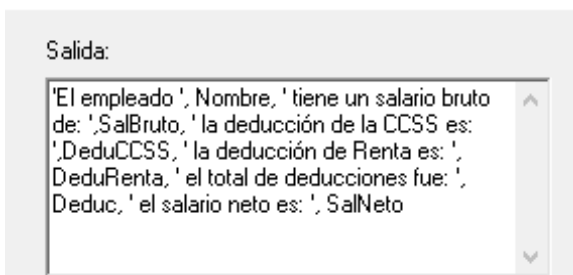
Paso 14

Asignación



Paso 15

Salida por pantalla



El paso 16, finaliza el diagrama.

17. Pseudocódigo con condición para viajar en bus o caminando.

```
Proceso ATiempo
  Definir tiempo Como Caracter;
  tiempo='s';
  Escribir '¿Llegó a tiempo a la estación del bus?';
  Leer tiempo;
  Si (tiempo='s' o tiempo= 'S') Entonces
  ...   Escribir 'Puede tomar el autobus'
  Sino
  ...   Escribir 'Debe irse caminando'
  Fin Si
FinProceso
```

Figura 64 Pseudocódigo para viajar en bus o caminando

18. Pseudocódigo con el promedio de tres números.

```
Proceso Promedio
  Definir num1, num2, num3 Como Entero;
  Definir prom como Real;
  Escribir 'Digite tres números';
  Leer num1, num2, num3;
  prom= (num1+num2+num3)/3
  Escribir 'El promedio es: ',prom;
FinProceso
```

Figura 65 Pseudocódigo para calcular el promedio de tres números

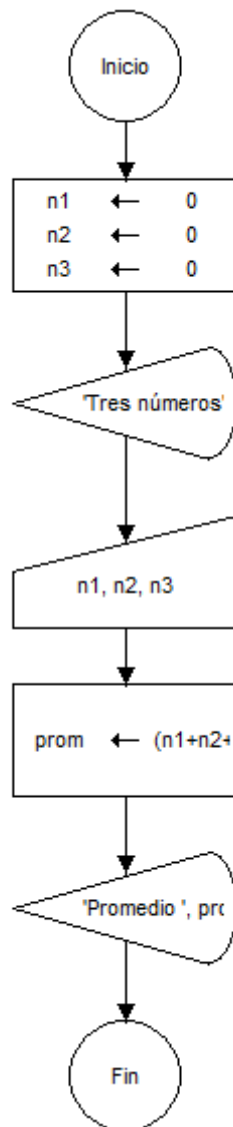


Figura 66 Diagrama de flujo para calcular el promedio de tres números

19. Diagrama de flujo de datos y pseudocódigo para determinar el mayor de dos números.

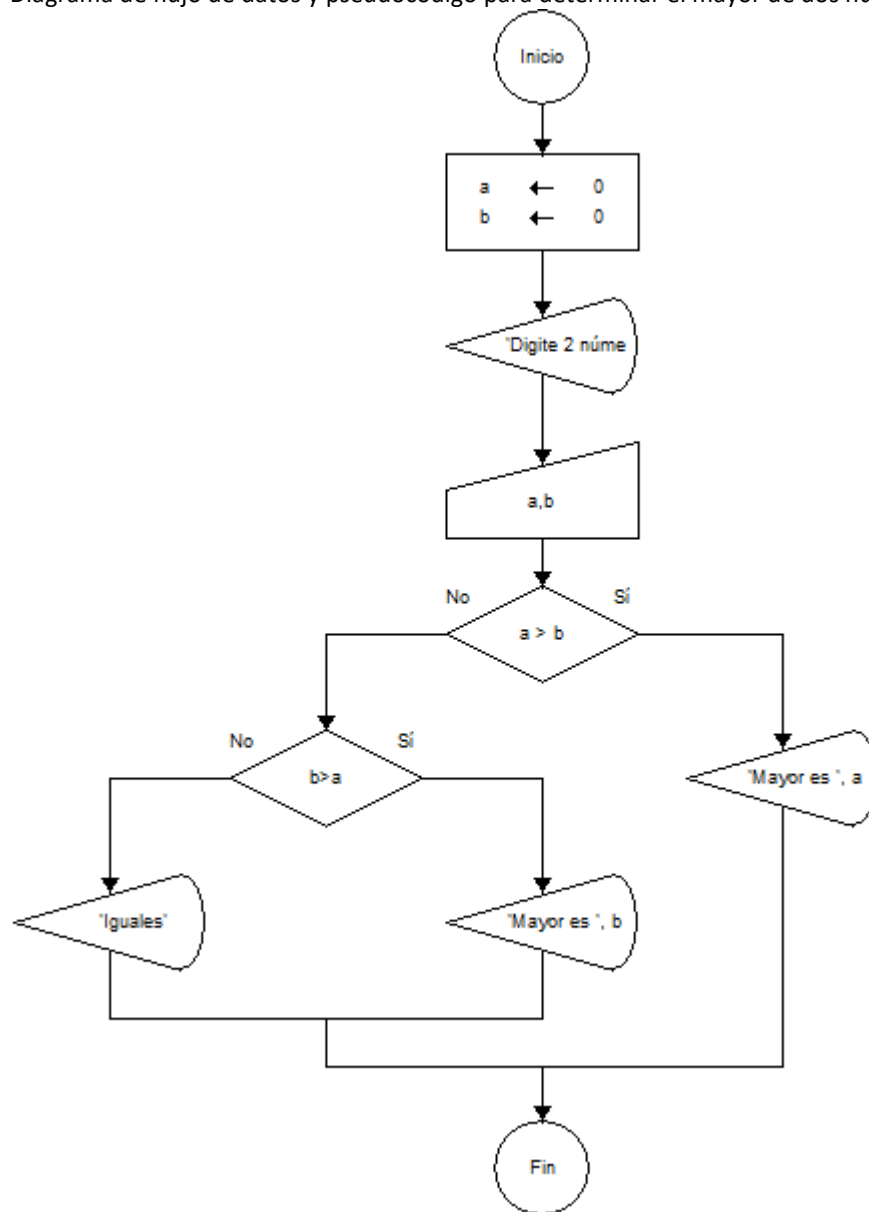


Figura 67 Diagrama de flujo de datos para determinar el mayor de dos números

```

1  Proceso MayorDosNumeros
2      Definir N1, N2 Como Entero;
3      Escribir "Escriba los números";
4      Leer N1, N2;
5      Si N1>N2 entonces
6          Escribir "El ", N1, " es el mayor";
7      FinSi
8      Si N2>N1 Entonces
9          Escribir "El ", N2, " es el mayor";
10     Sino
11         Escribir 'Los números son iguales';
12     Fin Si
13 FinProceso
  
```

Figura 68 Pseudocódigo para determinar el mayor de dos números

20. Pseudocódigo para calcular la edad de una persona

```
1  Proceso Edad_Usuario
2      Definir edad, anio, actual como Entero
3      Escribir 'Digite año nacimiento';
4      Repetir
5          Leer anio;
6      Hasta Que anio>0
7      Escribir 'Digite año actual';
8      Repetir
9          Leer actual;
10     Hasta Que actual>anio
11     edad=actual-anio;
12     Si edad>0 y edad <12 Entonces
13         Escribir 'Tiene ', edad, ' años. Es un niño.';
14     Sino
15         Si edad>=12 y edad <18 Entonces
16             Escribir 'Tiene ', edad, ' años.Es adolescente.';
17         Sino
18             Si edad>=18 Entonces
19                 Escribir 'Tiene ', edad, ' años. Es adulto';
20             Fin Si
21         Fin Si
22     Fin Si
23 FinProceso
```

Figura 69 Pseudocódigo para calcular la edad de una persona

21. Pseudocódigo para determinar el día de la semana

```
1  Proceso Dia_Semana
2      Definir dia como Entero;
3      Escribir 'Escriba un número del 1 al 7';
4      Leer dia;
5      Segun dia Hacer
6          1:
7              Escribir 'Hoy es lunes';
8          2:
9              Escribir 'Hoy es martes';
10         3:
11             Escribir 'Hoy es miércoles';
12         4:
13             Escribir 'Hoy es jueves';
14         5:
15             Escribir 'Hoy es viernes';
16         6:
17             Escribir 'Hoy es sábado';
18         7:
19             Escribir 'Hoy es domingo';
20         De Otro Modo:
21             Escribir 'Error. No está entre 1 y 7';
22         Fin Segun
23 FinProceso
```

Figura 70 Pseudocódigo para determinar el día de la semana

22. Pseudocódigo para determinar el día de la semana, validado con ciclos

```
1  Proceso Dia_Semana
2      Definir dia como Entero;
3      Escribir 'Escriba un número del 1 al 7';
4      Repetir
5          Leer dia;
6      Hasta Que dia>0 Y dia<8
7      Segun dia Hacer
8          1:
9              Escribir 'Hoy es lunes';
10         2:
11             Escribir 'Hoy es martes';
12         3:
13             Escribir 'Hoy es miércoles';
14         4:
15             Escribir 'Hoy es jueves';
16         5:
17             Escribir 'Hoy es viernes';
18         6:
19             Escribir 'Hoy es sábado';
20         7:
21             Escribir 'Hoy es domingo';
22         De Otro Modo:
23             Escribir 'Error. No está entre 1 y 7';
24         Fin Segun
25 FinProceso
```

Figura 71 Pseudocódigo para determinar el día de la semana, validando el día con ciclos

23. Pseudocódigo para mostrar los números pares del 0 al 200.

```
1  Proceso NumerosPares
2      Definir num como entero;
3      num ← 0;
4      Escribir 'Los primeros 100 números pares son: ';
5      Mientras num<=200 Hacer
6          Escribir num;
7          num←num+2;
8      Fin Mientras
9  FinProceso
```

Figura 72 Pseudocódigo mostrar con el ciclo “Mientras” los números pares del 0 al 200

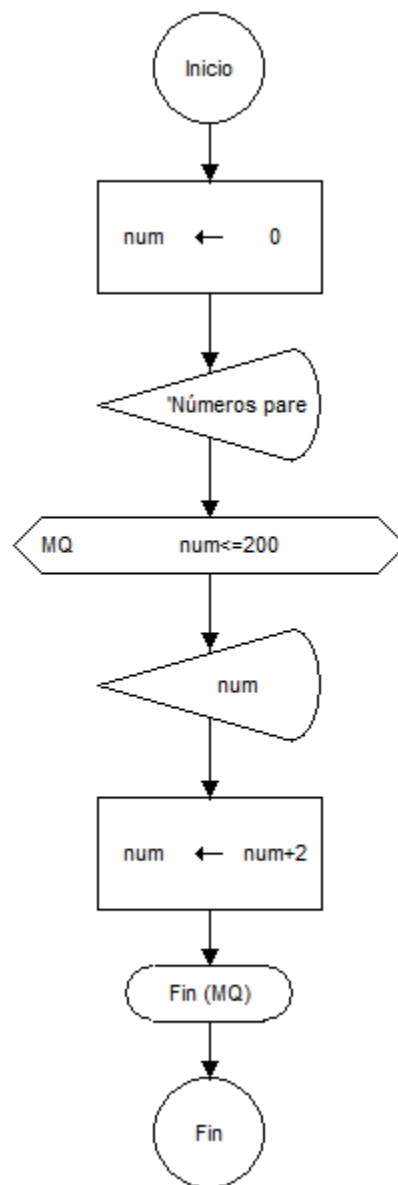


Figura 73 Diagrama de flujo mostrar con el ciclo “Mientras” los números pares del 0 al 200

24. Pseudocódigo para ejecutar “N” veces, el programa del número primo

```
1  Proceso NumerosPrimos
2      Definir i, num como Entero;
3      Definir bandera como Logico;
4      Definir seguir Como Caracter;
5      bandera=Verdadero;
6      Repetir
7          Escribir "Digite el número a valorar:";
8          Leer num;
9          Para i<-2 Hasta num-1 Con Paso 1 Hacer
10             Si num MOD i = 0 Entonces
11                 bandera=Falso;
12                 i=num-1; //obliga a terminar el ciclo
13             Sino
14                 Si Num MOD i = 1 Entonces
15                     bandera=Verdadero;
16                 FinSi
17             FinSi
18          FinPara
19          Si bandera=Falso Entonces
20              Escribir "El número ', num,' no es primo";
21          Sino
22              Escribir "El número ', num,' es primo";
23          Fin Si
24          Escribir "¿Desea repetir el programa?";
25          Leer seguir;
26          Hasta Que seguir="n" o seguir="N"
27  FinProceso
```

Figura 74 Pseudocódigo para determinar un número primo, con el uso de ciclos

25. Ejercicio de Calculadora, Tabla de multiplicar, División, Exponente y Áreas de dos figuras.

```
PROCESO SUMADIVISIONTABLAEXPONENTEAREA
  DEFINIR OPC1, OPC2, I COMO ENTERO;
  DEFINIR NUM1, NUM2, TOTCALC COMO REAL;
  //SELECCIONAR LA OPCIÓN DEL MENÚ, ENTRE 1 Y 5
  ESCRIBIR "BIENVENIDO A LA CALCULADORA, ELIJA UNA OPCIÓN:";
  ESCRIBIR "1. SUMA, 2. TABLA DE MULTIPLICAR, 3. DIVISIÓN, 4. EXPONENTE, 5. ÁREAS";
  REPETIR
    LEER OPC1;
    HASTA QUE OPC1>=1 Y OPC1<=5
    SEGUN OPC1 HACER
      1: ESCRIBIR "SUMA. DIGITE LOS NÚMEROS QUE DESEA SUMAR, PARA FINALIZAR LA
SUMA DIGITE 0";
        REPETIR
          LEER NUM1;
          TOTCALC=TOTCALC+NUM2;
          HASTA QUE NUM1=0
          ESCRIBIR "LA SUMA ES: ", TOTCALC;
        2: ESCRIBIR "TABLA DE MULTIPLICAR. DIGITE EL MULTIPLICANDO Y EL MÁXIMO
MULTIPLICADOR";
          LEER NUM1, NUM2;
          I=0;
          MIENTRAS I<=NUM2 HACER
            ESCRIBIR NUM1, " X", I, "= ", NUM1 * I;
            I=I+1;
          FIN MIENTRAS
        3: ESCRIBIR "DIVISIÓN. DIGITE EL DIVIDENDO Y EL DIVISOR";
          LEER NUM1;
          REPETIR
            LEER NUM2;
            HASTA QUE NUM2!=0 //VALIDACIÓN PARA QUE EL DENOMINADOR NO SEA CERO
            TOTCALC=NUM1/NUM2;
            ESCRIBIR "LA DIVISIÓN ES: ", TOTCALC;
          4: ESCRIBIR "EXPONENTE. DIGITE LA BASE Y EL EXPONENTE";
            LEER NUM1, NUM2;
            TOTCALC=1; //ACUMULADOR DEL RESULTADO
            I=1; //CONTADOR DEL CICLO
            MIENTRAS I<=NUM2 HACER
              TOTCALC=TOTCALC*NUM1;
              I=I+1;
            FIN MIENTRAS
            ESCRIBIR "EL RESULTADO ES: ", TOTCALC;
          5: ESCRIBIR "ELIJA UNA OPCIÓN DE MENÚ PARA CALCULAR EL ÁREA DE: 1.
RECTÁNGULO, 2. ROMBO";
            REPETIR
              LEER OPC2;
              HASTA QUE OPC2>=1 Y OPC2<=2 //VALIDA QUE ELIJA UN NÚMERO ENTRE 1 Y 2

              SI OPC2=1 ENTONCES
                ESCRIBIR "RECTÁNGULO. DIGITE AMBOS LADOS";
                LEER NUM1, NUM2;
                TOTCALC=NUM1*NUM2;
                ESCRIBIR "EL ÁREA DEL RECTÁNGULO ES: ", TOTCALC, " CM2";
              SINO
                ESCRIBIR "ROMBO. DIGITE AMBOS LADOS";
                LEER NUM1, NUM2;
                TOTCALC=(NUM1*NUM2)/2;
                ESCRIBIR "EL ÁREA DEL ROMBO ES: ", TOTCALC, " CM2";
```


FIN SI DE OTRO MODO: ESCRIBIR "NO ELIGIÓ UNA OPCIÓN VÁLIDA"; FIN SEGUN FINPROCESO

Figura 75 Pseudocódigo con menú para: suma, tabla de multiplicar, división, exponente y áreas, con ciclos y validaciones

Glosario de términos

A

Acumulador: variable que almacena varios valores en sí misma. Es empleada frecuentemente en los ciclos.

Aleatorio: valor que se desconoce. En todos los programas generalmente hay funciones que generan número aleatorios que serán utilizados en los algoritmos a manera de trivias. En el caso de PSeInt la función que genera números aleatorios se denomina Azar y en el caso de DFD la función se llama Random.

C

Ciclo: estructuras de programación que permiten a partir de condiciones, que una serie de instrucciones se repitan.

Contador: variable de tipo entera, que se emplea para contar de uno en uno, las veces que se ejecutan las instrucciones. Generalmente se emplea dentro de los ciclos. La única excepción a este conteo de uno en uno, es el ciclo Para, donde el incremento del contador puede modificarse a conveniencia, dependiendo del contexto del problema.

I

Indentación: también conocida como sangría en los documentos escritos, la indentación a nivel de un programa es el margen que se deja entre la instrucción “padre” y las instrucciones “hijas”. La indentación aplica únicamente para el pseudocódigo y no así para el diagrama de flujo de datos. En PSeInt cuando se programan las estructuras de decisión (Si, Sino, Switch) o de iteración (Mientras, Para, Repetir ... Hasta), las instrucciones posteriores que están dentro del ciclo o de las decisiones están indentadas a la derecha para que las personas identifiquen que ese bloque de instrucción se realizará bajo determinadas condiciones.

E

Eficaz: resolución de un algoritmo, pero sin la optimización de variables, contantes, procesos e instrucciones, que tienden a hacer el programa más extenso innecesariamente, lo que conlleva al gasto de recursos computacionales como disco duro y memoria.

Eficiente: resolución de un algoritmo, optimizando las variables, las contantes, y los procesos e instrucciones, que tienden a hacer el programa más concreto, lo que conlleva al ahorro de recursos computacionales como disco duro y memoria.

V

Validar: hacer que el valor de una variable se encuentre en un rango definido para evitar errores en las operaciones del algoritmo. Una validación clásica es que en una división, el denominador sea diferente de cero, pues la división entre cero no existe.