UNIVERSIDAD DE LOS ANDES

FACULTAD DE CIENCIAS FORESTALES Y AMBIENTALES

ESCUELA DE INGENIERÍA FORESTAL

# INTRODUCCIÓN A LA LÓGICA DE PROGRAMACIÓN

Conceptos básicos y ejercicios

**MARÍA ALEJANDRA QUINTERO MÉNDEZ**

Mérida – Venezuela

2016

# CONTENIDO

# PRESENTACIÓN

La idea de escribir este texto surge de la necesidad de disponer de un material adaptado a los requerimientos de la asignatura Informática de Escuela de Ingeniería Forestal de la Facultad de Ciencias Forestales y Ambientales de la Universidad de Los Andes, Mérida - Venezuela. Aunque existen numerosos libros de lógica programación, en la mayoría de los casos el nivel de dificultad está orientado a personas que estudian una carrera relacionada a las Ciencias de la Computación o que han tomado cursos avanzados de matemáticas.

El objetivo principal es presentar las estructuras básicas de programación de manera clara y sencilla, utilizando ejemplos y ejercicios resueltos que faciliten la comprensión de los conceptos. Se utiliza un enfoque algorítmico, por lo que se hace énfasis en la lógica y no en el uso de un lenguaje de programación específico. Los lenguajes de programación evolucionan constantemente y hoy día, gracias a Internet, es posible encontrar innumerables recursos, tutoriales y referencias sobre cualquier lenguaje de programación, razón por la cual este texto se orienta en el aprendizaje de la lógica de programación independientemente del lenguaje utilizado.

El texto está dividido en cuatro capítulos, en el primero se presentan los conceptos más importantes que el lector debe conocer antes de empezar a programar, en el capítulo 2 se aborda el proceso de construcción de algoritmos para computadoras y las estructuras secuenciales de programación, y en los siguientes capítulos se tratan las estructuras de decisión y las estructuras de repetición. En cada capítulo se exponen los fundamentos teóricos del tema con ejemplos ilustrativos, así como también se incluyen ejercicios resueltos y ejercicios propuestos. Muchos de los ejercicios que se presentan son clásicos en el aprendizaje de la programación, porque ayudan a adquirir destrezas específicas en algunos tópicos, otros ejercicios que se plantean en el texto están relacionados a la ingeniería forestal, lo cual facilita la compresión de los temas y permite vincular la programación con la carrera. Este último aspecto no impide que el texto también sea utilizado por estudiantes de otras carreras, pues al presentarse un ejercicio de este tipo se da una breve explicación de los conceptos forestales involucrados.

Por último, quisiera agradecer a todos mis alumnos, que con sus dudas e inquietudes me han motivado a escribir este texto. Espero que sea de utilidad a los futuros estudiantes y a aquellos que desean iniciarse en la programación de computadoras.

**MARÍA ALEJANDRA QUINTERO MÉNDEZ**

# CAPÍTULO 1
# CONCEPTOS BÁSICOS SOBRE PROGRAMACIÓN

## 1.1 Programa

Un programa para computadora o programa informático es una secuencia de instrucciones que le indican a la computadora cómo ejecutar una tarea específica. La mayoría de los programas permiten a la computadora procesar ciertos datos de entrada para convertirlos en datos de salida (información).

Por ejemplo, los datos de entrada de un programa pueden ser el diámetro y la altura de un árbol, después de realizar los cálculos correspondientes el programa puede arrojar como salida una estimación del volumen.

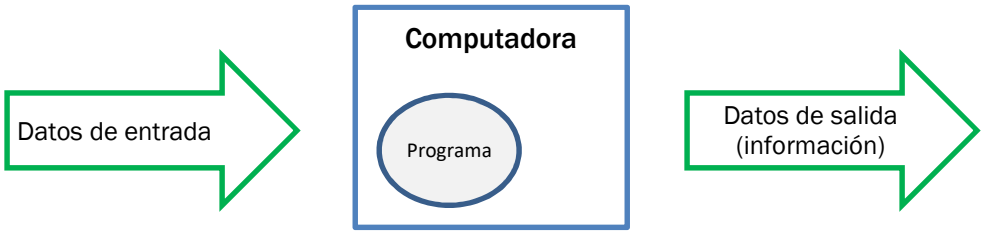



Figura 1.1 El procesamiento de datos

Los programas son muy importantes puesto que permiten utilizar las capacidades de procesamiento de una computadora para dar soluciones a problemas específicos.

## 1.2 Programación

Es la acción de escribir programas de computación con el fin de resolver un determinado problema. El arte de programar implica escribir instrucciones para decirle a la computadora cómo procesar cierta información.

## 1.3 Lenguaje de programación

Es un conjunto de símbolos, palabras y reglas utilizado para definir adecuadamente una secuencia de instrucciones que puedan ser interpretadas y ejecutadas en una computadora. En otras palabras, un lenguaje de programación es un tipo de software que brinda los elementos necesarios para crear nuevos programas.

Actualmente existe una gran cantidad de lenguajes de programación, de diferentes niveles de abstracción (máquina, bajo y alto nivel), con distintos paradigmas de programación (ej. programación estructurada, programación modular, orientados a objetos, orientados a eventos), de propósito general, de propósito específico, entre otras características distintivas. Algunos de los lenguajes de programación más conocidos son C, C++, C#, Java, PHP, Visual Basic, Perl, Python.

## 1.4 Etapas del proceso de programación

Para elaborar un programa de computación es necesario cumplir una serie de etapas que comienzan con la definición y análisis del problema, y conducen a la implantación de un programa que lo soluciona. Los pasos que generalmente sigue un programador a la hora de construir un programa son los siguientes:

**Análisis del problema:** tiene como finalidad conocer y comprender el problema. Es importante que el programador entienda a fondo la naturaleza del problema y cuáles son sus límites (dónde empieza y dónde termina). En esta fase se definen cuáles son los datos necesarios, qué debe hacer el programa y cuáles son los resultados deseados.

Una técnica que ayuda a realizar el análisis en forma ordenada es el análisis de entrada-proceso- salida, también llamado análisis E-P-S, el cual se describe en el capítulo 2.

**Diseño:** consiste en especificar cómo se resuelve el problema. Durante esta fase se establece la secuencia de pasos que debe seguirse para obtener la solución del problema. Esta secuencia de pasos es un esquema que servirá como guía para escribir el código del programa. Es importante probar el funcionamiento de los pasos especificados en el diseño antes de avanzar a la siguiente fase, para lo cual se pueden utilizar algunos datos y se verifica a mano que los resultados sean los esperados. Dos herramientas que se utilizan en el diseño del programa son los algoritmos y los diagramas de flujo.

**Codificación:** es la traducción de cada uno de los pasos especificados en el diseño a un lenguaje de programación, siguiendo las reglas de sintaxis y semántica del mismo. Si el diseño se ha hecho de manera detallada la etapa de codificación debería ser una tarea fácil. El resultado de esta fase será el programa escrito en un lenguaje entendible por la computadora, llamado también **código fuente**.

**Ejecución y pruebas:** consiste en ejecutar (correr) el programa para observar su funcionamiento y encontrar errores. En primer lugar se detecta si el programa tiene errores de sintaxis, estos ocurren cuando no se cumplen las reglas del lenguaje de programación, por ejemplo, se requiere que en instrucción haya una coma pero al transcribir el programador se equivocó y colocó un punto, o hay una palabra mal escrita, etc.; es bastante común cometer este tipo de errores cuando se codifica un programa. En caso de haber errores de sintaxis el lenguaje de programación indica cuáles son y dónde están.

Una vez corregidos todos los errores de sintaxis el programa está listo para funcionar. Durante esta etapa se recomienda probar el programa con una amplia variedad de datos para determinar si hay errores de lógica, este tipo de errores se presenta cuando el programa funciona pero produce resultados erróneos. Si esto ocurre, puede ser necesario revisar el diseño para determinar por qué el programa no se comporta de acuerdo a lo pautado, luego se procede a corregir el código.

El resultado esperado al finalizar los cuatro pasos antes descritos, es un programa de computación que funcione correctamente y que solucione el problema planteado.

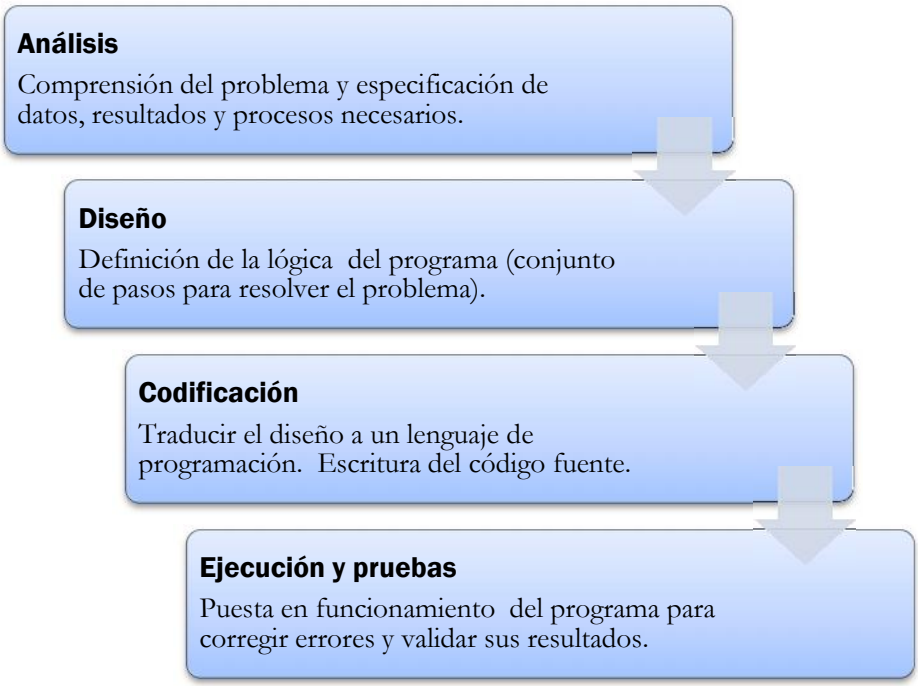



Figura. 1.2 Etapas del proceso de programación.

## 1.5 Algoritmo

Un algoritmo es una secuencia ordenada de pasos que llevan a la solución de un problema o a la ejecución de una tarea. Los pasos deben ser simples, claros y exactos, seguir un orden lógico, y tener un principio y un fin.

Muchas de las actividades que realizan los seres humanos en la vida diaria son algoritmos que han aprendido a seguir; en algunos casos se hacen de manera inconsciente o automática, por ejemplo, lavarse las manos, servirse un vaso de agua, revisar la cuenta de correo electrónico. En otras ocasiones un algoritmo puede incluir una serie de instrucciones a seguir, como preparar una receta de cocina, armar algún objeto (ej. juguete, mueble), buscar una dirección. En cualquier caso el algoritmo indica cada paso en el orden apropiado.

Ejemplo 1

Algoritmo para cepillarse los dientes

1. Dirigirse al lavamanos.
2. Tomar la crema dental y abrirla.
3. Agarrar el cepillo de dientes y colocar crema dental sobre las cerdas.
4. Colocar la crema dental sobre el lavamanos.
5. Abrir el agua y mojar las cerdas del cepillo de dientes.
6. Cerrar el agua.
7. Cepillar la parte externa de los dientes superiores e inferiores con movimientos cortos y verticales.
8. Cepillar la superficie interna de los dientes superiores e inferiores
9. Cepillar las muelas superiores e inferiores con un movimiento de adentro hacia afuera.
10. Cepillar suavemente la lengua.
11. Abrir el agua.
12. Enjuagar el cepillo y colocarlo sobre el lavamanos
13. Enjuagar la boca hasta que no queden restos de crema dental.
14. Cerrar el agua.

Es posible hacer este algoritmo más preciso y detallado, por ejemplo, podrían colocarse todos los pasos a seguir para enjuagar la boca. De esta manera cada paso del algoritmo sería más simple y conciso.

Ejemplo 2

Algoritmo para hacer un sándwich de jamón y queso, con salsa de tomate y mayonesa (opcional).

1. Colocar dos rebanadas de pan en un plato.
2. Si se desea salsa de tomate, colocar un poco de esta salsa en una rebanada de pan y untar con un cuchillo.

3. Si se desea mayonesa, con un cuchillo distribuir un poco de esta salsa en una rebanada de pan.
4. Colocar una lonja de jamón sobre una rebanada de pan.
5. Colocar una lonja de queso sobre la lonja de jamón.
6. Tapar el sándwich, colocando la otra rebanada de pan sobre el queso.
7. Calentar el sándwich.

Los anteriores son ejemplos de algoritmos informales, relacionados a actividades realizadas por los seres humanos en su vida diaria. Existen otro tipo de algoritmos llamados algoritmos para computadora, estos indican cuáles son los pasos que una computadora debe realizar para completar una tarea o resolver un problema concreto.

Los algoritmos para computadora o computacionales pueden incluir estructuras que permiten controlar el flujo de los pasos (estructuras secuenciales, decisiones, ciclos), fórmulas matemáticas y otros tipos de instrucciones. Los elementos básicos para la construcción de algoritmos para computadora que resuelven problemas sencillos son el tema principal de este texto.

## 1.6 Ejercicios propuestos

Diseñar un algoritmo que indique los pasos a seguir para completar cada una de las siguientes actividades:

1) Lavarse las manos.

2) Ir de su casa a la Universidad.

3) Cambiar un neumático dañado.

4) Hacer cotufas (palomitas de maíz).

5) Prestar un libro en la biblioteca.

6) Responder un correo electrónico.

7) Sembrar una planta.

# CAPÍTULO 2
# CONSTRUCCIÓN DE ALGORITMOS PARA COMPUTADORAS

Antes de comenzar a construir algoritmos computacionales es necesario conocer algunos elementos que se utilizan en su desarrollo. Un algoritmo requiere datos, maneja variables y constantes, probablemente necesita realizar operaciones aritméticas procesar los datos, y utiliza operaciones de entrada y salida de datos. Todos estos conceptos se explican en los siguientes apartados, así como también se indica su representación en pseudocódigo que es el lenguaje utilizado para escribir algoritmos computacionales.

Las últimas secciones de este capítulo están dedicadas a la resolución de problemas sencillos, desde el análisis hasta el diseño del algoritmo y su representación gráfica mediante diagramas de flujo.

## 2.1 Datos

Un dato es la representación de un hecho, evento o elemento del mundo real. Por ejemplo, un estudiante puede ser representado por varios datos: nombre, cédula de identidad, carrera que estudia, promedio, edad, sexo, etc.

Puede decirse que los datos son todos aquellos objetos que la computadora es capaz de procesar.

## 2.2 Tipos de datos

Un tipo de datos define un conjunto de valores, las operaciones que se pueden ejecutar con estos valores y la cantidad de memoria necesaria para su almacenamiento. Los lenguajes de

programación manejan diferentes tipos de datos y aunque estos pueden variar de un lenguaje a otro, todos tienen en común los tipos de datos básicos o primitivos.

Los tipos de datos básicos utilizados en computación son los siguientes:

- Entero
- Real
- Carácter
- Cadena de caracteres
- Lógico

**Datos de tipo entero:** son números que no tienen componentes fraccionarios o decimales. Pueden ser negativos o positivos.

Ejemplos de números enteros son:

2 25000
30 -1250

Ejemplos de dato tipo entero: edad de una persona, número de estudiantes en un salón, número de especies de árboles que se encuentran en un área.

**Datos de tipo real:** son números que tienen punto decimal, incluyen números positivos y negativos.

Ejemplos de números reales:

801.35 3550.5

3.0 -100.1999

Ejemplo de datos tipo real: área de un terreno, salario de una persona, altura de un árbol.

**Datos de tipo carácter:** son símbolos que el computador reconoce. Un carácter puede ser una letra (A, B, ....., Z, a, b,.....z), un dígito (1, 2, .....,9) o un símbolo ( ! , @ , # , $ , % , ^ , * , & , +, - , .....). También un espacio en blanco se considera un carácter.

Un dato de este tipo sólo contiene un carácter, y se acostumbra a representarlo entre comillas (“) o entre apóstrofes (‘), dependiendo del lenguaje de programación.

Ejemplos de datos tipo carácter:

Estado civil de una persona (“S”, “C”, “V”)
Sexo (“M”, “F”)
Calidad de la madera (“A”, “B”, “C”)

**Datos de tipo cadena de caracteres***:* contienen una sucesión de caracteres delimitada por comillas o apóstrofes.

Ejemplos de cadenas de caracteres:

"Universidad de Los Andes"
"Prof. Pedro Pérez"
"31 de diciembre de 2016"
"1000 $"

Ejemplos de datos tipo cadena de caracteres: nombre y dirección de una persona, nombre de una especie forestal, una clave.

**Datos de tipo lógico:** son datos que sólo pueden tomar uno de dos valores, verdadero o falso. Se conocen también como datos de tipo booleano. Este tipo de datos se utiliza para representar alternativas a determinadas condiciones (por ejemplo, si / no).

Ejemplo: se desea saber si un árbol presenta cierta condición o enfermedad, en este caso la respuesta será "si" o "no" y puede ser representada mediante un dato de tipo lógico.

## 2.3 Constantes y variables

Los datos que maneja un programa pueden ser constantes o variables.

**Constante:** es un valor o dato que no puede cambiar en la ejecución de un programa, son valores fijos. Una constante tiene dos atributos que la caracterizan: nombre y valor.

Ejemplos:

Pi = 3.1416
Mínimo = 20
Empresa = "Corporación M & M"
EdadMaxima= 50
Clase = "A"
Respuesta = Falso

El valor dado a una constante determina su tipo. Así por ejemplo, la constante de nombre Pi es tipo real ya que su valor 3.1416 es un número real. Las constantes Mínimo y EdadMáxima son de tipo entero, Empresa es una constante de tipo cadena de caracteres, Clase es tipo carácter y Respuesta es de tipo lógico.

**Variable:** es un dato que puede cambiar su valor durante la ejecución de un programa. Una variable representa una dirección o posición de memoria donde se guarda un dato. Todo

dato que vaya a ser introducido en la computadora, y todo valor que se calcule a partir de otros datos en un programa, deben manejarse como variables.

Una variable tiene dos atributos: un nombre que la identifica y el tipo de dato que describe su uso. Algunos ejemplos se muestran en la tabla 2.1

Tabla 2.1 Ejemplos de variables y sus atributos

| Nombre | Tipo |
|---|---|
| **Diámetro** | Real |
| **Nota** | Entero |
| **Ciudad** | Cadena de caracteres |

Una variable que es de cierto tipo solamente puede tomar valores de ese tipo. Por ejemplo, a la variable nota no podría dársele el valor 11.5 porque su tipo es entero y 11.5 es un número real; en este caso se originaría un error al ejecutar el programa.

Una vez que se elige el nombre y el tipo de dato de una variable, es necesario darle un valor. Existen varias maneras de dar valor a las variables, en este texto se trabajará con dos métodos:

- Solicitar al usuario que teclee un valor (operación básica de entrada).
- Asignar a la variable un valor mediante una instrucción del programa (instrucción de asignación).

Estos dos métodos se explican en las siguientes secciones.

## 2.4 Operaciones básicas de entrada/salida

En este texto se trabajará con operaciones de entrada/salida simple, esto es, se asume que los programas diseñados requieren de un usuario, el cual introducirá mediante el teclado los datos necesarios y luego observa en la pantalla los resultados que arrojará el programa una vez procesados los datos. Para realizar estas tareas se utilizan dos instrucciones: lectura y escritura de datos.

**Lectura de datos**: esta operación permite introducir datos a la computadora desde un dispositivo de entrada (ej. teclado, ratón) o desde un archivo. Tal como se indicó anteriormente, la lectura de datos en los ejemplos de este texto se hará desde teclado.

La forma de representar una instrucción de lectura en lenguaje algorítmico (pseudocódigo) es la siguiente:

```
Leer Nombre_Variable
```

Ejemplo:

Supóngase que la siguiente línea es un paso de un algoritmo:

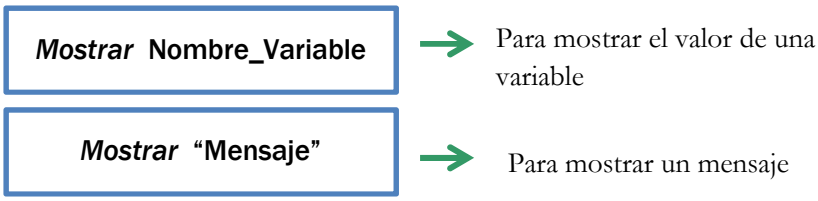
Leer peso

Esta instrucción significa que el usuario debe escribir un valor con el teclado, y éste se almacenará en una variable cuyo nombre es "peso". Si por ejemplo el usuario teclea 61, la variable peso tendrá almacenado en memoria ese valor.

La instrucción de lectura también se puede expresar usando las palabras *Obtener* o *Solicitar*, en vez de la palabra *Leer*, el programador decide cuál prefiere usar.

**Escritura de datos:** permite mostrar la salida (resultados) del programa, y cualquier mensaje que se considere necesario.

La forma de representar una instrucción de escritura en lenguaje algorítmico (pseudocódigo) es la siguiente:



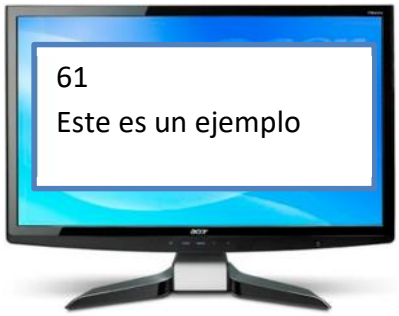

Ejemplos:

Supóngase que las siguientes líneas son instrucciones de un algoritmo

1. Mostrar peso
2. Mostrar "Este es un ejemplo"

La primera instrucción escribirá en pantalla el valor que tiene en memoria la variable peso. La segunda instrucción mostrará el mensaje que está entre comillas. Siempre un mensaje estará delimitado por comillas o apóstrofes, de acuerdo al lenguaje de programación que se utilice.

Para comprender mejor los ejemplos, supóngase que el valor de la variable peso es 61. En ese caso, si esas instrucciones se tradujeran a un lenguaje de programación se observaría en pantalla algo similar a lo siguiente:




La escritura de datos también se puede representar usando las palabras *Escribir* o *Imprimir*, en vez de la palabra *Mostrar*, dependerá del gusto del programador.

## 2.5 Instrucción de asignación

Permite darle un valor a una variable, el cual será almacenado en memoria. La forma de escribir en algoritmo (pseudocódigo) una instrucción de asignación es la siguiente:

***Nombre_variable* = valor o expresión**

Ejemplos:

1) Salario = 85000
   A la variable salario se le asigna el valor 85000

2) A = X+Y
   El resultado de sumar las variables X y Y, se le asigna a la variable A.

3) Especie = "Eucalipto"
   A la variable specie se le asigna el valor Eucalipto. Obsérvese que cuando se asigna una cadena de caracteres, ésta debe ir entre comillas.

4) A=B
   El contenido de la variable B se le asigna a la variable A.

Existen algunas reglas para las instrucciones de asignación que hay que tener presente:

1. Sólo un nombre de variable puede ir a la izquierda del signo igual, porque indica la variable (ubicación de memoria) que cambiará su valor.

2. El valor a la derecha del signo igual puede ser una constante (ejemplos 1 y 3), otra variable (ejemplo 4) o una fórmula o expresión que combine constantes y/o variables (ejemplo 2).

3. La variable y el valor que se le asigna deben ser del mismo tipo de datos. Por ejemplo si en el ejemplo 3, la variable especie es de tipo real se originaría un error porque se le está asignando una cadena de caracteres.

La instrucción de asignación se caracteriza por ser destructiva, esto significa que cuando se asigna un valor a una variable, cualquier valor anterior de esta variable se borra de la memoria y permanecerá solamente el último valor asignado.

Ejemplo: los valores de dos variables A y B son 5 y 8 respectivamente, y se tienen las siguientes instrucciones de asignación para darle valor a una variable C.

C= 10
C= B – A

¿Cuál es el valor de la variable C después de ejecutar estas dos instrucciones?

El valor es 3. Con la primera instrucción se le asigna a C el valor 10, pero al ejecutar la operación B-A, C cambia su valor a 3. El valor 10 se borra de memoria y permanece el último valor asignado.

Alas instrucciones de escritura de datos, lectura de datos y a la instrucción de asignación usualmente se les denomina **estructuras secuenciales**.

## 2.6 Operadores y expresiones aritméticas

Los operadores aritméticos permiten realizar cálculos sobre un conjunto de variables y/o constantes, se utilizan para construir fórmulas o expresiones que se incluyen en un programa como parte del procesamiento de los datos. Los operadores aritméticos básicos se muestran en la Tabla 2.2.

Tabla 2.2 Operadores Aritméticos

| Operador Aritmético | Operación | Ejemplo |
|---|---|---|
| + | Suma | 3 + 2 = 5 |
| - | Resta | 10 – 4 = 6 |
| * | Multiplicación | 8 * 3 = 24 |
| / | División | 7 / 2 = 3.5 |
| \ | División entera | 7 \ 2 = 3 |
| Mod | Resto de una división | 22 Mod 4 = 2 |
| ^ | Potenciación | 5 ^ 2 = 25 |

Los operadores división entera, resto de una división y exponenciación pueden ser diferentes de acuerdo al lenguaje de programación que se utilice. Por ejemplo, en algunos lenguajes la división entera se representa con la palabra Div, el operador Mod con el símbolo de porcentaje % y la exponenciación con doble asterisco **.

**Jerarquía de los operadores aritméticos:** indica el orden en el que deben resolverse las operaciones aritméticas. Es útil cuando hay dos o más operadores en una misma expresión, pues el orden en que se ejecutan las operaciones influye en el resultado. En la tabla 2.3 se indica la jerarquía de los operadores aritméticos.

Tabla 2.3 Jerarquía de los operadores aritméticos.

| Operador | Orden de precedencia |
|---|---|
| ( ) | 1 |
| ^ | 2 |
| * / | 3 |
| \ | 4 |
| Mod | 5 |
| + - | 6 |

Las reglas para resolver expresiones aritméticas con varios operadores son las siguientes:

- Se aplica primero el operador de mayor jerarquía, se resuelve esa operación y luego se aplica el operador que sigue en la jerarquía, y así sucesivamente.
- Si hay paréntesis se resuelven primero las expresiones que estén dentro de éstos, respetando la jerarquía dentro de los paréntesis. Si hay paréntesis dentro de otros se evalúan primero los paréntesis internos.
- Si en una expresión hay dos o más operadores con el mismo nivel de jerarquía u orden de precedencia, se resuelven las operaciones comenzando de izquierda a derecha.

Ejemplos:

En los siguientes ejercicios se resuelven varias expresiones aritméticas y se muestra el orden en el que se ejecutan las operaciones, tomando en cuenta a la jerarquía de los operadores aritméticos.

1) Y = ( 6 * 3 / 2 ) ^ 2
   = (18 / 2) ^ 2
   = 9 ^ 2
   = 81

2) Y = (7 * 8 * (16 mod 3) \ 5 )* 3 – 28

= (7 * 8 * 1 \ 5 )* 3 – 28
= (56 * 1 \ 5) * 3 – 28
= (56 \ 5) * 3 -28
= 11 * 3 -28
= 33 – 28
= 5

3) Y = 3 + 10 * (17 mod 3) \ 5 * 3 - 28
= 3 + 10 * 2 \ 5 * 3 - 28
= 3 + 20 \ 5 * 3 -28
= 3 + 20 \ 15 – 28
= 3 + 1 – 28
= 4 – 28
= -24

**Construcción de expresiones aritméticas:** cuando se programa, frecuentemente se requiere incluir fórmulas matemáticas o ecuaciones para describir el procesamiento de los datos. Estas fórmulas deben representarse mediante expresiones aritméticas escritas en un formato entendible por la computadora (formato de una línea), usando los operadores aritméticos y tomando en cuenta su jerarquía.

Ejemplo:
Escribir la ecuación $Z = \frac{X+3}{X-Y}$ en un formato entendible por la computadora (una línea).

La expresión en formato de una línea es:

$$Z = (x + 3)/(x - y)$$

El numerador y el denominador se encierran entre paréntesis para garantizar que se ejecuten primero estos cálculos antes de efectuar la división.

Si la ecuación se escribe de la siguiente manera:

$$Z = x + 3 / x - y$$

estaría incorrecta porque no representa la ecuación original, primero se efectuaría 3 / x y luego se harían la suma y la resta, según la jerarquía de los operadores aritméticos.

Para escribir correctamente una expresión aritmética, es importante considerar el orden de precedencia o jerarquía de los operadores, en caso contrario el programa podría calcular resultados erróneos.

## 2.7 Análisis de Entrada-Proceso-Salida

Antes de comenzar a diseñar los primeros algoritmos para computadora, se explicará cómo hacer el análisis del problema que es la primera etapa en el proceso de programación. Tal como se mencionó en el capítulo 1, en el análisis es importante comprender el problema, determinar cuáles son los datos necesarios, lo que debe hacer el programa y los resultados que mostrará al usuario. Una manera fácil y ordenada de realizar el análisis del problema, es dividirlo en tres partes: entrada, proceso y salida; a esta técnica se le denomina análisis de Entrada-Proceso-Salida o análisis E-P-S.

**Entrada:** en esta parte se especifican cuáles son los datos necesarios para resolver el problema. A cada dato de entrada se le coloca un nombre, se indica su significado y el tipo de dato más adecuado. Cada uno de estos datos será una variable en el programa.

**Proceso:** se indican los procesos que se van a realizar con los datos de entrada, a través de fórmulas y expresiones escritas de la manera más sencilla posible.

**Salida:** aquí se detallan cuáles son los resultados esperados. Se indica nombre, significado y tipo de dato de cada variable de salida.

Ejemplo

Se quiere construir un programa para calcular el área de un triángulo. El análisis del problema usando la técnica análisis de entrada – proceso – salida, es el siguiente:

*Entrada*

Los datos necesarios para resolver el problema son:

b: base del triángulo. Tipo: Real
h: altura del triángulo. Tipo: Real

*Proceso*

Calcular el área del triángulo usando la ecuación:

$$A = \frac{b \times h}{2}$$

*Salida*

A: área del triángulo. Tipo: real.

## 2.8 Construcción de algoritmos computacionales simples

En las secciones anteriores se han detallado todos los elementos básicos para construir algoritmos computacionales que resuelvan problemas sencillos. Este tipo de algoritmos, en la mayoría de los casos, comienzan con la lectura de los datos de entrada, continúan con el

procesamiento de los datos el cual puede incluir instrucciones de asignación con expresiones aritméticas, y finalizan con la escritura de los resultados para que puedan ser visualizados por el usuario del programa.

Los algoritmos se escriben en pseudocódigo, usando palabras o instrucciones escritas de una manera que facilite la posterior codificación a un lenguaje de programación.

Ejemplo

Algoritmo para calcular el área de un triángulo.

0. Inicio
1. Leer base del triángulo (b)
2. Leer altura del triángulo (h)
3. A = b * h/2
4. Mostrar el área (A)
5. Fin

Obsérvese que este algoritmo comienza con la lectura de los datos necesarios para resolver el problema o datos de entrada (pasos 1 y 2), estos datos serán introducidos por el usuario mediante teclado cuando el algoritmo se codifique en un lenguaje de programación; en las instrucciones de lectura se recomienda colocar entre paréntesis el nombre de las variables donde se almacenarán los datos. Luego en el paso 3, se describe el proceso colocando la ecuación correspondiente. Una vez especificado el proceso, está la instrucción *Mostrar* (paso 4), con la cual se indica que el resultado (área del triángulo) debe ser presentado al usuario del programa, además en esta instrucción se especifica entre paréntesis el nombre de la variable donde se almacena el resultado, esto facilitará la codificación.

Otra manera de escribir el algoritmo de forma más resumida es la siguiente:

0. Inicio
1. Leer b
2. Leer h
3. A = b * h / 2
4. Mostrar A
5. Fin

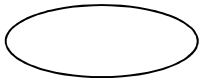
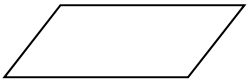
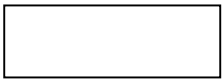
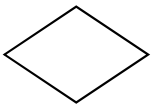
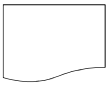
Acá no se especifica el significado de las variables de entrada y salida. En este texto se utilizará la primera forma porque facilita la posterior codificación en un lenguaje de programación. La segunda forma puede ser más conveniente cuando se tiene más experiencia programando y el algoritmo solo sirve como un primer esquema de lo que será el programa.

## 2.9 Diagrama de flujo

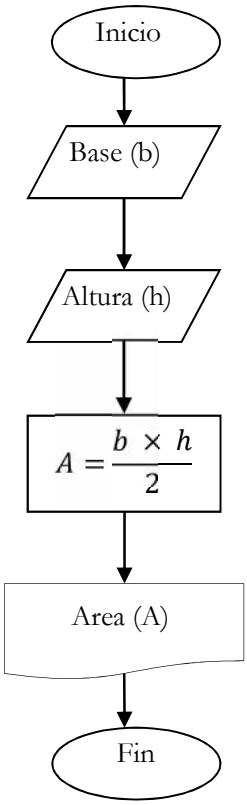
Un diagrama de flujo o flujograma es una representación gráfica de un algoritmo. Esta herramienta de diseño utiliza símbolos para indicar acciones y éstos se conectan a través de flechas que muestran el flujo o secuencia del programa.

En la tabla 2.4 se muestran los símbolos más utilizados en la construcción de un diagrama de flujo.

Tabla 2.4 Símbolos usados en los diagramas de flujo

| Símbolo | Significado |
|---|---|
|  | Inicio / Fin del programa |
|  | Entrada / Salida de datos |
|  | Procesos |
|  | Decisión |
|  | Conector para una misma página |
|  | Conector de página diferente |
|  | Salida de datos |

Ejemplo
Diagrama de flujo para calcular el área de un triángulo.




Obsérvese que el diagrama de flujo anterior incluye los mismos pasos que el algoritmo para calcular el área de un triángulo, pero cada paso se representa gráficamente a través de un símbolo cuya forma depende del tipo de instrucción.

## 2.10 Ejercicios resueltos

1) En la siguiente lista de variables, colocar el tipo de dato más conveniente de acuerdo al contenido que almacenará cada variable.

| Nombre de la variable | Contenido | Tipo de dato |
|---|---|---|
| PorcHMadera | Porcentaje de humedad en una pieza de madera | **Real** |
| Nom_Cien | Nombre científico de una especie forestal | **Cadena de caracteres** |
| NumRamas | Número de ramas en un árbol | **Entero** |
| TipoTrat | Tipo de tratamiento aplicado a una especie forestal (F, P, A) | **Carácter** |
| EdadArbol | Edad de un árbol | **Entero** |
| Costo | Costo de plantar un árbol | **Real** |

2) Determinar qué valor tienen las variables X, Y y Z después de ejecutar las siguientes instrucciones de asignación

| Instrucción | Operaciones | Valor de las variables | | |
|---|---|---|---|---|
| | | **X** | **Y** | **Z** |
| **X=3** | | 3 | | |
| **Y= 45 - X** | Y = 45 – 3 | | 42 | |
| **Z= Y / 7** | Z= 42 / 7 | | | 6 |
| **Y = X + Z** | Y = 3 + 6 | | 9 | |
| **Z = Y - Z** | Z = 9 - 6 | | | 3 |
| **X= X * 5** | X= 3 * 5 | 15 | | |

Los valores de las variables después de ejecutar las instrucciones de asignación son:

X = 15; Y = 9; Z = 3. Estos son los valores que quedan almacenados en memoria, los valores anteriores de las variables se borran al asignar nuevos valores.

3) Evaluar las siguientes expresiones aritméticas

a) 10 Mod 4 + 4 – 3 ^ 3 * 2

Solución
10 Mod 4 + 4 – 3 ^ 3 * 2
10 Mod 4 + 4 – 27 * 2
10 Mod 4 + 4 – 54
2 + 4 - 54
6 – 54
- 48

b) 25 * 3 – 47 \ (13 – 3 ^ 2)

Solución
25 * 3 – 47 \ (13 – 3 ^ 2)
25 * 3 – 47 \ (13 – 9)
25 * 3 – 47 \ 4
75 – 47 \ 4
75 – 11
64

c) (7 * 8 * (18 mod 5) / 5) * 3 – 28

Solución
(10 * 8 * (18 mod 5) / 5) * 3 – 28
(10 * 8 * 3 / 5) * 3 – 28
(80 * 3 / 5) * 3 – 28
(240 / 5) * 3 – 28
48 * 3 – 28
144 – 28
116

d) 11 + 3 / 2 + 5

Solución
11 + 3 / 2 + 5
11 + 1.5 + 5
12.5 + 5
17.5

e) (11 + 3) / 2 + 5

Solución
(11 + 3) / 2 + 5
14 /2 + 5
7 + 5
12

f) (11 + 3) / (2 + 5)

Solución
(11 + 3) / (2 + 5)
14 / (2 + 5)
14 / 7
2

g) 35 / 7 * 4 – 21 + 2 ^ 3 * 3

Solución
35 / 7 * 4 – 21 + 2 ^ 3 * 3
35 / 7 * 4 – 21 + 8 * 3
5 * 4 – 21 + 8 * 3
20 – 21 + 8 * 3
20 – 21 + 24

-1 + 24
23

4) Escribir las siguientes expresiones en formato de una línea

a) $X = 5 + \frac{Y}{3} - Z$

En formato de una línea: X = 5 + Y/3 – Z

b) $Y = \frac{(25-X)^2}{z+1}$

En formato de una línea: Y = (25 – X) ^ 2 / (z + 1)

c) $Z = \frac{3\,(X+2Y)-4}{X} + Y$

En formato de una línea: ( 3 * ( X + 2 * Y) – 4) / X + Y

5) Realizar análisis E-P-S, algoritmo y diagrama de flujo para calcular el salario semanal de un trabajador tomando en cuenta las horas trabajadas y el salario por hora.

**Análisis E-P-S**

*Entrada*

Nom: nombre del trabajador. Tipo: cadena de caracteres
nh: número de horas trabajadas en la semana. Tipo: Real
sh: salario por hora. Tipo: Real

*Proceso*

Calcular el salario del trabajador usando la ecuación:

S = nh x sh

*Salida*

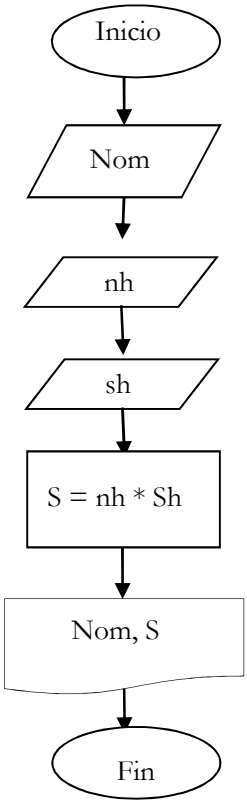
S: salario semanal del trabajador. Tipo: Real.

**Algoritmo**

0. Inicio
1. Leer nombre del trabajador (Nom)
2. Leer número de horas trabajadas en la semana (nh)
3. Leer salario por horas (sh)

4. S = nh * Sh
5. Mostrar nombre (nom) y salario semanal del trabajador (S)
6. Fin

**Diagrama de flujo**




6) Realizar análisis E-P-S, algoritmo y diagrama de flujo para calcular el precio total a pagar por la compra de un producto si se tienen como datos el precio de venta por unidad y la cantidad que se desea comprar. En el cálculo del precio total se debe considerar un descuento del 15% y un 12% de IVA.

**Análisis E-P-S**

*Entrada*

pu: precio unitario del producto. Tipo: Real
cant: cantidad a comprar. Tipo: Entero

*Proceso*

Calcular el precio sin descuento e IVA (Subtotal):

S = pu x cant

Calcular el descuento

D = S x 0.15

Calcular IVA

IVA = (S – D) x 0.12

Calcular precio total

PT = S – D + IVA

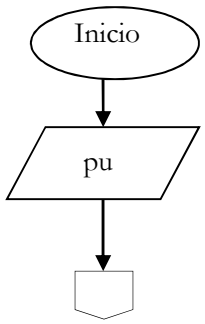
*Salida*

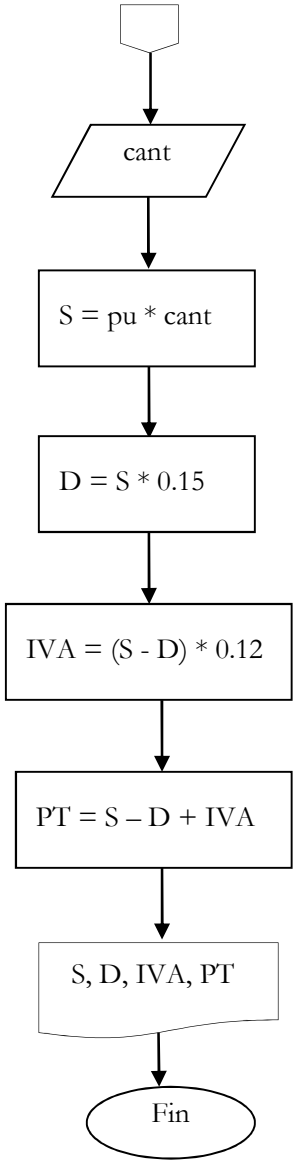
S: Subtotal;. Tipo: Real.
D: Descuento. Tipo: Real.
IVA: Impuesto. Tipo: Real.
PT: Precio total. Tipo: Real.

**Algoritmo**

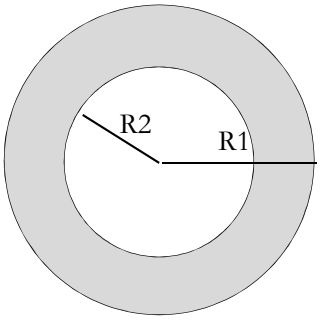
0. Inicio
1. Leer precio unitario del producto (pu)
2. Leer cantidad a comprar (cant)
3. S = pu * cant
4. D = S * 0.15
5. IVA = (S – D) * 0.12
6. PT = S – D + IVA
7. Mostrar subtotal (S)
8. Mostrar descuento (D)
9. Mostrar impuesto (IVA)
10. Mostrar precio total (PT)
11. Fin

**Diagrama de flujo**






7) Realizar análisis E-P-S, algoritmo y diagrama de flujo para calcular el área de una corona circular (área sombreada de la figura) teniendo como datos el radio de la circunferencia mayor y el radio de la circunferencia menor.




$$A = \pi\,(R1^2 - R2^2)$$

**Análisis E-P-S**

*Entrada*

R1: radio de la circunferencia mayor. Tipo: Real
R2: radio de la circunferencia menor. Tipo: Real

*Proceso*

Calcular el área:

$$A = \pi (R1^2 - R2^2)$$

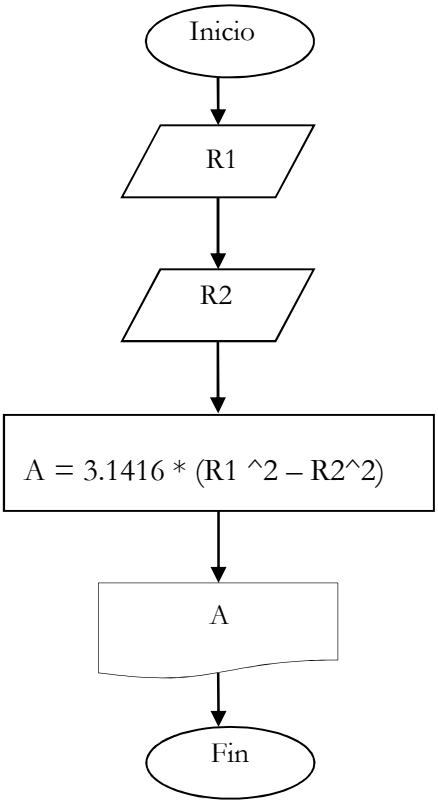
*Salida*

A: área de la corona circular. Tipo: Real.

**Algoritmo**

0. Inicio
1. Leer radio de la circunferencia mayor (R1)
2. Leer radio de la circunferencia menor (R2)
3. A = 3.1416 * ( R1 ^2 – R2^2)
4. Mostrar área (A)
5. Fin

**Diagrama de flujo**



8) Se desea calcular el precio de un terreno rectangular, teniendo como datos el ancho (en metros), la longitud (en metros) y el precio del metro cuadrado. Realizar análisis E-P-S, algoritmo y diagrama de flujo.

**Análisis E-P-S**

*Entrada*

a: ancho del terreno. Tipo: Real.
l: largo del terreno. Tipo: Real
pm: precio del metro cuadrado. Tipo: Real.

*Proceso*

Calcular el área del terreno:

$At = a \times l$

Calcular el precio

$P = pm \times At$

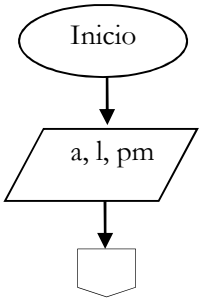
*Salida*

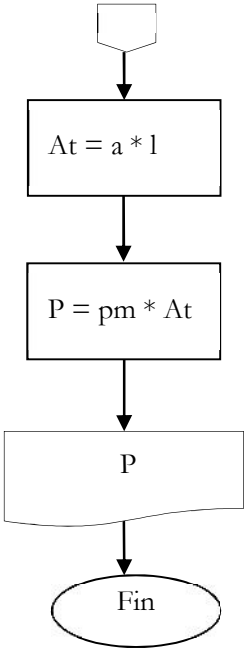
P: precio del terreno. Tipo: Real.

**Algoritmo**

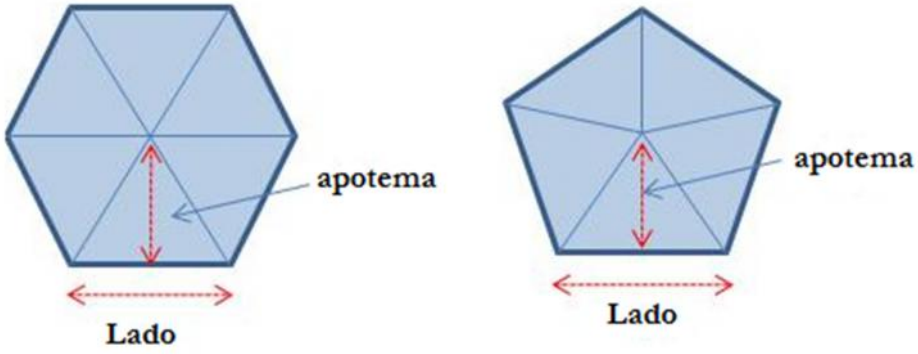
0. Inicio
1. Leer ancho del terreno (a)
2. Leer longitud del terreno (l)
3. Leer precio del metro cuadrado (pm)
4. At = a * l
5. P = pm * At
6. Mostrar precio del terreno (P)
7. Fin

**Diagrama de flujo**






9) El área de un polígono regular es igual a la mitad del producto de su perímetro por su apotema (en la figura se muestran dos ejemplos de polígonos regulares). Realizar un algoritmo que teniendo como datos el número de lados del polígono, la longitud de un lado y su apotema, calcule el área.




Antes de escribir el algoritmo, se recomienda realizar el análisis del problema para facilitar la definición de los pasos.

**Análisis E-P-S**

*Entrada*

nl: número de lados del polígono. Tipo: Entero.
L: longitud del lado del polígono. Tipo: Real.
a: apotema del polígono. Tipo: Real.

*Proceso*

Calcular el perímetro del polígono (suma de sus lados):

P = nl x L

Calcular el área

$$Ar = \frac{P\ x\ a}{2}$$

*Salida*

Ar: área del polígono. Tipo: Real.

**Algoritmo**

0. Inicio
1. Leer número de lados del polígono (nl)
2. Leer longitud del lado (L)
3. Leer apotema (a)
4. P = nl * L
5. Ar = P * a / 2
6. Mostrar área del polígono (Ar)
7. Fin

10) Se desea diseñar un programa que dada una cantidad expresada en metros la convierta a pulgadas, kilómetros y pies, de acuerdo con las siguientes equivalencias: 1 metro = 39.3701 pulgadas, 1 Km = 1000 metros, 1 pie = 12 pulgadas.

**Análisis E-P-S**

*Entrada*

Cm: cantidad en metros. Tipo: Real

*Proceso*

Calcular cantidad en pulgadas

Cpulg = Cm x 39.3701

Calcular cantidad en kilómetros

CKm =Cm /1000

Calcular cantidad en pies

Cpies = Cpulg / 12

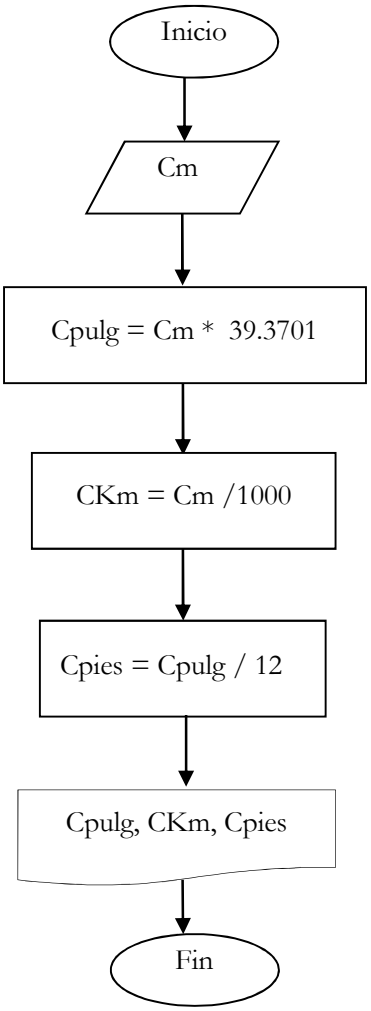
*Salida*

Cpulg: cantidad en pulgadas. Tipo: Real.

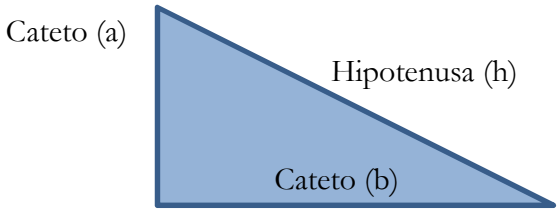
CKm: cantidad en kilómetros. Tipo: Real.
Cpies: cantidad en pies. Tipo: Real.

El diseño se puede hacer usando un algoritmo o un diagrama de flujo; en este problema se utilizará un diagrama de flujo, queda como ejercicio para el lector hacer el algoritmo correspondiente.

**Diagrama de flujo**




11) Realizar un algoritmo que calcule la hipotenusa de un triángulo rectángulo.



**Análisis E-P-S**

*Entrada*

a: longitud del cateto a. Tipo: Real.

b: longitud del cateto b. Tipo: Real.

*Proceso*

Calcular la hipotenusa usando el teorema de Pitágoras:

$$h = \sqrt{a^2 + b^2}$$

*Salida*

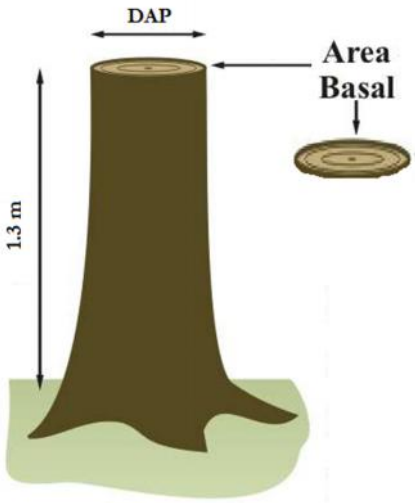
h: hipotenusa. Tipo: Real.

**Algoritmo**

0. Inicio
1. Leer longitud del cateto a (a)
2. Leer longitud del cateto b (b)
3. h = raíz ( a^2 + b^2)
4. Mostrar hipotenusa (h)
5. Fin

En el paso 4 del algoritmo se usa la función raíz() para expresar la raíz cuadrada. Cuando el algoritmo se codifique en un lenguaje de programación se utilizará una función para realizar esta operación. En algunos lenguajes se usa la función Sqr o Sqrt para calcular la raíz cuadrada de un número o expresión.

12) El área basal de un árbol es al área en metros cuadrados del corte transversal de un árbol a la altura del pecho, es decir, medido a 1,30 m.




Se obtiene a partir de la fórmula del área del círculo, expresada como:

$$AB = \frac{\pi}{4}\left(\frac{dap^2}{100}\right)$$

donde dap es el diámetro a la altura de pecho medido en centímetros (cm).

Elabore un algoritmo que calcule el área basal de un árbol.

**Algoritmo**

0. Inicio
1. Leer diámetro a la altura de pecho (dap)
2. AB = 3.1416 / 4 * dap^2/100
3. Mostrar área basal (AB)
4. Fin

Se deja como ejercicio para el lector verificar la fórmula del área basal escrita en formato de una línea ¿está correctamente escrita la expresión?

13) Realizar análisis E-P-S, algoritmo y diagrama de flujo para calcular el crecimiento periódico anual en volumen de un árbol durante un periodo de tiempo expresado en años. Este indicador se calcula como la diferencia de volumen del árbol entre el comienzo y el final del período de tiempo, dividido por el número de años del período.

**Análisis E-P-S**

*Entrada*

V1: volumen al inicio del período de tiempo. Tipo: Real.
V2: volumen al final del período de tiempo. Tipo: Real.
na: número de años considerado. Tipo: Entero.

*Proceso*

Calcular el crecimiento periódico anual

$$CPA = \frac{V2 - V1}{na}$$

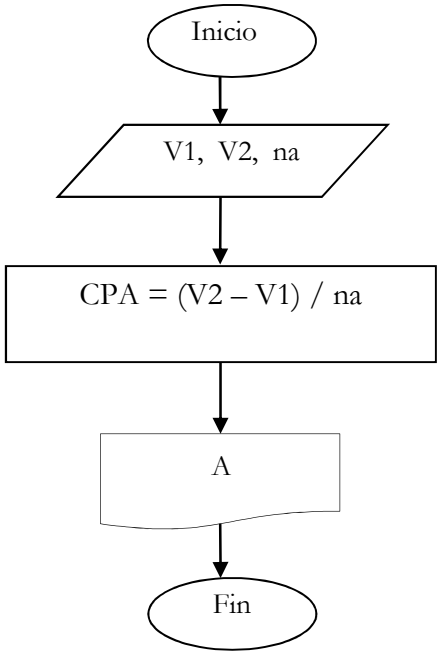
*Salida*

CPA: Crecimiento periódico anual en volumen. Tipo: Real.

**Algoritmo**

0. Inicio
1. Leer volumen al inicio del período de tiempo (V1)
2. Leer volumen al final del período de tiempo (V2)
3. Leer número de años considerado (na)
4. CPA = (V2 – V1) / na
5. Mostrar crecimiento periódico anual en volumen (CPA)
6. Fin

**Diagrama de flujo**




Obsérvese que en este diagrama de flujo se colocaron todos los datos de entrada en un mismo símbolo, esto también es válido hacerlo y en problemas largos permite ahorrar tiempo y espacio a la hora de hacer el diagrama. Sin embargo, cuando se comienza a programar, puede ser útil colocar cada lectura de datos en un símbolo para facilitar la codificación al lenguaje de programación.

14) Una forma de calcular la cantidad de carbono almacenada en un árbol es a partir del peso seco de la madera. Se estima que el carbono almacenado es cercano a la mitad del peso seco. Una ecuación básica para determinar el peso seco de la madera de un árbol es:

Peso seco = densidad de la madera x densidad del agua x volumen del árbol

Realizar un algoritmo que calcule la cantidad de carbono almacenada en un árbol de cierta especie, a partir de la densidad de la madera y el volumen del árbol. Se asume que la densidad del agua es una constante igual a 1000 $Kg/m^3$.

**Análisis E-P-S**

*Entrada*

Esp: nombre de la especie. Tipo: Cadena de caracteres.
dm: densidad de la madera. Tipo: Real.
V: volumen del árbol. Tipo: Real

*Proceso*

Calcular el peso seco de la madera:

$Ps = dm \times 1000 \times V$

Calcular la cantidad de carbono

$C = 0.5 \times Ps$

*Salida*

C: cantidad de carbono almacenada en el árbol. Tipo Real.

**Algoritmo**

0. Inicio
1. Leer Especie (Esp)
2. Leer densidad de la madera en $Kg/m^3$ (dm)
3. Leer volumen del árbol en $m^3$ (v)
4. Ps = dm * 1000 * V
5. C = 0.5 * Ps
6. Mostrar la cantidad de carbono en Kg (C)
7. Fin

En este algoritmo el manejo de las unidades se hace mediante las instrucciones de entrada y salida, se le indica al usuario las unidades de los datos de entrada (densidad de la madera en $Kg/m^3$, volumen en $m^3$) y de los resultados (carbono en Kg), en concordancia a la unidad utilizada para la densidad del agua que es una constante (1000 $Kg/m^3$). De esta manera, se le advierte al usuario del programa las unidades que debe usar para evitar que los resultados sean erróneos. Igualmente, es importante observar que en las ecuaciones del algoritmo no se colocan unidades, pues al momento de codificar, el lenguaje de programación solo acepta en una expresión aritmética: variables, constantes, funciones y operadores; no acepta unidades.

## 2.11 Ejercicios propuestos

1) A continuación se da una lista de variables y una descripción de su contenido. Asignar a cada variable el tipo de dato más conveniente.

| **Variable** | **Contenido** | **Tipo** |
|---|---|---|
| Suma | Resultado de sumar tres números enteros | |
| Saldo | Monto en bolívares disponible en una cuenta bancaria. | |
| Nom | Nombre de una persona | |
| Enfermo | Indica si una persona presenta o no una determinada enfermedad | |
| NumArbol | Cantidad de árboles de una misma especie presentes en un área determinada | |
| Edo_Fit | Estado fitosanitario de un árbol (B, R, M) | |

2) ¿Qué valor tiene la variable Z después de ejecutar las siguientes operaciones de asignación?

a)
```
X = 3
Y = 4
Z = X - Y
```

b)
```
Z= 5
X = 2 + Z
Z = 3
Z = Z + X
```

3) ¿Qué valor tienen las variables Z y W después de ejecutar las siguientes operaciones de asignación?

a)
```
Z = 8
W = 4
Y = 2
W = W + 5
Z = Z - Y + W
```

b)
```
Z = 4
W = 6
Y = Z + W
Z = W + Y
W = Z + W
```

4) Obtener el valor de cada una de las siguientes expresiones aritméticas y mostrar el orden de ejecución si hay más de un operador.

a) 69 \ 8
b) 69 mod 8
c) 12 \ 3
d) 12 mod 3

e) 7 * 10 – (5 mod 3) * 4 + 9
f) (7 * (10 – 5) mod 3) * 4+ 9
g) (12 + 3) + 8 * 3 mod 5 + 4 * 3
h) A * B / C * C -1 si A = 4, B = 3, C = 2

5) Escribir las siguientes expresiones en formato de una línea

a) $y = \frac{a+b}{c-a}$ b) $w = \frac{x^2 + y^2}{z^2}$

c) $d = \sqrt{(x-y)^2 + (z-w)^2}$ d) $y = \frac{b}{3-a}(b+5a)$

6) Realizar análisis E-P-S, algoritmo y diagrama de flujo para calcular el salario mensual de un trabajador, si se tienen como datos el número de horas trabajadas, el salario por hora hora y además se sabe que se le descuenta el 10% por concepto de caja de ahorros. El trabajador también recibirá un bono de 600 Bs. por cada hijo.

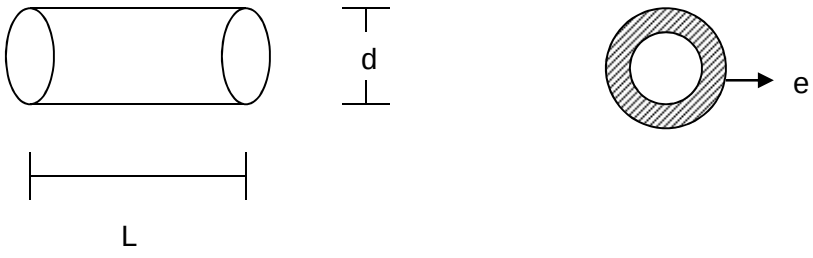
7) Realizar análisis E-P-S, algoritmo y diagrama de flujo para convertir una temperatura dada en grados centígrados a grados Fahrerenheit y Kelvin. Las fórmulas de conversión son:

$$F = \frac{9}{5}C + 32$$

$K = C + 273.1$

8) Realizar análisis E-P-S, algoritmo y diagrama de flujo para calcular el área y el volumen de un cubo.

9) Realizar análisis E-P-S, algoritmo y diagrama de flujo para calcular el volumen de una tubería de acero. Se tienen como datos el diámetro (d), el espesor (e) y la longitud de la tubería (L).



10) Realizar análisis E-P-S, algoritmo y diagrama de flujo para calcular la calificación promedio de un alumno que presenta tres exámenes.

11) Escribir un algoritmo para intercambiar los valores de dos variables numéricas.

12) Realizar un diagrama de flujo que dados dos números reales, calcule la suma, resta y multiplicación de dichos números.

13) Escribir un algoritmo que dado un valor de x, calcule el valor del siguiente polinomio:

$p(x) = 3X + 5X2 – 1.5 X + 2$

14) Escribir un algoritmo que dada una cantidad expresada en dólares calcule su equivalente en bolívares.

15) Escribir un algoritmo que dada un cantidad expresada en minutos, determine su equivalente en segundos y días.

16) Escribir un algoritmo que calcule el área de un triángulo si se conocen las coordenadas en el plano XY de los puntos P1, P2 y P3, los cuales corresponden a los vértices del triángulo.

# CAPÍTULO 3

# ESTRUCTURAS DE DECISIÓN

Las estructuras de decisión se utilizan en un algoritmo cuando se debe elegir entre dos o más alternativas, cada una de las cuales representan pasos o instrucciones diferentes que conducen a la solución del problema considerando distintas condiciones.

En una estructura de decisión se evalúa una expresión lógica la cual puede ser verdadera o falsa, dependiendo de su valor, se ejecutan las instrucciones correspondientes. Por ejemplo, supóngase que en un algoritmo que calcula la nota definitiva de un estudiante se desea indicar si está reprobado o aprobado; en ese caso la condición está relacionada a la nota definitiva, si ésta es 10 o más el algoritmo debe señalar que el estudiante está aprobado, y si es menor a 10 debe indicar que está reprobado (en una escala de 0 a 20). Para resolver situaciones como esta, es necesario utilizar una estructura de decisión.

En este capítulo se muestra cómo utilizar estructuras de decisión en la resolución de problemas mediante algoritmos. Primeramente se explican los elementos necesarios para construir las condiciones que utilizan las estructuras de decisión, estos son: operadores relacionales, operadores lógicos y expresiones lógicas. Luego, se definen y ejemplifican cuatro tipos de estructuras de decisión: simple, doble, anidadas y múltiple.

## 3.1 Operadores relacionales

Son operadores que permiten hacer comparaciones entre constantes y variables. En la tabla 3.1 se muestran los operadores relacionales usados en programación, su significado y el equivalente en notación matemática.

Tabla 3.1 Operadores relacionales

| Operador | Significado | Equivalente matemático |
|---|---|---|
| > | Mayor que | > |
| < | Menor que | < |
| >= | Mayor o igual que | ≥ |
| <= | Menor o igual que | ≤ |
| = | Igual a | = |
| <> | Diferente a | ≠ |

## 3.2 Operadores lógicos

Los operadores lógicos básicos son AND, OR y NOT. Estos operadores se aplican a operandos lógicos (booleanos), que son variables o constantes que pueden tener el valor verdadero o falso.

**Operador AND (Y):** relaciona dos operandos booleanos. Da como resultado un valor verdadero (V) si los dos operandos son verdaderos (V); en caso contrario proporciona un resultado falso (F).

El operador AND se usa de la siguiente manera en una expresión:

```
Operando 1  AND  Operando 2
```

Las posibles combinaciones de resultados se muestran en la Tabla 3.2.

Tabla 3.2. Resultados de un operador AND.

| Operando 1 | Operando 2 | Resultado |
|---|---|---|
| V | V | **V** |
| V | F | **F** |
| F | V | **F** |
| F | F | **F** |

Ejemplo: sean X y Y variables lógicas cuyo valor es falso (F) y verdadero (V) respectivamente, determinar el valor de la expresión X AND Y.

X AND Y = F AND V = F

**Operador OR (o):** al igual que AND, el operador OR relaciona dos operandos booleanos. El resultado es un valor verdadero (V) si cualquiera de los dos operandos es verdadero (V); si los dos operandos son falsos (F) el resultado es falso (F).

El operador And se usa de la siguiente manera

```
Operando 1  OR  Operando 2
```

Los resultados que pueden obtenerse al aplicar un operador OR, se muestran en la Tabla 3.3.

Tabla 3.3. Resultados de un operador OR.

| Operando 1 | Operando 2 | Resultado |
|---|---|---|
| V | V | **V** |
| V | F | **V** |
| F | V | **V** |
| F | F | **F** |

Ejemplo: sean X y Y variables lógicas cuyo valor es verdadero (V) y falso (F) respectivamente, determinar el valor de la expresión X OR Y.

X OR Y = V OR F = V

**Operador NOT (no):** este operador se aplica a un operando lógico y da como resultado el valor opuesto al que tiene el operando. Esto es, si el operando es verdadero el resultado es falso, y si el operando es falso el resultado es verdadero.

El operador NOT se usa de la siguiente manera:

```
NOT  Operando
```

Los posibles resultados de un operador NOT se muestran en la Tabla 3.4.

Tabla 3.4. Resultados de un operador NOT.

| Operando | Resultado |
|---|---|
| V | **F** |
| F | **V** |

Ejemplo: sean X una variable lógica cuyo valor es verdadero (V), determinar el valor de la expresión NOT X.

NOT X = NOT V = F

## 3.3 Expresiones lógicas

Este tipo de expresiones se forma al combinar variables, constantes, operadores relacionales y operadores lógicos. Se llaman expresiones lógicas o booleanas porque al ser evaluadas el resultado siempre será verdadero o falso.

**Expresiones lógicas con operadores relacionales:** las expresiones lógicas más simples se forman al combinar variables y/o constantes con operadores relacionales. Ejemplos de este tipo de expresiones se muestran en la Tabla 3.5.

Tabla 3.5 Ejemplos de expresiones lógicas simples

| Expresión lógica | Valor de la expresión si X=5 y Y=2 |
|---|---|
| X < 3 | **Falso** |
| Y > X - 4 | **Verdadero** |
| Y <= X | **Verdadero** |
| X = Y | **Falso** |

Las variables de tipo cadena de caracteres también se pueden comparar, para ello la computadora examina carácter por carácter de izquierda a derecha, y relaciona el valor ASCII de cada letra. De acuerdo al código ASCII:

a < b < c < … < z

A < B < C < … < Z

Minúsculas > Mayúsculas

En la Tabla 3.6 se muestran algunos ejemplos.

Tabla 3.6. Ejemplos de expresiones lógicas con cadenas de caracteres

| Expresión lógica | Valor de la expresión |
|---|---|
| "Mari" < "Marianela" | Verdadero |
| "Ana" > "José" | Falso |
| "Doris" > "Doria" | Verdadero |

**Expresiones lógicas con operadores lógicos:** es posible construir expresiones lógicas más complejas usando los operadores lógicos AND, OR y NOT.

Ejemplos

Sea X= Verdadero(V) y Y= Falso (F).

a) X AND Y = F

b) X OR Y = V

Se pueden utilizar varios operadores lógicos en una misma expresión. En este caso, para determinar el valor de la expresión es necesario conocer la jerarquía (orden de precedencia) de los operadores lógicos, la cual se muestra en la Tabla 3.7

Tabla 3.7. Jerarquía de los operadores lógicos

| Operador | Jararquía |
|---|---|
| ( ) | 1 |
| NOT | 2 |
| AND | 3 |
| OR | 4 |

Ejemplos

Determinar el valor de las siguientes expresiones lógicas y mostrar el orden de ejecución, suponiendo que A= V, B=V, C= F y D=F.

a) NOT A OR B AND C

NOT V OR V AND F

F OR V AND F

F OR F = F

El valor de esta expresión es falso. Para determinar el valor de la expresión se evaluó el NOT, luego el AND y por último el OR de acuerdo al orden de precedencia de las

operaciones (en cada línea se efectuó una operación, la cual se encuentra subrayada para facilitar la comprensión).

b) A AND C OR B AND ( NOT D OR A)

V AND F OR V AND (NOT F OR V)

V AND F OR V AND (V OR V)

V AND F OR V AND V

F OR V AND V

F OR V = V

El valor de esta expresión es verdadero.

**Expresiones lógicas con operadores relacionales y operadores lógicos:** cuando se está programando es bastante usual tener que construir expresiones lógicas que combinen operadores relacionales y operadores lógicos, como por ejemplo (A >= 5) AND (B < 8).

En expresiones de este tipo se evalúan inicialmente los operadores relacionales y luego los operadores lógicos. Si además existen operadores aritméticos, éstos se evalúan primero.

Ejermplos

Determinar el valor de las siguientes expresiones lógicas, considerando que A = 8 y B = 2.

1) (A >= 5) AND (B < 8)

(8 >= 5) AND (2 < 8)

V AND (2 < 8)

V AND V

V

2) (A+2 < 11) OR (B>=2) AND (A – B = 4)

(8 +2 < 11) OR (2>=2) AND (8 – 2 = 4)

(10 < 11) OR (2>=2) AND (8 – 2 = 4)

(10 < 11) OR (2>=2) AND (6 = 4)

V OR (2>=2) AND (6 = 4)

V OR V AND (6 = 4)

V OR V AND F

V OR F

V

## 3.4 Estructuras de decisión simple

En este tipo de estructura se evalúa una condición o expresión lógica, si es verdadera se ejecutan un conjunto de instrucciones, si la condición es falsa se ignoran estas instrucciones.

Una estructura de decisión simple se utiliza cuando la ejecución de algunas instrucciones está condicionada, pero no hay instrucciones alternativas.

La forma de escribir una estructura de decisión simple en pseudocódigo es la siguiente:

```
Si condición entonces
    Instrucciones a ejecutar si la condición
    es verdadera
Fin de si
```

La Figura 3.1 ilustra el diagrama de flujo de este tipo de instrucción.

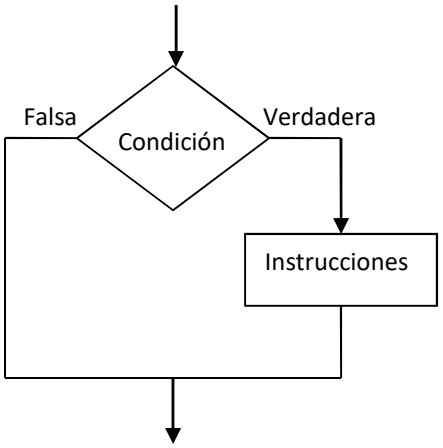



Figura 3.1. Diagrama de flujo de una estructura de decisión simple

Si la expresión lógica es verdadera, se ejecutarán las instrucciones. Si es falsa, no sucede nada y la ejecución del programa continua en las instrucciones que van después de la estructura de decisión simple.

Ejemplo:

Diseñar un programa para calcular la nota definitiva de un estudiante en una asignatura. Los datos son la nota de la parte teórica de la asignatura que representa el 70% de la definitiva y la nota de la práctica que representa el 30%. El programa debe mostrar un mensaje que indique si el estudiante está reprobado.

**Análisis E-P-S**

*Entrada*

NT: nota de teoría. Tipo: Real.
NP: nota de práctica. Tipo: Real.

*Proceso*

Calcular la nota definitiva
Ndef = 0.7 x NT + 0.3x NP

Determinar si el estudiante está reprobado (Ndef < 9.5), en caso afirmativo mostrar un mensaje.

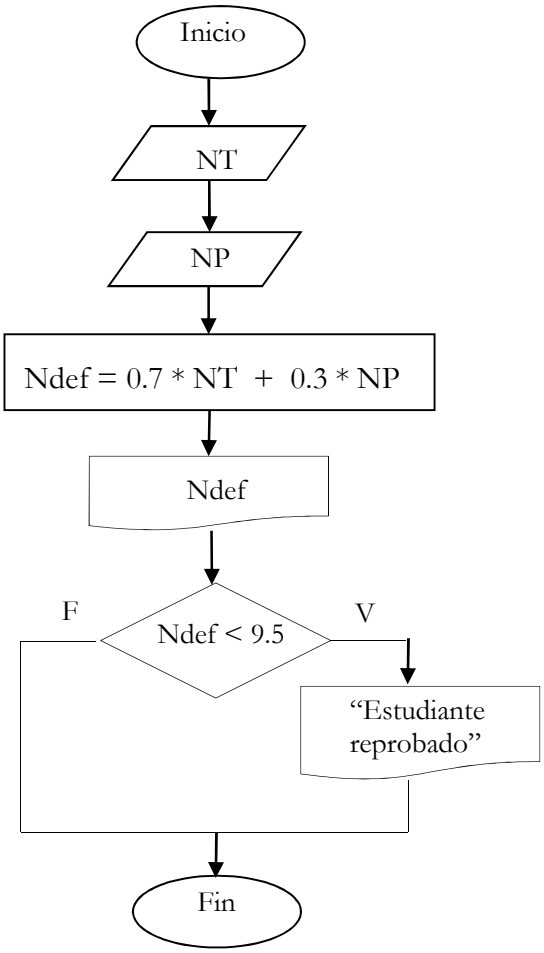
*Salida*

Ndef: nota definitiva. Tipo: Real.
Mensaje que indica si el estudiante está reprobado en la asignatura.

**Algoritmo**

0. Inicio
1. Leer nota de teoría (NT)
2. Leer nota de práctica (NP)
3. Ndef = 0.7 * NT + 0.3 * NP
4. Mostrar nota definitiva (Ndef)
5. Si Ndef < 9.5 entonces
   5.1 Mostrar "El estudiante está reprobado"
   Fin de si
6. Fin

**Diagrama de flujo**




En este ejemplo, si la condición lógica de la estructura de decisión es verdadera solo se muestra como salida un mensaje, en otros problemas pueden colocarse otros tipos de instrucciones en la estructura de decisión tales como ecuaciones, lectura de datos, escritura de variables, incluso varias instrucciones si el problema lo requiere.

## 3.5 Estructuras de decisión doble

Se utilizan cuando en un algoritmo o programa se debe elegir entre dos alternativas dependiendo de una condición. Una estructura de decisión doble evalúa una expresión lógica, si ésta es verdadera se ejecuta un conjunto de instrucciones, y si es falsa se ejecuta otro conjunto de instrucciones diferente.

Una estructura de decisión doble se escribe de la siguiente manera en pseudocódigo:

**Si** condición **entonces**

Instrucciones a ejecutar si la condición es verdadera

**de lo contrario**

Instrucciones a ejecutar si la condición es falsa

**Fin de si**

En la Figura 3.2 se presenta el diagrama de flujo de este tipo de estructura de programación. En el mismo, se observa claramente como el diagrama de flujo se divide en dos caminos o cursos de acción.

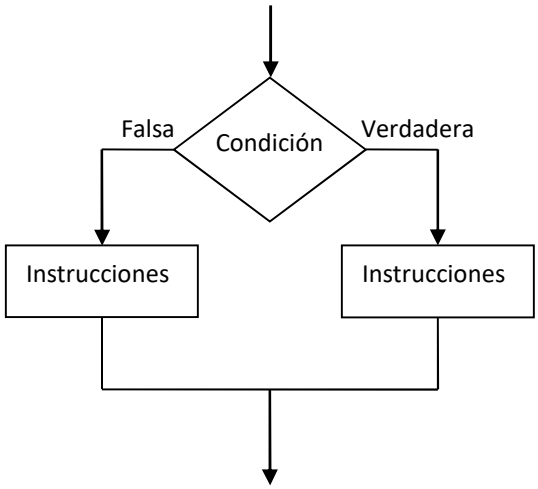



Figura 3.2 Diagrama de flujo de una estructura de decisión doble

Ejemplo:

Diseñar un programa para calcular la nota definitiva de un estudiante en una asignatura. Los datos son la nota de la parte teórica de la asignatura que representa el 70% de la definitiva y la nota de la práctica que representa el 30%. El programa debe mostrar un mensaje que indique si el estudiante está aprobado o reprobado

**Análisis E-P-S**

*Entrada*

NT: nota de teoría. Tipo: Real.
NP: nota de práctica. Tipo: Real.

*Proceso*

Calcular la nota definitiva
Ndef = 0.7 x NT + 0.3x NP

Determinar si el estudiante está aprobado o reprobado
Ndef ≥ 9.5 Estudiante aprobado
Ndef < 9.5 Estudiante reprobado

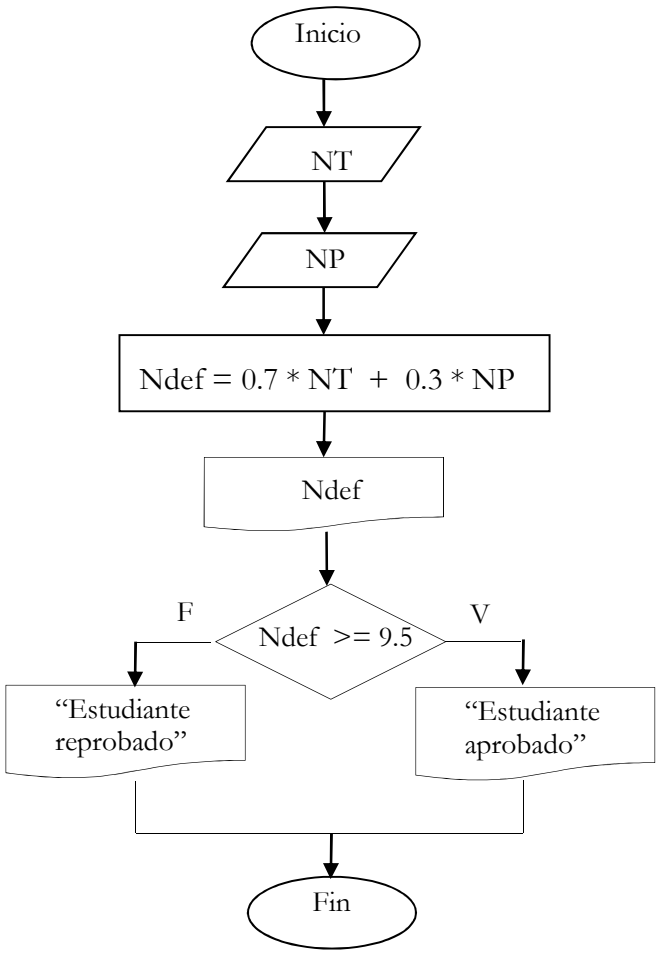
*Salida*

Ndef: nota definitiva. Tipo: Real.
Mensaje que indica si el estudiante está aprobado o reprobado en la asignatura.

**Algoritmo**

0. Inicio
1. Leer nota de teoría (NT)
2. Leer nota de práctica (NP)
3. Ndef = 0.7 * NT + 0.3 * NP
4. Mostrar nota definitiva (Ndef)
5. Si Ndef >= 9.5 entonces
   5.1 Mostrar "El estudiante está aprobado"
   de lo contrario
   5.2 Mostrar "El estudiante está reprobado"
   Fin de si
6. Fin

**Diagrama de flujo**




## 3.6 Estructuras de decisión anidadas

Se utilizan en problemas en los que hay más de dos alternativas entre las cuales se puede elegir. Cuando en una estructura de decisión alguna de sus instrucciones es otra estructura de decisión se dice que las estructuras están anidadas.

En pseudocódigo las estructuras de decisión anidadas utilizan varias estructuras si / entonces/ de lo contrario, unas dentro de otras. Representan situaciones en las cuales se toma una decisión y dependiendo del resultado es necesario tomar otra decisión. El proceso puede repetirse varias veces dependiendo del problema.

Hay diferentes maneras en las que pueden anidarse las estructuras de decisión. Obsérvese el siguiente ejemplo:

```
Si  condición 1  entonces
        Instrucciones a ejecutar si la condición 1 es verdadera
de lo contrario
        Si condición 2 entonces
                Instrucciones a ejecutar si la condición  2 es verdadera
        de lo contrario
                Instrucciones a ejecutar si la condición  2 es falsa
        Fin de si
Fin de si
```

En este caso se evalúa la condición 1, si es verdadera se ejecuta un conjunto de instrucciones y luego finaliza la estructura de decisión, pero si la condición 1 es falsa, es necesario evaluar la condición 2, y dependiendo de si ésta es verdadera o falsa se ejecutarán ciertas instrucciones. El diagrama de flujo correspondiente se muestra en la Figura 3.3

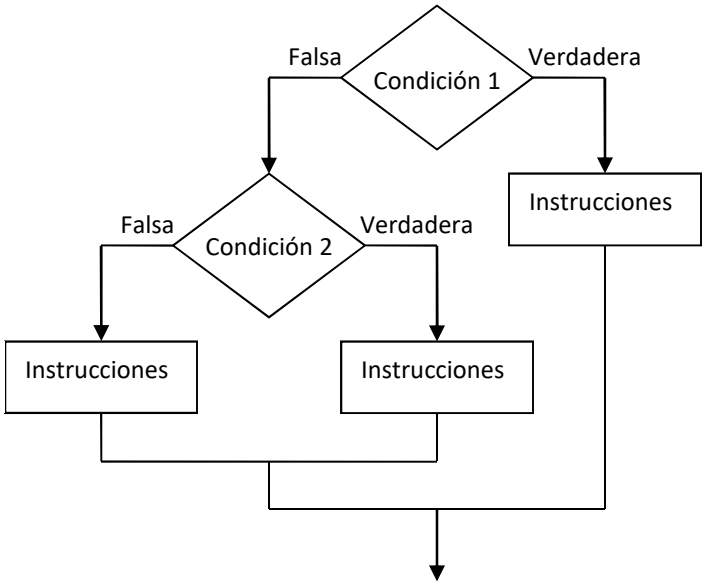



Figura 3.3 Diagrama de flujo de una estructura de decisión anidada (ejemplo 1)

Otro ejemplo en pseudocódigo de una estructura de decisión anidada es el siguiente:

```
Si  condición 1  entonces
    Si condición 2 entonces
        Instrucciones a ejecutar si la condición  2 es verdadera
    de lo contrario
        Instrucciones a ejecutar si la condición  2 es falsa
    Fin de si
de lo contrario
    Si condición 2 entonces
        Instrucciones a ejecutar si la condición  2 es verdadera
    de lo contrario
        Instrucciones a ejecutar si la condición  2 es falsa
    Fin de si
Fin de si
```

Acá se evalúa una condición 1 y si el resultado es verdadero, se evalúa la condición 2, dependiendo si ésta es verdadera o falsa se ejecutan las instrucciones correspondientes. Ahora, si al evaluar la condición 1 el resultado es falso, es necesario evaluar la condición 3, para determinar de acuerdo al resultado las instrucciones a ejecutar. El diagrama de flujo correspondiente se muestra en la Figura 3.4.

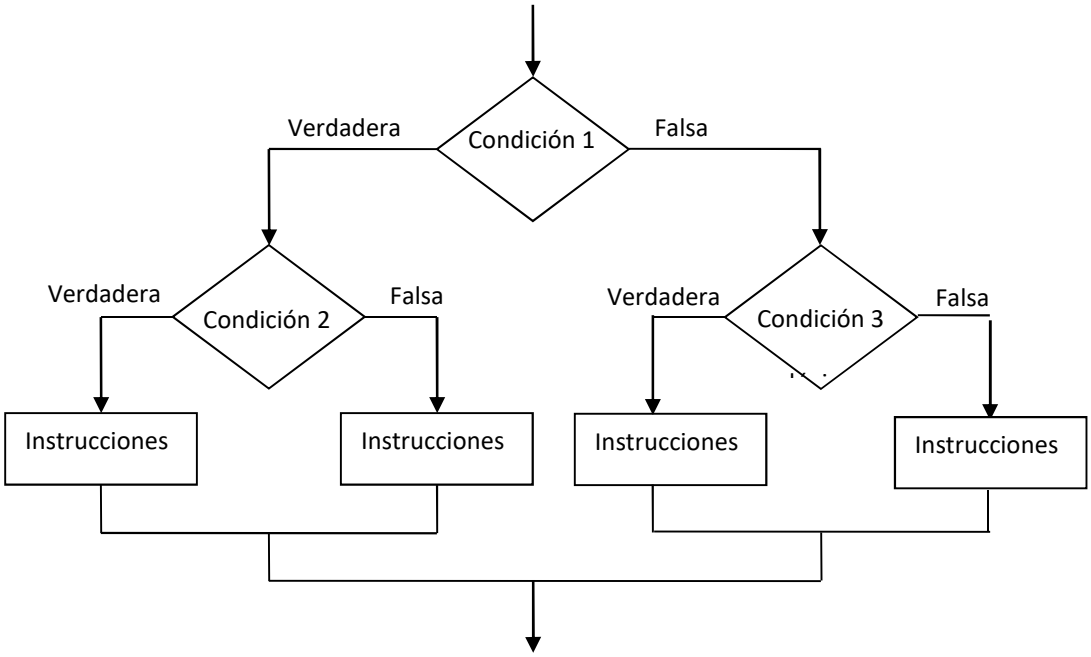



Figura 3.4 Diagrama de flujo de una estructura de decisión anidada (Ejemplo 2)

Las estructuras de decisión anidadas representadas en las Figuras 3.3 y 3.4 son solo dos ejemplos, la forma en las que se anidan las estructuras de decisión puede ser muy variada y depende del problema que se esté resolviendo, por lo tanto no hay un formato único.

Ejemplo:

Diseñar un programa para calcular la nota definitiva de un estudiante en una asignatura. Los datos son la nota de la parte teórica de la asignatura que representa el 70% de la definitiva y la nota de la práctica que representa el 30%. El programa debe mostrar los siguientes mensajes de acuerdo a la nota definitiva del estudiante:

| Nota definitiva | Mensaje |
|---|---|
| <9.5 | Reprobado |
| >= 9.5 y < 16 | Aprobado |
| >= 16 y < 19 | Aprobado - Distinguido |
| >= 19 | Aprobado - Sobresaliente |

**Análisis E-P-S**

*Entrada*

NT: nota de teoría. Tipo: Real.
NP: nota de práctica. Tipo: Real.

*Proceso*

Calcular la nota definitiva
Ndef = 0.7 x NT + 0.3x NP

Determinar clasificación del estudiante:
Ndef < 9.5 Estudiante reprobado
Ndef >= 9.5 y < 16 Estudiante aprobado
Ndef >= 16 y < 19 Estudiante aprobado – distinguido
Ndef >= 19 Estudiante aprobado - sobresaliente

*Salida*

Ndef: nota definitiva. Tipo: Real.
Mensaje que indica el tipo de estudiante según su nota.

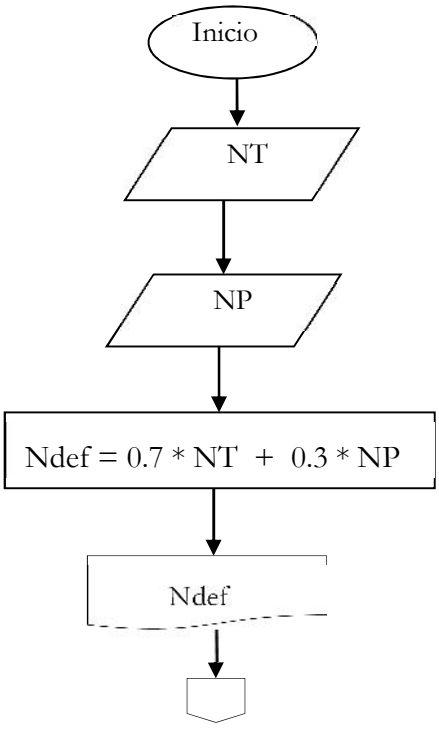
**Algoritmo**

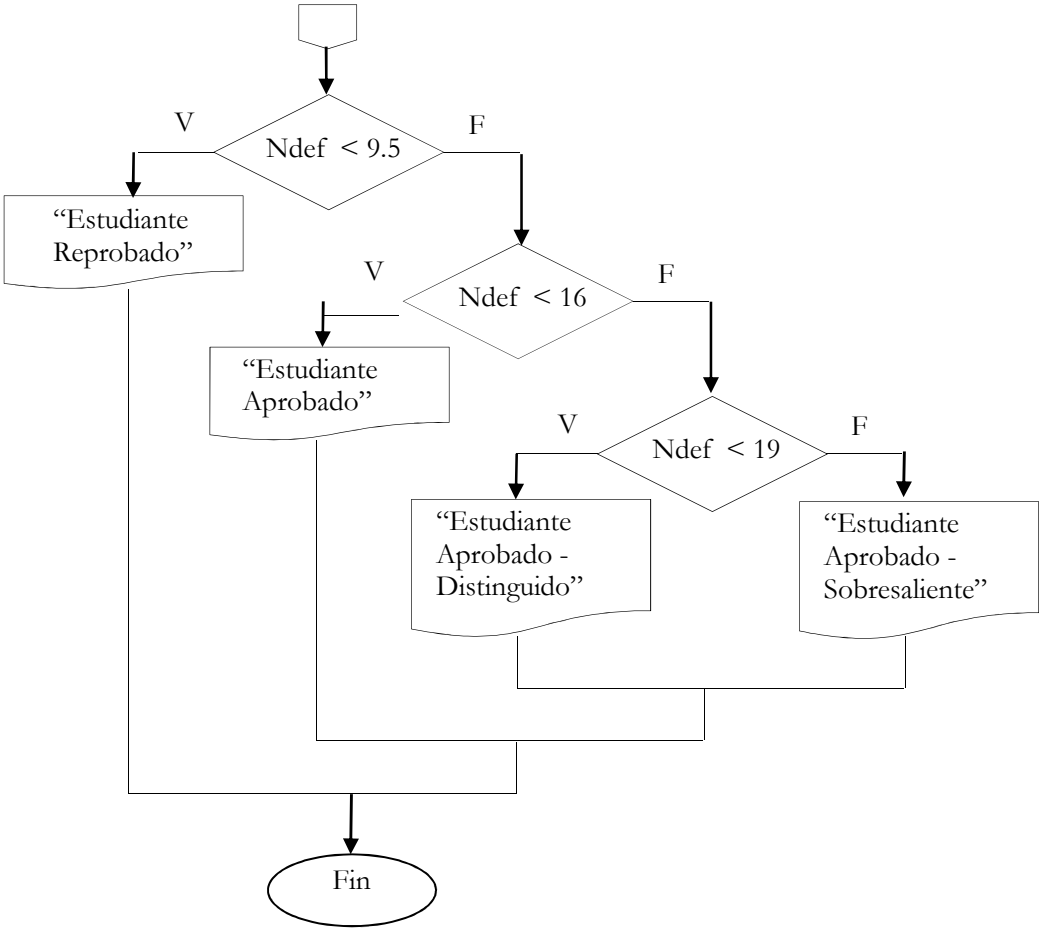
0. Inicio
1. Leer nota de teoría (NT)
2. Leer nota de práctica (NP)
3. Ndef = 0.7 * NT + 0.3 * NP
4. Mostrar nota definitiva (Ndef)
5. Si Ndef < 9.5 entonces
   5.1 Mostrar "Estudiante Reprobado"
   de lo contrario
   5.2 Si Ndef < 16 Entonces
   5.2.1 Mostrar "Estudiante Aprobado"
   de lo contrario
   5.2.2 Si Ndef < 19 Entonces
   5.2.2.1 Mostrar "Estudiante Aprobado - Distinguido"
   de lo contrario
   5.2.2.2 Mostrar "Estudiante Aprobado - Sobresaliente"
   Fin de si 5.2.2
   Fin de si 5.2
   Fin de si 5
6. Fin

En el algoritmo anterior puede observarse lo siguiente:

- Uso de la numeración y las sangrías (indentación) para denotar el anidamiento de las estructuras.
- En la instrucción 5.2 no es necesario colocar la condición Ndef >= 9.5 and Ndef < 16, porque la frase "de lo contrario" que está antes del paso 5.2 indica que la condición Ndef < 9.5 es falsa y por consiguiente Ndef será >= 9.5, por tanto solo se requiere en el paso 5.2 colocar Ndef < 16.
- Igualmente, al finalizar cada estructura se coloca el número de la estructura de decisión que se está cerrando (por ejemplo, Fin de si 5.2.2).

**Diagrama de flujo**






## 3.7 Estructuras de decisión múltiple

Al igual que las estructuras de decisión anidadas, las estructuras de decisión múltiple se utilizan cuando se quiere elegir entre varias alternativas. En una estructura de este tipo se especifica una variable, la cual puede tomar diferentes valores, dependiendo del valor que tenga se ejecutarán las instrucciones pertinentes.

Una manera de especificar una estructura de decisión múltiple en pseudocódigo es la siguiente:

**Seleccionar caso** Variable

**V1:** instrucciones a ejecutar si la variable toma el primer valor

**V2:** instrucciones a ejecutar si la variable toma el segundo valor

.
.
.

**Vn:** instrucciones a ejecutar si la variable toma el n-ésimo valor

**de lo contrario**

Instrucciones a ejecutar si la variable no toma alguno de los valores antes especificados

**Fin de seleccionar**

La estructura de decisión múltiple (seleccionar caso) chequea cuál es el valor de la variable cuyo nombre aparece en la primera línea de la estructura, dependiendo de su valor ejecuta la instrucción correspondiente. Si la variable no toma alguno de los valores que aparecen listados, se ejecutan las instrucciones que están después de la sentencia "de lo contrario". Esta última parte es opcional, es decir, si no se coloca el "de lo contrario" y la variable no toma alguno de los valores especificados, simplemente no se ejecutan instrucciones en la estructura de decisión múltiple y el programa continúa su ejecución en las instrucciones que se encuentren después de "fin de seleccionar".

En la Figura 3.5 se observa el diagrama de flujo de este tipo de estructura.

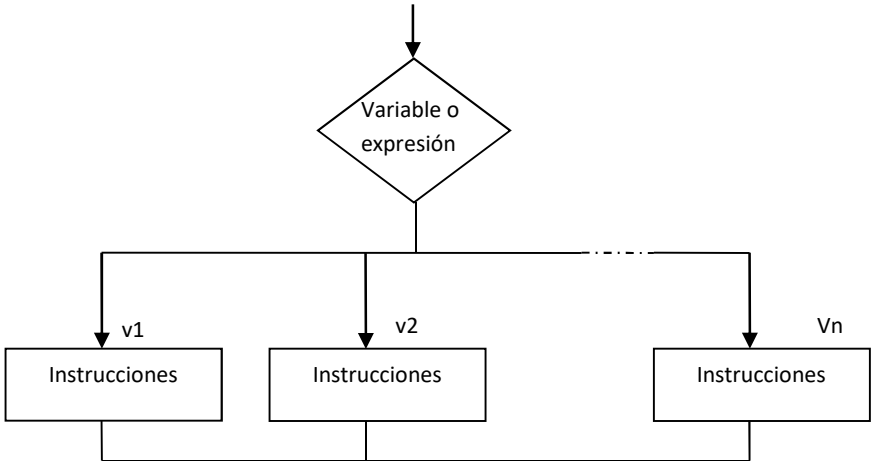



Figura. 3.5 Diagrama de flujo de una estructura de decisión múltiple

Ejemplo:

Diseñar un programa que reciba como dato un número entre 1 y 7, y muestre un mensaje que indique el día de la semana correspondiente.

**Análisis E-P-S**

*Entrada*

Num: número entre 1 y 7. Tipo: Entero.

*Proceso*

Determinar el día de la semana correspondiente

Num = 1 Día Domingo
Num = 2 Día Lunes
Num = 3 Día Martes
Num = 4 Día Miércoles
Num = 5 Día Jueves
Num = 6 Día Viernes
Num = 7 Día Sábado

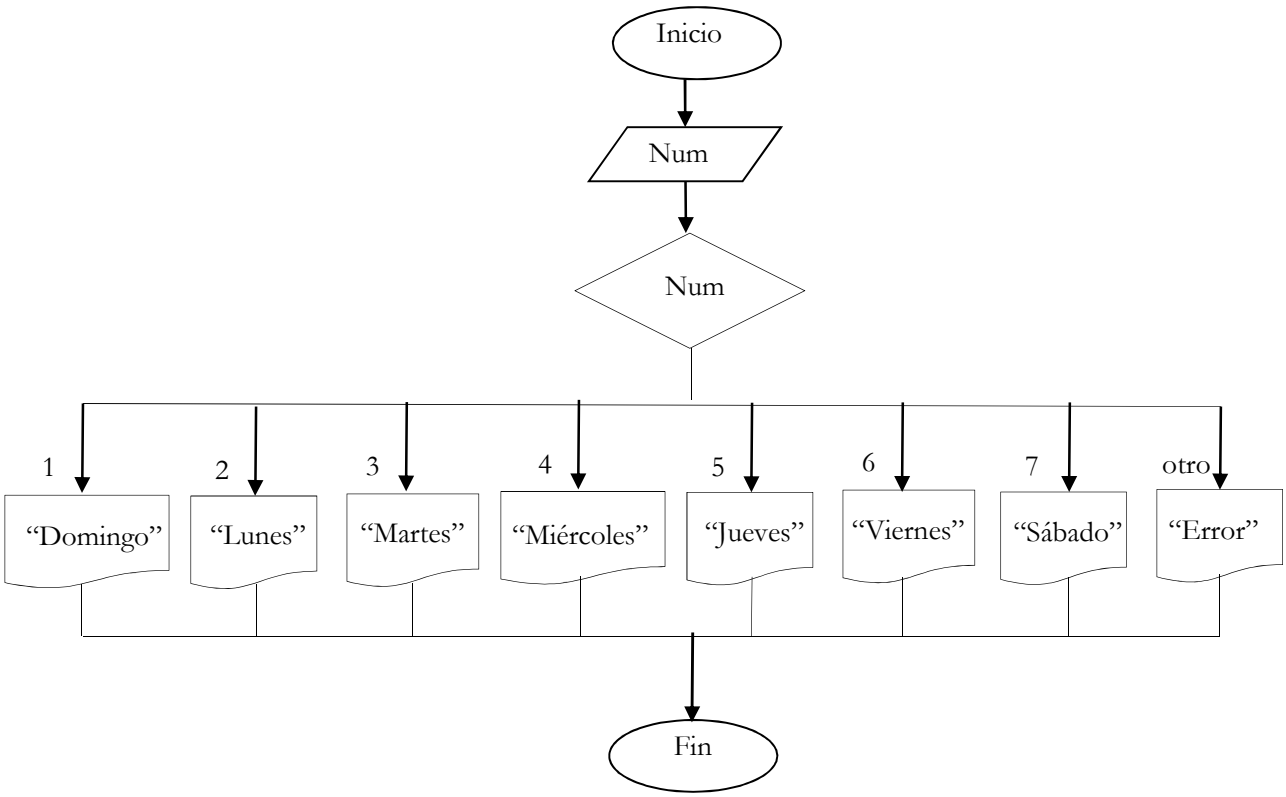
*Salida*

Mensaje que indica el día de la semana correspondiente al número indicado por el usuario.

**Algoritmo**

0. Inicio
1. Leer número (Num)
2. Seleccionar caso Num
   1: Mostrar "Domingo"
   2: Mostrar "Lunes"
   3: Mostrar "Martes"
   4: Mostrar "Miércoles"
   5: Mostrar "Jueves"
   6: Mostrar "Viernes"
   7: Mostrar "Sábado"
   De lo contrario
      Mostar "Error: el número debe estar entre 1 y 7"
   Fin de Seleccionar
3. Fin

**Diagrama de flujo**




En una estructura de decisión múltiple también se pueden colocar varios valores separados por coma en una misma línea, para indicar que si la variable toma alguno de esos valores se ejecuta una misma instrucción.

Ejemplo:

Escribir un algoritmo que dado un número cuyo valor está entre 1 y 10, diga si el número es par o impar.

**Algoritmo**

0. Inicio
1. Leer número (N)
2. Seleccionar caso N
   2, 4, 6, 8 10: Mostrar "Es un número par"
   1, 3, 5, 7, 9: Mostrar "Es un número impar"
   Fin de Seleccionar
3. Fin

En este ejemplo no se colocó "de lo contrario" en la estructura de decisión múltiple, si el usuario teclea un número que no está entre 1 y 10, el programa no muestra resultados. Se deja como ejercicio para el lector realizar el diagrama de flujo para este problema.

## 3.8 Ejercicios resueltos

1) Resolver las siguientes expresiones lógicas

a) A OR (NOT (B AND C) OR A) AND C si A = F, B = F, C = V

Solución

F OR (NOT (F AND V) OR F) AND V
F OR (NOT F OR F) AND V
F OR (V OR F) AND V
F OR V AND V
F OR V
V

b) ((X > 7) OR (Y < 3)) AND (Z >= 2) si X = 8, Y = 2, Z = 1

Solución

((8 > 7) OR (2 < 3)) AND (1 >=2)
(V OR (2 < 3)) AND (1 >=2)
(V OR V) AND (1>=2)
(V OR V) AND F
V AND F
F

2) Construir una expresión lógica cuyo valor sea verdadero si un número X se encuentra en el intervalo [3, 10). El resultado de la expresión debe ser falso si X se encuentra fuera de este intervalo.

La expresión lógica es: (X >= 3) AND (X < 10)

3) Construir una expresión lógica cuyo valor sea verdadero si la altura de un árbol es superior a 20 metros o su diámetro es 50 cm o menos. Si no se cumple ninguna de estas condiciones el resultado debe ser falso.

La expresión lógica es: (altura > 20) OR (diámetro <= 50)

4) Realizar un algoritmo y un diagrama de flujo para calcular el salario neto de un trabajador, teniendo como entrada su salario base y el número de hijos. Al trabajador se le descuenta el

5% de su salario base por concepto de seguro social, pero si tiene más de dos hijos se le pagan 5000 Bs. adicionales.

En estos ejercicios resueltos no se mostrará el análisis E-P-S, éste se deja como ejercicio para el lector. En cambio, se hace un listado con los nombres de variables utilizados en el algoritmo, su significado y su tipo de dato.

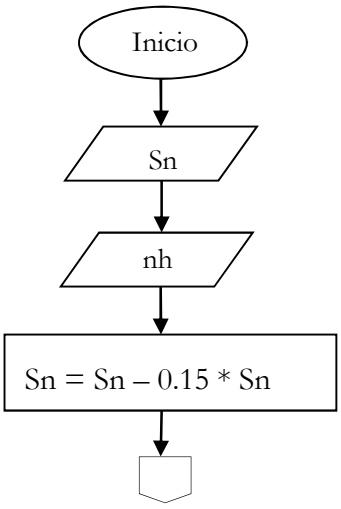
**Variables**

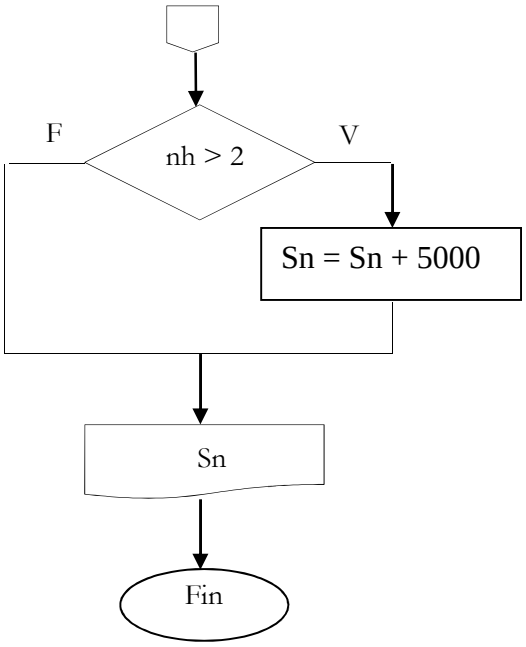
Sb: Salario base. Tipo: Real.
nh: Número de hijos. Tipo: Real.
Sn: Salario neto. Tipo: Real.

**Algoritmo**

0. Inicio
1. Leer salario base (Sb)
2. Leer número de hijos (nh)
3. Sn = Sb – 0.05 * Sb
4. Si nh > 2 Entonces
   4.1 Sn = Sn + 5000
   Fin de si
5. Mostrar salario neto (Sn)
6. Fin

**Diagrama de flujo**






5) Elaborar un algoritmo para calcular el precio a pagar por un lote de tablones de madera. Se tienen como datos la cantidad de metros cúbicos de madera a comprar y el precio por metro cúbico. Si la madera se ve defectuosa, el vendedor da un descuento del 12%.

**Variables**

cmc: cantidad de metros cúbicos de madera a comprar. Tipo: Real.
pmc: precio por metro cúbico de madera. Tipo: Real.
cal_mad: calidad de la madera, "D" si es defectuosa, "B" si no es defectuosa. Tipo: Carácter.
Desc: descuento. Tipo: Real.
Pt: precio total. Tipo: Real.

**Algoritmo**

0. Inicio
1. Leer cantidad de metros cúbicos a comprar (cmc)
2. Leer precio por metro cúbico (pmc)
3. Leer calidad de la madera (cal_mad)
4. Pt = cmc * pmc
5. Si cal_mad = "D" entonces
   5.1 Desc = 0.12 * pt
   5.2 Mostrar descuento (Desc)
   5.3 Pt = Pt – Desc
   Fin de si
6. Mostrar precio total a pagar (Pt)
7. Fin

6) Dada la siguiente función por partes:

$$f(x) = \begin{cases} x^2 - 9 & si\ x < 3 \\ -x + 3 & si\ x \geq 3 \end{cases}$$

Diseñar un programa que dado un valor de x determine el valor de la función f(x).

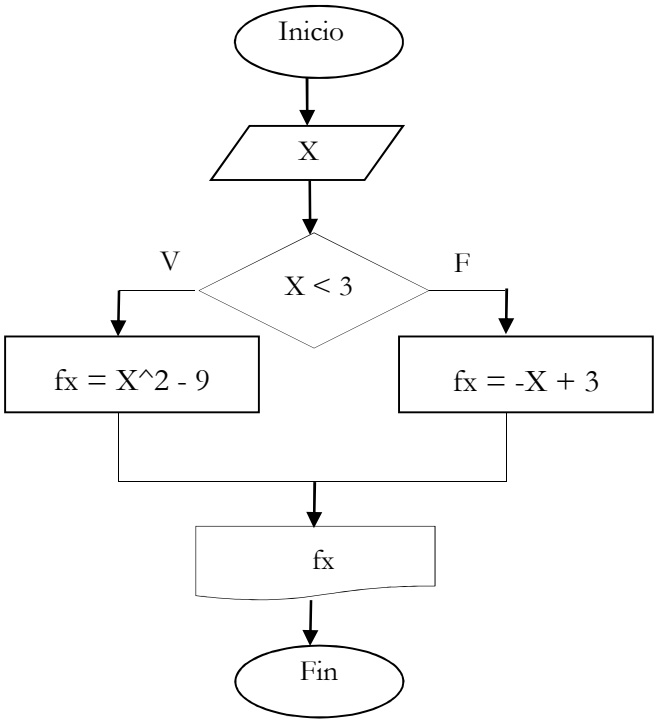
**Variables**

X: número utilizado para evaluar la función f(x). Tipo Real.
fx: valor de la función. Tipo: Real.

**Algoritmo**

0. Inicio
1. Leer (X)
2. Si x < 3 entonces
   2.1 fx = X^2 - 9
   de lo contrario
   2.2 fx = - X + 3
   Fin de si 2
3. Mostrar valor de la función (fx)
4. Fin

**Diagrama de flujo**



7) En una universidad se desea determinar si una persona es aceptada o no para estudiar una determinada carrera. Para que el aspirante sea aceptado se requiere que haya obtenido una calificación superior a 85 puntos en la prueba de admisión, en una escala de 0 a 100, o que el promedio de notas de bachillerato sea superior a 17.5 puntos. Si no cumple alguno de estos requisitos el aspirante no es aceptado y deberá someterse a una segunda prueba de admisión. Elaborar un algoritmo para determinar si un aspirante es o no aceptado.

**Variables**

Nom_asp: nombre del aspirante. Tipo: Cadena de caracteres.
npa: nota obtenida en la prueba de admisión. Tipo: Real.
pnb: promedio de notas de bachillerato. Tipo: Real.

**Algoritmo**

0. Inicio
1. Leer nombre del aspirante (Nom_asp)
2. Leer nota obtenida en la prueba de admisión (npa)
3. Leer promedio de notas de bachillerato (pnb)
4. Si npa > 85 OR pnb > 17.5 entonces
   4.1 Mostrar nombre del aspirante (Nom_asp)
   4.2 Mostrar "Aceptado"
   de lo contrario
   4.3 Mostrar nombre del aspirante (Nom_asp)
   4.4 Mostrar "No fue aceptado"
   Fin de si
5. Fin

8) Diseñar un programa que tenga como datos de entrada los coeficientes a, b, c de una ecuación de segundo grado $aX^2 + bX + c = 0$ y calcule sus raíces. Si las raíces son números complejos el programa debe indicarlo mediante un mensaje.

**Variables**

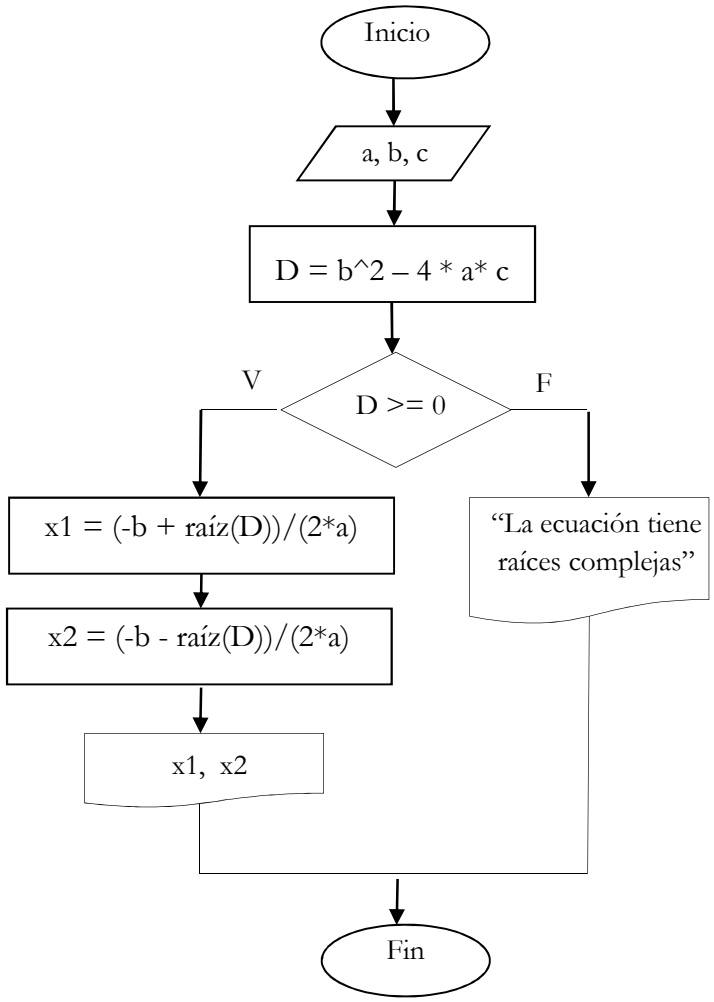
a: coeficiente cuadrático. Tipo: Real.
b: coeficiente lineal. Tipo: Real.
c: término independiente. Tipo: Real.
D: discriminante ($b^2 – 4ac$). Tipo: Real.
x1: raíz 1. Tipo: Real.
x2: raíz 2. Tipo: Real.

**Algoritmo**

0. Inicio
1. Leer coeficiente cuadrático (a)

2. Leer coeficiente lineal (b)
3. Leer término independiente (c)
4. D = b^2 – 4 * a * c
5. Si D >= 0 entonces
   5.1 x1 = (-b + raíz(D))/(2*a)
   5.2 x2 = (-b - raíz(D))/(2*a)
   5.3 Mostrar valor de x1 (x1)
   5.4 Mostrar valor de x2 (x2)
   De lo contrario
   5.5 Mostrar "La ecuación tiene raíces complejas"
   Fin de si
6. Fin

**Diagrama de flujo**



Se deja como ejercicio para el lector modificar el ejercicio anterior de tal manera que en caso de que la ecuación tenga raíces complejas, el algoritmo también las calcule.

9) Una empresa que distribuye tableros aglomerados a nivel nacional cuenta con varios vendedores. Un vendedor además de su sueldo base, mensualmente recibe una comisión de acuerdo a las ventas que realiza, si en un mes vende más de 200000 Bs. le corresponde una comisión del 5% sobre las ventas y si vende 200000 o menos la comisión es del 2%. Elaborar un algoritmo que calcule el salario neto mensual de un vendedor considerando su sueldo base y la comisión.

**Variables**

NomV: nombre del vendedor. Tipo: cadena de caracteres.
Sb: sueldo base. Tipo: Real.
V: monto de las ventas realizadas por un vendedor en un mes. Tipo: Real.
Com: comisión. Tipo: Real.
Sn: salario neto.

**Algoritmo**

0. Inicio
1. Leer nombre del vendedor (NomV)
2. Leer sueldo base (Sb)
3. Leer monto de las ventas (V)
4. Si V > 200000 entonces
   4.1 Com = 0.05 * V
   de lo contrario
   4.2 Com = 0.02 * V
   Fin de si
5. Sn = Sb + Com
6. Mostrar salario neto (Sn)
7. Fin

10) Elaborar un algoritmo para clasificar una especie forestal de acuerdo a su resistencia. El dato de entrada es el porcentaje de pérdida de peso de la especie y la salida es uno de los siguientes mensajes.

| % pérdida de peso | Mensaje |
|---|---|
| ≤1 | **Altamente resistente** |
| (1 - 5] | **Resistente** |
| (5 – 10] | **Moderadamente resistente** |
| (10- 30] | **Muy poco resistente** |
| Más de 30 | **No resistente** |

**Variables**

Pp: porcentaje de pérdida de peso de la madera.

**Algoritmo**

0. Inicio
1. Leer porcentaje de pérdida de peso de la madera (Pp)
2. Si Pp <= 1 entonces
   2.1 Mostrar "Altamente resistente"
   de lo contrario
   2.2 Si Pp <= 5 entonces
   2.2.1 Mostrar "Resistente"
   de lo contrario
   2.2.2 Si Pp <= 10 entonces
   2.2.2.1 Mostrar "Moderadamente resistente"
   de lo contrario
   2.2.2.2 Si pp <= 30 entonces
   2.2.2.2.1 Mostrar "Muy poco resistente"
   de lo contrario
   2.2.2.2.2 Mostrar "No resistente"
   Fin de si 2.2.2.2
   Fin de si 2.2.2
   Fin de si 2.2
   Fin de si 2
3. Fin

11) Diseñar un programa que dados tres números diferentes determine cuál es el mayor.

**Variables**

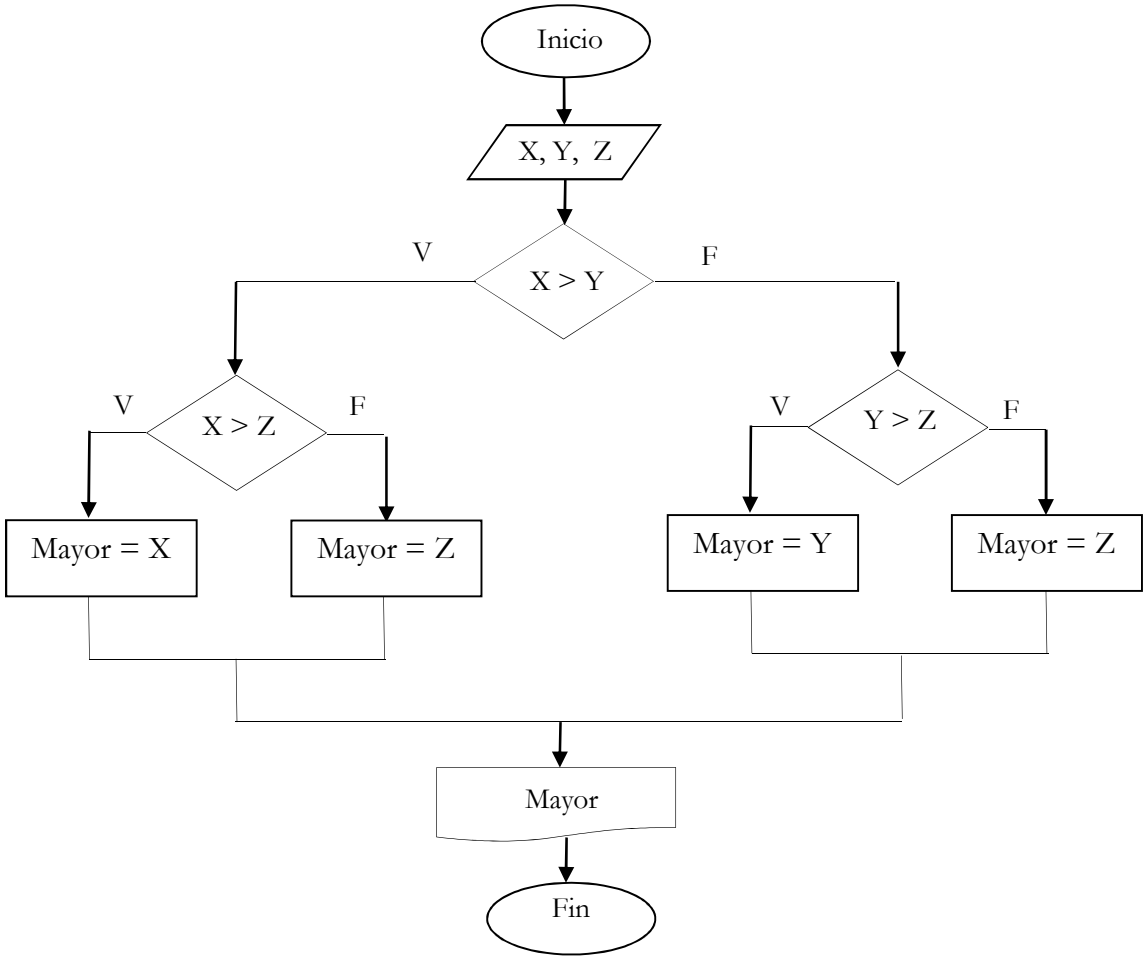
X: primer número. Tipo: Real.
Y: segundo número. Tipo: Real.
Z: tercer número. Tipo: Real

**Algoritmo**

0. Inicio
1. Leer primer número (X)
2. Leer segundo número (Y)
3. Leer tercer número (Z)
4. Si X > Y entonces
   4.1 Si X > Z entonces

```
            4.1.1 Mayor = X
        de lo contrario
            4.1.2 Mayor = Z
        Fin de si 4.1
    de lo contrario
        4.2 Si Y > Z entonces
            4.2.1 Mayor = Y
        de lo contrario
            4.2.1 Mayor = Z
        Fin de si 4.2
    Fin de si 4
5. Mostrar número mayor (Mayor)
6. Fin
```

**Diagrama de flujo**



12) Una empresa de producción forestal planta dos especies: Eucalipto (*Eucaliptus grandis*) y Pino pátula (*Pinus patula*). Se desea diseñar un programa que calcule el volumen total con corteza y el volumen total sin corteza para un árbol. El programa debe permitir al usuario indicar la especie y de acuerdo a esto aplicará las ecuaciones correspondientes:

| Especie | Volumen con corteza | Volumen sin corteza |
|---|---|---|
| **Eucalipto** | VCC = 0.0185 + 0.32084($d^2h$) | VSC = -0.0047 + 0.8947 VCC |
| **Pino pátula** | VCC = 0.00653 + 0.0000355($d^2h$) | VSC = -0.005 + 0.9128 VCC |

d: diámetro a la altura de pecho en cm. h: altura total del árbol.

**Variables**

Esp: especie del árbol, 1 si eucalipto y 2 si es pino pátula. Tipo: Entero.
d: diámetro a la altura de pecho en cm. Tipo: Real.
h: altura total del árbol. Tipo: Real.
Vcc: volumen con corteza. Tipo: Real.
Vsc: volumen sin corteza. Tipo: Real.

**Algoritmo**

0. Inicio
1. Leer especie del árbol (Esp)
2. Leer diámetro a la altura de pecho (d)
3. Leer altura total (h)
4. Si Esp = 1 entonces
   4.1 Vcc = 0.0185 + 0.32084 * d^2 * h
   4.2 Vsc = -0.0047 + 0.8947 * Vcc
   de lo contrario
   4.3 Si Esp = 2 entonces
      4.3.1 Vsc = 0.00653 + 0.000035 * d^2 * h
      4.3.2 Vcc = -0,005 + 0,9128 * Vcc
      de lo contrario
      4.3.3 Mostrar "Especie no definida"
      Fin de si 4.3
   Fin de si 4
5. Mostrar Volumen con corteza (Vcc)
6. Mostrar volumen sin corteza (Vsc)
7. Fin

13) Escribir un algoritmo para calcular el precio a pagar por la compra de madera en un aserradero. Los datos de entrada son la cantidad de metros cúbicos a comprar, el precio por metro cúbico y el tipo de madera. La madera está clasificada en tres tipos (A, B y C). Si la cantidad a comprar es superior a 30 metros cúbicos, se aplica el siguiente esquema de descuento:

| Tipo de madera | Descuento |
|---|---|
| A | 4% |
| B | 8% |
| C | 10% |

Si la cantidad comprada es inferior a 30 metros cúbicos el descuento es del 2%, independientemente del tipo de madera.

**Variables**

Cmc: cantidad de metros cúbicos de madera a comprar. Tipo: Real.
Pmc: precio de la madera por metro cúbico. Tipo: Real.
Tmad: tipo de madera (A, B, C). Tipo: carácter.
Desc: descuento. Tipo: Real.
Pre_total: precio total. Tipo: Real.

**Algoritmo**

0. Inicio
1. Leer cantidad de metros cúbicos de madera a comprar (Cmc)
2. Leer precio de la madera por metro cúbico (Pmc)
3. Leer tipo de madera (Tmad)
4. Pre_total = Cmc * Pmc
5. Si Cmc > 30 entonces
   5.1 Si Tmad = "A" entonces
   5.1.1 Desc = 0.04 * Pre_total
   de lo contrario
   5.1.2 Si Tmad = "B" entonces
   5.1.2.1 Desc = 0.08 * Pre_total
   de lo contrario
   5.1.2.2 Si Tmad = "C" entonces
   5.1.2.2.1 Desc= 0.1 * Pre_total
   Fin de si 5.1.2.2
   Fin de si 5.1.2
   Fin de si 5.1
   de lo contrario
   5.2 Desc = 0.02 * Pre_total

Fin de si 5
6. Pre_total = Pre_total - Desc
7. Mostrar descuento (Desc)
8. Mostrar precio total a pagar (Pre_total)
9. Fin

Se deja como ejercicio para el lector realizar el diagrama de flujo de este ejercicio y responder las siguientes preguntas:

a) ¿Qué descuento aplicaría el algoritmo si el tipo de madera es cualquier otra letra diferente a A, B, o C?

b) ¿Qué descuento aplicaría el algoritmo si el tipo de madera es a, b, o c (en minúsculas)? ¿Cómo resolvería esta situación?

14) Los impuestos que una empresa debe pagar en cierto municipio depende de sus ingresos y de la cantidad de empleados que tiene, tal como se indica en la siguiente tabla.

| Ingresos | Impuestos | |
|---|---|---|
| | Menos de 25 empleados | 25 empleados o más |
| **Menos de 800000** | 1% de los ingresos | 0.7% de los ingresos |
| **800000 - 1500000** | 2% de los ingresos | 1.5% de los ingresos |
| **Más de 1500000** | 30000 más el 3% de la cantidad excedente a 1500000 | 22500 más el 2.5% de la cantidad excedente a 1500000 |

Realizar un algoritmo que calcule los impuestos que una empresa debe pagar.

**Variables**

Ing: ingresos de la empresa. Tipo: Real.
CE: cantidad de empleados. Tipo: Entero.
Imp: impuesto a pagar. Tipo: Real.

**Algoritmo**

0. Inicio
1. Leer ingresos de la empresa (Ing)
2. Leer cantidad de empleados que tiene la empresa (CE)
3. Si CE < 25 entonces
   3.1 Si Ing < 800000 entonces
      3.1.1 Imp = 0.01 * Ing
   De lo contrario
      3.1.2 Si Ing <= 1500000 entonces

3.1.2.1 Imp = 0.02 * Ing
de lo contrario
3.1.2.2 Imp = 30000 + 0.03 * (Ing – 1500000)
Fin de si 3.1.2
Fin de si 3.1
de lo contrario
3.2 Si Ing < 800000 entonces
3.2.1 Imp = 0.007 * Ing
De lo contrario
3.2.2 Si Ing <= 1500000 entonces
3.2.2.1 Imp = 0.015 * Ing
de lo contrario
3.2.2.2 Imp = 22500 + 0.025 * (Ing – 1500000)
Fin de si 3.2.2
Fin de si 3.2
Fin de si 3

4. Mostrar impuesto a pagar (Imp)
5. Fin

15) Diseñar un programa para un campamento ecológico el cual desea calcular el monto total que sus huéspedes deben pagar por la estadía en sus instalaciones. El campamento dispone de habitaciones que pueden alojar hasta un máximo de 4 personas. Las tarifas del campamento por noche dependen del número de personas alojadas tal como se muestra en la siguiente tabla:

| No. personas / habitación | Bs / noche |
|---|---|
| 1 | 2000 |
| 2 | 3000 |
| 3 | 4200 |
| 4 | 5500 |

En el cálculo del precio total a pagar se deben considerar dos impuestos, el 1% del fondo de turismo y el 12% de IVA. Se desea mostrar como salida el subtotal (precio sin impuestos), los impuestos y el precio total a pagar.

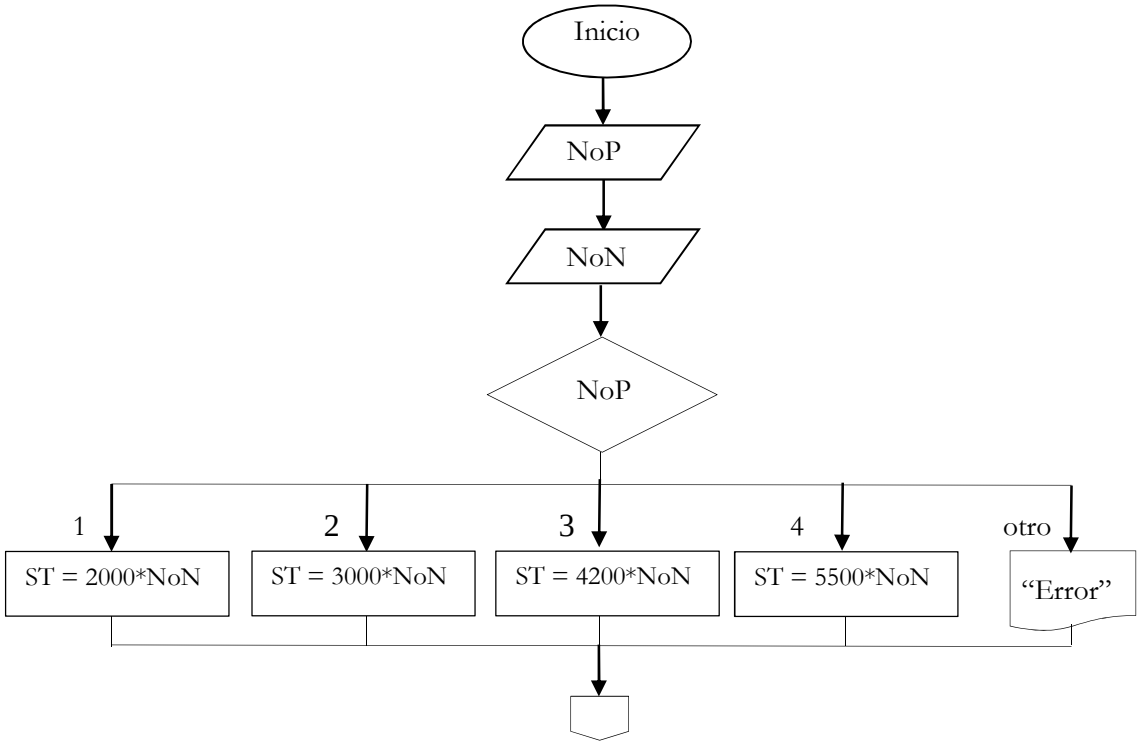
**Variables**

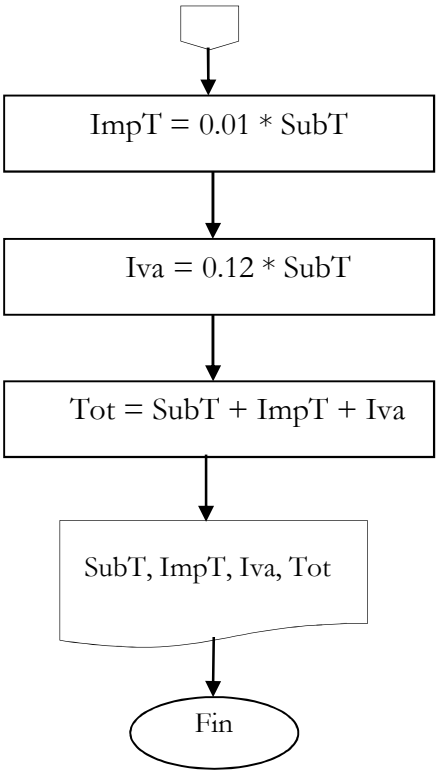
NoP: número de personas alojadas. Tipo: Entero.
NoN: número de noches de estadía. Tipo: Entero.
SubT: subtotal. Tipo: Real.
ImpT: impuesto del fondo de turismo. Tipo: Real.
Iva: impuesto al valor agregado. Tipo: Real.
Tot: total a pagar. Tipo: Real.

**Algoritmo**

0. Inicio
1. Leer número de personas alojadas (NoP)
2. Leer número de noches de estadía (NoN)
3. Seleccionar caso NoP
   1: SubT = 2000 * NoN
   2: SubT = 3000 * NoN
   3: SubT = 4200 * NoN
   4: SubT = 5500 * NoN
   de lo contrario
   Mostrar "Error: el número de personas debe estar entre 1 y 4 "
   Fin de seleccionar
4. ImpT = 0.01 * SubT
5. Iva = 0.12 * SubT
6. Tot = SubT + ImpT + Iva
7. Mostrar subtotal (SubT)
8. Mostrar impuesto del fondo de turismo (ImpT)
9. Mostrar impuesto al valor agregado (Iva)
10. Mostrar total a pagar (Tot)
11. Fin

**Diagrama de flujo**






16) Un productor forestal planta varios tipos de especies madereras. Dependiendo del tipo de madera y de la cantidad de árboles presentes en su plantación al momento de la corta final, es posible calcular la producción estimada en $m^3$, tal como se muestra en la siguiente tabla:

| Tipo de madera | $m^3$ / árbol |
|---|---|
| Muy valiosa (M) | 0.754 |
| Valiosa (V) | 0.512 |
| Regular (R) | 0.424 |
| Ordinaria (O) | 0.188 |

Realizar un algoritmo que dado el tipo de madera y la cantidad de árboles a cortar, estime la producción en $m^3$.

**Variables**

Tm: tipo de madera (M, V, R, O). Tipo: Carácter.
Ca: cantidad de árboles a cortar. Tipo: Entero.
Prod: producción estimada en $m^3$. Tipo: Real.

**Algoritmo**

0. Inicio
1. Leer tipo de madera (Tm)
2. Leer cantidad de árboles a cortar (Ca)
3. Seleccionar caso Tm
   "M": Prod = 0.754 * Ca
   "V": Prod = 0.512 * Ca
   "R": Prod = 0.424 * Ca
   "O": Prod = 0.188 * Ca
   de lo contrario
   Mostrar "Tipo de madera erróneo"
   Fin de seleccionar
4. Mostrar producción estimada en $m^3$ (Prod)
5. Fin

Obsérvese que en la estructura Seleccionar caso los posibles valores de la variable Tm (tipo de madera) se colocan entre comillas ("M", "V", "R", "O"), esto se debe a que la variable Tm es de tipo carácter.

17) Escribir un algoritmo para calcular el área de un triángulo, de un rectángulo o de un círculo. Se debe elegir una figura y de acuerdo a esto el algoritmo solicita los datos necesarios y aplica la fórmula correspondiente.

**Variables**

Fig: figura a elegir (T, R, C). Tipo: Carácter.
b: base del triángulo. Tipo: Real.
h: altura del triángulo. Tipo: Real.
l: largo del rectángulo. Tipo: Real.
a: ancho del rectángulo. Tipo: Real.
r: radio de un círculo. Tipo: Real.
AreaFig: área de la figura. Tipo: Real.

**Algoritmo**

0. Inicio
1. Leer figura (Fig)
2. Seleccionar caso Fig
   "T", "t": Leer base del triángulo (b)
   Leer altura del triángulo (h)
   AreaFig = b * h / 2

"R", "r": Leer largo del rectángulo (l)
Leer ancho del rectángulo (a)
AreaFig = l * a
"C", "c": Leer radio del círculo (r)
AreaFig = 3.1416 * $r^2$
de lo contrario
Mostrar "Opción equivocada"
Fin de seleccionar
3. Mostrar área de la figura (AreaFig)
4. Fin

18) Una compañía consultora ofrece sus servicios de asesoría financiera a diversas empresas. El precio viene dado por el tipo de asesoría, tal como se indica en la siguiente tabla:

| Tipo de asesoría | Costo ($/hora) |
|---|---|
| A | 3000 |
| B | 2000 |
| C | 1000 |
| D | 500 |

En estos momentos, la compañía desea aplicar un descuento especial que depende del número de horas de asesoría:

| Número de horas | Descuento |
|---|---|
| < 5 | No hay descuento |
| [5, 10) | 5% |
| [10, 15) | 8% |
| >= 15 | 10% |

Elaborar un algoritmo que calcule el precio total a pagar por una asesoría.

**Variables**

Ta: tipo de asesoría (A, B, C, D). Tipo: carácter.
Nh: número de horas de asesoría. Tipo: Real.
D: descuento. Tipo: Real.
Pt: precio total a pagar. Tipo: Real.

**Algoritmo**

0. Inicio
1. Leer tipo de asesoría (Ta)
2. Leer número de horas de asesoría (Nh)
3. Seleccionar caso Ta
   "A", "a": Pt = 3000 * Nh
   "B", "b": Pt = 2000 * Nh
   "C", "c": Pt = 1000 * Nh
   "D", "d": Pt = 500 * Nh
   Fin de seleccionar
4. Si nh < 5 entonces
   4.1 D = 0
   de lo contrario
   4.2 Si nh < 10 entonces
   4.2.1 D = 0.05 * Pt
   de lo contrario
   4.2.2 Si nh < 15 entonces
   4.2.2.1 D = 0.08 * Pt
   de lo contrario
   4.2.2.2 D = 0.1 * Pt
   Fin de se 4.2.2
   Fin de si 4.2
   Fin de si 4
5. Pt = Pt – D
6. Mostrar descuento (D)
7. Mostrar precio total a pagar por la asesoría (Pt)
8. Fin

## 3.9 Ejercicios propuestos

1) Resolver las siguientes expresiones lógicas

a) NOT (X AND Y OR X) si X= Verdadero y Y=Falso

b) NOT (A > B) OR (B > 4) AND (A < = 1) si A= 5 y B= 8

2) Construir una expresión lógica cuyo valor sea verdadero, si la edad de una persona es mayor a 60 años o menor a 12 años. El resultado de la expresión debe ser falso si la edad no cumple ninguna de estas condiciones.

3) Construir una expresión lógica cuyo valor sea verdadero, si una persona es de sexo femenino y tiene más de 25 años. En otros casos, el resultado de la expresión debe ser falso.

4) Diseñar un programa para calcular el precio a pagar por la compra de un artículo conocido su precio de venta. Si su precio de venta es mayor a 25000 Bs se debe realizar un descuento del 1%. El IVA a pagar es del 12% .

5) Los empleados de una fábrica trabajan en uno de los siguientes turnos: diurno o nocturno. Escriba un algoritmo que calcule el salario de un trabajador de acuerdo con lo siguiente:

- El pago diurno es de 1000 Bs/hora

- El pago nocturno es de 1500 Bs/hora

6) Diseñar un programa que dado un número entero determine si es par o impar.

7) Realizar un algoritmo para calcular el valor de *Y*, el cual está dado por la siguiente ecuación $Y = \frac{x^3 + 5}{x}$. Si $x = 0$ debe dar un mensaje de error e indicar que no puede realizarse el cálculo.

8) Escribir un algoritmo que dado el diámetro de un árbol de cierta especie diga cuál es su posible uso de acuerdo a la siguiente tabla:

| Diámetro (cm) | Uso |
|---|---|
| <10 | Leña |
| [10, 20) | Estantillos |
| [20 – 25) | Puntales y varetas |
| >= 25 | Madera para aserrío |

9) Escribir un algoritmo que dado un valor de x calcule el valor de la función f(x).

$$f(x) = \begin{cases} 3x & si\ x < 10 \\ x^2 - 9 & si\ 10 \leq x < 20 \\ \sqrt{x} & si\ x \geq 20 \end{cases}$$

9) Una empresa que distribuye cierto producto cuenta con varios vendedores, los cuales ganan una comisión mensual de acuerdo a sus ventas. Diseñar un programa para calcular la comisión que le corresponde a un vendedor utilizando a los siguientes criterios:

| Ventas (Bs.) | Comisión (% sobre las ventas) |
|---|---|
| < 500000 | 0.5 |
| [500000 - 1000000) | 1 |
| [1000000 – 1500000) | 2 |
| ≥ 1500000 | 2.5 |

11) Diseñar un programa que permita introducir un número entre 1 y 12 correspondiente a un mes y muestre el número de días que tiene ese mes.

12) Escribir un algoritmo que tenga como dato de entrada una calificación alfabética A, B, C, D, E, o F, y muestre la correspondiente calificación numérica 20, 17, 14, 10, 5, 0, respectivamente.

13) Diseñar un programa para calcular el salario total de un trabajador teniendo en cuenta su salario base y el número de horas extra trabajadas. Para el pago de las horas extra se debe considerar la categoría del trabajador, de acuerdo a la siguiente tabla:

| Categoría | Precio de la hora extra |
|---|---|
| A | 2000 |
| B | 1750 |
| C | 1500 |
| D | 1000 |

Además si el trabajador tiene más de 15 años en la empresa se le dará un bono de 3500 Bs.

14) Un negocio mayorista que vende tablas de madera de ciertas medidas y especie, tiene clasificado a sus clientes en tres tipos: 1 (si paga a tiempo), 2 (si se retrasa con los pagos) y 3 (si es un cliente nuevo). Se requiere diseñar un programa que dado el número de tablas que un cliente compra, el precio unitario (precio de una tabla) y el tipo de cliente, calcule el precio total que dicho cliente debe pagar, considerando un descuento. Si el cliente es tipo 1 el descuento es del 15%, si es tipo 2 tiene un descuento del 5%, y si es tipo 3 tiene un descuento del 2%.

15) Para cubicar o estimar el volumen de una troza obtenida al tumbar un árbol se pueden utilizar varias fórmulas, asemejando la forma de las trozas a ciertos sólidos de revolución como cilindro, paraboloide, cono, o neiloide. Tres de las fórmulas más usadas son las siguientes:

Fórmula de Smalian

$$V = \frac{\pi}{8}(d1 + d2)^2 \times L$$

Fórmula de Huber

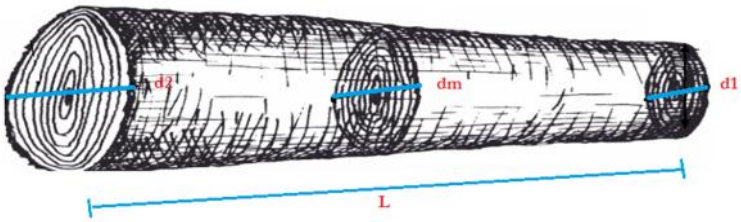
$$V = \frac{\pi}{4} dm \times L$$

Fórmula de Newton

$$V = \frac{\pi}{24} (d1^2 + dm^2 + d2^2) \times L$$

donde:

V: volumen de la troza (en metros cúbicos)
d1: diámetro menor de la troza (en metros)
d2: diámetro mayor de la troza (en metros)
dm: diámetro a la mitad de la longitud de la troza (en metros)
L: longitud de la troza (en metros)




Diseñar un programa que permita elegir una de las tres fórmulas anteriores y estime el volumen de una troza.

# CAPÍTULO 4
# ESTRUCTURAS DE REPETICIÓN

Las estructuras de repetición, también llamadas "bucles" o "lazos", permiten que un programa ejecute un conjunto de instrucciones varias veces, tantas como sea necesario. Son útiles cuando se desean procesar un conjunto de datos. Por ejemplo, supóngase que para cada árbol de una plantación se conocen una serie variables que fueron medidas en el campo y a partir de éstas se desea calcular mediante un programa de computación otras variables de interés. Si hay *n* cantidad de árboles, en el algoritmo no es necesario colocar para cada árbol una instrucción de lectura de cierta variable o escribir *n* veces una ecuación, si fuera así el programa sería demasiado extenso y muy tedioso de codificar. Las instrucciones comunes solo se escriben una vez pero dentro de una estructura de repetición, ésta indicará cuántas veces se debe repetir el proceso.

Existen tres tipos básicos de estructuras de repetición: "para", "mientras" y "hasta". Dependiendo del lenguaje de programación utilizado hay variantes para cada una de ellas. En las próximas secciones se explica la lógica de las estructuras de repetición (válida para cualquier lenguaje) y se presentan ejemplos de su aplicación.

## 4.1 Repetir Para

Se utiliza cuando es posible conocer de antemano el número exacto de repeticiones, el cual puede ser un dato de entrada o un valor dado en el planteamiento del problema.

La forma de escribir una estructura "Repetir Para" en pseudocódigo es la siguiente:

```
Repetir Para  variable_índice = Vi  hasta  Vf
        Instrucciones a repetir
Fin de RP
```

La estructura "Repetir Para" ejecuta un conjunto de instrucciones un determinado número de veces. El número de repeticiones se define mediante una variable, a veces llamada índice, la cual tiene un valor inicial (Vi) y un valor final (Vf) declarados en la estructura de repetición. El número de repeticiones viene dado por:

$$\textit{Número de repeticiones} = Vf - Vi + 1$$

Por ejemplo, si se desea repetir 100 veces un conjunto de instrucciones el valor inicial de la variable índice puede ser 1 y el valor final 100. También podría usarse cualquier otra combinación de valores que al ser introducidos en la fórmula anterior dé como resultado 100 repeticiones, por ejemplo, valor inicial 50 y valor final 149.

Cada vez que se ejecutan las instrucciones que están dentro de la estructura de repetición, la variable índice se incrementa en 1 hasta que su valor sea mayor que el valor final, cuando esto ocurre finaliza la ejecución del bucle. El incremento de la variable índice siempre es 1, aunque algunos lenguajes de programación permiten indicar un valor diferente para el incremento.

La Figura 4.1 ilustra el diagrama de flujo de un "Repetir Para".

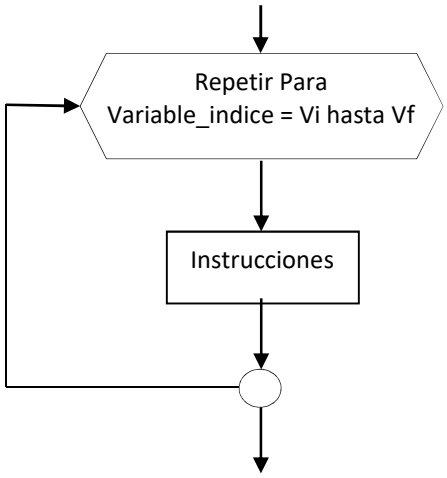



Figura 4.1. Diagrama de flujo de una estructura de repetición tipo "Repetir Para"

Ejemplo 1 "Repetir Para":

Dada la altura en metros y el diámetro en cm de los 1000 árboles de una parcela, calcular el volumen de cada árbol usando la siguiente ecuación:

$$V = 0.26\, x \left( \left( \frac{d}{100} \right)^2 x\, h \right)^{0.96}$$

donde $V$ es el volumen del árbol en $m^3$, $d$ es el diámetro a la altura de pecho en cm y $h$ es la altura total expresada en m.

**Análisis E-P-S**

*Entrada*

Para cada árbol se requiere los siguientes datos de entrada:
d: diámetro a la altura de pecho. Tipo: Real.
h: altura total del árbol. Tipo: Real.

*Proceso*

Calcular para cada árbol el volumen:
Repetir 1000 veces el siguiente cálculo
V = 0.26 * ((d/100)^2 *h)^0.96

*Salida*

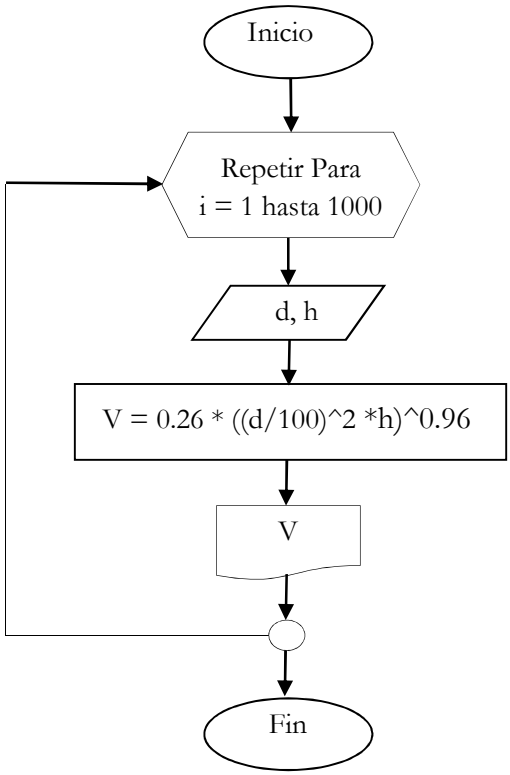
Para cada árbol de la plantación se mostrará como salida:
V: volumen del árbol. Tipo: Real

**Algoritmo**

0. Inicio
1. Repetir Para i= 1 hasta 1000
   1.1 Leer diámetro del árbol i (d)
   1.2 Leer altura del árbol i (h)
   1.3 V = 0.26 * ((d/100)^2 *h)^0.96
   1.4 Mostrar volumen del árbol i (V)
   Fin de RP
2. Fin

Nota: la variable " i " es la variable índice del Repetir Para, su valor inicial es 1 y en cada ejecución de la estructura de repetición su valor se incrementa en 1.

**Diagrama de flujo**




Ejemplo 2 "Repetir Para":

Diseñar un programa que dada la nota de cada uno de los estudiantes de una asignatura calcule el promedio general del curso. Asumir que el número de estudiantes es una variable de entrada.

**Análisis E-P-S**

*Entrada*

ne: número de estudiantes que cursan la asignatura. Tipo: Entero.

De cada estudiante se requiere:

Nota: nota del estudiante en la asignatura. Tipo: Real.

*Proceso*

1) Repetir para cada estudiante (*ne* veces):
   Leer la nota y sumarla a la suma acumulada de las notas de los estudiantes anteriores.
2) Calcular el promedio general del curso:

$$Prom = \frac{Suma\ de\ las\ notas}{ne}$$

*Salida*

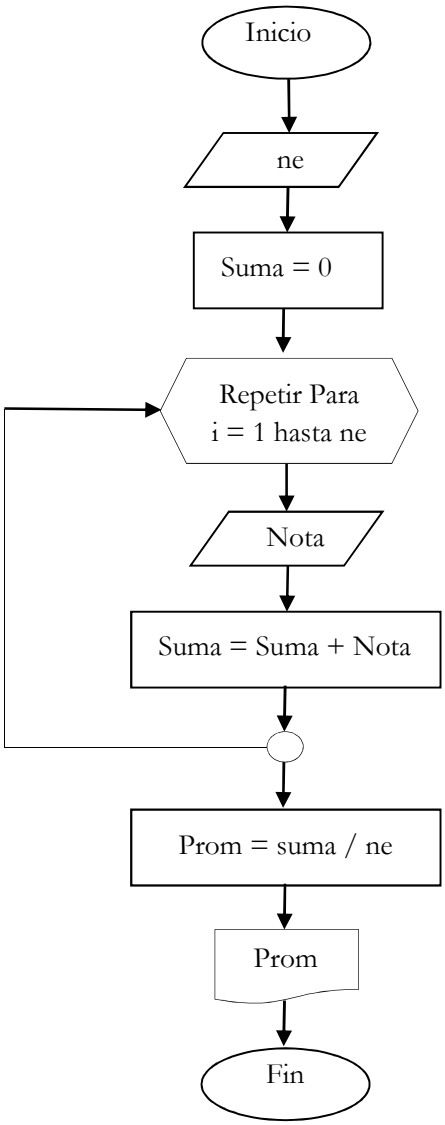
Prom: promedio general del curso. Tipo: Real

**Algoritmo**

0. Inicio
1. Leer número de estudiantes (ne)
2. Suma = 0
3. Repetir Para j = 1 hasta ne
   3.1 Leer nota del estudiante j (nota)
   3.2 Suma = Suma + nota
   Fin de RP
4. Prom = Suma / ne
5. Mostrar promedio del curso (prom)
6. Fin.

La variable Suma de este ejemplo es un **acumulador**. Una variable acumuladora se usa cuando se necesita sumar valores sucesivos de una variable dentro de una estructura de repetición. Generalmente se inicializa en cero.

**Diagrama de flujo**



## 4.2 Repetir Mientras

Esta estructura repite la ejecución de un conjunto de instrucciones mientras que una expresión lógica (condición) es verdadera. Cuando la expresión lógica es falsa se detiene el ciclo de repeticiones.

Al igual que el "Repetir Para", un "Repetir Mientras" puede utilizarse cuando se conoce el número de repeticiones o éste puede obtenerse como un dato de entrada. También un "Repetir Mientras" es adecuado cuando no se conoce de antemano el número de repeticiones o la cantidad de datos a procesar, pues el fin del ciclo de repeticiones se controla mediante una expresión lógica.

En pseudocódigo una estructura "Repetir Mientras" se escribe de la siguiente manera:

```
Repetir Mientras  expresión lógica
        Instrucciones a repetir
Fin de RM
```

En la Figura 4.2 se muestra el diagrama de flujo correspondiente a un "Repetir Mientras".

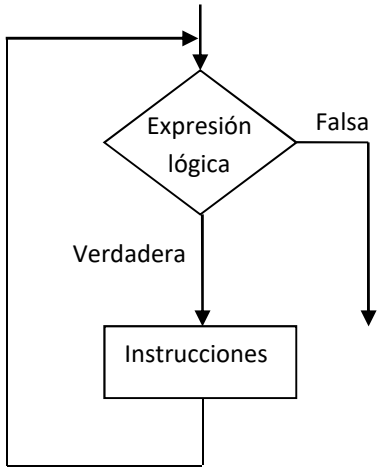



Figura 4.2 Diagrama de flujo de una estructura "Repetir Mientras"

Como un "Repetir Mientras" se ejecuta solo si la expresión lógica es verdadera, antes de una estructura de este tipo se debe colocar una instrucción que garantice que inicialmente la expresión lógica sea verdadera, para que las instrucciones dentro de la estructura de repetición se ejecuten al menos una vez.

Luego, dentro del repetir deben estar las instrucciones necesarias para que en algún momento la expresión lógica sea falsa y las repeticiones se detengan. De no incluir estas instrucciones se puede tener un ciclo de repeticiones infinito, pues la expresión lógica siempre será verdadera.

Ejemplo 1 "Repetir Mientras":

Diseñar un programa que dada la nota de cada uno de los estudiantes de una asignatura calcule el promedio general del curso. Asumir que el número de estudiantes es una variable de entrada.

Este mismo problema fue resuelto usando "Repetir Para", por tanto el lector puede observar la diferencia entre ambas estructuras de repetición.

**Análisis E-P-S**

*Entrada*

ne: número de estudiantes del curso. Tipo: Entero.
De cada estudiante se requiere:
Nota: nota del estudiante en la asignatura. Tipo: Real.

*Proceso*

1) Inicializar en 1 una variable (j) que cuente las repeticiones (j=1)
2) Repetir mientras j<=ne
   Leer la nota de un estudiante y sumarla a la suma acumulada de las notas de los estudiantes anteriores.
3) Calcular el promedio general del curso:

$$Prom = \frac{Suma\ de\ las\ notas}{ne}$$

*Salida*

Prom: promedio general del curso. Tipo: Real

**Algoritmo**

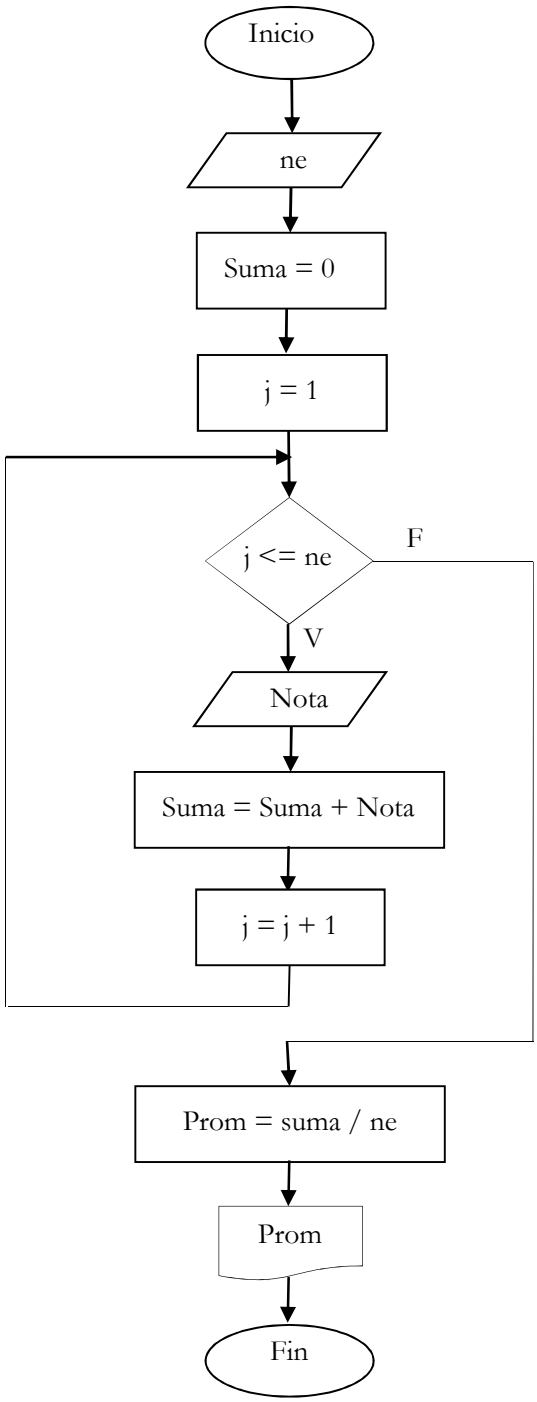
0. Inicio
1. Leer número de estudiantes (ne)
2. Suma = 0
3. j=1
4. Repetir mientras j <= ne
   4.1 Leer nota del estudiante j (nota)

La variable j cuenta el número de repeticiones, se inicializa en 1 para que la condición j <=ne sea verdadera y se procese el primer estudiante.

4.2 Suma = Suma + nota
4.3 j= j+ 1
Fin de RM
5. Prom = Suma / ne
6. Mostrar promedio del curso (Prom)
7. Fin.

Se incrementa la variable j cada vez que se procesa la nota de un estudiante. Cuando esta variable se hace mayor que el número de estudiantes (ne), la condición del "Repetir Mientras" es falsa y finalizan las repeticiones.

**Diagrama de flujo**



Ejemplo 2 "Repetir Mientras":

Diseñar un programa que reciba como entrada el volumen de cada árbol presente en una plantación y calcule el volumen total (sumatoria de los volúmenes), asumiendo que se desconoce cuántos árboles se van a procesar. Además del volumen total, el programa debe mostrar como salida el número de árboles que fueron procesados.

En este ejemplo el número de repeticiones es desconocido, pues no se sabe cuántos árboles van a ser procesados. Hay varias formas de manejar esta situación, una de ellas es usar un valor centinela, esto es, se define un valor que el usuario utilizará para indicar que no desea introducir más datos (en este ejemplo los datos corresponden al volumen). Así, la terminación de las repeticiones dependerá de un dato de entrada.

El valor centinela que se eligió en este ejemplo es -1, cuando el usuario no desee introducir más datos escribirá como volumen ese valor. Al introducir -1 el programa no solicitará más datos de entrada y mostrará la salida. También puede usarse otro valor centinela, siempre y cuando no sea un valor posible para el volumen.

**Análisis E-P-S**

*Entrada*

Para cada árbol se requiere:
vol: volumen del árbol. Tipo: Real.

*Proceso*

1) Inicializar la variable vol en 0 (vol = 0)
2) Repetir mientras vol ≠ -1
   Leer volumen de un árbol y sumarlo al volumen acumulado de los árboles anteriores.
   Cada vez que se lea un volumen incrementar una variable n en 1, esta llevará la cuenta del número de árboles procesados.

*Salida*

n: número de árboles procesados. Tipo: Real
volTotal: suma de los volúmenes de todos los árboles de la plantación. Tipo: Real.

**Algoritmo**

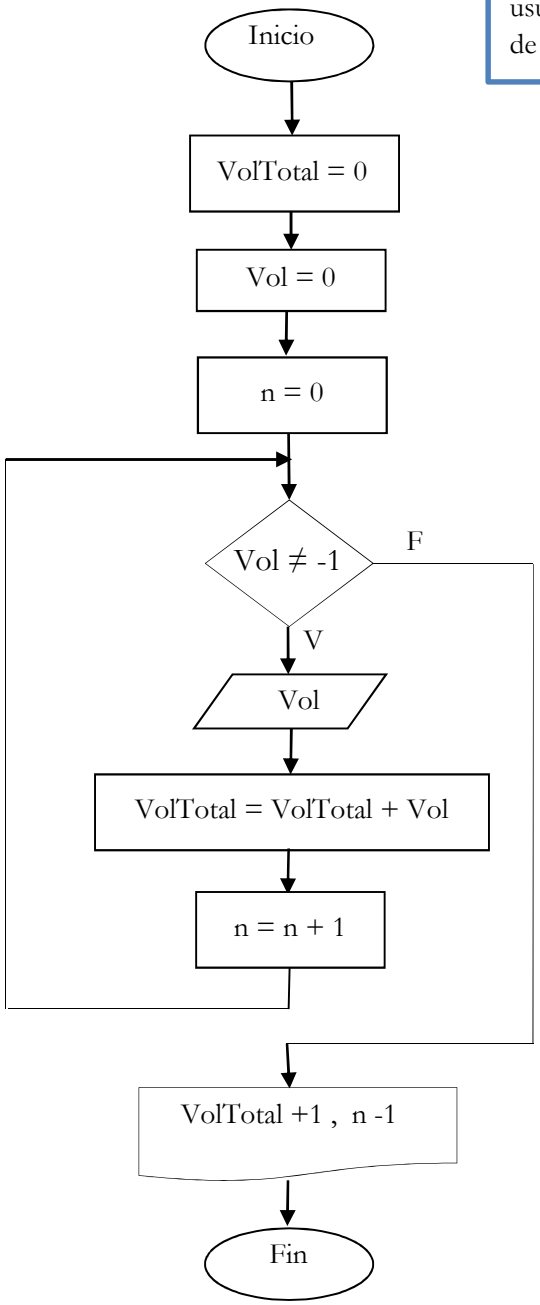
0. Inicio
1. volTotal = 0
2. vol = 0
3. n=0
4. Repetir mientras vol ≠ -1
   3.1 Leer volumen del árbol (vol)

3.2 volTotal = volTotal + vol
3.3 n= n+ 1
Fin de RM
5. Mostrar volumen total (volTotal +1)
6. Mostrar número de árboles procesados (n -1)
7. Fin.

Se incrementa la variable n cada vez que se introduce el volumen de un árbol, al finalizar el algoritmo el valor de n indica el número de árboles procesados. A este tipo de variables se le denomina **contador**.

A la variable volTotal se le suma 1 y a n se le resta 1, para que su valor real no se vea afectado por el valor centinela que el usuario introdujo para finalizar la entrada de datos.

**Diagrama de flujo**



## 4.3 Repetir Hasta

Esta estructura permite repetir la ejecución de un conjunto de instrucciones hasta que una expresión lógica sea verdadera, o lo que es igual, repite mientras una expresión lógica es falsa. Cuando la expresión lógica es verdadera, el bucle deja de ejecutarse y el algoritmo continúa en el paso siguiente.

Un "Repetir Hasta" se puede utilizar en los mismos casos que se usa "Repetir Mientras", sin importar si el número de repeticiones es conocido o desconocido. La lógica del problema indicará cuál de las dos estructuras es más conveniente emplear, o si es indiferente utilizar cualquiera de ellas.

Una estructura "Repetir Hasta" se escribe de la siguiente manera en pseudocódigo:

```
Repetir
        Instrucciones a repetir
Hasta  expresión lógica
```

En la Figura 4.3 se muestra la representación mediante diagrama de flujo de un "Repetir Hasta".

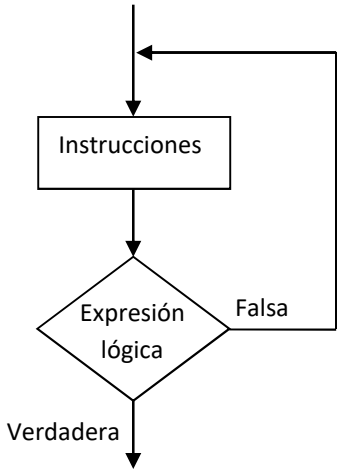



Figura 4.3 Diagrama de flujo de una estructura "Repetir Hasta"

Ejemplo 1 "Repetir Hasta":

Diseñar un programa que dada la nota de cada uno de los estudiantes de una asignatura, calcule el promedio general del curso. Asumir que el número de estudiantes que cursa la asignatura es una variable de entrada.

Antes se resolvió este problema usando "Repetir Para" y "Repetir Mientras", ahora se escribirá un algoritmo que incluye una estructura "Repetir Hasta".

**Análisis E-P-S**

*Entrada*

ne: número de estudiantes del curso. Tipo: Entero.

De cada estudiante se requiere:

Nota: nota del estudiante en la asignatura. Tipo: Real.

*Proceso*

1) Inicializar en 1 una variable (j) que cuente las repeticiones ( j=1).
2) Repetir

   Leer la nota y sumarla a la suma acumulada de las notas anteriores.

   Hasta j > ne
3) Calcular el promedio general del curso:

$$Prom = \frac{Suma\ de\ las\ notas}{ne}$$

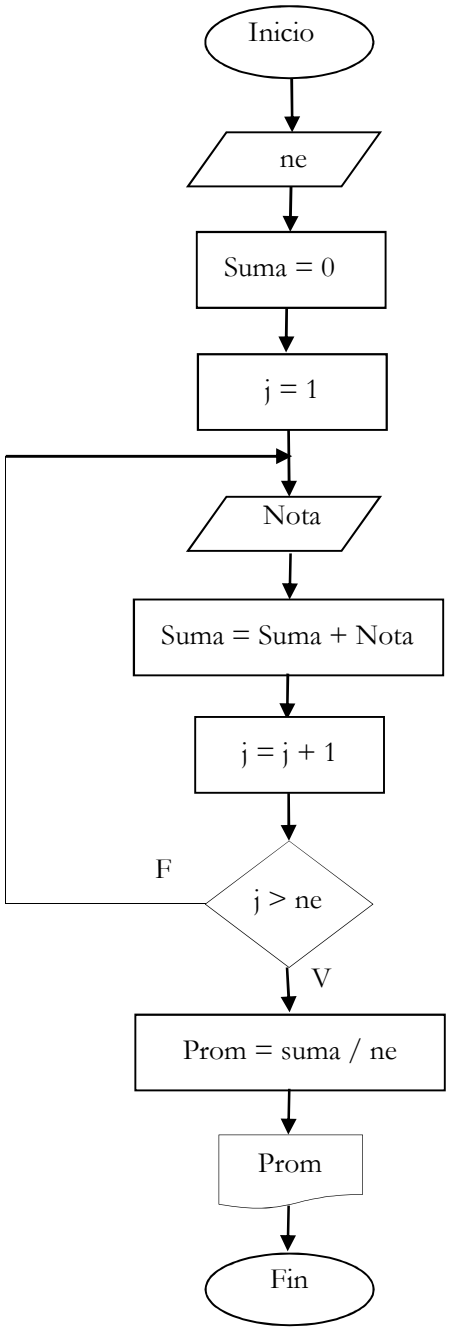
*Salida*

Prom: promedio general del curso. Tipo: Real

**Algoritmo**

0. Inicio
1. Leer número de estudiantes (ne)
2. Suma = 0
3. j=1
4. Repetir
   4.1 Leer nota del estudiante j (nota)
   4.2 Suma = Suma + nota
   4.3 j= j+ 1
   Hasta j > ne
5. Prom = Suma / ne
6. Mostrar promedio del curso (prom)
7. Fin.

**Diagrama de flujo**




En un "Repetir Hasta" el ciclo de repeticiones se ejecuta al menos una vez antes de comprobar la expresión lógica o condición de repetición. Esto hace que en algunos casos, sea más conveniente este tipo de estructura que un "Repetir Mientras", tal como se muestra en el siguiente ejemplo.

Ejemplo 2 "Repetir Hasta":

Diseñar un programa para calcular la suma y la media aritmética de un conjunto de números. Se asume que no se conoce de antemano cuántos números se van a procesar.

En este ejemplo el número de repeticiones es desconocido. El uso de un valor centinela para controlar el ciclo de repeticiones no conviene en este ejemplo. Recuérdese que el valor centinela es un valor no válido para los datos, y en este caso los datos son números cualesquiera, el problema no especifica un rango de valores para los mismos por lo que no es posible elegir un valor centinela adecuado.

Para detener las repeticiones se preguntará al usuario si desea introducir más números, si su respuesta es afirmativa se ejecuta la estructura de repetición y si su respuesta es negativa, se termina el ciclo.

**Análisis E-P-S**

*Entrada*

num: cada uno de los números a procesar. Tipo: Real.

*Proceso*

1) Repetir
   Leer un número y sumarlo a los números antes leídos.
   Cada vez que se lea un número incrementar una variable *n* en 1, esta llevará la cuenta la cantidad de números procesados.
   Preguntar al usuario si desea procesar otro número
   Hasta que el usuario indique que no desea procesar más números.
3) Calcular media aritmética

$$M = \frac{\sum_{i=1}^{n} num}{n}$$

*Salida*

Suma: sumatoria de los números. Tipo: Real
Media: media aritmética de los números. Tipo: Real.

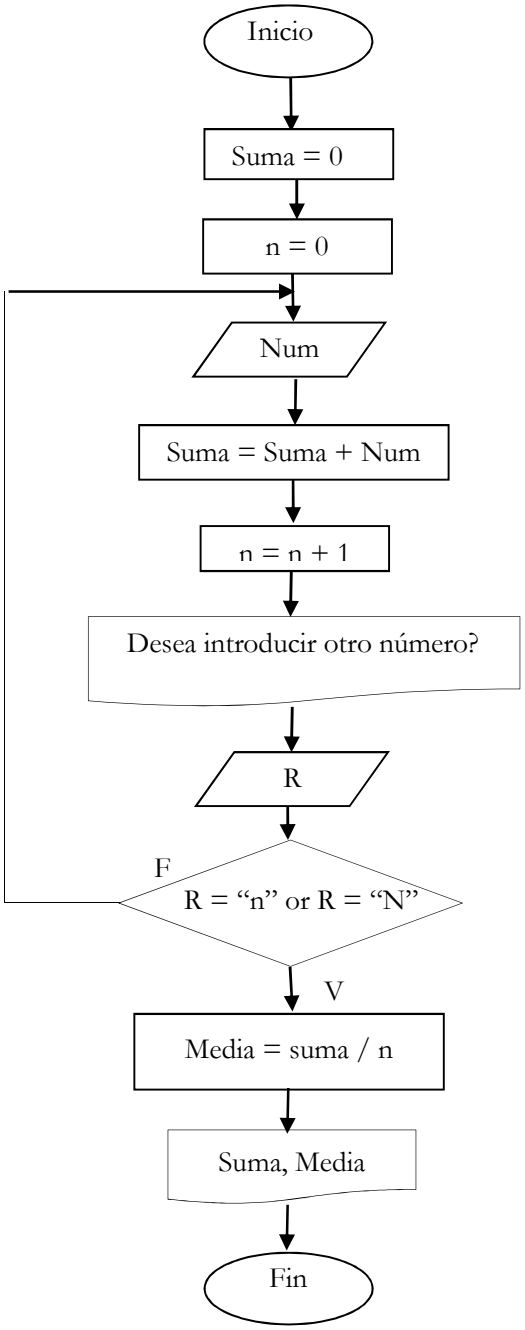
**Algoritmo**

0. Inicio
1. Suma = 0
2. n = 0
3. Repetir
   3.1 Leer número (num)
   3.2 Suma = suma + num
   3.3 n= n+ 1

3.4 Mostrar mensaje (“Desea introducir otro número S / N ?”)
3.5 Leer respuesta (R)
Hasta R=” N ” or R = “n”
4. Media = suma / n
5. Mostrar suma de los números (suma)
6. Mostrar media aritmética (media)
7. Fin.

La respuesta es una variable tipo carácter cuyos valores son: “S” o ”s” si es afirmativa, “N” o “n” si es negativa.

**Diagrama de flujo**



## 4.4 Ejercicios resueltos

A continuación se presentan algunos ejercicios resueltos referentes a estructuras repetición. Tal como se hizo en el capítulo 3, no se mostrará el análisis E-P-S pues se deja como ejercicio para el lector. Todos los problemas planteados pueden solucionarse de manera diferente a la acá mostrada. Algunos pueden resolverse usando dos de las estructuras de repetición vistas, otros pueden plantearse mediante los tres "Repetir". Se recomienda intentar resolver los ejercicios usando estructuras de repetición distintas.

1) Elaborar un algoritmo para calcular el valor de la siguiente serie

$$1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \ldots\ldots + \frac{1}{n}$$

donde $n$ es un valor introducido por el usuario.

**Variables**

n: número de términos de la serie. Tipo: Entero.
Suma: variable acumuladora que suma todos los términos de la serie. Tipo: Real.
i: variable que controla el número de repeticiones. Tipo: Entero.

**Algoritmo**

0. Inicio
1. Leer número de términos de la serie (n)
2. Suma = 0
3. Repetir Para i= 1 hasta n
   3.1 Suma = Suma + 1/i
   Fin de RP
4. Mostrar el valor de la serie (Suma)
5. Fin

2) Diseñar un programa que calcule el salario de cada uno de los trabajadores de una empresa. Para cada trabajador se tienen los siguientes datos: número de horas trabajadas en tiempo regular y número de horas extra. La hora en tiempo regular se paga a 300 Bs. y la hora en tiempo extra se paga a 450 Bs. Además el programa debe calcular el valor de la nómina (suma de todos los salarios). Se asume que el número de trabajadores es conocido por el usuario del programa.

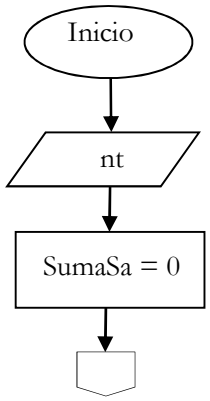
**Variables**

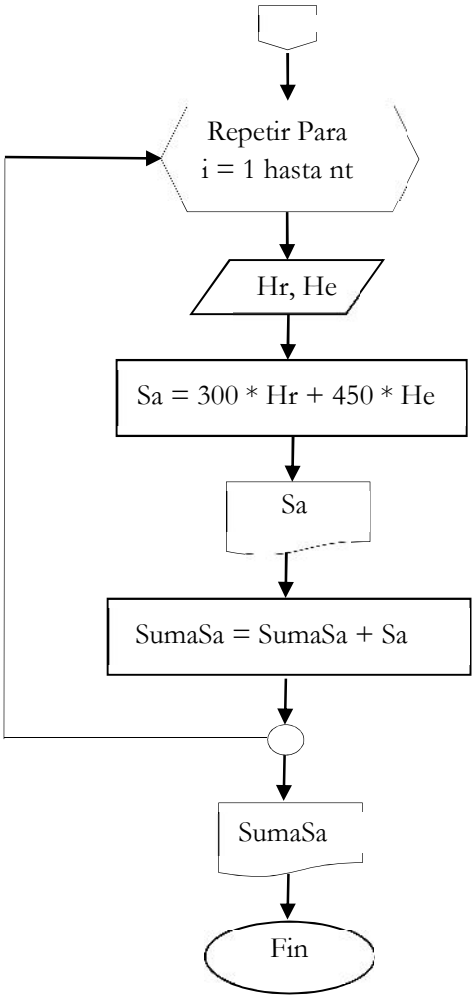
nt: número de trabajadores de la empresa. Tipo: Entero.
Hr: número de horas trabajadas por un empleado en tiempo regular. Tipo: Real.
He: número de horas trabajadas por un empleado en tiempo extra. Tipo: Real.
Sa: salario de un trabajador. Tipo: Real.
SumaSa: variable acumuladora que suma todos los salarios. Tipo: Real.
i: variable que controla el número de repeticiones. Tipo: Entero.

**Algoritmo**

0. Inicio
1. Leer número de trabajadores en la empresa (nt)
2. SumaSa = 0
3. Repetir Para i= 1 hasta nt
   3.1 Leer número de horas en tiempo regular del trabajador i (Hr)
   3.2 Leer número de horas extra del trabajador i (He)
   3.3 Sa = 300 * Hr + 450 * He
   3.4 Mostrar salario del trabajador i (Sa)
   3.5 SumaSa = SumaSa + Sa
   Fin de RP
4. Mostrar el valor de la nómina (SumaSa)
5. Fin

**Diagrama de flujo**






3) Escribir un algoritmo que lea los diámetros a la altura de pecho medidos en cm correspondientes a 500 árboles de una plantación y determine:

a) Cantidad de árboles con diámetro menor a 10 cm.
b) Cantidad de árboles con diámetro entre 10 y 18 cm (ambos inclusive).
c) Cantidad de árboles con diámetro superior a 18 cm.
d) Diámetro promedio.

**Variables**

i: variable que controla el número de repeticiones. Tipo: Entero.
Dap: diámetro a la altura de pecho de un árbol. Tipo: Real
SumaD: variable acumuladora que suma todos los diámetros. Tipo: Real.
C1: cantidad de árboles con diámetro menor a 10 cm. Tipo: Entero.
C2: cantidad de árboles con diámetro entre 10 y 18 cm. Tipo: Entero.
C3: Cantidad de árboles con diámetro superior a 18 cm. Tipo: Entero.

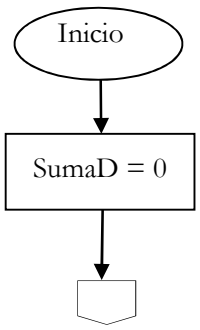
Dprom: diámetro promedio. Tipo: Real.

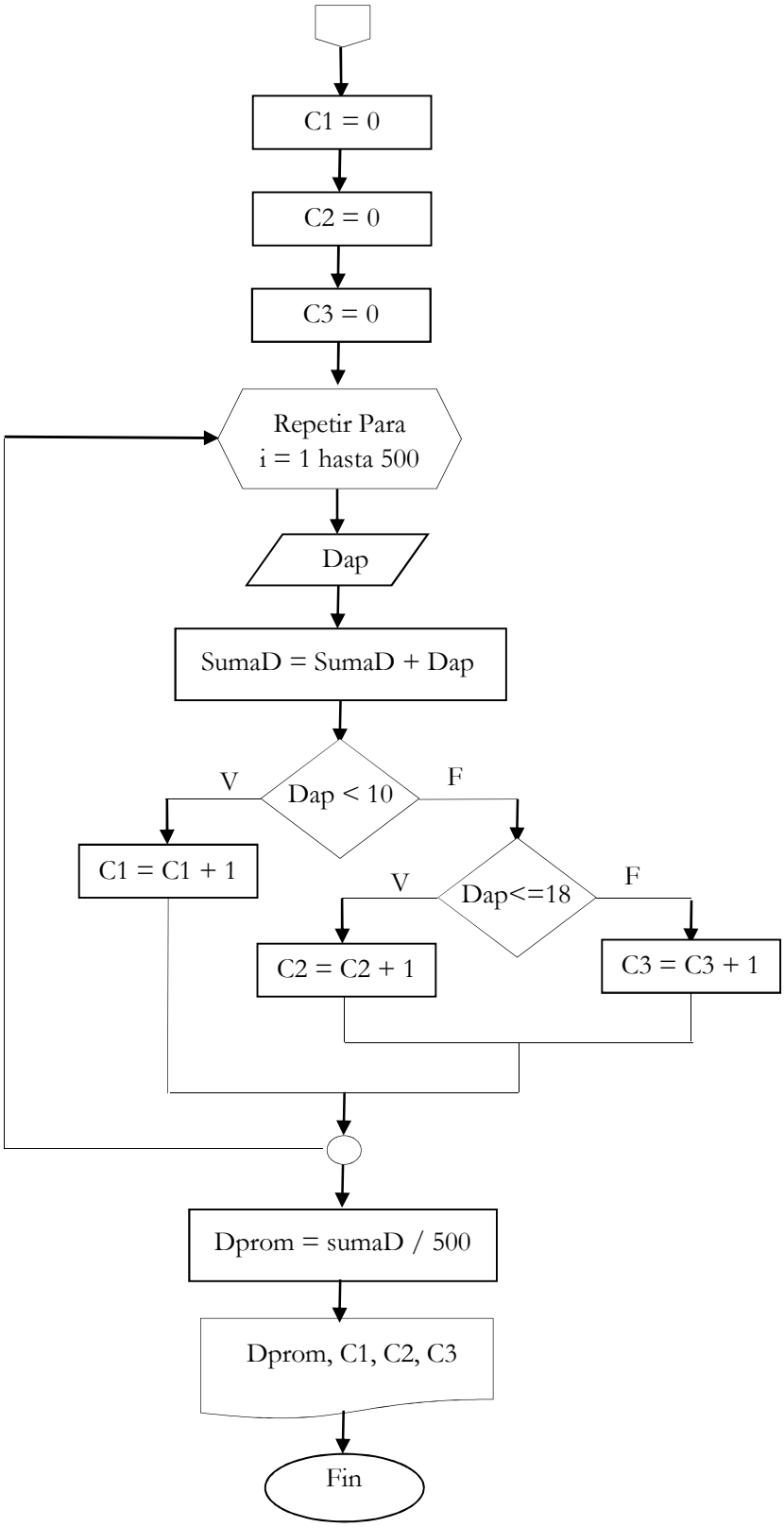
**Algoritmo**

0. Inicio
1. SumaD = 0
2. C1 = 0
3. C2 = 0
4. C3 = 0
5. Repetir Para i= 1 hasta 500
   3.1 Leer diámetro a la altura de pecho del árbol i (Dap)
   3.2 SumaD = SumaD + Dap
   3.4 Si Dap < 10 Entonces
   3.4.1 C1 = C1 + 1
   De lo contrario
   3.4.2 Si dap < = 18 Entonces
   3.4.2.1 C2 = C2 + 1
   De lo contrario
   3.4.2.2 C3 = C3 + 1
   Fin de si 3.4.2
   Fin de si 3.4
   Fin de RP
6. Dprom = SumaD / 500
7. Mostrar cantidad de árboles con diámetro menor a 10 cm (C1)
8. Mostrar cantidad de árboles con diámetro entre 10 y 18 cm C(C2)
9. Mostrar Cantidad de árboles con diámetro superior a 18 cm (C3)
10. Mostrar diámetro promedio (Dprom)
11. Fin

Las variables C1, C2 y C3 son **contadores**. Este tipo de variables cuenta la cantidad de individuos, datos u objetos que hay en una lista. Estos elementos en ocasiones cumplen cierta condición lógica, como en este ejemplo, los árboles se clasifican en tres grupos de acuerdo a su diámetro y se cuentan haciendo uso de estas variables. Los contadores siempre se incrementan en 1 cada vez que se cumple la condición lógica que los define.

**Diagrama de flujo**




C1 = 0
C2 = 0
C3 = 0
Repetir Para
i = 1 hasta 500
Dap
SumaD = SumaD + Dap
Dap < 10
V
F
C1 = C1 + 1
Dap<=18
V
F
C2 = C2 + 1
C3 = C3 + 1
Dprom = sumaD / 500
Dprom, C1, C2, C3
Fin

4) Diseñar un programa que tenga como datos de entrada una lista de 250 números enteros y calcule la suma de los números pares y la suma de los números impares. Además el programa debe indicar cuántos números pares e impares hay en el conjunto de números.

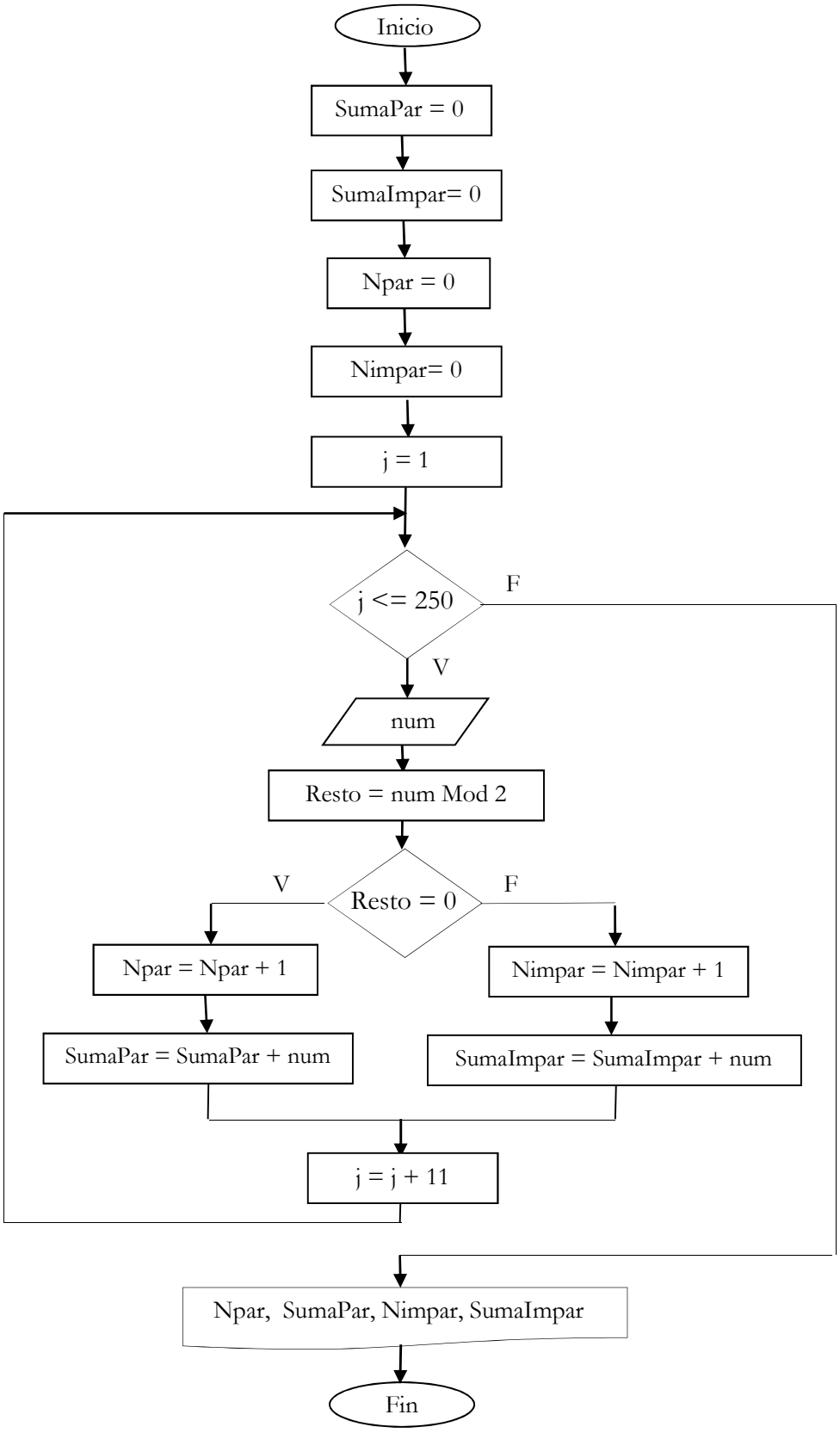
**Variables**

j: variable que controla en número de repeticiones. Tipo: Entero.
num: cualquier número de la lista. Tipo: Entero.
Resto: resto de dividir un número entre 2, este permite determinar si un número es par o impar. Tipo: Entero.
SumaPar: variable acumuladora que almacena la suma de los números pares. Tipo: Entero.
SumaImpar: variable acumuladora que almacena la suma de los números impares. Tipo: Entero.
Npar: variable contadora que almacena la cantidad de números pares. Tipo: Entero.
Nimpar: variable contadora que almacena la cantidad de números impares. Tipo: Entero.

**Algoritmo**

0. Inicio
1. SumaPar = 0
2. SumaImpar = 0
3. Npar = 0
4. Nimpar = 0
5. j = 1
6. Repetir mientras j <= 250
   6.1 Leer (num)
   6.2 Resto = num Mod 2
   6.3 Si Resto = 0 entonces
      6.3.1 Npar = Npar + 1
      6.3.2 SumaPar = SumaPar + num
   de lo contrario
      6.3.3 Nimpar = Nimpar + 1
      6.3.4 SumaImpar= SumaImpar + num
   Fin de si 6.3
   6.4 j = j + 1
   Fin de RM
7. Mostrar cantidad de números pares (Npar)
8. Mostrar suma de los números pares (SumaPar)
9. Mostrar cantidad de números impares (Nimpar)
10. Mostrar suma de los números impares (SumaImpar)
11. Fin.

**Diagrama de flujo**



5) Diseñar un algoritmo para procesar los datos mensuales de producción de una fábrica que elabora cierto producto. La fábrica cuenta con *n* empleados, cada uno de ellos puede elaborar cierta cantidad de productos (unidades) en un mes. La cantidad de unidades producidas por un empleado depende de sus habilidades y destrezas, hay que considerar que en ocasiones algunos productos una vez terminados tienen ciertos defectos por lo que deben ser descartados. De cada empleado se conocen los siguientes datos: cantidad total de unidades producidas y cantidad de unidades defectuosas producidas en un mes.

El algoritmo debe calcular:

a) Total de unidades producidas por todos los empleados.
b) Total de unidades defectuosas producidas por todos los empleados.
c) Número de empleados que produjeron más de 10 unidades defectuosas
d) Porcentaje de unidades defectuosas producidas por la fábrica.

**Variables**

n: número de empleados de la fábrica. Tipo: Entero.
Upro : cantidad de unidades producidas al mes por un empleado. Tipo: Entero.
Udef: cantidad de unidades defectuosas producidas al mes por un empleado. Tipo: Entero.
SumaUni: variable acumuladora que suma las unidades producidas por todos los empleados. Tipo: Entero.
SumaDef: variable acumuladora que suma las unidades defectuosas producidas por todos los empleados.
Tipo: Entero.
Ne: variable que cuenta el número de empleados que produjeron más de 10 unidades defectuosas. Tipo: Entero.
PorcDef: porcentaje de unidades defectuosas producidas por la fábrica. Tipo: Real.
i: variable que controla el número de repeticiones. Tipo: Entero.

**Algoritmo**

0. Inicio
1. Leer número de empleados de la fábrica (n)
2. SumaUni = 0
3. SumaDef = 0
4. Ne = 0
5. i = 1
6. Repetir mientras i <= n
   6.1 Leer cantidad de unidades producidas al mes por el empleado i (Upro)
   6.2 Leer cantidad de unidades defectuosas producidas al mes por el empleado i (Udef)

```
        6.3 SumaUni = SumaUni + Upro
        6.4 SumaDef = SumaDef + Udef
        6.5 Si Udef > 10 Entonces
                6.5.1 Ne = Ne + 1
            Fin de si 6.5
        6.6 i = i + 1
        Fin de RM
7.  PorcDef = SumaDef / SumaUni  * 100
8.  Mostrar total de unidades producidas por todos los empleados (SumaUni)
9.  Mostrar total de unidades defectuosas producidas por todos los empleados (SumaDef)
10. Mostrar número de empleados que produjeron más de 10 unidades defectuosas (Ne)
11. Mostrar porcentaje de unidades defectuosas producidas por la fábrica (PorcDef)
12. Fin
```

6) El gobierno regional realizó una encuesta a cierto número de familias de distintos sectores de la ciudad de Mérida para estudiar los ingresos que éstas perciben mensualmente. Se necesita diseñar un programa que lea el ingreso de cada familia y calcule:

a) Ingreso promedio de las familias de Mérida.
b) Número de familias con ingresos de 50000 Bs. o menos.
c) Número de familias con ingresos entre 50000 y 100000 Bs.
d) Número de familias con ingresos superiores a 100000 Bs.

Para facilitar el manejo de los datos se asumirá un número de repeticiones desconocido. El usuario deberá introducir uno a uno los ingresos de cada familia y cuando no tenga más datos que procesar, debe escribir un valor no posible para el ingreso (valor centinela) que detenga la ejecución de la estructura de repetición, por ejemplo -5 (el ingreso no puede ser un valor negativo por ello este puede ser un valor centinela).

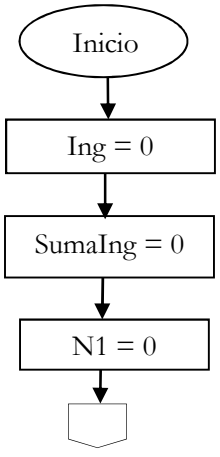
**Variables**

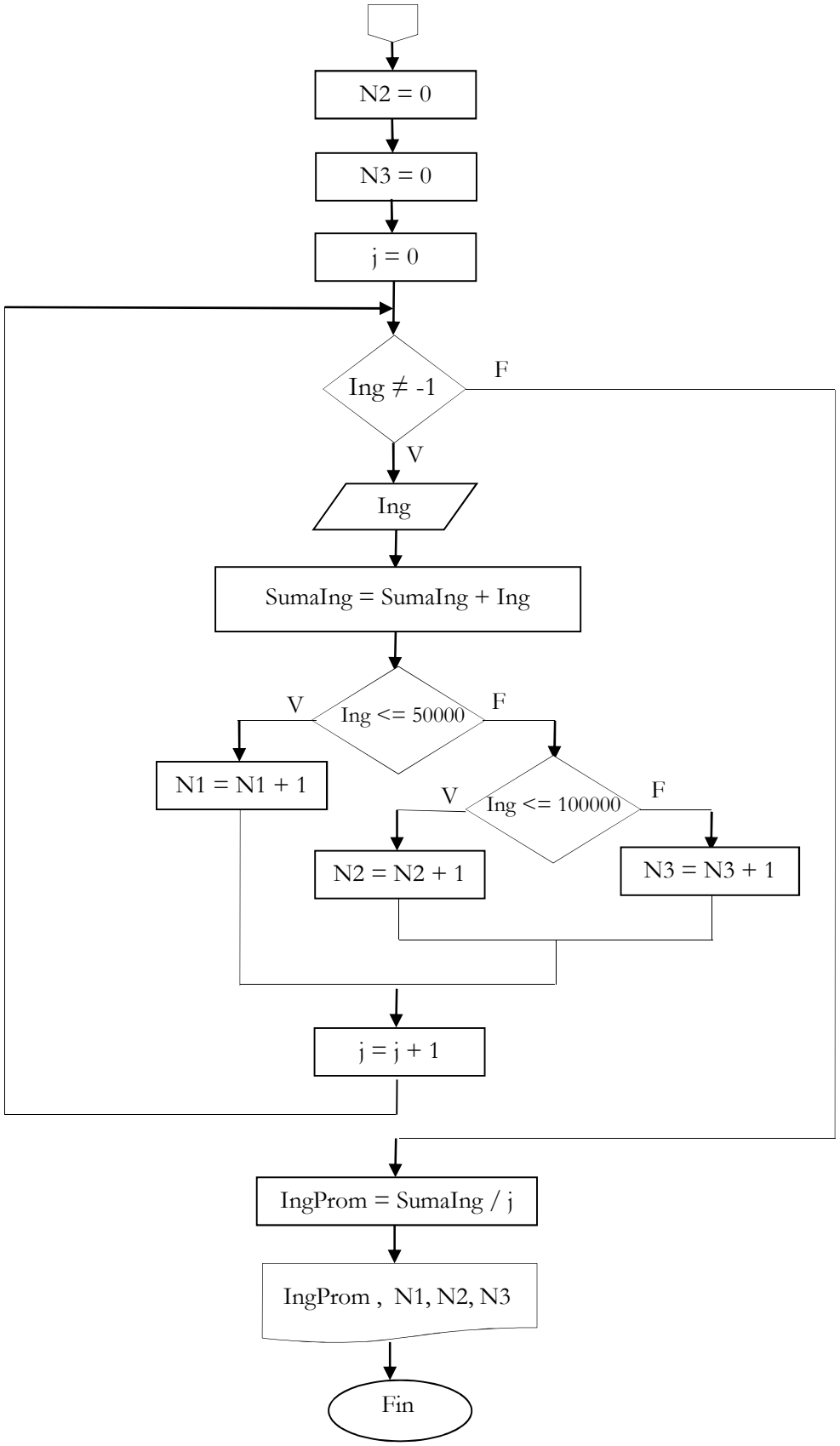
j: variable que cuenta el número de familias encuestadas. Tipo: Entero.
Ing: ingreso de una familia. Tipo: Real
SumaIng: variable acumuladora que suma los ingresos de todas las familias. Tipo: Real.
N1: número de familias con ingresos de 50000 Bs. o menos. Tipo: Entero.
N2: número de familias con ingresos entre 50000 y 100000 Bs. Tipo: Entero.
N3: número de familias con ingresos superiores a 100000 Bs. Tipo: Entero.
IngProm: ingreso promedio de las familias de Mérida. Tipo: Real.

**Algoritmo**

0. Inicio
1. Ing=0
2. SumaIng = 0
3. N1 = 0
4. N2 = 0
5. N3 = 0
6. j = 0
7. Repetir mientras Ing ≠ -5
   7.1 Leer (Ing)
   7.2 SumaIng = SumaIng + ing
   7.3 Si Ing <= 50000 entonces
   7.3.1 N1 = N1 + 1
   de lo contrario
   7.3.2 Si Ing <= 100000 Entonces
   7.3.2.1 N2 = N2 + 1
   de lo contrario
   7.3.2.2 N3 = N3 +1
   Fin de si 7.3.2
   Fin de si 7.3
   7.4 j = j + 1
   Fin de RM
8. IngProm = SumaIng / j
9. Mostrar ingreso promedio de las familias de Mérida (IngProm)
10. Mostrar cantidad de familias con ingresos inferiores o iguales a 50000 Bs. (N1)
11. Mostrar cantidad de familias con ingresos entre 50000 y 100000 Bs.(N2)
12. Mostrar cantidad de familias con ingresos superiores a 100000 Bs. (N3)
13. Fin

**Diagrama de flujo**




N2 = 0
N3 = 0
j = 0
Ing ≠ -1
F
V
Ing
SumaIng = SumaIng + Ing
V
Ing <= 50000
F
N1 = N1 + 1
V
Ing <= 100000
F
N2 = N2 + 1
N3 = N3 + 1
j = j + 1
IngProm = SumaIng / j
IngProm , N1, N2, N3
Fin

7) La Facultad de Ciencias Forestales y Ambientales necesita un programa para obtener la siguiente información acerca de sus estudiantes:

a) Número de estudiantes del sexo masculino
b) Número de estudiantes del sexo femenino
c) Edad promedio
d) Promedio de notas general de la facultad.

Diseñar un programa que permita introducir los datos individuales de cada estudiante y muestre como salida la información que requiere la facultad. Para resolver el problema se puede asumir que la Facultad conoce la cantidad de estudiantes que tiene inscritos.

**Variables**

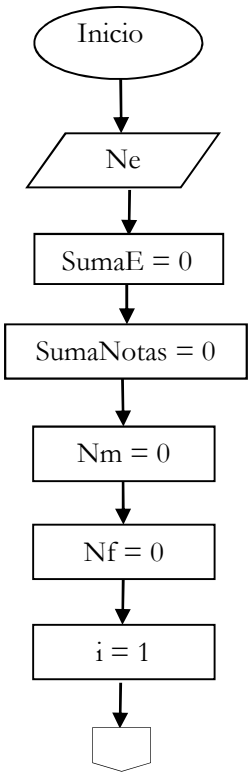
Ne: número de estudiantes. Tipo: Entero.
i: variable que controla el número de repeticiones. Tipo: Entero.
Ed: edad de un estudiante. Tipo: Entero.
Sx: indica si un estudiante es masculino (M, m) o femenino (F, f). Tipo: Carácter.
Prom: promedio de un estudiante.
SumaE: variable acumuladora que suma las edades de todos los estudiantes. Tipo: Entero.
SumaNotas: variable acumuladora que suma el promedio de todos los estudiantes. Tipo: Real.
Nm: número de estudiantes del sexo masculino. Tipo: Entero.
Nf: número de estudiantes del sexo femenino. Tipo: Entero.
Eprom: edad promedio de los estudiantes de la facultad. Tipo: Real.
PromF: nota promedio de los estudiantes de la facultad. Tipo: Real.

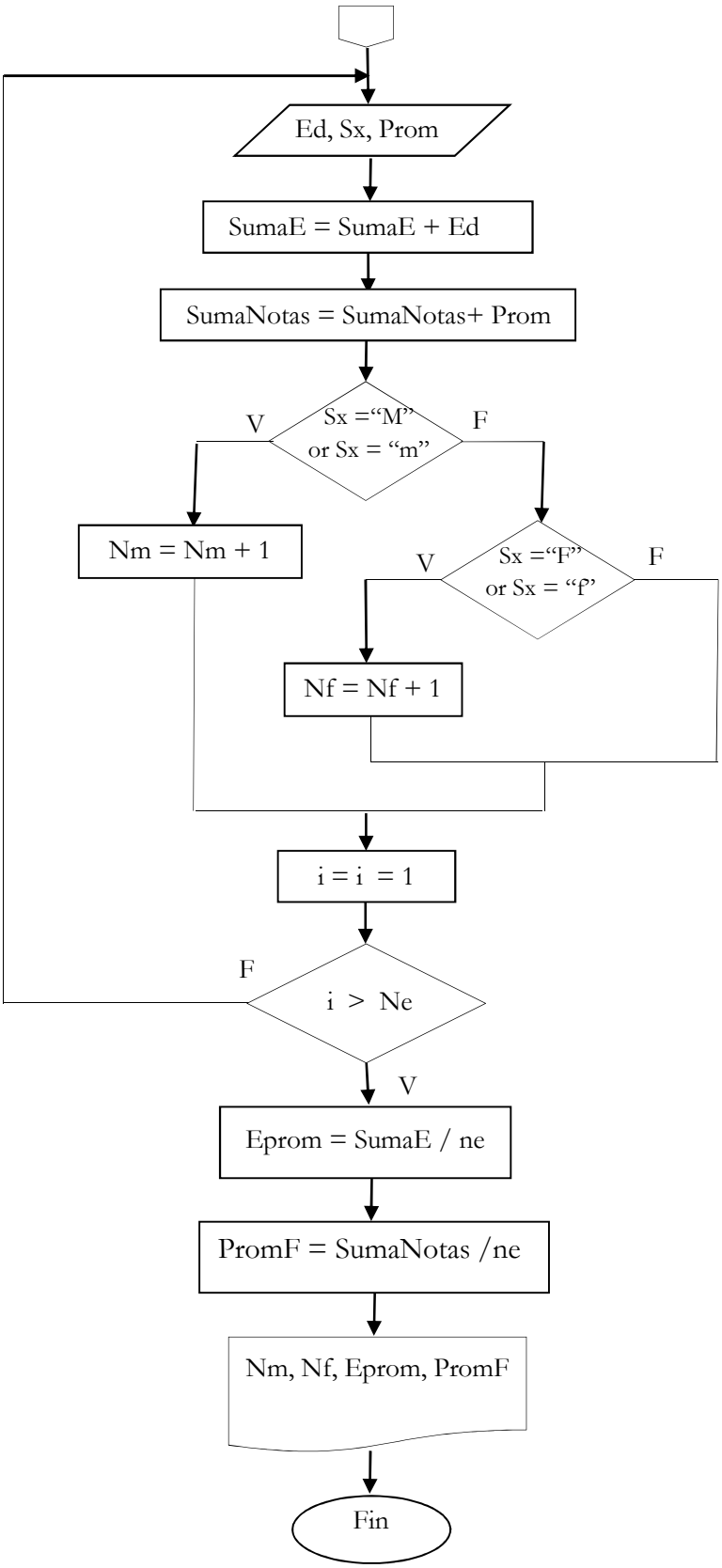
**Algoritmo**

0. Inicio
1. Leer (Ne)
2. SumaE = 0
3. SumaNotas = 0
4. Nm = 0
5. Nf = 0
6. i = 1
7. Repetir
   7.1 Leer (Ed)
   7.2 Leer (Sx)
   7.3 Leer (Prom)
   7.4 SumaE = SumaE + Ed
   7.5 SumaNotas = SumaNotas + Prom
   7.6 Si Sx = "M" or Sx = "m" entonces
      7.6.1 Nm = Nm + 1

De lo contrario
7.6.2 Si Sx = "F" or Sx = "f" entonces
7.6.2.1 Nf = Nf + 1
Fin de si 7.6.2
Fin de si 7.6
7.7 i = i + 1
Hasta i > ne
8. Eprom = SumaE / ne
9. PromF = SumaNotas /ne
10. Mostrar número de estudiantes del sexo masculino (Nm)
11. Mostrar número de estudiantes del sexo femenino (Nf)
12. Mostrar edad promedio (Eprom)
13. Mostrar promedio de notas general de la facultad (PromF)
14. Fin

**Diagrama de flujo**




Ed, Sx, Prom
SumaE = SumaE + Ed
SumaNotas = SumaNotas+ Prom
Sx ="M"
or Sx = "m"
V
F
Nm = Nm + 1
Sx ="F"
or Sx = "f"
V
F
Nf = Nf + 1
i = i = 1
F
i > Ne
V
Eprom = SumaE / ne
PromF = SumaNotas /ne
Nm, Nf, Eprom, PromF
Fin

Para responder sobre el problema 7:

¿Qué sucede si al introducir los datos de un estudiante, el usuario por error escribe cualquier otra letra para el sexo, diferente de M, m, F, f ?

¿Cómo mejoraría el diseño del programa para solventar este problema?

8) Diseñar un algoritmo que tenga como datos de entrada *n* números enteros y determine:

a) Cuántos de los números introducidos por el usuario fueron positivos, cuántos negativos y cuántos cero.
b) Promedio de los números positivos
c) promedio de todos los números

**Variables**

N: cantidad de números a procesar. Tipo: Entero.
i: variable que controla el número de repeticiones. Tipo: Entero.
Num: Número cualquiera introducido por el usuario. Tipo: Entero.
Cp: cantidad de números positivos. Tipo: Entero.
Cn: cantidad de números negativos. Tipo: Entero.
Cc: cantidad de ceros. Tipo: Entero.
SumaP: variable acumuladora que suma los números positivos. Tipo: Entero.
Suma: variable acumuladora que suma todos los números. Tipo: Entero.
Promp: promedio de números positivos. Tipo: Real.
Prom: promedio de todos los números. Tipo: Real.

**Algoritmo**

0. Inicio
1. Leer (N)
2. Cp = 0
3. Cn = 0
4. Cc = 0
5. Suma = 0
6. SumaP = 0
7. k = 1
8. Repetir mientras k <= N
   8.1 Leer (Num)
   8.2 Suma = Suma + Num
   8.3 Si Num > 0 entonces
      8.3.1 Cp = Cp + 1
      8.3.2 SumaP = SumaP + Num

```
        de lo contrario
          8.3.3  Si Num = 0 entonces
                 8.3.3.1 Cc = Cc + 1
                 de lo contrario
                 8.3.3.2 Cn = Cn + 1
                 Fin de si 8.2.2
        Fin de si 8.2
      8.4  k = k + 1
      Fin de RM
9.  Prom = Suma / N
10. Promp = SumaP / Cp
11. Mostrar cantidad de números positivos (Cp)
12. Mostrar cantidad de números negativos (Cn)
13. Mostrar cantidad de ceros (Cc)
14.  Mostrar promedio general (Prom)
15. Mostrar promedio de los números positivos (Promp)
16. Fin
```

9) Diseñar un programa para calcular el promedio de notas de un estudiante que presenta tres evaluaciones. El programa debe validar los datos de entrada, es decir, solamente debe aceptar notas entre 0 y 20. Cualquier valor fuera de este rango no debe ser aceptado, en tal caso el programa debe indicarlo y solicitar nuevamente la nota correspondiente.

**Variables**

nota1: nota de la evaluación 1. Tipo: Real.
nota2: nota de la evaluación 2. Tipo: Real.
nota3: nota de la evaluación 3. Tipo: Real.
P: promedio de notas. Tipo: Real.

**Algoritmo**

```
0.  Inicio
1.  Repetir
      1.1  Leer (nota1)
      1.2  Si nota1 < 0  or  nota1 > 20 entonces
           1.2.1  Mostrar "Error!!  La nota de la evaluación 1 debe estar entre 0 y 20"
      Fin de si 1.2
    Hasta  nota1 >=0 and nota1 <=20

2. Repetir
      2.1 Leer (nota2)
```

La nota de cada evaluación se lee dentro de un Repetir Hasta. Si el usuario introduce una nota fuera del rango permitido (0-20), se muestra un mensaje de error y se repite la lectura, es decir, el usuario debe introducir un nuevo valor. Este proceso se repite hasta que se introduzca un valor válido para la nota.

2.2 Si nota2 <0 or nota2 > 20 entonces

2.2.1 Mostrar "Error!! La nota de la evaluación 2 debe estar entre 0 y 20"

Fin de si 2.2

Hasta nota2 >= 0 and nota2 <= 20

3. Repetir

3.1 Leer (nota3)

3.2 Si nota3 <0 or nota3 > 20 entonces

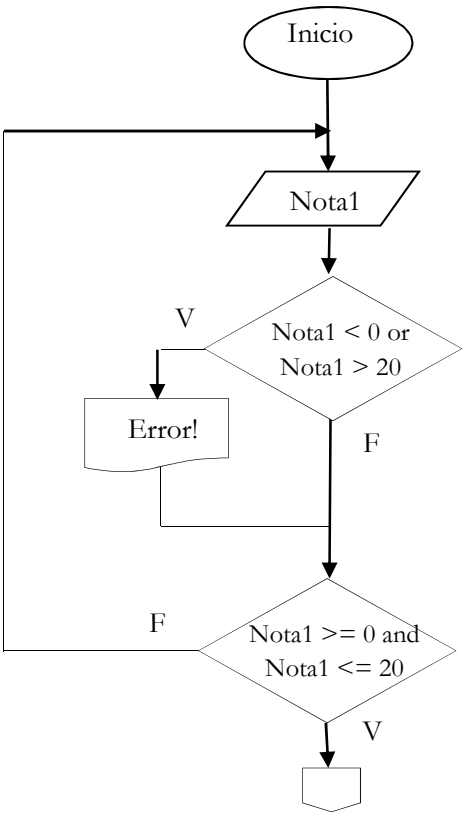
3.2.1 Mostrar "Error!! La nota de la evaluación 3 debe estar entre 0 y 20"

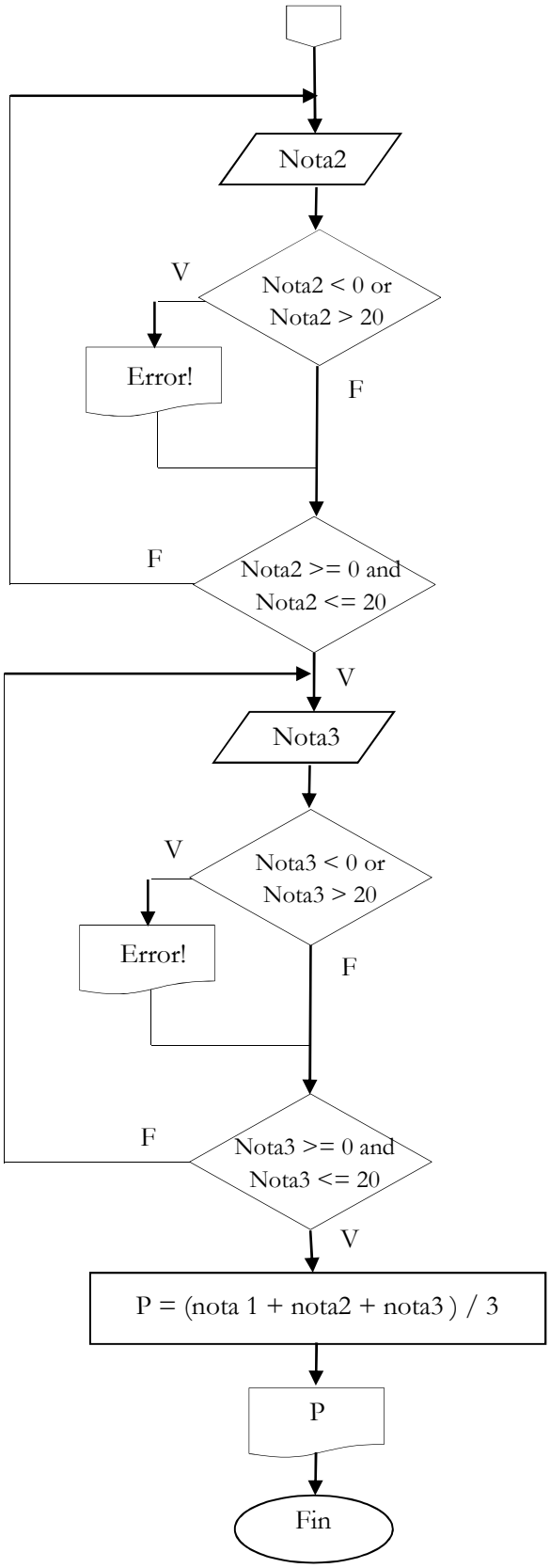
Fin de si 3.2

Hasta nota3 >= 0 and nota3 <= 20

4. P = (nota1 + nota2 + nota3) / 3
5. Mostrar promedio (P)
6. Fin

**Diagrama de flujo**




Nota2
V
Nota2 < 0 or
Nota2 > 20
Error!
F
F
Nota2 >= 0 and
Nota2 <= 20
V
Nota3
V
Nota3 < 0 or
Nota3 > 20
Error!
F
F
Nota3 >= 0 and
Nota3 <= 20
V
P = (nota 1 + nota2 + nota3 ) / 3
P
Fin

10) Escribir un algoritmo que permita calcular la suma de los gastos mensuales de una persona. Se asume que no se conoce de antemano cuántos gastos son, por ello después de leer cada gasto el programa debe preguntar al usuario si tiene más gastos que procesar. Cuando la respuesta sea negativa el programa mostrará la suma de los gastos.

**Variables**

G: gasto. Tipo: Real.
SumaG: variable acumuladora que almacena la suma de los gastos. Tipo: Real.
R: respuesta cuyo valor es "S" o "s" si el usuario tiene más gastos que procesar, N" o "n" si no hay más gastos para procesar. Tipo: carácter.

**Algoritmo**

0. Inicio
1. SumaG = 0
2. Repetir
   2.1 Leer (G)
   2.2 SumaG = SumaG + G
   2.3 Mostrar "¿Tiene más gastos para procesar S/N?"
   2.4 Leer (R)
   Hasta R = "N" or R = "n"
3. Mostrar suma de los gastos mensuales (SumaG)
4. Fin

Se recomienda al lector resolver nuevamente este problema usando un valor centinela para detener la estructura de repetición. Además se deja como ejercicio modificar el algoritmo para que muestre un mensaje de alerta si la suma de los gastos es mayor a los ingresos de la persona.

11) Elaborar un algoritmo que reciba como entrada 24 números reales que representan las temperaturas registradas en una estación meteorológica en un período de 24 horas, y calcule: temperatura promedio y temperatura más alta.

**Variables**

T: temperatura medida en cierta hora del día. Tipo: Real.
SumaT: variable acumuladora que almacena la suma de las temperaturas. Tipo: Real.
Tprom: temperatura promedio. Tipo: Real.
Tmayor: temperatura más alta. Tipo: Real.
i: variable contadora que indica el número de repeticiones. Tipo: Entero

**Algoritmo**

0. Inicio

1. SumaT = 0
2. Tmayor = -1000
3. Repetir Para i = 1 a 24
   3.1 Leer (T)
   3.2 SumaT = SumaT + T
   3.3 Si T > Tmayor entonces
   3.3.1 Tmayor = T
   Fin de si 3.3
   Fin de RP
4. Tprom = SumaT / 24
5. Mostrar temperatura promedio (Tprom)
6. Mostrar temperatura más alta (Tmayor)
7. Fin

Para determinar la temperatura mayor, una estrategia es inicializar la variable Tmayor en un valor muy bajo, un valor que no sea factible para la temperatura.

Queda como ejercicio para el lector modificar el algoritmo para que también determine la temperatura más baja.

12) En un experimento se mide el PH del suelo en dos zonas diferentes (A y B). Elaborar un algoritmo que lea el PH medido en 20 muestras de suelo en la zona A y 20 muestras de la zona B, y determine:

a) PH promedio de cada zona
b) Zona con mayor PH promedio

**Variables**

i: variable contadora que indica el número de repeticiones. Tipo: Entero
PHA: PH de una muestra de la zona A. Tipo: Real.
PHB: PH de una muestra de la zona B. Tipo: Real.
SumaPHA: suma de los PH del suelo de la zona A. Tipo: Real.
SumaPHB: suma de los PH del suelo de la zona B. Tipo: Real.
PHpromedioA: PH promedio de la zona A. Tipo: Real.
PHpromedioB: promedio de la zona B. Tipo: Real.

**Algoritmo**

0. Inicio
1. SumaPHA = 0
2. SumaPHB = 0
3. i = 1
4. Repetir mientras i <= 20
   4.1 Leer (PHA)
   4.2 SumaPHA = SumaPHA + PHA
   4.3 i = i + 1

```
    Fin RM
5.  i = 1
6.  Repetir mientras i <= 20
    6.1 Leer (PHB)
    6.2 SumaPHB = SumaPHB + PHB
    Fin RM
    6.3  i = i + 1
7.  PHpromedioA = SumaPHA / 20
8.  PHpromedioB = SumaPHB / 20
9.  Mostrar PH promedio de la zona A (PHpromedioA)
10. Mostrar PH promedio de la zona B (PHpromedioB)
11. Si PHpromedioA > PHpromedioB entonces
        11.1 Mostrar "La zona A tiene mayor PH promedio"
    De lo contrario
        11.2 Si PHpromedioB > PHpromedioA entonces
                11.2.1 Mostrar " La zona B tiene mayor PH promedio"
            De lo contrario
                11.2.2 Mostrar " Zonas A y B tienen igual PH promedio"
            Fin de si 11.2
    Fin de si 11
12. Fin
```

## 4.5 Ejercicios propuestos

1) Elaborar tres algoritmos (usando Repetir Para, Repetir Mientras y Repetir Hasta) para resolver cada uno de los siguientes problemas:

a) Obtener la suma de *n* números introducidos por el usuario.

b) Calcular la suma de los cuadrados de los 50 primeros números naturales (enteros positivos).

2) Diseñar un programa que calcule independientemente la suma de los números pares e impares comprendidos entre 1 y 50.

3) Escribir un algoritmo que calcule la suma:

$$1+\frac{1}{2^2}+\frac{1}{3^2}+\frac{1}{4^2}+\frac{1}{5^2}+.....+\frac{1}{n^2}$$ donde *n* es un número especificado por el usuario

4) Diseñar un programa que calcule el factorial de un número entero positivo. El factorial de un número *n* (se denota *n!)* se define como el producto de todos los números enteros positivo desde 1 hasta *n*. Ejemplo: 6! = 1 x 2 x 3 x 4 x 5 x 6.

5) Se tienen las calificaciones de n alumnos en la asignatura Informática, la cual consta de tres cursos: Programación, Excel y Word. Diseñar un programa que lea las notas de cada alumno en cada curso y calcule su nota en la asignatura (promedio de los 3 cursos). Igualmente, el programa debe calcular el número de estudiantes aplazados, número de estudiantes aprobados y el promedio general.

6) Elaborar un algoritmo que tenga como datos de entrada los salarios de 50 empleados de una empresa y determine: a) el número de empleados que ganan menos de 80000 Bs., b) el número de empleados que ganan entre 80000 y 120000 Bs., c) el número de empleados que ganan más de 120000 Bs., d) El valor de la nómina (suma de todos los salarios).

7) En un centro meteorológico se registran las precipitaciones mensuales caídas en tres zonas del país: Occidente, Centro y Oriente. Se pide diseñar un programa que reciba como datos de entrada las precipitaciones registradas en los 12 meses de un año para cada región y determine: a) precipitación anual en cada región, b) región con mayor precipitación anual, c) región con menor precipitación anual.

8) Elaborar un algoritmo que lea las precipitaciones diarias caídas en una ciudad en un determinado mes y determine: a) precipitación mensual, b) Mayor precipitación y el día en que se registró.

9) Se desea diseñar un programa que contabilice una cuenta de ahorros. Al comienzo el programa debe leer el nombre del titular de la cuenta y el saldo inicial. A continuación se permite hacer depósitos y retiros sucesivos, el usuario debe escribir una "d" si desea depositar o una "r" si desea retirar. Cuando es depósito se incrementa al saldo y cuando es retiro se resta, luego de cada operación debe mostrarse el saldo. El programa finalizará cuando ya no se desee hacer más movimientos. Al terminar, el programa debe mostrar el saldo final.

10) Escribir un algoritmo que calcule el incremento medio anual (IMA) de un árbol y que valide los datos de entrada. El incremento medio anual (IMA) se obtiene al dividir el tamaño de un árbol (diámetro, altura, volumen u otra medida de tamaño) entre su edad. En este caso se desea que el algoritmo calcule el incremento medio anual en altura, considerando que la altura es un valor positivo menor o igual a 50 metros, y la edad máxima es 100 años.

11) Los estudiantes de Ingeniería Forestal de una universidad evaluaron el estado fitosanitario de los árboles de una parcela. En la evaluación cada árbol fue clasificado en: bueno, regular, malo o muerto, de acuerdo a sus características y condiciones fitosanitarias. Diseñar un programa que lea el estado fitosanitario de cada árbol de la parcela y determine: el número y porcentaje de árboles que se ubican en cada grupo (bueno, malo, regular, muerto).

12) Diseñar un programa para procesar los datos de una plantación que tiene *n* árboles. De cada árbol se conoce: altura, diámetro a la altura de pecho y la posición que ocupa dentro de la parcela usando un sistemas de coordenadas X, Y. El programa debe leer los datos de cada árbol y calcular:

a) Diámetro promedio
b) Altura promedio
c) Mayor diámetro y las coordenadas X, Y del árbol con mayor diámetro
d) Mayor altura y las coordenadas X, Y del árbol con mayor altura

13) Diseñar un algoritmo que tenga como dato de entrada un número entero positivo y que muestre como salida la tabla de multiplicar correspondiente a ese número. Por ejemplo, si el usuario introduce el número 5, el algoritmo debe mostrar lo siguiente:

Tabla 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

14) Escribir un algoritmo que permita al usuario introducir los datos de un inventario forestal realizado en un área boscosa. Los datos corresponden a mediciones de árboles de tres especies (A, B, C). Se asume que la cantidad total de árboles es un dato de entrada, pero se desconoce de antemano cuántos árboles hay de cada especie.
De cada árbol se tiene su diámetro a la altura de pecho (en cm) y altura (en m). El algoritmo debe calcular número y porcentaje de árboles de cada especie que se encuentran en el área, así como también diámetro promedio, altura promedio y área basal promedio para cada especie.
Nota: en el ejercicio resuelto número 12 del capítulo 2 se explica cómo calcular el área basal de un árbol.

# BIBLIOGRAFÍA


Bettinger P, Boston K, Siry JP, Grebner DL (2009) *Forest Management and Planning*. Academic Press Elseiver. San Diego, EEUU. 331 pp.

Cairó O (2005) *Metodología de la programación*: *Algoritmos, diagramas de flujo y programas*. 3ra edición. Grupo editor Alfaomega. México DF. 476 pp.

Caro S (2003) *Lógica de programación y algoritmos*. Editorial Universidad de Boyacá. Boyacá, Colombia. 335 pp.

Cerrada J, Collado H (2010) *Fundamentos de programación*. Editorial Universitaria Ramón Areces. Madrid, España. 488 pp.

García JJ, Montoya F, Fernández JC, Majado MJ (2005) *Una introducción a la programación: Un enfoque algorítmico*. Ediciones Paraninfo. Madrid, España. 632 pp.

Hernández RJ, Rodas CA, Ospina CM, Urrego JB, Godoy JA, Aristizábal FA (2006) *Guías silviculturales para el manejo de especies forestales con miras a la producción de madera en la zona andina colombiana: El eucalipto*. Cenicafé, Colombia. [documento en línea]. Disponible desde internet en: www.cenicafe.org/es/publications/eucalipto.pdf.

Hurtado N, Laguía M, Silva EL (2010) *Introducción a la programación*. Servicio de publicaciones de la Universidad de Cádiz. España. 177 pp.

Joyanes L (2003) *Fundamentos de programación: Libro de problemas*. Segunda edición. McGraw Hill /Interamericana de España. Madrid, España. 433 pp.

Joyanes L (2008) *Fundamentos de programación: Algoritmos, estructuras de datos y objetos*. Cuarta edición. McGraw Hill /Interamericana de España. Madrid, España. 938 pp.

Juganaru M (2014) *Introducción a la programación*. Grupo editorial Patria. México DF. 288 pp.

Mata-Toledo R, Cushman P (2001) *Introducción a la programación con ejemplos en Visual Basic, C, C++ y Java*. McGraw Hill / Interamericana Editores. México DF. 331 pp.

Ospina CM, Hernández RJ, Rincón EA, Sánchez FA, Urrego JB, Rodas CA, Ramírez CA, Riaño NM (2011). *Guías silviculturales para el manejo de especies forestales con miras a la producción de madera en la zona andina colombiana: El pino pátula*. Cenicafé, Colombia. [documento en línea]. Disponible desde internet en: www.cenicafe.org/es/publications/pinus.pdf.

Romahn CF, Ramírez H (2010) *Dendrometría*. Universidad Autónoma de Chapingo, División de Ciencias Forestales. [documento en línea]. Disponible desde internet en: http://dicifo.chapingo.mx/licenciatura/publicaciones/dendrometria.pdf

Tamayo F (2012) *Fundamentos de la lógica de programación: Conceptos fundamentales, demostraciones y ejercicios*. Editorial Académica Española. Madrid, España. 132 pp.

Ugalde L (1981) *Conceptos básicos de Dasometría*. Centro Agronómico Tropical de Investigación y Enseñanza, CATIE. Manuscrito. 22 pp.