Capítulo 3

Primitivas algorítmicas

Se utilizan en la solución de un problema en forma de algoritmo para posteriormente ser codificado de acuerdo con la sintaxis de un lenguaje de programación. En los algoritmos se usan palabras y frases del lenguaje natural sujetas a unas determinadas reglas. Todo algoritmo consta básicamente de un conjunto de primitivas, las cuales se pueden clasificar en la siguiente forma:

- Primitivas de Inicio y Fin
- Primitivas de Asignación
- Primitivas de entrada / salida
- Primitivas condicionales no repetitivas
- Primitivas condicionales repetitivas

De igual forma, debe permitir la declaración de datos, tipos de datos, constantes, variables, expresiones, archivos y cualquier otro objeto que pueda ser utilizado en un programa de computador.

3.1 Estructuras de entrada/salida y asignación

Las estructuras básicas para la construcción de la lógica de control algorítmico que desempeñan la función básica de entrada de entrada y salida al computador. Estas se identifican por acciones de lectura (*Lea*) de datos que ingresan al computador sobre una o un conjunto de variables y por la acción de escritura (*Escriba*) de las informaciones derivadas del procesamiento aritmético/lógico del computador, las cuales son almacenadas en un conjunto de variables. Las informaciones derivadas también son acompañadas de literales encerrados entre comillas; por ejemplo: **Escriba**("Resultados de la función $f(x) =$ "), que sirven para identificar los comentarios de los resultados que se le presenta al usuario junto con las variables de salida.

La sintaxis de la estructura de lectura, que sirve para entrar datos al computador mediante un dispositivo de lectura, es:

Lea $V_1, V_2, \ldots, V_i, \ldots, V_n$, donde V_i son variables para $1 \leq i \leq n$.

Ejemplo 3.1 Lectura de datos

Lea las variables de base y altura para calcular el área de un rectángulo.

Análisis: El área (A) de un rectángulo se define matemáticamente por la expresión matemática $A = b \times h$, donde b es la base y h es la altura de la estructura geométricas el área. La expresión del área (A) no se puede calcular sino se tienen la base y la altura; entonces, la notación en algoritmos de la lectura de las variables necesarias para calcular es:

Lea *base*
Lea *altura*

Lo cual es equivalente a las instrucciones algorítmicas

Lea *base*, *altura*
Lea b, h (siendo las variables b igual a la base y h la altura).

La estructura básica de escritura, la cual permite presentar resultados de los datos procesados, los cuales al ser calculados por el computador se convierten en información de resultado para el usuario, especificada en su notación es:

Escribir $V_1, V_2, \ldots, V_n$

Ejemplo 3.2 Escritura de informaciones

Escriba la variable *area*, resultado del cálculo del valor del área del rectángulo.

Análisis: Recordando que la fórmula $A = h \times b$ sirve para calcular el área del rectángulo, entonces una vez el usuario ha leído los valores de la altura (h) y la base (b), y calculada la expresión matemática del área en la variable $Area\_Resultado \leftarrow b \times h$ (donde $\times$ es el operador de la multiplicación), el valor resultado del área se escribe de las siguientes formas:

Escribir *Area_Resultado* // o en su lugar, mediante la estructura algorítmica:

Escribir "El área del rectángulo es = ", *Area_Resultado*

En el último caso, el escriba está acompañado del literal encerrado entre comillas. Se aclara al lector que siempre que se utilice los caracteres seguidos // son comentarios que acompañan a las estructuras algorítmicas.

La entrada de datos al computador utilizando la estructura de lectura permite la construcción de resultados de información derivados de los procesos de cálculos que se hagan sobre los datos de entrada, y que deben ser asignados a nuevas variables de procesamiento, en este caso de salida. Generando de esta forma un sistema de Entrada (E), Proceso (P) y Salida (S), representado por E/P/S, de la máquina de procesamiento electrónica de datos (computador); sistema que hace que las salidas (S) sean datos procesados por computador, lo cual gráficamente se representa en la figura 3.1.

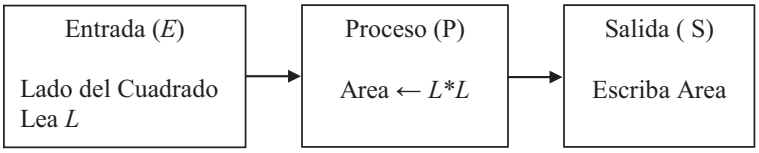


Figura 3.1 Sistema de proceso E/P/S

Ejemplo 3.3 Cálculo del área del cuadrado

Diseñe una gráfica representativa de las variables de entrada, el proceso y las salidas que permitan a partir de la lectura de la variable L o lado del cuadrado obtener como resultado el área del cuadrado.

Análisis
Entrada: Está representada por la variable L, dato de entrada o el lado del cuadrado.
Proceso: El proceso de cálculo del área es la multiplicación de la variable por sí misma $L \times L$, que da como resultado una nueva variable *Area*, representativa del área del cuadrado $Area = L \times L$ como fórmula matemática; lo cual es representado como estructura algorítmica en la forma $Area \leftarrow L \times L$; ello significa que a la variable *Area* se le asigna ($\leftarrow$) su lado al cuadrado.
Salida: El resultado de la salida es la variable *Area*, la cual contiene la información de cálculo generada por la operación aritmética del producto de de la variable L. Gráficamente, el sistema de procesamiento se representa por la figura 3.1.

La notación de la estructura básica de asignación, la cual envía el valor calculado de la expresión aritmética o lógica a la variable V, está representada por

$$V \leftarrow \text{expresión}$$

En la cual el valor de la expresión es asignada a la variable V, y la expresión está construida con base en operandos y operadores aritméticos o lógicos; en notación algorítmica, dentro del algoritmo se estila escribir únicamente la variable V a la cual

se le asigna ($\leftarrow$) la expresión calculada. Es de la mayor importancia resaltar que la notación algorítmica ($\leftarrow$) implica, con relación a la flecha denotada por Izquierda $\leftarrow$ Derecha, que la parte derecha de la estructura de asignación se calcula y se asigna a la variable de la parte izquierda de la estructura, según se muestra en los ejemplos siguientes:

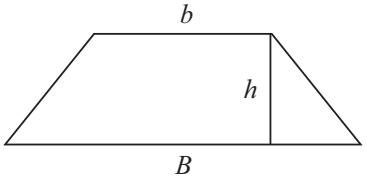
Ejemplo 3.4 Cálculo de expresiones matemáticas

Diseñe una expresión algorítmica de asignación para calcular el área de un trapecio.

Análisis: Matemáticamente, el área de un trapecio está dada por la base mayor (B) más la base menor (b) sobre dos, multiplicado por la altura (h), lo cual se representa por la fórmula matemática

$$A = \left(\frac{B+b}{2}\right) h$$

Expresión matemática representativa de la siguiente estructura geométrica:



La expresión en notación algorítmica del área del trapecio es $Area\_Trapecio \leftarrow ((Base\_Mayor + base\_menor)/2) \times altura$.

Las variables $Base\_Mayor$, $base\_menor$ y $altura$ deben haber sido leídas con la estructura algorítmica de entrada Lea.

Ejemplo 3.5 Construya una expresión en notación de algoritmos representativa del cálculo de la apotema de un cuadrado

Análisis: El cálculo aritmético de la apotema de un cuadrado requiere primero la constante numérica del divisor 2 que divide al lado (L) del cuadrado. En algoritmo denotada la expresión por

$Apotema\_Cuadrado \leftarrow L/2$; // La variable L debe ser leída, y el 2 es una constante.

Ejemplo 3.6 Diseñe una expresión algorítmica representativa del volumen de un cono

Análisis: El volumen de un cono analíticamente es igual al tercio de su altura multiplicada por el área del círculo de la base; lo que conduce a la expresión matemática

$$A = \frac{1}{3}h\pi r^2$$

Expresión que matemáticamente se representa en notación algorítmica por $Volumen\_Cono \leftarrow (3.1416 \times radio \times radio \times altura)/3$.

Se debe tener en cuenta que: *i*) 3.1416 es una constante que representa el valor de π; y el denominador es también una constante. *ii*) La altura (h) y el radio (r) son variables que se deben leer. *iii*) Note que la expresión matemática tiene unos paréntesis, que aseguran primero el cálculo del numerador ($3.1416 \times radio \times radio \times altura$), para posteriormente dividir la parte de expresión calculada entre paréntesis por el valor de 3, que representa el denominador.

Las estructuras básicas de control algorítmico de lectura, asignación y escritura permite la construcción de resultados derivados procesos de cálculo con operadores aritméticos o lógicos construidos en la parte de la expresión (parte derecha) para ser asignados a la variable V (parte izquierda del algoritmo).

Las estructuras básicas mencionadas con relación al sistema E/P/S generan un algoritmo cuyo diseño sirve para dar un resultado para el usuario; resultado que es la solución a un problema tratable computacionalmente utilizando la teoría de algoritmos.

La identidad del problema que va a ser solucionado en este libro denota un procedimiento (en este caso un procedimiento principal o **Proc**: (*main*)), el cual tiene un inicio y un final, y como procedimiento tiene las R_i reglas que conforman la lógica del control del algoritmo, lo que equivale al "... paradigma de programación procedimental o estructurado" (García, 2010, p. 5); con la característica de que el procedimiento en su proceso de cálculo debe ser finito en el tiempo; o sea, debe terminar **Fin_proc** para dar una respuesta al usuario.

La notación del procedimiento algorítmico (Proc) en su principio y fin es la siguiente:

Proc: Nombre del procedimiento // Equivalente al inicio
| **Lea** V_1, V_2 // Variables de entrada
| $V \leftarrow$ Expresión // Expresión lógica o matemática
| **Escriba** V // Variable de salida respuesta del proceso
Fin_proc // Equivale al fin del algoritmo

Ejemplo 3.7 Cálculo del área de un triángulo

Diseñar un algoritmo para calcular el área de un triángulo.

Análisis: El área de un triangulo está en función de la longitud de su base y su altura y está representada por la expresión aritmética $A = (base \times altura)/2$.

Proc: Área del triángulo
| **Entero:** $base$, $altura$, $area$
| **Lea** $base, altura$ // Lee los datos de entrada base y altura
| $area \leftarrow (base \times altura)/2$ // Calcula el área
| **Escriba** "El área del triángulo es ", $area$
| // Escribe los resultados del área
Fin_proc

Ejemplo 3.8 Cálculo de la longitud de una circunferencia de radio r

Diseñe un algoritmo para calcular la longitud de una circunferencia de radio r.

Análisis: La longitud de una circunferencia de radio r está dada por la ecuación

$$L = 2\pi r$$

El valor de π es una constante igual a 3.1416; y la variable del radio (r) debe ser leída.

Proc: Longitud de circunferencia
| **Real:** r, Pi, L
| **Lea** r // Lee el radio (r) de la circunferencia
| $Pi \leftarrow 3.1416$ // Se inicializa la constante π
| $L \leftarrow 2 \times Pi \times r$ // Calcula la longitud de la circunferencia
| **Escriba** "La longitud de la circunferencia es igual a", L
| // Escribe los resultados del cálculo de la longitud L de la circunferencia
Fin_proc

Ejemplo 3.9 Cálculo del volumen de un cono

Diseñar un algoritmo para calcular el volumen de un cono teniendo como datos de entrada la generatriz (g) del cono y su radio (r).

Análisis: La generatriz (g) del cono en función de la altura del cono (h) y de su radio (r) es igual a $g^2 = h^2 + r2$; luego $h = \sqrt{g^2 - r^2}$, derivado de lo cual, teniendo la altura, se puede calcular el volumen del cono, el cual es igual a $V = (\pi r^2 h)/3$.

Proc: Volumen del cono
Real: g, r, Pi, h, V
Lea g, r // Lee los datos de entrada, la generatriz
// y el radio
$Pi \leftarrow 3.1416$ // Se inicializa la constante π
$h \leftarrow \sqrt{g * g - r * r}$ // Cálculo de la altura (h) en función de la
// generatriz y el radio
$V \leftarrow (Pi \times r \times r \times h)/3$ // Calcula del volumen (V) del cono
Escriba "El volumen del cono es ", V
// Escribe el resultado volumen
Fin_proc

Ejemplo 3.10 Número total de conexiones entre los servidores de una red

Dado un conjunto de n servidores que apoyen una red de computadores, diseñe un algoritmo para calcular el número total de conexiones entre los servidores que soportan la red.

Análisis: Un servidor (*server*) es, a su vez, un computador que al estar inserto en una red de computadores suministra servicios de procesamiento de información a otros computadores que reciben el nombre de "clientes". El número de conexiones entre los servidores a través de la red, siendo el número de servidores n, está representado por la tabla 3.1.

Tabla 3.1 Generación del número de conexiones (C) derivadas de un conjunto de servidores (N)

Número de servidores (n)	Número total de conexiones (C)
0	0
1	0
2	1
3	3
4	6
⋮	⋮
N	$\binom{N}{2} = \frac{N(N-1)}{2}$

Proc: Total conexiones entre n servidores
Entero: n, c
Lea n // Lee n, el número de servidores de la red
$c \leftarrow n \times (n-1)/2$ // Calcula la expresión del número de conexiones
Escriba "Para una cantidad de servidores $n =$ ", n
// Escribe el número de servidores
Escriba "Existe un número total de conexiones $=$ ", c
// Número total de conexiones
Fin_proc

Ejemplo 3.11 Cálculo del número de metros cúbicos contaminados por una pila

Si es posible que una sola pila contamine 175 000 litros de agua, ¿cuántos metros cúbicos de agua serán contaminados por la población de una ciudad de n millones de habitantes en un año si cada persona utiliza 2 pilas al semestre?

Análisis: En los dos semestres del año cada persona utilizará 4 pilas. Luego, el número de litros que contaminan los n millones de personas es de $4 \times n \times 175000$ (n en millones). Por lo tanto, el número de metros cúbicos contaminados (Mcc) es igual a $4 \times 175 \times n$, porque un metro cúbico tiene 1000 litros.

```
Proc: Metros cúbicos contaminados
  Entero: n, Mcc
  Lea n  // Lee en millones n, el número de
  // habitantes de la ciudad
  Mcc ← 4 × 175 × n  // Calcula la expresión del número de m^3
  // contaminados
  Escriba "Una ciudad con un número de = ", n, " millones de habitantes"
  // Escribe el resultado de la contaminación
  Escriba "Contamina = ", Mcc, " millones de metros cúbicos de H2O"
  // Si n = 1 millón, 700 millones de metros cúbicos de agua
  // son contaminados
Fin_proc
```

Ejemplo 3.12 Cálculo de la resistencia total de un circuito en paralelo

Suponga que un circuito eléctrico tiene dos resistencias en paralelo, R_1 y R_2, diseñe un algoritmo para calcular la resistencia total del circuito.

Análisis: La resistencia total de un circuito en paralelo de n resistencias está representada por la ecuación

$$\frac{1}{R_t} = \frac{1}{R_1} + \frac{1}{R_2} + \cdots + \frac{1}{R_n}.$$

Luego, la resistencia total del circuito está dada por la fórmula

$$R_t = \frac{1}{\frac{1}{R_1} + \frac{1}{R_2} + \cdots + \frac{1}{R_n}}.$$

Por lo tanto, cuando se tienen dos resistores conectados en paralelo, la resistencia total es

$$R_t = \frac{1}{\frac{1}{R_1} + \frac{1}{R_2}}$$

Proc: Cálculo de la resistencia total de dos resistores conectados en paralelo
Real: $R1$, $R2$, Rt
Lea $R1, R2$ // Lee los resistores R1 y R2 del circuito
// Calcula la expresión del número de conexiones
$Rt \leftarrow 1/((1/R1) + (1/R2))$
Escriba "Para un circuito con dos resistores = ", $R1$, ", ", $R2$
// Escribe el valor de las resistencia conectadas en paralelo R1 y R2
Escriba "conectado en paralelo, la resistencia total Rt es igual a = ", Rt
// Escribe el cálculo de la resistencia total
Fin_proc

Ejemplo 3.13 Área del trapecio circular

Dado el radio mayor (R) y el radio menor (r) de un trapecio circular de amplitud en grados igual a g grados, diseñe un algoritmo que calcule el área del trapecio circular.

Análisis: La ecuación representativa del área (A) del trapecio circular es

$$A = \frac{\pi(R^2 - r^2)g}{360^\circ}$$

Proc: Cálculo del área del trapecio circular
// Lee primero el radio mayor (Rma), el radio menor (Rme) y
// la amplitud en grado (g)
// Se debe tener en cuenta diferenciar las variables
// de lecturas de los dos radios
Entero: Rma, Rme, g, Pi, A
Lea Rma, Rme, g
$Pi \leftarrow 3.1416$ // Se inicializa la constante Pi
$A \leftarrow (Pi \times (Rma \times Rma - Rme \times Rme) \times g)/360$
// Calcula el área del trapecio circular
Escriba "Para un trapecio de radio mayor y de radio menor
iguales a = ", Rma, ", ", Rme
Escriba "Si el trapecio tiene una amplitud en grados de = ", g
Escriba "El área del trapecio circular es igual a = ", A
// Escribe el área del trapecio circular
Fin_proc

Ejemplo 3.14 Cálculo del valor presente

Suponga que un inversionista invierte un capital C a una tasa de interés t durante n años. Si el interés de inversión es del 7 %, diseñe un algoritmo que calcule el valor presente del capital invertido al final del año K de la inversión.

Análisis: Un capital C que se invierte a una tasa de interés del t a n años al término de los años mencionados recibirá un capital que traído a valor presente (VP) es igual a

$$VP = \frac{C}{(1+t)^n}$$

Proc: Valor presente de una inversión
| **Entero:** C, K
| **Real:** t, VP
| **Lea** C, K // Se lee el capital C y el número de años K
| $t \leftarrow 7/100$ // Se inicializa la constante del interés
| // de inversión
| $VP \leftarrow C/(1+t) * *K$ // Se calcula el valor presente
| // de la inversión en los K años
| // La notación ** es el símbolo del operador
| // exponencial
| **Escriba** "El valor presente de la inversión es = ", VP
Fin_proc

Ejemplo 3.15 Método de ciframiento del Caesar

Suponga el sistema de números digitales [0 al 9]; en un sistema de seguridad utilizan el método de cifrado llamado *Caesar Method* con un desplazamiento de n ($1 \leq n \leq 9$); si se captura un dígito fuente igual a d, diseñe una algoritmo para cifrar el dígito.

Análisis: Los número digitales son 0 1 2 3 4 5 6 7 8 9; si cada uno de ellos se cifra con el método del César con desplazamiento de 1 unidad, los resultados son 1 2 3 4 5 6 7 8 9 10; si a estos números se le aplica el módulo 10, el resultado de la función da un texto cifrado de números igual a 1 2 3 4 5 6 7 8 9 0; luego, si el código es 72815234 si se cifran sus dígitos, con la función de cifrado anteriormente expuesta, da como resultado 83926345; por lo tanto, la función de Encriptación (E) que recibe un dígito d y se desplaza n posiciones de cifrado está representada por la ecuación

$$E_n(d) = (d+n) \,\mathbf{mod}\, 10$$

Proc: *Caesar Method* con desplazamiento n
| **Entero:** d, n, c
| **Lea** d, n // Se lee el dígito que se va a cifrar d y el
| // desplazamiento aplicado n
| $c \leftarrow (d+n) \,\mathbf{mod}\, 10$ // cifrado del dígito con desplazamiento
| // n sobre la variable c
| // **mod** es la función módulo
1

```
1
| Escriba "El dígito de entrada = ", d
| Escriba "Cifrado con el método del César con un desplazamiento = ", d
| Escriba "El dígito cifrado es igual a = ", c
Fin_proc
```

3.2 Concepto de primitivas básicas para la construcción de algoritmos

La lógica de control de un algoritmo se basa en un conjunto de estructuras lógicas que se ejecutan secuencialmente desde la primera estructura hasta la última con una cantidad de esfuerzo o trabajo de y en una cantidad finita de tiempo.

Luego las estructuras de entrada/salida y asignación son constructos lógicos primarios, que permiten la elaboración de lógicas de programación más complejas. Los constructos lógicos primarios reciben también el nombre de primitivas lógica (p_i), no porque sean antiguas, sino porque son la base para la construcción de estructuras lógicas más complejas.

Definición de primitiva lógica (p_i): Una primitiva lógica para la construcción de algoritmos es una instrucción lógica que permite la ejecución de una o varias acciones en el computador; acciones que, a su vez, pueden ser otras primitivas lógicas; primitivas que tienen la característica de ordenar acciones al computador, y cuando dichas acciones son codificadas en un lenguaje de programación reciben el nombre de instrucciones de un lenguaje de programación. Como instrucciones de un lenguaje de programación permiten ser ejecutadas en un computador y consumen tanto un espacio de memoria como una cantidad finita de tiempo para su ejecución.

Con base en lo anterior, las estructuras lógicas de entrada/salida y asignación reciben también el nombre de primitiva de entrada/salida y proceso.

Ejemplo 3.16 Residuo y cociente de división entera

División entera. Dados dos números enteros numerador y denominador, diseñe un algoritmo que dé como resultado el residuo y el cociente de su división entera.

Análisis: Suponga dos números enteros que se leen en las variables numerador = 7 y denominador = 2. El residuo de la división entera 7/2 es igual a 1 (residuo de división entera que en el algoritmo se representa por el operador %); y el cociente de la división entera 7/2 es igual a 3.

La tabla 3.2 presenta la relación directa entre las primitivas lógicas de control algorítmico para el caso de la División entera con el lenguaje de programación C.

Tabla 3.2 Relación entre las primitivas lógicas algorítmicas con el lenguaje de programación C

Algoritmo	Lenguaje de programación C
Proc: División entera	`void main() // División entera`
	`{`
	`int numerador,denominador,residuo,cociente; // Variables enteras int`
(p_1) Lea numerador	`cout<<"Digite el valor del numerador : \n";` `cin>>numerador;`
(p_2) Lea denominador	`cout<<"Digite el valor del denominador : \n";` `cin>>denominador;`
(p_3) residuo ← numerador**%**denominador	`residuo=numerador%denominador;`
(p_4) cociente ← numerador/denominador	`cociente=numerador/denominador;`
(p_5) Escriba “Residuo =”residuo	`cout<<"\t Resultado del residuo = \t"<<residuo;` `cout<<"\n";` `cout<<"\n";`
(p_6) Escriba “Cociente =”cociente	`cout<<"\t Resultado del cociente = \t"<<cociente;` `cout<<"\n";` `cout<<"\n";`
Fin_proc.	`getch();` `}`

Las primitivas lógicas descritas en la tabla mencionada son:

- Primitivas lógicas de entrada: Son las primitivas de lectura o entrada de datos representas por Lea numerador (p_1) y Lea denominador (p_2).
- Primitivas lógicas de asignación: Son las estructuras lógicas primitivas representadas por las asignaciones de cálculo del residuo (p_3) y del cociente (p_4).
- Primitiva lógica de escritura: Son las estructuras de control representadas por las variables que contienen los resultados de las operaciones de asignación: Escriba "Residuo = " residuo (p_5) y Escriba "Cociente = " cociente (p_6).

3.3 Estructura lógica condicional simple

La lógica de programación se basa en estructuras simples, tales como las estructuras de entrada, asignación y salida (desarrolladas en la sección anterior), a las cuales se les agregan estructuras lógicas más elaboradas; una de las cuales es la estructura condicional.

La "condición" se refiere a una decisión que se debe tomar en la lógica de control del algoritmo. La "decisión" implica la solución a una pregunta que se estructura en la condición. Si la resolución a la pregunta es verdadera, entonces se ejecutan las estructuras lógicas o primitivas lógicas de control que se escriben en la condición verdadera del algoritmo.

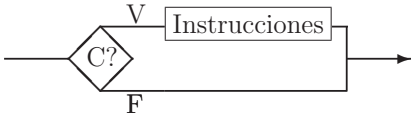


Figura 3.2 Esquema condicional simple

La estructura condicional simple en función de las primitivas lógicas se define como:

Condicional Simple. Sean $p_1, p_2, \ldots, p_n$ un conjunto de primitivas, la condicional simple está representada por

Si(condición)**entonces**
| p_1
| p_2
| $\vdots$
| p_n
Fin_si

Esta condicional permite la ejecución de las primitivas: $p_1, p_2, \ldots, p_n$ si la condición es verdadera o se cumple.

Los ejemplos de estructuras lógicas de programación que se presentan en esta sección corresponden a la primitiva condicional simple; pero se aclara al lector que todos los algoritmos ejemplificados para este caso tienen su contraparte que no es verdadera; contraparte de primitivas por la opción de falso que se desarrollarán con la estructura lógica de control algorítmico identificada como la condicional compuesta.

Ejemplo 3.17 Chequeo de un número par

Dado un número n, el cual es leído, diseñe un algoritmo para chequear si el número es par; y en el caso de que sea par, que imprima un mensaje que diga "número par" e imprima el número que es par.

Análisis: Suponga un número dado $n = 8$; entonces, si a 8 se divide entre dos utilizando la división entera 8/2, la respuesta es 4; ahora, si al resultado de la división entera, o sea, 4, lo multiplicamos por el número dos $(4 \times 2 = 8)$, por lo tanto se construye la condición de que si $(\lfloor n/2 \rfloor \times 2 = n)$ utilizando la notación de $\lfloor x \rfloor$ para representar la parte entera de x, es decir, el mayor entero menor o igual que x.

Proc: Chequeo de número par
| **Entero:** n
| **Lea** n // Leer el número en la variable entera n, para decidir si es par
| // Pregunta condicional que chequea si el número dado es par
| **Si**$((\lfloor n/2 \rfloor \times 2 = n)$**Entonces** // Inicio de la estructura condicional Si
| | **Escriba** "El número leído es par = ", n
| | // Primitiva que se ejecuta si la condición es verdadera
| **F.Si** // Cierre del condicional Si
Fin_proc

Ejemplo 3.18 Conversión de grados centígrados a grados Fahrenheit y a grados Kelvin

Diseñe un algoritmo que lea la temperatura en grados centígrados (°C) y la convierta a grados Farenheit (°F) y a grados Kelvin (°K), siempre y cuando la temperatura dada en grados centígrados sea positiva.

Análisis: La conversión de los grados centígrados a grados Farenheit y a grados Kelvin se realiza solo cuando el valor de los grados centígrados es mayor que cero; de otro modo no se hace la conversión. Las ecuaciones de conversión son las siguientes:

De grados Farenheit a grados centígrados: $°\mathrm{F} = \frac{9}{5}(°\mathrm{C} + 32)$.

De grados Kelvin a grados centígrados: $°\mathrm{K} = °\mathrm{C} + 273.15$.

Proc: Conversor de grados centígrados positivos a Farenheit y Kelvin
| **Real:** C, F, K
| **Lea** C // Leer la temperatura en grados centígrados (°C)
| // Chequea si la temperatura leída es positiva
| **Si**$(C > 0)$**Entonces**
| | $F \leftarrow (9/5) \times (C + 32)$
| | $K \leftarrow C + 273.15$
| | **Escriba** "La temperatura en grados Centígrados = ", C
| | **Escriba** "Es igual a la temperatura en grados Farenheit = ", F
| | **Escriba** "Es equivalente a la temperatura en grados Kelvin = ", K
| **F.Si**
Fin_proc

Ejemplo 3.19 Conversión horaria de 24 horas a 12 horas

Dados dos enteros: horas (h) y minutos (m), representativos del tiempo en horas y minutos dado por un reloj de 24 horas, convierta el tiempo dado por el reloj de 24 horas a un reloj que obtenga el tiempo cada 12 horas.

Análisis: En primer lugar se debe verificar si los datos en horas (h) y minutos (m) dados por el usuario son correctos; luego el rango horario debe estar en el intervalo cerrado [0;24] horas, y para el caso de los minutos, dentro del intervalo [0;60] minutos. En segundo lugar se debe hacer la conversión.

Suponga que los datos de entrada son 17 horas y 57 minutos. Entonces, el equivalente en tiempo horario de 12 horas se calcula de la siguiente forma: se realiza la división entera entre el número de horas dado entre 13; luego $17/13 = 1$ en su cociente; adicionalmente, al cociente calculado se le suma el residuo del módulo 13 aplicado sobre el número de horas recibidas; entonces 17 **mod** 13 (en lenguaje C $h\,\%13$) es igual 4. Por lo tanto, el tiempo en horas convertido a 12 horas es igual $5 = 17\,\%13 + 17/13$ (siendo esta división entera) $= 4 + 1 = 5$, lo cual es válido en el sentido de que las 17 horas corresponde a las 5 horas.

En el caso de que el tiempo leído en minutos sea igual a 60 minutos, el algoritmo diseñado considera 1 hora, y se la suma al tiempo horario, colocando los minutos a cero.

Proc: Conversor horario de 24 horas a 12 horas
| **Entero:** h, m, $Tiempo12$
| **Lea** h, m // Leer las horas y los minutos en el reloj de 24 horas
| **Si**$(((h \geq 0)$ y $(h \leq 24))$**Entonces** // Condicional compuesta
| | // con el operador y
| | $Tiempo12 \leftarrow (h/13) + (h$ **mod** $13)$ // División entera, **mod** = %
1 2

```
1 2
|  |  Si((m ≥ 0) y (m ≤ 60))Entonces
|  |  |  Si(m = 60)Entonces
|  |  |  |  Tiempo12 ← Tiempo12 + 1
|  |  |  |  m ← 0
|  |  |  F.Si
|  |  F.Si
|  |
|  |  Escriba "El tiempo dado de ", h, " horas y ", m, " minutos"
|  |  Escriba "Equivale a = ", Tiempo12, " horas y ", m, " minutos"
|  F.Si
Fin_proc
```

Ejemplo 3.20 Ubicación del semestre de un estudiante en función del número de créditos

Un programa académico de pregrado tiene un total de 150 créditos; suponga que por semestre el alumno puede cursar 15 créditos; si C es el número de créditos totales aprobados por el estudiante, diseñe un algoritmo que calcule el semestre de ubicación del estudiante, validando que el número de créditos leídos esté en el intervalo cerrado [1;150] créditos.

Análisis: Se debe primeramente verificar si el número de créditos aprobados por el alumno (C) está en el rango válido del intervalo cerrado [1;150] créditos. Luego se deben dividir los créditos totales aprobados por el alumno (C) entre el número de créditos que puede cursar por semestre, que son 15, y el cociente de la división entera es el semestre en el cual está ubicado el alumno.

Proc: Ubicación del semestre de un alumno en función del número de créditos aprobados

```
|  Entero: C, Se
|  Lea C  // Leer el número de créditos aprobados por el alumno.
|  Si((C ≥ 1) y (C ≤ 150))Entonces  // Validar créditos aprobados
|  |  Se ← C/15  // División entera que calcula el semestre
|  |  Escriba "El alumno con ", C, " créditos aprobados"
|  |  Escriba "Está ubicado en el semestre ", Se
|  F.Si
Fin_proc
```

Ejemplo 3.21 Raíces reales de una ecuación cuadrática

Diseñe un algoritmo para calcular las raíces reales de una ecuación cuadrática.

Análisis: La ecuación cuadrática está representada por $ax^2 + bx + c = 0$, y su solución es la expresión $x = \dfrac{-b \pm \sqrt{b^2 - 4ac}}{2a}$. El discriminante de la solución de la ecuación cuadrática es $b^2 - 4ac$; luego, si el discriminante es positivo, las raíces son reales; de otro modo son imaginarias.

Proc: Cálculo de las raíces reales de una ecuación cuadrática
  Real: a, b, c, D, $R1$, $R2$
  Lea a, b, c // Leer los coeficientes de la ecuación cuadrática
  $D \leftarrow b * b - 4 * a * c$ // Cálculo del discriminante
  Si$(D \geq 0)$**Entonces** // Validar si el discriminante es positivo
    $R1 \leftarrow (-b + \sqrt{D})/(2 * a)$ // Calcular la primera raíz de la ecuación
    $R2 \leftarrow (-b - \sqrt{D})/(2 * a)$ // Calcular la segunda raíz de la ecuación
    Escriba "Las raíces de ecuación cuadrática con los coeficientes ",
    Escriba a, ", ", b, " y ", c, " son las siguientes: ", $R1$, " y ", $R2$
  F.Si
  // Se aclara al lector que hay una contraparte cuando $D < 0$,
  // para la cual se requiere la condicional compuesta, la cual
  // se desarrolla en la siguiente sección
Fin_proc

3.4 Estructura lógica condicional compuesta

Teniendo como bases iniciales para la construcción de algoritmos las estructuras básicas de control de lectura, asignación, escritura y condicional simple, la estructura condicional compuesta amplía las posibilidades de la lógica de control algorítmica.

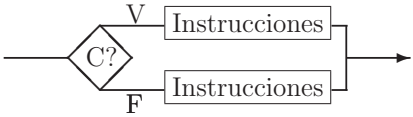


Figura 3.3 Esquema Condicional Compuesto

La decisión de control en la condicional simple permitió la ejecución de las primitivas cuando se cumple la pregunta en su condición de verdad, y la condicional compuesta permite la ejecución de estructuras de control que se diferencian en el cumplimiento de la condición de la siguiente forma: cuando la condición es verdadera, solo se ejecutan las primitivas de control diseñadas en la parte verdadera de la condición (p_1, p_2, p_3, …, p_i, …, p_n); y cuando la condición es falsa se ejecutan solo las estructuras de control diseñadas en la parte falsa de cumplimiento de la condición (q_1, q_2, q_3, …, q_i, …, q_n), generándose, por lo tanto, dos conjuntos disyuntos de primitivas tales que (p_1, p_2, …, p_n) $\cap$ (q_1, q_2, …, q_n) $= \emptyset$ ($\emptyset$, símbolo de vacío). Lo cual implica que el flujo de control del algoritmo si se ejecuta en el cumplimiento de

verdad de la condición, no se ejecuta en el cumplimiento de la condición falsa; caso que implica que los dos grupos de primitivas (p, q) son disyuntas en la ejecución del algoritmo.

La notación algorítmica de la estructura lógica condicional compuesta es: Dados dos conjuntos disyuntos de primitivas p_1, p_2, ..., p_n y q_1, q_2, ..., q_n, la condicional compuesta representada por

Si(condiciones)**Entonces**
| $p_1, p_2, \ldots, p_n$
Sino
| $q_1, q_2, \ldots, q_n$
F.Si

permite la ejecución del conjunto de primitivas p_1, p_2, ..., p_n, si la condición es verdadera, o la ejecución del conjunto q_1, q_2, ..., q_n, en el caso de que la condición sea evaluada como falsa.

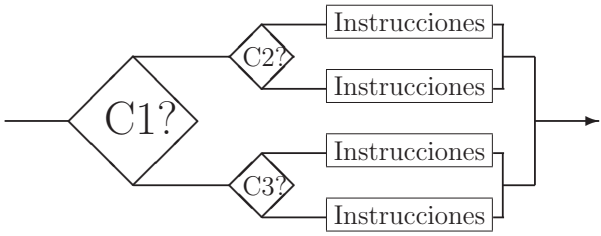


Figura 3.4 Estructuras condicionales compuestas (anidados)

Ejemplo 3.22 Chequeo de un número par o impar

Dado un número n, diseñe un algoritmo para clasificar el número como par o impar.

Análisis: El algoritmo fue analizado y diseñado en el ejemplo 3.17; por lo tanto, se completará correspondiente a la condición falsa del algoritmo cuando el número es impar.

Proc: Chequeo de número par o impar
| **Entero:** n
| **Lea** n // Leer el número en la variable entera n, para decidir si es par
| **Si**$(\lfloor n/2 \rfloor * 2 = n)$**Entonces**
| | **Escriba** "El número leído es par"
| | // Primitiva que se ejecuta si la condición es verdadera
1 2

```
1 2
| Sino
| | Escriba "El número es impar"
| | // Primitiva que se ejecuta si la condición es falsa
| F.Si
Fin_proc
```

Ejemplo 3.23 Cálculo del valor absoluto de un número n

Diseñe un algoritmo para calcular el valor absoluto de un número leído n.

Análisis: Si el número es positivo, el valor absoluto del número es el mismo; por otra parte, si el número es negativo, su valor absoluto es el número multiplicado por -1; así $|-5| = (-1) * (-5) = 5$.

```
Proc: Cálculo del valor absoluto de n
| Entero: n, a
| Lea n
| // Pregunta condicional que chequea si el número dado es mayor
| // o menor que cero
| Si(n < 0)Entonces
| | a ← −1 ∗ n
| | // Primitiva que se ejecutan si la condición es verdadera
| Sino
| | a ← n
| | // Primitiva que se ejecutan si la condición es falsa
| F.Si
| Escriba "Para el número leído n = ", n
| Escriba "Su valor absoluto es ", a
Fin_proc
```

Ejemplo 3.24 Clasificación de un número dado en la recta real

Dado un número leído X, el cual está ubicado en la recta real, diseñe un algoritmo para clasificar si el número es positivo, negativo o igual a cero.

Análisis: La recta real en el intervalo $(-\infty, +\infty)$ clasifica los números en tres subintervalos: el negativo $(-\infty, 0)$, el positivo $(0, +\infty)$ y el igual a cero (0).

```
Proc: Clasificación del número X en la recta real
| Real: X
| Lea X  // Leer el número que se va a clasificar en la recta real.
| // Pregunta condicionales de clasificación del número en la recta real
1
```

```
1
| Si(X < 0)Entonces // Clasifica el número como negativo
| | Escriba "El número es negativo ", X
| Sino Si(X = 0)
| | // Clasifica el número como igual a cero
| | Escriba "El número es igual a cero ", X
| Sino
| | // Clasifica el número como positivo
| | Escriba "El número es positivo", X
| F.Si
Fin_proc
```

Ejemplo 3.25 Cálculo de una función $F(x)$ por intervalos

Sea $F : [0, +\infty) \longrightarrow \mathbb{R}$ tal que

$$F(x) = \begin{cases} x^2 & \text{si } 0 \leq x \leq 2 \\ -x^2 + 4 & \text{si } 2 < x \leq 4 \\ x + 4 & \text{si } x > 4 \end{cases}$$

Diseñe un algoritmo para calcular el valor de la función $F(x)$ para un valor de entrada x, el cual es leído.

Análisis: La función $F(x)$ genera tres estados de cálculo: *i*) Si el valor leído x pertenece al intervalo cerrado $[0, 2]$, entonces el valor de la función es igual a $F(x) = x^2$. *ii*) Si el valor de la función está en el intervalo $(2, 4]$, entonces el valor de la función es $F(x) = -x^2 + 4$. Finalmente, en el caso de que el valor de entrada esté en un rango de valores mayores que 4, entonces el valor de la función es $F(x) = x + 4$.

```
Proc: Cálculo de la función matemática F(x) por intervalos
| Real: x, FX
| Lea x // Leer el valor de entrada de la función F(x)
| // Pregunta condicionales que clasifican F(x) en los intervalos definidos
| Si((x ≥ 0) y (x ≤ 2))Entonces
| | // Calcula F(x) en el intervalo [0, 2]
| | FX ← x * x
| | Escriba "El valor de la función F(x) es igual a = ", FX
| Sino Si((x > 2) y (x ≤ 4))
| | // Calcula F(x) en el intervalo (2, 4]
| | FX ← −x * x + 4
| | Escriba "El valor de la función F(x) es igual a = ", FX
| Sino Si((x > 4))
| | // Calcula F(x) en el intervalo (4, +∞)
| | FX ← x + 4
1 2
```

```
1 2
| | Escriba "El valor de la función es = ", FX
| F.Si
Fin_proc
```

Ejemplo 3.26 Clasificación de variables en tipos binario, intervalo y proporción

Dadas dos variables numéricas X y Y, las cuales son leídas, diseñe un algoritmo para clasificar las variables en los tipos binario, intervalo y proporción.

Análisis: Las variables nominales o binarias son aquellas que establecen categorías de clasificación en la escala de medición binaria $(X = 1)$ es diferente de $(Y = 0)$, implican las variables binarias el estado de switch de control, en el caso de su estado activo en uno dejando pasar la corriente, o no activo en cero impidiendo el paso de la corriente. Las variables intervalo, además de contener relaciones de orden $X > Y$ o $X < Y$, permiten establecer la distancia entre los puntos X y Y. Es el caso en el cual la edad de María es de 8 años y la de Pedro es de 19; luego, se establece una relación ordinal en la cual se puede concluir que María es menor que Pedro, o lo que es equivalente, $8 < 19$; en el caso de un orden ascendente de orden de edades, o en un sentido equivalente, Pedro es mayor en edad que María, justificado por el hecho de que $19 > 8$ en una relación de orden de edades descendente.

Finalmente, las variables de tipo proporción permiten establecer la razón entre las variables X y Y; en tal sentido, $X = k * Y$ o $Y = k * X$, donde k es la constante de proporcionalidad entre las dos variables. El caso se presenta en el sistema solar, si se tiene en cuenta que el número de lunas del Marte (M) son 2 y el número de lunas del planeta Neptuno (N) son 8; entonces se puede concluir que $N/M = 8/2 = 4$, o lo que es equivalente, $N = 4 * M$, o lo que es equivalente, $8 = 4 * 2$.

El algoritmo que se va a diseñar se construye en función de la estructura lógica de control condicional simple, y considera que variables del tipo intervalo también pueden ser clasificadas como variables de tipo proporción. Luego, si $X = 8$ y $Y = 2$, en la clasificación de por tipo intervalo la distancia de Y a X es $(X - Y = 8 - 2 = 6)$; pero en las clasificación tipo razón o proporción, si se calcula la constante de proporcionalidad $k = X/Y = 8/2 = 4$, se puede conceptuar que las variable X es k veces la variable Y; o lo que es equivalente, $X = 8 = 4 * 2$, con $Y = 2$ y constante de proporcionalidad $k = 4$.

```
Proc: Clasificador de variables binarias, intervalo y proporción
| Entero: X, Y, D, K
| Lea X, Y  // Leer el valor de las variables que van a ser clasificadas
| // Estructura de primitivas condicionales que clasifican las variables
1
```

```
1
| Si((X = 0 o X = 1) y (Y = 1 o Y = 0))Entonces
| | Escriba "Las variables pertenecen a la escala binaria"
| Sino Si((X ≠ 0 o X ≠ 1) y (Y ≠ 1 o Y ≠ 0))
| | // Clasificador intervalo
| | Si(X > Y)Entonces
| | | D ← X − Y
| | | Escriba "Las variables son del tipo intervalo"
| | | Escriba "La distancia de Y a X es = ", D
| | Sino Si(X < Y)
| | | D ← Y − X
| | | Escriba "Las variables son del tipo intervalo"
| | | Escriba "La distancia entre de X a Y es = ", D
| | F.Si
| Sino
| | // Clasificador proporción
| | Si(X > Y)Entonces
| | | K ← X/Y
| | | Escriba "Las variables son del tipo proporción"
| | | Escriba "La variable X es ", K, " veces mayor que la variable Y"
| | Sino Si(Y > X)
| | | K ← Y/X
| | | Escriba "Las variables son del tipo proporción"
| | | Escriba "La variable Y es ", K, " veces mayor que la variable X"
| | F.Si
| F.Si
Fin_proc
```

Ejemplo 3.27 Intersecto de dos funciones

Dadas dos funciones $F1(x) = x$ y $F2(x) = x^2$. Suponga que se lee un valor X, diseñe un algoritmo para identificar si las dos funciones se intersectan en el valor leído.

Análisis: Si las dos funciones se intersectan en el argumento de entrada X, entonces significa que $F1(x) = F2(x)$. Suponga que $X = 1$, entonces $F1(x) = 1$ y $F2(x) = 1 * 1$; por lo tanto, en el punto 1 las dos funciones son iguales.

```
Proc: Intersecto de funciones en el punto X
| Real: x, F1, F2
| Lea x // Leer el valor para el cual se va a calcular el intersecto
| F1 ← x // Cálculo del valor de la primera función
| F2 ← x * x // Cálculo del valor de segunda función
1
```

```
1
| Si(F1 = F2)Entonces // Condicional que indica el intersecto
| | Escriba "La función F1(x) = ", x
| | Escriba "La función F2(x) = ", x * x
| | Escriba "Se intersectan en la ordenada Y = ", F2
| | Escriba "Junto con la abscisa X = ", x
| | // Primitivas que se ejecutan si la condición es verdadera
| Sino
| | Escriba "Las funciones F1(x) = x y F2(x) = x * x"
| | Escriba "NO se intersentan en la abscisa X = ", x
| | Escriba "Luego, el valor de las dos funciones para el punto X"
| | Escriba "es diferente"
| | // Primitivas que se ejecutan si la condición es falsa
| F.Si
Fin_proc
```

Ejemplo 3.28 Equivalencia de una escala cuantitativa a cualitativa

Suponga que la escala de calificaciones de una universidad se muestra en la tabla 3.3.

Tabla 3.3 Escala de calificaciones de una universidad

Numérica	Cualitativa
$[4.6, 5.0]$	Excelente
$[4.1, 4.5]$	Muy bueno
$[3.6, 4.0]$	Bueno
$[3.3, 3.5]$	Aceptable
$[3.0, 3.2]$	Aprobado
$[2.6, 2.9]$	Deficiente
$[2.1, 2.5]$	Malo
$[0.0, 2.0]$	Por mejorar y considerablemente

Diseñe un algoritmo que escriba la equivalencia cualitativa de una nota numérica leída n, considerando que el alumno aprobó la calificación.

Análisis: Como el estudiante aprobó la calificación, entonces la nota debe ser superior a 3.0. Una vez se verifique que la nota es superior a 3.0, entonces se clasifica en el respectivo intervalo, utilizando estructuras lógicas condicionales compuestas.

```
Proc: Conversor de escalas aprobatorias de calificaciones
| Real: n
| Lea n
| // Condicionales que cualifican la nota del alumno
| Si(n ≥ 3.0)Entonces // La nota es válida como aprobada
| |
1 2
```

```
1 2
| | Si((n ≥ 4.6) o (n ≤ 5.0))Entonces
| | | Escriba "Estudiante Excelente"
| | Sino
| | | Si((n ≥ 4.1) o (n ≤ 4.5))Entonces
| | | | Escriba "El alumno es Muy Bueno"
| | | Sino
| | | | Si((n ≥ 3.6) o (n ≤ 4.0))Entonces
| | | | | Escriba "El estudiante es Bueno"
| | | | Sino
| | | | | Si((n ≥ 3.3) o (n ≤ 3.5))Entonces
| | | | | | Escriba "El alumno registró un desempeño Aceptable"
| | | | | Sino
| | | | | | Si((n ≥ 3.0) o (n ≤ 3.2))Entonces
| | | | | | | Escriba "Alumno Aprobado"
| | | | | | F.Si
| | | | | F.Si
| | | | F.Si
| | | F.Si
| | F.Si
| | Sino
| | | Escriba "El alumno registra una calificación no aprobatoria"
| | F.Si
Fin_proc
```

Ejemplo 3.29 Ubicación de coordenadas en los cuadrantes del plano cartesiano

Dados dos puntos X y Y, que son las coordenadas del plano cartesiano, haga un algoritmo para clasificar el cuadrante del plano donde está ubicada la coordenada (X, Y).

Análisis: Los ejes del plano cartesiano clasifican los puntos X y Y en cuatro cuadrantes: 1) Si $X > 0$ y $Y > 0$, entonces la coordenada está en el primer cuadrante; 2) Si $X < 0$ y $Y > 0$, la coordenada pertenece al segundo cuadrante; 3) en el caso de que $X < 0$ y $Y < 0$, la coordenada perteneces al tercer cuadrante; y 4) adicionalmente, si $X > 0$ y $Y < 0$, entonces la coordenada pertenece al cuarto cuadrante. Finalmente, si $X = Y = 0$, la coordenada está ubicada en el origen; de otro modo la coordenada está situada en uno de los ejes cartesianos.

```
Proc: Clasificador de la coordenada (X, Y) en el plano cartesiano
| Real: X, Y
| Lea X, Y  // Leer las coordinadas X y Y a clasificar en el
| // plano en mención
| // Estructura de primitivas condicionales que clasifican la coordenada
1
```

```
1
| Escriba "La coordenada (" X, ",", Y, ")"
| Si(X > 0 y Y > 0)Entonces
| | Escriba " está en el primer cuadrante"
| Sino
| | Si(X < 0 y Y > 0)Entonces
| | | Escriba " está en el segundo cuadrante"
| | Sino
| | | Si(X < 0 y Y < 0)Entonces
| | | | Escriba " está en el tercer cuadrante"
| | | Sino
| | | | Si(X > 0 y Y < 0)Entonces
| | | | | Escriba " está en el cuarto cuadrante"
| | | | Sino
| | | | | Si(X = 0 y Y = 0)Entonces
| | | | | | Escriba " está en el origen"
| | | | | Sino
| | | | | | Si(X = 0)Entonces
| | | | | | | Si(Y > 0)Entonces
| | | | | | | | Escriba " está en el eje de las Y positivo"
| | | | | | | Sino
| | | | | | | | Escriba " está en el eje de las Y negativo"
| | | | | | | F.Si
| | | | | | Sino
| | | | | | | Si(X < 0)Entonces
| | | | | | | | Escriba " está en el eje de las X negativo"
| | | | | | | Sino
| | | | | | | | Escriba " está en el eje de las X positivo"
| | | | | | | F.Si
| | | | | | F.Si
| | | | | F.Si
| | | | F.Si
| | | F.Si
| | F.Si
| F.Si
Fin_proc
```

Ejemplo 3.30 Orden de crecimiento de funciones para grandes valores de n

Dadas dos funciones $F(x) = 2^x$ y $G(x) = x^x$, diseñe un algoritmo que compruebe cuál de las dos funciones es mayor para un valor grande de x tal que x supere a 1 000 000.

Análisis: Los cálculos de funciones para grandes valores de n son importantes, pues indican la tendencia de crecimiento de la función, lo cual sirve para realizar el análisis asintótico de las funciones. Luego, para el valor dado de x se calcula el valor de las dos funciones y se comparan para decidir cuál de las dos funciones es mayor.

Proc: Ordenador asintótico de funciones $F(x)$ y $G(x)$
| // Leer el valor de X para el cual se van a calcular $F(x)$ y $G(x)$
| **Real:** X, FX, GX
| **Lea** X
| **Si**($X \geq 1000000$)**Entonces** // Chequea para grandes valores de X
| | $FX \leftarrow 2 * *X$ // Cálculo de la función $F(x)$
| | $GX \leftarrow X * *X$ // Cálculo de la función $G(x)$
| | **Si**($FX \geq GX$)**Entonces**
| | | **Escriba** "La función FX es mayor que GX"
| | **Sino**
| | | **Escriba** "La función FX es menor que GX"
| | **F.Si**
| **F.Si**
Fin_proc

Ejemplo 3.31 Operaciones lógicas booleanas

Dadas dos variables booleanas a, b, diseñe un algoritmo que imprima el valor de la tabla de verdad de las operaciones OR, AND y NOT de a y b, sujeto a una opción de impresión, la cual es leída en la variable Op, y que discrimina la impresión así: Si la opción es 1, se imprime la tabla de verdad OR; si la opción es 2, se imprime la tabla de verdad AND; y en caso contrario se imprime la tabla de verdad del operador NOT.

Análisis: Las tablas de verdad de las operaciones mencionadas se presentan en la tabla 3.4. Luego, de acuerdo con la opción leída se distribuye el flujo de control del algoritmo para imprimir la respectiva tabla.

Tabla 3.4 Tabla de verdad de las operaciones booleanas

OR $(+)$	**AND** $(.)$	**NOT**
$a + b$	$a \,.\, b$	Si $a = 0$, entonces $\overline{a} = 1$
$0 + 0 = 0$	$0 \,.\, 0 = 0$	Si $a = 1$, entonces $\overline{a} = 0$
$0 + 1 = 1$	$0 \,.\, 1 = 0$	
$1 + 0 = 1$	$1 \,.\, 0 = 0$	
$1 + 1 = 1$	$1 \,.\, 1 = 1$	

```
Proc: Impresor de tablas de verdad OR, AND, NOT
  Entero: a, b, Op, B
  Lea a, b // Leer las variables booleanas
  Lea Op // Leer la opción del operador booleano
  Si(Op = 1)Entonces // Se calcula el valor de a OR b
    Si(a = 0 o b = 0)Entonces
      B ← 0 // Valor booleano (B) resultado
    Sino
      B ← 1
    F.Si
    Escriba a, " OR ", b, " es igual a ", B
  Sino
    Si(Op = 2)Entonces // Se calcula el valor de a AND b
      Si(a = 1 y b = 1)Entonces
        B ← 1
      Sino
        B ← 0
      F.Si
      Escriba a, " AND ", b, " es igual a ", B
    Sino
      Si(Op = 3)Entonces // Se calcula el valor de NOT A
        Si(a = 1)Entonces
          B ← 0
        Sino
          B ← 1
        F.Si
        Escriba "El valor negado de ", a, " es igual a ", B
      F.Si
    F.Si
  F.Si
Fin_proc
```

Ejemplo 3.32 Diseño de la ecuación de la línea recta

Dados dos puntos en el plano cartesiano, representados por las coordenadas $(X1, Y1)$ y $(X2, Y2)$, construya un algoritmo para diseñar la ecuación de la línea de recta y escribir si su pendiente es creciente o decreciente.

Análisis: La ecuación de la línea recta es $y = mx + b$. En función de las coordenadas dadas, las ecuaciones son $y_1 = x_1 + b$ y $y_2 = x_2 + b$; resolviendo las ecuaciones de la primera restando la segunda se calcula el valor de la pendiente de la recta, que es igual a $m = \dfrac{y_1 - y_2}{x_1 - x_2}$. Si la pendiente es mayor que cero, es creciente la ecuación de la línea recta; de lo contrario es decreciente. Finalmente, la ecuación de la línea recta en función de las

coordenadas dadas es

$$y = \frac{y_1 - y_2}{x_1 - x_2}(x - x_1) + y_1. \quad x_1 \neq x_2$$

Proc: Diseño de la ecuación de la línea recta en función de dos coordenadas
| **Real:** $X1$, $Y1$, $X2$, $Y2$, m
| **Lea** $X1, Y1, X2, Y2$ // Leer las coordenadas de los puntos
| $m \leftarrow (Y1 - Y2)/(X1 - X2)$
| **Escriba** "La ecuación de la línea recta es $y =$ "
| **Si**$(m \neq 0)$**Entonces**
| | **Escriba** m, "$(x-$", $X1$, "$) + $", $Y1$
| | **Si**$(m > 0)$**Entonces**
| | | **Escriba** "La pendiente de la recta es positiva"
| | **Sino**
| | | **Escriba** "La pendiente de la recta es negativa"
| | **F.Si**
| **Sino**
| | **Escriba** $Y1$
| **F.Si**
Fin_proc

Ejemplo 3.33 Escoger el mayor de tres números

Dados tres números distintos, n, m, k, los cuales son leídos, diseñe un algoritmo utilizando primitivas lógicas Si compuestas anidadas para seleccionar el mayor de los tres números.

Análisis: Suponga que los números son $n = 9$, $m = 7$ y $k = 3$; entonces el algoritmo debe realizar todas las combinaciones de comparación; sin hacer una enumeración exhaustiva, una de las combinaciones es si $n > m$, lo cual implica que $9 > 7$, luego la respuesta es Si; entonces dentro de ese sí preguntamos que Si $(n > k)$, pregunta que corresponde a $9 > 3$, y como la respuesta es verdadera, entonces se concluye que el mayor de los tres números es 9.

En caso de que $n > m$ sea falso, implicaría que $m > n$; entonces se debe preguntar cómo es m con respecto a k. Luego, Si $m > k$, se concluye que el mayor valor es m; de lo contrario k es mayor que m, y por lo tanto, el número mayor es k.

Proc: Número mayor entre n, m y k
| **Entero:** n, m, k
| **Lea** n, m, k // Leer los tres números
| // Estructura de primitivas condicionales seleccionan el número mayor
1

```
1
| Si(n > m)Entonces
| | Si(n > k)Entonces
| | | Escriba "El número mayor es ", n
| | Sino
| | | Escriba "El número mayor es ", k
| | F.Si
| Sino
| | Si(m > k)Entonces
| | | Escriba "El número mayor es ", m
| | Sino
| | | Escriba "El número mayor es ", k
| | F.Si
| F.Si
Fin_proc
```

Ejemplo 3.34 Operaciones sobre la descomposición de un número n de d dígitos

Dado un número n de tres dígitos, diseñe un algoritmo para descomponer el número en sus cifras de unidades (u), decenas (d) y centenas (c) y realizar las siguientes operaciones: *i*) ¿cuántas cifras del número son iguales a cero?, *ii*) ¿cuántas cifras del número son pares?, *iii*) ¿cuántas cifras del número son impares?, *iv*) la suma de las cifras del número y, *v*) la productoria de las cifras del número leído.

Análisis: Un número $n = 704$ en el sistema decimal es igual a 4 unidades, 0 decenas y 7 centenas. *i*) El número de cifras iguales a cero (cc) del número es 1. *ii*) El número de cifras pares del número (cp) es 1, representada por el número 4. *iii*) La cantidad de cifras impares del número es 1, equivalente al número 7. *iv*) La suma de las cifras del número dado es $7 + 0 + 4 = 11$. *v*) Finalmente, el producto de las cifras del número $7 * 0 * 4 = 0$.

```
Proc: Operaciones sobre la descomposición de un número de tres cifras
| Entero: n, cc, cp, ci, Suma, Producto, u, d, c
| Lea n // Leer el número que se va a descomponer
| // Se inicializan los contadores
| cc ← 0 // Contador de cifras iguales a cero
| cp ← 0 // Contador de cifras pares
| ci ← 0 // Contador de cifras impares
| // Inicializar los contadores de sumatoria y productoria de las cifras
| Suma ← 0 // La suma de las cifras se inicializa en cero
| Producto ← 1 // El producto de las cifras se inicializa en uno
| // Descomposición del número en sus cifras
1
```

```
1
| u ← n mod 10 // Unidades del número n
| d ← n/10 mod 10 // Decenas del número , n/10 es división entera
| c ← n/100 mod 10 // Centenas del número, n/100 es división entera
| // Parte i)
| Si(u = 0)Entonces
| | cc ← cc + 1
| F.Si
| Si(d = 0)Entonces
| | cc ← cc + 1
| F.Si
| Si(c = 0)Entonces
| | cc ← cc + 1
| F.Si
| // Partes ii) y iii)
| Si(u mod 2 = 0)Entonces
| | cp ← cp + 1
| Sino
| | ci ← ci + 1
| F.Si
| Si(d mod 2 = 0)Entonces
| | cp ← cp + 1
| Sino
| | ci ← ci + 1
| F.Si
| Si(c mod 2 = 0)Entonces
| | cp ← cp + 1
| Sino
| | ci ← ci + 1
| F.Si
| Suma ← u + d + c // Cálculo de la suma de las cifras
| Producto ← u ∗ d ∗ c // Cálculo de la multiplicación de las cifras
| // Impresión de resultados
| Escriba "El número de entrada ", n, " está compuesto por: "
| Escriba "Número de unidades: ", u
| Escriba "Número de decenas: ", d
| Escriba "Número de centenas: ", c
| Escriba "Número de cifras cero del número: ", cc
| Escriba "Número de cifras pares del números: ", cp
| Escriba "Número de cifras impares del número: ", ci
| Escriba "Suma de las cifras: ", Suma
| Escriba "Producto de las cifras: ", Producto
Fin_proc
```

3.5 Estructura lógica Dependiendo De

La estructura condicional compuesta permite la ejecución de un conjunto de primitivas de control (p_i), en la que el cumplimiento de la condición de la estructura en mención hace posible la ejecución de las instrucciones entre muchas alternativas; no solo por el cumplimiento verdadero de la condición, punto en el cual se ejecutan las primitivas p_1, p_2, p_3, ..., p_n, sino por el cumplimiento de la condición cuando es falsa, en cuyo lugar se procesan las primitivas q_1, q_2, q_3, ..., q_n, siendo el conjunto de primitivas disyuntas. Estas alternativas se amplían considerablemente cuando las estructuras condicionales se encuentran anidadas. La condicional 'Dependiendo De' es una alternativa a la condicional compuesta cuando es necesario en la comparación el cumplimiento de la condición con un valor específico. Luego, si se cumple el valor específico, se ejecutan únicamente las primitivas asociadas con la selección de este valor, y adicionalmente, las primitivas asociadas con un valor son completamente diferentes a las asociadas con el cumplimiento de otro valor.

Formalmente, la estructura lógica *control unitario* se denota por Dependiendo_De(opción) o DD(opción). Sea V_i una variable que contiene el cumplimiento de V_i opciones ($1 \leq i \leq n$); suponga que cada una de las opciones tiene asociadas un conjunto de primitivas disyuntas identificadas como V_1: p_1^1, p_2^1, ..., p_k^1 para la opción uno; V_2: p_1^2, p_2^2, ..., p_k^2 si la opción toma el valor igual a dos; V_i: p_1^i, p_2^i, ..., p_k^i, en el caso de que cumpla la variable V_i; y finalmente V_n: p_1^n, p_2^n, ..., p_k^n para el cumplimiento de la variable n. Al final de la ejecución de primitivas de la opción seleccionada hay un rompimiento de la ejecución del algoritmo (*break*), en cuyo caso el flujo de control del algoritmo termina.

La estructura lógica compuesta DD es útil en el diseño de algoritmos que requieran el diseño de menús de opciones, o diseños que requieran la modularización parcial de la lógica de control del algoritmo. La lógica de programación modular se desarrolla en el cuarto capítulo.

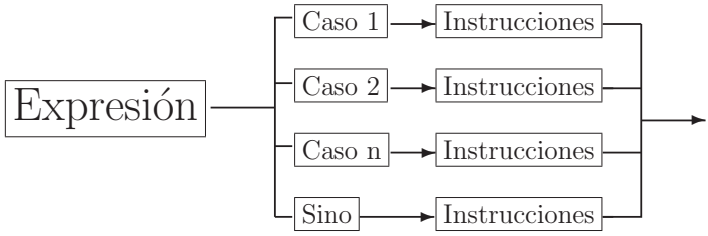


Figura 3.5 Estructura Dependiendo De

La condicional Dependiendo De en su notación algorítmica se escribe de la siguiente forma:

Dependiendo_De(V)**Haga**
V$_1$:
$p_1^1, p_2^1, \ldots, p_k^1$
Fin
V$_2$:
$p_1^2, p_2^2, \ldots, p_k^2$
Fin
$\vdots$
V$_n$:
$p_1^n, p_2^n, \ldots, p_k^n$
Fin
Fin_Dependiendo_De

La primitiva Dependiendo De (DD) es útil cuando se requiere el diseño de algoritmos con opciones de menús de alternativas, en los que cada alternativa es disyunta y se ejecuta independientemente de las otras.

Ejemplo 3.35 Administrador de funciones trigonométricas

Diseñe un algoritmo que ejecute las funciones trigonométricas $\sin(x)$, $\cos(x)$ y $\tan(x)$, sujetas a la selección de las opciones asociadas así: opción uno para la función Seno, opción dos para la función Coseno y opción tres para el cálculo de la Tangente, siendo el valor de x un argumento de entrada dado en grados.

Análisis: Se lee la opción y el valor en grados, se estructura el menú de opciones, y finalmente con el Dependiendo De, sujeto al valor de la opción se ejecuta el cálculo de la función con el argumento (x) de entrada.

Proc: Administrador de funciones trigonométricas
| **Entero:** OP
| **Real:** x, R
| **Escriba** "Administrador de funciones trigonométricas básicas"
| **Escriba** " 1. Seno"
| **Escriba** " 2. Coseno"
| **Escriba** " 3. Tangente"
| **Escriba** "Por favor digite una de las opciones 1, 2 o 3 únicamente: "
| **Lea** OP // Variable que contiene la opción trigonométrica seleccionada
| **Lea** x // Valor en grado del argumento de entrada de la función
| **DD**(OP)**Haga**
| | **Opción** 1 :
| | | $R \leftarrow \sin(x)$
| | | **Escriba** "El valor de la función $\sin(x)$ es igual a ", R
| | **Fin**
| | **Opción** 2 :
| | | $R \leftarrow \cos(x)$
1 2 3

```
1 2 3
|  |  | Escriba "El valor de la función cos(x) es igual a ", R
|  | Fin
|  | Opción 3 :
|  |  | R ← tan(x)
|  |  | Escriba "El valor de la función tan(x) es igual a ", R
|  | Fin
|  Fin_DD
Fin_proc
```

Ejemplo 3.36 Calculadora básica

Diseñe un algoritmo que dados dos números, n y m, permita el cálculo de la suma, resta, multiplicación, división y exponenciación de los dos números, dependiendo de la selección de opciones en un menú identificadas como: la primera opción para la suma, la segunda para la resta, la tercera para la multiplicación, la cuarta para la división y, finalmente, la quinta para la potenciación n^m.

Análisis: En primer lugar, la calculadora básica se organiza a través de un menú de acuerdo con las opciones dadas: se lee la opción del menú de la calculadora (supuesta válida en el rango $[1, 5]$); en segundo lugar se leen los números n y m; finalmente, de acuerdo con la opción, se ejecuta el cálculo de la operación seleccionada con los argumentos dados.

```
Proc: Administrador de funciones trigonométricas
|  Entero: OP
|  Real: n, m, R
|  Escriba "Calculadora Básica"
|  Escriba " 1. Suma"
|  Escriba " 2. Resta"
|  Escriba " 3. Multiplicación"
|  Escriba " 4. División"
|  Escriba " 5. Exponenciación"
|  Escriba "Por favor, digite una de las opciones únicamente: "
|  Lea OP // Contiene la opción que ejecutará la calculadora
|  Lea n, m // Valor de los operandos (n y m)
|  DD(OP)Haga
|  |  Opción 1 :
|  |  |  R ← n + m
|  |  |  Escriba "El valor de la suma es igual a ", R
|  |  Fin
|  |  Opción 2 :
|  |  |  R ← n − m
1 2 3
```

```
1 2 3
|  |  | Escriba "El valor de la resta es igual a ", R
|  | Fin
|  | Opción 3 :
|  | | R ← n * m
|  | | Escriba "El valor de la multiplicación es igual a ", R
|  | Fin
|  | Opción 4 :
|  | | R ← n/m
|  | | Escriba "El valor de la división es igual a ", R
|  | Fin
|  | Opción 5 :
|  | | R ← n * *m
|  | | Escriba "El valor de la exponenciación es igual a ", R
|  | Fin
| Fin_DD
Fin_proc
```

Ejemplo 3.37 Clasificador de alimentos

Suponga que un restaurante ofrece servicios de alimentos a través de Internet. Los tipos de comidas que ofrece son vegetarianas, no vegetarianas y rápidas. Las vegetarianas tienen un menú especificado por sopas de vegetales, ensaladas y jugos. Las no vegetarianas, por bandejas de carne, pollo y cerdo; y las comidas rápidas, de perros calientes y hamburguesas. Diseñe un algoritmo que permita al cliente seleccionar la comida de su preferencia y dar el valor de la compra, suponiendo que el restaurante asigna a través de precios fijos el valor de la compra de cada plato, a lo cual se le debe agregar el IVA de la compra, que es leído. Adicionalmente, si la comida es vegetariana, el restaurante tiene un descuento del 20 %; si es no vegetariana, del 10 %; y si es rápida, del 5 %.

Análisis: Se organiza el restaurante en un sistema de menús así: El primer menú son los tipos de comida: 1. Vegetarianas, 2. No vegetarianas, y 3. Rápidas. El segundo menú, dependiendo de la opción seleccionada, se diseña para cada opción un segundo menú dispuesto como: el vegetariano, compuesto por dos opciones: 1. Sopas de vegetales, 2. Ensaladas y jugos. El menú de carnes, compuesto por: 1. Pollo, 2. Res, y 3. Cerdo; y finalmente, el rápido, compuesto por: 1. Perros calientes y 2. Hamburguesas. Posteriormente se solicita al cliente el número de comidas que se va a comprar y se realiza la liquidación de la cuenta.

```
Proc: Sistema de restaurante
| Entero: Sopas, Ensaladas, Pollo, Res, Cerdo, PerrosCalientes
1
```

```
1
| Entero: Hamburguesas, Iva, OP, OP1, n
| Real: V
| // Inicialización de precios fijos de menús
| Sopas ← 10000
| Ensaladas ← 25000
| Pollo ← 28000
| Res ← 30000
| Cerdo ← 33000
| PerrosCalientes ← 5000
| Hamburguesas ← 7000
| Lea Iva
| Escriba "Sistema de Restaturante"
| Escriba "1. Vegetariano"
| Escriba "2. No Vegetariano"
| Escriba "3. Comida Rápida"
| Escriba "Por favor, digite una de las opciones únicamente: "
| Lea OP // Variable que contiene la opción que ejecutará el restaurante
| DD(OP)Haga
| | Opción 1 :
| | | Escriba "1. Sopa Vegetariana"
| | | Escriba "2. Ensalada y jugo"
| | | Lea OP1
| | | Escriba "¿Cuántas comidas quiere comprar?"
| | | Lea n
| | | DD(OP1)Haga
| | | | Opción 1 :
| | | | | V ← n * Sopas
| | | | Fin
| | | | Opción 2 :
| | | | | V ← n * Ensaladas
| | | | Fin
| | | Fin_DD
| | | V ← V − 0.2 * V
| | | V ← V + V * Iva/100
| | | Escriba V
| | Fin
| | Opción 2 :
| | | Escriba "1. Pollo"
| | | Escriba "2. Carne"
| | | Escriba "3. Cerdo"
| | | Lea OP1
| | | Escriba "¿Cuántas comidas quiere comprar?"
| | | Lea n
1 2 3
```

```
1 2 3
|  |  |  DD(OP1)Haga
|  |  |  |  Opción 1 :
|  |  |  |  |  V ← n * Pollo
|  |  |  |  Fin
|  |  |  |  Opción 2 :
|  |  |  |  |  V ← n * Carne
|  |  |  |  Fin
|  |  |  |  Opción 3 :
|  |  |  |  |  V ← n * Cerdo
|  |  |  |  Fin
|  |  |  Fin_DD
|  |  |  V ← V − 0.1 * V
|  |  |  V ← V + V * Iva/100
|  |  |  Escriba V
|  |  Fin
|  |  Opción 3 :
|  |  |  Escriba "1. Perro Caliente"
|  |  |  Escriba "2. Hamburguesa"
|  |  |  Lea OP1
|  |  |  Escriba "¿Cuántas comidas quiere comprar?"
|  |  |  Lea n
|  |  |  DD(OP1)Haga
|  |  |  |  Opción 1 :
|  |  |  |  |  V ← n * PerroCaliente
|  |  |  |  Fin
|  |  |  |  Opción 2 :
|  |  |  |  |  V ← n * Hamburguesa
|  |  |  |  Fin
|  |  |  Fin_DD
|  |  |  V ← V − 0.05 * V
|  |  |  V ← V + V * Iva/100
|  |  |  Escriba V
|  |  Fin
|  Fin_DD
Fin_proc
```

Ejemplo 3.38 Simulador de cajero automático

Un sistema de cajero automático tiene cuatro opciones: cambio de clave, consulta de saldos, retiros y transferencias bancarias. Las posibilidades de retiro son: el cliente puede retirar de tres opciones del cajero 500 000, 1 000 000 o cantidad deseada. La opción de transferencia bancaria pide el valor que se va a transferir de la cuenta del cliente a otra cuenta.

Diseñe un algoritmo que simule el sistema de cajero, teniendo en cuenta que para la consulta del saldo y otras transacciones se debe validar la clave del cliente.

Análisis: Se organizan dos sistemas de menús: el primero con las cuatro opciones ofrecidas por el banco y el segundo con las opciones de retiro; y en cada una de ella se hace la respectiva acción de la transacción.

Proc: Simulador de cajero automático
| **Entero:** OP, $Clave$, $NuevaClave$, $Saldo$, N, T, STC
| **Escriba** "Opciones Bancarias"
| **Escriba** "1. Cambio de Clave"
| **Escriba** "2. Consulta de Saldo"
| **Escriba** "3. Retiros"
| **Escriba** "4. Transferencia de Cuentas"
| **Escriba** "Por favor, digite una de las opciones únicamente: "
|
| **Lea** OP
| **DD**(OP)**Haga**
| | **Opción** 1 :
| | | **Escriba** "Digite la clave actual"
| | | **Lea** $Clave$
| | | **Si**($Clave = Valida$)**Entonces**
| | | | **Escriba** "Digite Nueva Clave"
| | | | **Lea** $NuevaClave$
| | | | $Clave \leftarrow NuevaClave$
| | | **Sino**
| | | | **Escriba** "Clave inválida"
| | | **F.Si**
| | **Fin**
| | **Opción** 2 :
| | | **Escriba** "Digite la clave actual"
| | | **Lea** $Clave$
| | | **Si**($Clave = Valida$)**Entonces**
| | | | **Lea** $Saldo$
| | | | // El saldo se encuentra en un dispositivo secundario
| | | | **Escriba** "El valor del saldo es ", $Saldo$
| | | **Sino**
| | | | **Escriba** "Clave inválida"
| | | **F.Si**
| | **Fin**
| |
| | **Opción** 3 :
| | |
1 2 3

```
1 2 3
|  |  |  Escriba "Opciones de Retiro de Efectivo"
|  |  |  Escriba "1. 500.000"
|  |  |  Escriba "2. 1'000.000"
|  |  |  Escriba "3. Cantidad diferente"
|  |  |  Escriba "Digite la opción de retiro: "
|  |  |  Lea OP1
|  |  |  DD(OP1)Haga
|  |  |  |  Opción 1 :
|  |  |  |  |  Lea Clave
|  |  |  |  |  Si(Clave = Valida)Entonces
|  |  |  |  |  |  Lea Saldo
|  |  |  |  |  |  Saldo ← Saldo − 500000
|  |  |  |  |  |  Escriba "El valor entregado es 500.000"
|  |  |  |  |  |  Escriba "El valor del saldo es ", Saldo
|  |  |  |  |  Sino
|  |  |  |  |  |  Escriba "Clave inválida"
|  |  |  |  |  F.Si
|  |  |  |  Fin
|  |  |  |  Opción 2 :
|  |  |  |  |  Lea Clave
|  |  |  |  |  Si(Clave = Valida)Entonces
|  |  |  |  |  |  Lea Saldo
|  |  |  |  |  |  Saldo ← Saldo − 1000000
|  |  |  |  |  |  Escriba "El valor entregado es 1'000.000"
|  |  |  |  |  |  Escriba "El valor del saldo es ", Saldo
|  |  |  |  |  Sino
|  |  |  |  |  |  Escriba "Clave inválida"
|  |  |  |  |  F.Si
|  |  |  |  Fin
|  |  |  |  Opción 3 :
|  |  |  |  |  Lea Clave
|  |  |  |  |  Si(Clave = Valida)Entonces
|  |  |  |  |  |  Lea Cantidad // Cantidad a retirar
|  |  |  |  |  |  Lea Saldo
|  |  |  |  |  |  Saldo ← Saldo − Cantidad
|  |  |  |  |  |  Escriba "El valor entregado es ", Cantidad
|  |  |  |  |  |  Escriba "El valor del saldo es ", Saldo
|  |  |  |  |  Sino
|  |  |  |  |  |  Escriba "Clave inválida"
|  |  |  |  |  F.Si
|  |  |  |  Fin
|  |  |  Fin_DD
|  |  Fin
1 2
```

```
1 2
|  | Opción 4 :
|  |  Lea Clave
|  |  Si(Clave = Valida)Entonces
|  |  |  Lea N  // Número de la cuenta
|  |  |  Lea T  // Valor que se va a transferir
|  |  |  Lea Saldo
|  |  |  Saldo ← Saldo − T
|  |  |  Lea STC  // Lea saldo de la cuenta
|  |  |  STC ← STC + T
|  |  |  Escriba "La transferencia fue realizada correctamente"
|  |  Sino
|  |  |  Escriba "Clave inválida"
|  |  F.Si
|  | Fin
|  Fin_DD
Fin_proc
```

3.6 Estructura lógica repetitiva Para

La primitiva de control Para, permite la ejecución de un conjunto de estructuras de control de una forma repetitiva que se ubican dentro del Para, comenzando con un valor inicial, terminando en un valor final, y ejecutando el conjunto de primitivas internas del Para con un incremento de una constante o variable j.

La notación algorítmica de la estructura de control en mención es la siguiente:

Dado un conjunto de primitivas $p_1, p_2, p_3, \ldots, p_i, \ldots, p_n$, se escribe como

Para $i = 1$ **hasta** n con incrementos de j haga

$p_1, p_2, \ldots, p_n$

Fin_Para

Permite la ejecución repetitiva del conjunto de primitivas $p_1, p_2, p_3, \ldots, p_i, \ldots, p_n$ hasta el valor n, o tope del ciclo de control Para, con incrementos de j, los cuales afectan la variable índice i.

Los ciclos repetitivos en el diseño de algoritmos son útiles en el caso de:

- La generación de secuencias de números.
- Cálculo de sumatorias $\sum$, tales como $\sum_{i=1}^{n} i^2$ o $\sum_{i=1}^{n} f(i)$

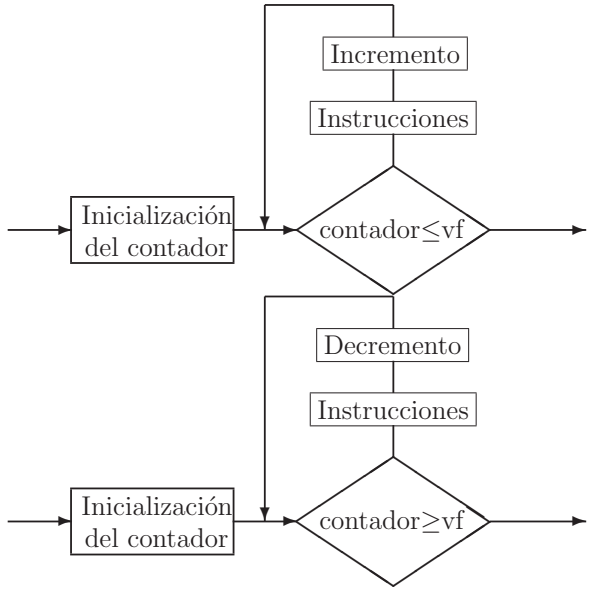


Figura 3.6 Estructura Repetitiva Para

- Cálculo de series; por ejemplo, la serie de Taylor de la función $f(x) = \sin(x)$,

$$\sin(x) = \frac{x!}{1!} - \frac{x!}{3!} + \frac{x!}{5!} - \cdots = \sum_{k=0}^{\infty} \frac{(-1)^k x^{2k+1}}{(2k+1)!}.$$

- Cálculo de productorias $\prod$, como es el caso de $n! = \prod_{i=1}^{n} i = 1 * 2 * 3 * \cdots * n$.
- Generación de series y sucesiones.
- Generación de funciones trigonométricas, entre otros casos, de repetición de un término de una sumatoria, de una productoria, o de una función matemática repetitiva sujeta a la variación de un índice, entre otros casos.
- Acciones de repetición de un conjunto de actividades desde un punto inicial a un punto final, como es el caso de lectura de calificaciones, contadores (luz, agua), censos poblacionales, o encuestas, entre otros; siempre y cundo se tenga el tope o límites del número de actividades que se van a considerar en la repetición (valor de n).
- Administración de vectores, y
- Administración de matrices.

Se debe tener en cuenta que el incremento de la estructura Para puede ser positivo o negativo, y se controla con el tope definido dentro de la estructura.

Ejemplo 3.39 Generación de los números naturales

Diseñe un algoritmo para imprimir los n primeros números naturales, validando que el valor de n, el cual debe ser leído, es positivo.

Análisis: Los números naturales son la generación de los dígitos 1, 2, 3, ..., y sus sucesores hasta el valor de n, verificando que el valor de n sea mayor que cero.

Proc: Impresor de números naturales hasta el valor de n
| **Entero:** n, i
| **Lea** n // Valor límite o tope de impresión de los números naturales
| **Si**$(n \leq 0)$**Entonces**
| | **Escriba** "El número n no es positivo"
| **Sino**
| | **Escriba** "Los números naturales desde el valor 1 hasta el valor"
| | **Escriba** "de $n =$ ", n, " son "
| | **Para**$(i = 1, n, 1)$**Haga**
| | | **Escriba** "Valor del número natural = ", i
| | **Fin_Para**
| **F.Si**
Fin_proc

Ejemplo 3.40 Cálculo de la potencia n de un número por multiplicaciones sucesivas

Diseñe un algoritmo para calcular la expresión matemática $r = x^n$ para valores de x y n leídos, suponiendo que $n > 1$, por multiplicaciones sucesivas, validando que el valor de n es positivo.

Análisis: La expresión $r = x^n$ por multiplicaciones sucesivas es igual a

$$r = \underbrace{x * x * x * \cdots * x}_{n \text{ veces}}.$$

Proc: Potencia del número n por multiplicaciones sucesivas
| **Entero:** n, i
| **Real:** x, r
| **Lea** n // Valor límite o tope de cálculo de la máxima potencia de n
| **Lea** x // Valor a ser elevado a la potencia n
| **Si**$(n \leq 0)$**Entonces**
| | **Escriba** "El número n no es positivo"
| **Sino**
| | // Iniciar el resultado de las sucesivas multiplicaciones en la base x
| | $r \leftarrow x$
1 2

```
1 2
|  |  Para(i = 1, n − 1, 1)Haga
|  |  | r ← r ∗ x
|  |  Fin_Para
|  |  Escriba "El valor resultado de ", x, " elevado a la potencia ", n
|  |  Escriba "es igual a ", r
|  F.Si
Fin_proc
```

Ejemplo 3.41 Impresor descendente de números desde el valor de n hasta k

Diseñe un algoritmo para imprimir los número desde un valor n hasta un valor k en orden descendente de incrementos igual a $-i$, suponiendo que $n \gg k$.

Análisis: Teniendo en cuenta que n es mucho mayor ($\gg$) que k, se debe realizar un impresión o escritura de los números desde el valor n hasta el valor k con incrementos de $-i$. Supuesto $n = 15$, $k = 3$ e $i = -3$, entonces los números que se deben escribir son: $15, 12, 9, 6, 3$.

```
Proc: Impresor de números
|  Entero: n, k, i, j
|  Lea n, k, i
|  Si(n ≤ k)Entonces
|  |  Escriba "Valores de entrada errados porque n debe ser "
|  |  Escriba "mucho mayor que k"
|  Sino
|  |  // Ciclo repetitivo que hace la impresión de los números de n hasta k
|  |  // con incrementos negativos de i
|  |  Escriba "La impresión de los números desde el valor ", n
|  |  Escriba " hasta el valor de ", k
|  |  Escriba " con decrementos de ", i, " son "
|  |  Para(j = n, k, −i)Haga
|  |  |  Escriba j
|  |  Fin_Para
|  F.Si
Fin_proc
```

Ejemplo 3.42 Cálculo del factorial de número n donde $n \in \mathbb{Z}^+$

Construya un algoritmo para calcular el factorial de un número n que pertenece a los enteros positivos, el cual se debe leer y verificar si pertenece a los enteros positivos.

Análisis: La definición matemática de $n!$ es $n = 1 * 2 * 3 * 4 \cdots * (n-1) * n$. Supuesto $n = 7$, entonces el valor del factorial es $1 * 2 * 3 * 4 * 5 * 6 * 7 = 720$.

```
Proc: Cálculo del factorial de un número entero positivo
| Entero: n, Factorial, i
| Lea n // Valor límite o tope de cálculo de la máxima potencia de n
| Si(n ≤ 0)Entonces
| | Escriba "El número n no es un entero positivo"
| Sino
| | // Iniciar el resultado en el módulo de la multiplicación
| | Factorial ← 1
| | Para(i = 1, n, 1)Haga
| | | Factorial ← factorial * i
| | Fin_Para
| | Escriba "El valor de ", n, "! es ", Factorial
| F.Si
Fin_proc
```

Ejemplo 3.43 Generación de la serie de los números de Fibonacci y su respectiva suma

Diseñe un algoritmo para generar la serie de los números de Fibonacci y su respectiva suma desde el valor 0 hasta el valor n, donde n, que pertenece a los enteros positivos, es leído.

Análisis: La sucesión de Fibonacci (F_n) está dada por la fórmula recursiva

$$F_n = F_{n-1} + F_{n-2}. \quad \text{con } F_1 = 1,\ F_2 = 1$$

La serie de Fibonacci corresponde, entonces, a la suma de los n primeros términos de la sucesión, es decir,

$$S_n = \sum_{i=1}^{n} F_i = F_1 + F_2 + \cdots + F_n$$

Por ejemplo, para el valor de $n = 5$ se tiene $F(1) = 1$, $F(2) = 1$, $F(3) = 2$, $F(4) = 3$ y $F(5) = 5$; por lo tanto, la serie de 5 corresponde a la suma $1 + 1 + 2 + 3 + 5 = 12$.

```
Proc: Generación de la serie de Fibonnaci con su respectiva suma
| Entero: n, Primero, Segundo, Suma, Tercero
| Lea n // Valor tope de la serie de Fibonacci
| Si(n ≤ 0)Entonces
| | Escriba "El número no es un entero positivo
| Sino
| | Primero ← 1 // Iniciar el primer número de la serie de Fibonacci
| | Segundo ← 2 // Iniciar el segundo número de la serie de Fibonacci
| | // Ciclo repetitivo que genera las serie de Fibonacci y su suma
1 2
```

```
1 2
|  |  Suma ← 1 + 2 // Iniciar la serie de Fibonacci
|  |  Escriba "Sucesión de Fibonacci"
|  |  Escriba "Término 1: ", Primero
|  |  Escriba "Término 2: ", Segundo
|  |  Para(i = 3, n, 1)Haga
|  |  |  Tercero ← Primero + Segundo
|  |  |  Suma ← Suma + Tercero
|  |  |  Escriba "Término ", i, ": ", Tercero
|  |  |  Primero ← Segundo
|  |  |  // Ahora el segundo término pasa a ser el primero
|  |  |  Segundo ← Tercero // y el nuevo pasa a ser el segundo
|  |  |  Escriba "Suma: ", Suma
|  |  Fin_Para
|  |  F.Si
Fin_proc
```

Ejemplo 3.44 Escoger el mayor y el menor número de un conjunto de L números

Sea un conjunto de L números ($L > 0$), los cuales son leídos, diseñe un algoritmo que calcule el número mayor y el número menor del conjunto dado. Asuma que el usuario ingresa un entero positivo L.

Análisis: Suponga $L = 10$, o conjunto de números cuyos valores son 10, 29, 87, 25, 67, 77, 87, 98, 99, 23. El algoritmo debe sacar el mayor de los números, es decir, 99, y el menor de los números, es decir, el valor de 10.

```
Proc: El mayor y el menor de L números
|  Entero: L, i
|  Real: n, Mayor, Menor
|  Lea L // Leer el valor del número de números a ser leídos
|  Lea n // See lee el primer número
|  Mayor ← n // Se asigna este número como el mayor
|  Menor ← n // Se asigna este número como el menor
|  // Al leer más números, se pregunta si es mayor o menor que
|  // Mayor y Menor, respectivamente
|  L ← L − 1 // Se reduce el valor en 1, debido a que ya se leyó
|  // un primer número
|  Para(i = 1, L, 1)Haga
|  |  Lea n // See lee el nuevo número
|  |  Si(n > Mayor)Entonces
|  |  |  // Debido a que se encontró un número mayor que Mayor
|  |  |  // se actualiza el valor de este último
|  |  |  Mayor ← n
|  |  F.Si
1 2
```

```
1 2
|  |  Si(n < Menor)Entonces
|  |  |  // Debido a que se encontró un número menor que Menor
|  |  |  // se actualiza el valor de este último
|  |  |  Menor ← n
|  |  F.Si
|  Fin_Para
|  Escriba "El mayor número leído es ", Mayor
|  Escriba "El menor número leído es ", Menor
Fin_proc
```

Ejemplo 3.45 Cálculo del promedio de un conjunto de n números

Sea n un conjunto de números, los cuales son leídos, diseñe un algoritmo que calcule el promedio de los n números, verificando que $n \geq 1$ y mostrando el número que va a participar en el promedio.

Análisis: Suponga $n = 5$ el conjunto de números a los cuales se les va a calcular el promedio. Si los números dados son 1, 0, 2, 1, y 6, entonces el valor de la suma de los número es igual a 10; luego, el valor del promedio del número de números leídos es $10/5 = 2$.

```
Proc: Cálculo del promedio de un conjunto de números
|  Entero: n, i
|  Real: Suma, num, Promedio
|  Lea n  // Leer el número de números a los cuales se le va
|  // a calcular el promedio
|  Suma ← 0  // Iniciar el numerador del promedio, en el módulo
|  // de la suma
|  // Ciclo repetitivo que permite leer los n números y acumular su suma
|  Para(i = 1, n, 1)Haga
|  |  Lea num  // Número que va participar en el promedio
|  |  Escriba "El número que participa en el promedio es ", num
|  |  Suma ← Suma + num
|  Fin_Para
|  // Cálculo del promedio de los n números
|  Promedio ← Suma/n
|  Escriba "El valor del promedio es igual a ", Promedio
Fin_proc
```

Ejemplo 3.46 Impresión de las tablas de multiplicación y división del número n hasta k

Dados dos números n y k, pertenecientes a los enteros positivos, diseñe un algoritmo que imprima la tabla de multiplicar del número n hasta el número

k, y simultáneamente imprima la tabla de la división de la multiplicación del número n empezando desde k hasta 1, de acuerdo con la siguiente imagen de salida (obviar la verificación de que n y k pertenecen a los enteros positivos):

$$\begin{array}{ll} 5 \times 1 = 5 & 60/12 = 5 \\ 5 \times 2 = 10 & 55/11 = 5 \\ 5 \times 3 = 15 & 50/10 = 5 \\ 5 \times 4 = 20 & 45/9 = 5 \\ 5 \times 5 = 25 & 40/8 = 5 \\ 5 \times 6 = 30 & 35/7 = 5 \\ 5 \times 7 = 35 & 30/6 = 5 \\ 5 \times 8 = 40 & 25/5 = 5 \\ 5 \times 9 = 45 & 20/4 = 5 \\ 5 \times 10 = 50 & 15/3 = 5 \\ 5 \times 11 = 55 & 10/2 = 5 \\ 5 \times 12 = 60 & 5/1 = 5 \end{array}$$

Análisis: Suponga que va a generar la tabla de la multiplicación del número $n = 5$ hasta el valor de $k = 12$; y simultáneamente se requiere la tabla de la división de la multiplicación de $k * n$, o sea, $12 * 5 = 60$, pero su resultado se divide entre k. Se nota que la tabla de la multiplicación va incrementando el valor por el cual se multiplica n, y la tabla de la división tiene incrementos negativos desde el valor de k hasta el valor de 1.

Proc: Tabla de multiplicación y división
  Entero: n, k, j, i
  Lea n, k
  $j \leftarrow k$ // Valor temporal para controlar la tabla de la división
  Para$(i = 1, k, 1)$**Haga**
    $m \leftarrow n * i$ // Valor de la multiplicación
    $r \leftarrow n * j$ // Valor resultado, base del numerador de la tabla de división
    $d \leftarrow r/j$ // Resultado de la división
    Escriba n, " * ", i, " = ", m
    Escriba r, " / ", j, " = ", d
    $j \leftarrow j - 1$ // Se actualiza el valor del contador j
  Fin_Para
Fin_proc

Ejemplo 3.47 Simulador de un contador digital

Diseñe un algoritmo para simular un contador digital de unidades (U), decenas (D) y centenas (C) desde el conteo 000 hasta 999, de tal manera que cada vez que se alcance el dígito 9 en las unidades se avance al siguiente dígito en las decenas; y cada vez que se alcance el número 9 en las decenas se avance un dígito más en las centenas hasta completar el tope del contador.

Análisis: El simulador del contador parte desde un punto inicial 000 y llega hasta un punto final 999, límite en el cual se han cubierto todos los dígitos de las unidades, de las decenas y de las centenas. Luego inicio implica para el contador que unidades, decenas y centenas inician en ceros 000, luego incrementamos el contador de unidades 001, 002, 003, ..., 009; cuando el contador de unidades llega a 9, es necesario sumar uno a las decenas o sea 010 y seguir contando en las unidades 011, 012, 013, ..., 099; ahora, cuando las decenas lleguen a 9, se necesita aumentar las centenas e iniciar nuevamente las unidades 101, y así sucesivamente.

Proc: Simulador de contador digital compuesto por unidades, decenas y centenas

```
Entero: c, d, u
Para(c = 0, 9, 1)Haga
    Para(d = 0, 9, 1, )Haga
        Para(u = 0, 9, 1)Haga
            Escriba "Centenas: ", c
            Escriba "Decenas: ", d
            Escriba "Unidades: ", u
        Fin_Para
    Fin_Para
Fin_Para
Fin_proc
```

Ejemplo 3.48 Cálculo de la varianza muestral de un conjunto de datos

Construya un algoritmo que calcule la varianza muestral de un conjunto de n datos, los cuales son leídos.

Análisis: La ecuación de la varianza muestral está representada por

$$S_x^2 = \frac{\sum_{i=1}^{n} x_i^2}{n} - \overline{x}^2$$

con

$$\overline{x} = \frac{\sum_{i=1}^{n} x_i}{n}.$$

Luego, *i*) se debe leer el valor del número de datos de entrada (n) para el cual se va a calcular la varianza muestral; *ii*) posteriormente se calcula la suma al cuadrado del número de datos leídos y se divide por el número de datos; *iii*) Adicionalmente se tiene que calcular el promedio de los datos ($\overline{x}$, lo cual se lee x barra o x trazo), ello se eleva al cuadrado y se resta del resultado de la operación de *ii*).

Proc: Cálculo de la varianza muestral
Entero: n, i
Real: $Suma1$, $Suma2$, x, P, PSC, V
Lea n
$Suma1 \leftarrow 0$ // Suma de los datos elevados al cuadrado Xi * Xi
$Suma2 \leftarrow 0$ // Suma de los datos
// Ciclo repetitivo que calcula la varianza muestral
Para$(i = 1, n, 1)$**Haga**
  Lea x // Leer el dato que va a participar en la varianza muestral
  $Suma1 \leftarrow Suma1 + x * x$
  $Suma2 \leftarrow Suma2 + x$
Fin_Para
$P \leftarrow Suma2/n$
$PSC \leftarrow Suma1/n$
// PSC corresponde al promedio de la suma de los cuadrados
$V \leftarrow \sqrt{PSC - P * P}$
// V contiene la varianza
Escriba "El valor de la varianza muestral es ", V
Fin_proc

Ejemplo 3.49 Liquidador de nómina

Diseñe un algoritmo representativo de un sistema liquidador de nómina con las siguientes características: una organización requiere liquidar la nómina de un conjunto de T trabajadores, de los cuales se tiene el NIT del funcionario (nit) y su nombre (nom). Cada trabajador tiene por mes un básico (b) y tiene un número de horas extras diurnas (hed), horas extras nocturnas (hen) y un valor de recargo nocturno (vrn); igualmente, el trabajador tiene horas extras diurnas festivas ($hedf$), horas extras nocturnas festivas ($henf$) y el recargo en días festivos (vrf). Los valores de las horas: diurna (vhd), nocturna (vhn), festiva diurna ($vhfd$) y nocturna ($vhfn$) se deben leer, y son los mismos valores para todos los trabajadores. Adicionalmente, los trabajadores tienen un conjunto de deducciones: deducción por salud (des), deducción por pensión (dep) y deducción por solidaridad (del), siempre y cuando el trabajador gane más de cuatro salarios mínimos legales (SML) sobre el valor básico de ganancia. Teniendo en cuenta que el salario mínimo legal se debe leer (sml), el algoritmo liquidador de nómina debe calcular los devengados o ganancias reales (g), los deducidos (d) y el neto a pagar (p), considerando que el recargo nocturno y en días festivos es del 10 % del valor liquidado en horas extras.

Análisis: La liquidación de nómina para un NIT ejemplo se muestra a continuación:

- $nit = 10$, $nom = A$, $b = 1000000$

- $hed = 10$, $vhd = 1$, $hen = 0$
- $vrn = 10\,\%$
- $hedf = 0$, $henf = 0$
- $vrf = 10\,\%$
- $des = 20000$, $dep = 10000$, $del = 0$
- $p = 0$

Para cada trabajador el neto a pagar (p) es igual a devengados (g) menos deducidos (d). Luego $p = g - d$. Las ganancias reales o los valores devengados en el caso del trabajador son iguales al básico más los valores devengados en las horas extras (e), es decir, $g = b + e$. El valor de las horas extras son iguales a

$$e = (hed * vhd) + (hen * vhn) + (hedf * vhfd) + (hefn * vhfn);$$

y el valor final de las horas extras es $e = e + 0.1 * e$.

Por otra parte, el valor de los deducidos es $d = des + dep + del$. Suponiendo que el valor de la hora diurna es una unidad, entonces el valor neto para el ejemplo es

$$p = 1000000 + 10 * 1 + 10 * 1 * 0.1 - (20000 + 10000) = 970011.$$

Proc: Liquidador de nómina
| **Entero:** t, vhd, vhn, $vhfd$, $vhfn$, i, b, hed, hen, $hedf$, $henf$
| **Real:** sml, del, des, dep, e, g, d, p
| **Cadena:** nit, nom
| **Lea** t // Leer el número de trabajadores para quienes se les
| // liquidará la nómina
| **Lea** $vhd, vhn, vhfd, vhfn$ // Leer los valores de las horas extras
| **Lea** sml // Leer el salario mínimo legal - SML
| **Lea** del // Leer el descuente a aplicar por solidaridad
| // Ciclo repetitivo que calcula la nómina para cada trabajador
| **Para**$(i = 1, t, 1)$**Haga**
| | **Lea** nit, nom // Leer el nit y el nombre del trabajador
| | **Lea** $b, hed, hen, hedf, henf$ // Leer el básico y número de horas extras
| | **Lea** des, dep // Leer deducciones fijas salud y pensión
| | // Cálculo extras
| | $e \leftarrow (hed * vhd) + (hen * vhn) + (hedf * vhfd) + (henf * vhfn)$
| | $e \leftarrow e * 0.1$ // Recargo de horas extras
| | $g \leftarrow b + e$ // Ganancias del trabajador
| | **Si**$(b > 4 * sml)$**Entonces**
| | | $d \leftarrow des + dep + del$ // Sumar el descuento de solidaridad
1 2 3

```
1 2 3
|  |  Sino
|  |  |  d ← des + dep // No se suma el descuesto de solidaridad
|  |  F.Si
|  |  p ← g − d // Neto a pagar para el trabajador i
|  |  Escriba "Valor de la nómina para el trabajador ", nit
|  |  Escriba "Nombre: ", nom
|  |  Escriba "Devengados: " g
|  |  Escriba "Deducidos: ", d
|  |  Escriba "Neto (Devengados - Deducidos) a pagar: ", p
|  Fin_Para
Fin_proc
```

Es importante tener en cuenta en el desarrollo del liquidador de nómina que los datos leídos en el ciclo Para son variables temporales que se van actualizando dentro del ciclo; luego, es necesario desarrollar el concepto de estructuras de datos, con el fin de que siendo almacenadas las variables temporales en dichas estructuras, se pueda tener la totalidad de los datos de la empresa a efectos de chequeo y verificación de los totales de una nómina mensual contra los valores de una entidad bancaria. Tópico de estructuras de datos que será desarrollado en el siguiente capítulo.

Ejemplo 3.50 Coeficiente de Correlación

Dada una clase de Algoritmia y Programación, la cual tiene n estudiantes, para cada estudiante se leen dos variables: X_i, definida como el coeficiente intelectual, y Y_i, identificada como rendimiento académico. Diseñe un algoritmo para calcular el Coeficiente de Correlación (R_{xy}), que es un índice que mide la relación lineal entre dos variables cuantitativas y está definido por la expresión

$$R_{xy} = \frac{\dfrac{\sum_{i=1}^{n} x_i y_i}{n} - \overline{xy}}{S_x S_y}.$$

Los valores de S_x y S_y están definidos por las expresiones

$$S_x = \sqrt{\frac{\sum_{i=1}^{n} x_i x_i}{n} - \overline{x}^2}$$

donde el promedio está definido como

$$\overline{x} = \frac{\sum_{i=1}^{n} x_i}{n}.$$

Análogamente,

$$S_y = \sqrt{\frac{\sum_{i=1}^{n} y_i y_i}{n} - \overline{y}^2}, \quad \overline{y} = \frac{\sum_{i=1}^{n} y_i}{n}.$$

Análisis: En primer lugar se lee el número de estudiantes del curso (n); en segundo lugar se lee para cada estudiante los datos de las variables del coeficiente intelectual (x_i) y del rendimiento académico (y_i). Seguidamente se va calculando las sumatorias requeridas de los promedios, las dos desviaciones estándar y el coeficiente de correlación; para finalmente construir la expresión matemática del coeficiente.

Proc: Coeficiente de correlación
Entero: n, i
Real: $sumac$, $sumar$, $sumapc$, $sumapr$, $sumapcr$, x, y, $promc$, $promc$
Real: $promr$, $prom$, $prompc$, $prompr$, $prompcr$, $desvc$, $desvr$, C
Lea n // Leer el número de estudiantes del curso

$sumac \leftarrow 0$ // Suma del coeficiente intelectual
$sumar \leftarrow 0$ // Suma del rendimiento académico
$sumapc \leftarrow 0$ // Suma del producto de coeficiente intelectual
$sumapr \leftarrow 0$ // Suma del producto del rendimiento académico
$sumapcr \leftarrow 0$ // Suma del producto del coeficiente por
// el rendimiento

// Ciclo repetitivo para calcular las sumatorias del coeficiente
Para$(i = 1, n, 1)$**Haga**
  Lea x // Leer el coeficiente intelectual asociado al alumno i
  Lea y // Leer el rendimiento académico asociado al alumno i

  $sumac \leftarrow sumac + x$
  $sumar \leftarrow sumar + y$
  $sumapc \leftarrow sumapc + x * x$
  $sumapr \leftarrow sumapr + y * y$
  $sumapcr \leftarrow sumapcr + x * y$
Fin_Para

// Cálculo de promedios
$promc \leftarrow sumac/n$
$promr \leftarrow sumar/n$
$prompc \leftarrow sumapc/n$
$prompr \leftarrow sumapr/n$
$prompcr \leftarrow sumapcr/n$

1

```
1
   // Raiz es la función matemática de la raíz cuadrada
   desvc ← √(prompc − (promc ∗ promc))
   desvr ← √(prompr − (promr ∗ promr))

   // Cálculo del coeficiente R_xy
   C ← ((prompcr) − (promc ∗ promr))/(desvc ∗ desvr)
   Escriba "El valor del coeficiente de correlación es", C
Fin_proc
```

3.7 Estructura lógica repetitiva Mientras que

La lógica de control de la estructura repetitiva Mientras que, cuya identidad en el algoritmo se abrevia con la sigla Mq, permite la realización de un conjunto de estructuras lógicas básicas o primitivas lógicas (p_i), las cuales se ejecutan cuando la condición del Mientras que es verdadera.

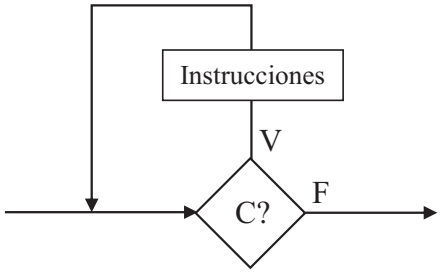


Figura 3.7 Estructura Repetitiva Mientras que

Implica, por lo tanto, en la estructura Mientras que el cumplimiento cuando sea verdadera de una condición, lo cual en la notación de algoritmos se expresa como:

Dado un conjunto de primitivas p_1, p_2, p_3, ..., p_i, ..., p_n, la repetitiva Mq se denota en el algoritmo como

Mientras_que (condición) **haga**

$p_1, p_2, \ldots, p_n$

Fin_Mientras_que

O, en su defecto, también se puede presentar como

Mq (condición) **haga**

$p_1, p_2, \ldots, p_n$

Fin_Mq

Esta estructura de control de repetición de otras estructuras algorítmicas permite la ejecución de las primitivas p_1, p_2, p_3, ..., p_n, siempre y cuando la condición sea verdadera; condición que "... es necesario que no se quede ejecutando indefinidamente..." (Rueda, 1981, p. 21), por cuanto el algoritmo, utilizando la estructura Mientras que, necesariamente debe terminar, a fin de dar los resultados al usuario.

Ejemplo 3.51 Descomposición de un número k en sus dígitos

Dado un número k que pertenece a los enteros positivos, diseñe un algoritmo para descomponer el número en sus dígitos decimales de unidades, decenas, centenas, etc. Si el número es 789, el resultado será 9 (unidades), 8 (decenas) y 7 centenas.

Análisis: Se guarda el número original (k) en una variable temporal ($k1$). Suponga que el número original es $k = k1 = 789$, este número en el sistema de numeración digital es igual a $7*10^2+8*10^1+9*10^0$; entonces al número le sacamos su residuo utilizando la función Módulo 10 (789 **mod** 10 es 9); el residuo de la aplicación de la función módulo 10 es el dígito posicionado en el lugar de las unidades (1), decenas (2), centenas (3), etc., cuantas veces hayamos aplicado la función módulo sobre la variable temporal ($k1$) ; luego actualizamos la variable temporal, dividiéndola por 10, haciendo el proceso siempre que el cociente sea mayor que cero.

```
Proc: Descomposición de un k en sus dígitos
  Entero: k, ki, j, Residuo
  Lea k // Leer el número a descomponer
  k1 ← k // Se almacena el número a descomponer en una variable
  // temporal
  j ← 0
  // j contiene el exponente de posicionamiento de la descomposición
  Mq(k1 > 0)Haga
    // Se calcula el residuo, utilizando la función Módulo 10
    Residuo ← k1 mod 10
    Escriba Residuo, "*10^", j
    // El residuo es el dígito ya descompuesto de k
    k1 ← k1/10 // Se actualiza la variable temporal
    j ← j + 1 // Actualiza el exponente de posicionamiento del residuo
  Fin_Mq
Fin_proc
```

Cabe resaltar que la división $k1/10$ es entera, dado que $k1$ y 10 son variables enteras. Por ejemplo, si $k1 = 123$, entonces $k1/10 = 123/10 = 12$.

Ejemplo 3.52 Función $\log_2 n$ para potencias de 2

Dado un número n, con las características de que sea par, potencia de dos y perteneciente a los enteros positivos, diseñe un algoritmo utilizando la estructura Mientras que para calcular el número de potencia del número n.

Análisis: Como el número es par y potencia de dos, suponga $n = 8$. Si a este número lo dividimos sucesivamente por 2 hasta 1, encontramos en la división las potencia de n, de la siguiente forma: $8/2 = 4$; $4/2 = 2$; $2/2 = 1$; luego, se realizaron tres divisiones; por lo tanto, $2^3 = 8$, con lo cual se concluye que el número de potencias de 8 son tres.

Proc: Función $\log_2 n$ para potencias de 2
| **Entero:** n, $n1$, p
| **Lea** n
| $n1 \leftarrow n$ // Guardar el número que se va a descomponer en una variable
| // temporal $n1$
| $p \leftarrow 0$ // Inicializar el contador de potencia de dos en cero
| **Mq**$(n1 \neq 1)$**Haga**
| | $n1 \leftarrow n1/2$ // División entera que parte cada vez el número en dos
| | $p \leftarrow p + 1$ // Actualizar el contador de potencias
| **Fin_Mq**
| **Escriba** "El logaritmo en base 2 de ", n
| **Escriba** "Es igual a ", p
Fin_proc

Ejemplo 3.53 Impresión de series alternantes de números y sus sumas

Utilizando la estructura de control repetitiva Mientras que (Mq) diseñe un algoritmo para calcular e imprimir los k primeros términos de la serie $S1 = 12 - 11 + 24 - 22 + 36 - 33 + 48 - 44 \ldots$ y adicionalmente los n primeros términos de la serie $S_2 = 10 - 20 + 30 - 40 \ldots$. Escribiendo el resultado de la suma de cada una de las series.

Análisis: La primera serie corresponde a una sucesión alternante (+ y −) de los múltiplos de 12 y 11; por otra parte, la segunda sucesión también es alternante pero de los múltiplos de 10.

Proc: Impresión de series oscilantes
| **Entero:** k, $S1$, $S2$, i, j, $T1$, $T2$, c
| **Lea** k
| $S1 \leftarrow 0$ // Inicializa la primera serie
| $S2 \leftarrow 0$ // Inicializa la segunda serie
| // Estructura repetitiva que permite el cálculo de la primera serie
1

```
1
  i ← 1 // Controla el ciclo Mq con el número de términos de la serie
  j ← 1 // Coeficiente base para calcular cada término
  Mq(i ≤ k)Haga
    T1 ← j * 12 // Calcula el término positivo
    T2 ← j * 11 // Calcula el término negativo
    S1 ← S1 + T1 − T2 // Se le suma los términos calculados
    j ← j + 1 // Coeficiente base para el otro término
    i ← i + 2 // Se incrementa i para alcanzar el valor k
  Fin_Mq
  // Vea que el ciclo anterior asigna de 2 en 2
  // A continuación se tiene en cuenta si k es un número impar
  Si(k mod 2 = 1)Entonces
    T1 ← c * j * 12
    S1 ← S1 + T1
  F.Si

  j ← 1 // Controla el segundo Mq, que calcula la serie dos
  c ← 1 // Signo del coeficiente del término de la serie
  Mq(j ≤ k)Haga
    T ← c * j * 10 // Calcula el término de la segunda serie
    S2 ← S2 + T // Calcula la suma de la serie dos
    c ← c * (−1) // Varia el signo como base para calcular otro término
    j ← j + 1 // Actualiza el contador de términos para alcanzar n
  Fin_Mq
  Escriba "El valor de las serie alternante uno es ", S1
  Escriba "El valor de las serie alternante uno es ", S2
Fin_proc
```

Ejemplo 3.54 Multiplicación rusa

Dados dos números n y m, identificados como multiplicando (n) y multiplicador (m), el algoritmo de la multiplicación rusa divide por dos el multiplicador hasta que su valor es 1 y multiplica por dos el multiplicando; adicionalmente, para obtener el resultado de la multiplicación suma todos los multiplicandos correspondientes a los multiplicadores impares, y esta suma es el resultado de la multiplicación. Diseñe un algoritmo para multiplicar dos números dados utilizando la multiplicación rusa.

Análisis: Sea $n = 12$ el multiplicando; y sea $m = 10$ el multiplicador. Entonces, la multiplicación normal da como resultado $12 \times 10 = 120$ y la multiplicación rusa utiliza el siguiente procedimiento:

Multiplicando			Multiplicador		
n	$=$	12	m	$=$	10
12×2	$=$	**24**	$10/2$	$=$	**5**
24×2	$=$	48	$5/2$	$=$	2 (se utiliza la división entera)
48×2	$=$	**96**	$2/2$	$=$	**1**

El resultado de la multiplicación es entonces $24 + 96 = 120$ (porque son los multiplicandos de los multiplicadores que dan resultados impares).

Proc: Multiplicación rusa
Entero: n, m, $Suma$, Mul, Dor
Lea n, m
$Suma \leftarrow 0$ // Acumula el valor de la multiplicación rusa
// Estructura repetitiva que permite el control de las multiplicaciones
// y las divisiones al igual que va sumando los multiplicandos que
// corresponden a los multiplicadores impares
$Mul \leftarrow n$ // Variable temporal correspondientes al
// Multiplicando y sus valores
$Dor \leftarrow m$ // Variable temporal que almacena el multiplicador
// y sus valores
Mq($Dor \geq 1$)**Haga**
  // Comprueba si el multiplicador es impar
  Si(Dor **mod** $2 = 1$)**Entonces**
    // Adiciona el multiplicando por 2 al resultado
    $Suma \leftarrow Suma + Mul$
  F.Si
  $Mul \leftarrow Mul * 2$
  $Dor \leftarrow Dor/2$
Fin_Mq
Escriba "El valor de la multiplicación rusa de ", n
Escriba "por ", m, " es ", $Suma$
Fin_proc

Ejemplo 3.55 Juego: Adivinar números entre dos jugadores: A y B

Dados dos jugadores, A y B, suponga que el jugador A tiene n números cuyos rangos están ubicados en el intervalo de los enteros positivos de $[1, k]$. El jugador B debe adivinar el número que el jugador A tiene en mente. Diseñe un algoritmo que lea los n números del jugador A y dé la posibilidad al jugador B de dar un número, el cual es leído hasta que coincida con el que tiene el jugador A. El algoritmo debe contar el número de veces en que el jugador B utilizó para adivinar el número del jugador A, al igual que calcular el promedio del número de veces que empleó el jugador B en adivinar todos los números del jugador A.

```
Proc: Juego de adivinanzas
| Entero: n, k, i, Sp, nA, sw, cont, nB
| Real: p
| Lea n, k
| i ← 1 // Indice que permite controlar el número de números
| // propuestos por A
| Sp ← 0 // Variable que se va a utilizar para sacar el promedio
| Mq(i ≤ n)Haga
| | Lea nA // Número dado por el jugador A
| | Si((nA ≥ 1) y (nA ≤ k))Entonces
| | | sw ← 0 // Controla si el número B adivina el número de A
| | | cont ← 0 // Intentos que emplea B en adivinar el número
| | | Mq(sw = 0)Haga
| | | | Lea nB // Número de B
| | | | cont ← cont + 1
| | | | Si(nA = nB)Entonces
| | | | | // El jugador B ha adivinado el número de A
| | | | | sw ← 1 // Se varía el switch de coincidencia
| | | | | // Esta variación obliga a que se cierre el ciclo Mq interno
| | | | F.Si
| | | | Escriba "Intentos: ", cont
| | | | Sp ← Sp + cont
| | | | i ← i + 1
| | | Fin_Mq
| | Sino
| | | Escriba "El número de A no está en el intervalo [1,", k, "]"
| | F.Si
| Fin_Mq
| // Cálculo del promedio
| p ← Sp/n
| Escriba "El número promedio de intentos que empleó B para adivinar"
| Escriba "los n números de A es igual a ", p
Fin_proc
```

3.8 Estructura lógica repetitiva Haga Hasta

La estructura de control repetitiva Haga Hasta permite la ejecución de un conjunto de estructuras de control o primitivas p_1, p_2, p_3, ..., p_i, ..., p_n, hasta que se cumpla la condición explicitada en la clausula de control del Haga Hasta.

Las primitivas de control contenidas dentro del ciclo repetitivo Haga Hasta se ejecutan hasta que la condición sea verdadera. Lo anterior implica que en esta estructura como mínimo las primitivas de control p_1, p_2, p_3, ..., p_i, ..., p_n, se ejecutan

al menos una vez.

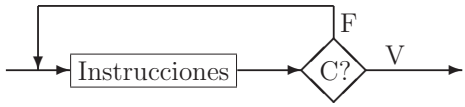


Figura 3.8 Estructura Repetitiva Haga Hasta

El ciclo repetitivo Haga Hasta se abrevia con HH, y su estructura en el algoritmo se denota como

Haga_Hasta

$p_1, p_2, p_3, \ldots, p_n$

Fin_Haga_Hasta(Condición)

Estructura de control que es equivalente en su notación de algoritmos a

HH

$p_1, p_2, p_3, \ldots, p_n$

Fin_HH(Condición)

Las estructuras de control repetitivas Mientras que y Haga Hasta expuestas permiten la ejecución de un conjunto de primitivas algorítmicas; pero en las dos estructuras relacionadas (Mq) y (HH) hay una diferencia fundamental: en la estructura Mientras que la condición de verdad para ejecutar las instrucciones que van en el interior se evalúa antes de ejecutar las primitivas; luego, es posible que el conjunto de primitivas en el interior del ciclo (Mq) no se ejecute ni una sola vez si la condición no se cumple. En el caso de la primitiva Haga Hasta, las instrucciones contenidas dentro del ciclo repetitivo (HH) se ejecutan al menos una vez, y después sí se evalúa la condición que controla la repetición del Haga Hasta.

Ejemplo 3.56 Verificador digital

Un sistema de codificación digital considera como códigos válidos los situados en el intervalo cerrado $[10'000\,000, 99'999\,999]$; los códigos ubicados en rangos diferentes al intervalo dado son considerados errados. Diseñe un algoritmo que haga el proceso de verificación de contar e imprimir códigos válidos, terminando el proceso cuando ingresen un código ubicado en un rango diferente al rango dado. (Los algoritmos de verificación son útiles para garantizar la entrada de NIT (cédulas) a un sistema de información).

Análisis: Suponga los códigos de entrada $10'000\,000$, $50'500\,500$, $99'784\,777$, 9999, entonces los tres primeros códigos dados son válidos, y en tal sentido, el sistema verificador capturaría solo tres códigos válidos. Ahora, el código 9999 no es válido porque su valor es inferior a $10'000\,000$.

```
Proc: Verificador digital
| Entero: cont, cod
| cont ← 0 // Cuenta el número de códigos digitales válidos en el rango
| // dado en el intervalo [10′000 000, 99′999 999]
| // Ciclo repetitivo que lee, cuenta e imprime los códigos
| HH
| | Lea cod // Código que se debe verificar si está en el rango dado
| | Si(((cod ≥ 10000000) y (cod ≤ 99999999))Entonces
| | | Escriba "Código válido"
| | | cont ← cont + 1
| | Sino
| | | Escriba "El código digitado NO es válido"
| | | Escriba "Fin del proceso de verificación del código ", cod
| | F.Si
| Fin_HH(((cod < 10000000) o (cod > 99999999))
| Escriba "Número de códigos válidos = ", cont
Fin_proc
```

Ejemplo 3.57 Controlador de tanques

Un sistema de tanques para riego está compuesto de un tanque principal (Tp) y tres tanques secundarios ($T1$, $T2$, y $T3$). El tanque principal se llena a través de una tubería madre con un flujo de entrada de k litros por segundo, y a su vez alimenta a los tanques secundarios dispuestos en secuencia para su llenado en el orden $T1 \rightarrow T2 \rightarrow T3$. La velocidad de entrada del tanque principal al tanque uno es de m litros de agua por segundo, del tanque uno al tanque dos es de n litros por segundo y del tanque dos al tres de p litros por segundo. Los tanques secundarios tienen tres flujos de salida: el primero, un flujo de riego de z litros por segundo; el segundo, un flujo de riego de a litros por segundo; y el tercero un caudal de riego de t litros/segundo. Los flujos de riego y alimentación de los tanques secundarios operan únicamente hasta que cada uno de ellos llegue a un determinado nivel de reserva. Los niveles de reserva de los tres tanques son: nivel de reserva del tanque uno ($N1$), nivel de reserva del tanque dos ($N2$) y nivel de reserva del tanque tres ($N3$); niveles de reserva que deben ser leídos como datos de entrada. Teniendo en cuenta que $k \gg m \gg n \gg p \gg z \gg a \gg t$, y que cada uno de los tanques tiene un control que identifica el tope o nivel límite del tanque, con el fin de que el agua no se desperdicie, diseñe un algoritmo que controle el sistema de tanques.

```
Proc: Controlador de tanques
| Real: N1, N2, N3, k, m, n, p, z, a, t, Vp, V1, V2, V3
| Entero: SwTp, Sw1, Sw2, Sw3
| Lea N1, N2, N3 // Leer los niveles de reserva de los tanques
| Lea k, m, n, p // Leer los flujos de entrada de los tanque en
1
```

```
1
| // litros por segundo
| Lea z, a, t // Caudal de riego de los tanques T1(z), T2(a) y T3(t)
| // Volúmenes de tanques principal y secundarios vacíos
| Vp ← 0, V1 ← 0, V2 ← 0, V3 ← 0
| SwTp ← 0 // Switch que controla el llenado del tanque principal
| // Controladores del llenado de los tres tanques
| Sw1 ← 0, Sw2 ← 0, Sw3 ← 0
| // Ciclo repetitivo que controla el nivel del tanque principal
| HH
| | Vp ← Vp + k // Volumen del tanque principal con carga de entrada k
| | HH
| | | Vp ← Vp − m // Reducción del volumen del tanque principal
| | | V1 ← V1 + m // Volumen del tanque secundario uno
| | | Si(V1 > N1 + z)Entonces
| | | | V1 ← V1 − z // Volumen de tanque mayor que nivel de reserva
| | | | HH
| | | | | V1 ← V1 − n
| | | | | V2 ← V2 + n
| | | | | Si(V2 > N2 + a)Entonces
| | | | | | V2 ← V2 − a
| | | | | | HH
| | | | | | | V2 ← V2 − p
| | | | | | | V3 ← V3 + p
| | | | | | | Si(V3 > N3 + t)Entonces
| | | | | | | | V3 ← V3 − t
| | | | | | | Sino
| | | | | | | | Escriba "Tanque 3 sin reserva"
| | | | | | | F.Si
| | | | | | | Lea Sw3
| | | | | | Fin_HH(Sw3 = 1)
| | | | | Sino
| | | | | | Escriba "Tanque 2 sin reserva"
| | | | | F.Si
| | | | | Lea Sw2
| | | | Fin_HH(Sw2 = 1)
| | | Sino
| | | | Escriba "Tanque 1 sin reserva"
| | | F.Si
| | | Lea Sw1
| | Fin_HH(Sw1 = 1)
| | Lea SwTp
| Fin_HH(SwTp = 1)
Fin_proc
```

Ejemplo 3.58 Evaluador bancario de empleados

Un sistema bancario atiende a sus clientes a nivel personalizado a través de tres empleados que desempeñan las funciones de cajeros manuales. El banco está interesado en conocer el nivel de atención de sus empleados para con los clientes, para lo cual requiere la siguiente información:

i) Número total de clientes atendidos por el banco durante una jornada horaria.

ii) Número total de clientes atendidos por cada cajero.

iii) Monto total de transacciones por cada cajero.

iv) Promedio de transacciones de cada cajero.

v) El promedio de transacciones del banco.

vi) El cajero que realizó el mayor número de transacciones y su monto.

vii) El cajero que realizó el menor número de transacciones y su monto.

Teniendo en cuenta que *i*) un cliente puede realizar más de una transacción ante el cajero; *ii*) el cliente puede decidir pasar de un cajero a otro, con la condición de que solo consulta cada cajero manual una sola vez; *iii*) las operaciones bancarias se realizan hasta que la entidad bancaria se encuentre abierta. Diseñe un algoritmo representativo del evaluador bancario.

Análisis: El algoritmo de control del sistema de evaluación de empleados del sector bancario requiere tres procesos principales: *i*) el proceso de inicialización de variables; *ii*) el proceso de control bancario de atención a los clientes; *iii*) el proceso de arqueo o consolidación estadística de las transacciones bancarias.

En el proceso de inicialización de variables se colocan en cero los valores representativos de las estadísticas bancarias y los *flags* o *switchs* que apoyan el proceso del control del banco. La inicialización de las estadísticas bancarias requeridas son: los contadores de clientes atendidos por cada cajero ($c1$, $c2$ y $c3$), el monto de las transacciones recibidas por cada cajero ($m1$, $m2$ y $m3$), los contadores del número de transacciones por cajero ($ct1$, $ct2$ y $ct3$) y la suma base para el promedio de transacciones totales del banco st. Adicionalmente se requiere el switch que controla la apertura del banco (SwA) y el switch que controla las transacciones de los clientes (SwT).

El proceso de control bancario, previamente habiendo inicializado los switches mencionados, es controlado algorítmicamente por dos estructuras de control: la primera un Haga Hasta (HH) con control de finalización del estado de apertura del banco (SwA), que en el caso de estar abierta la entidad lee las variables que controlan los cajeros del tipos uno ($t1$), dos ($t2$) y tres ($t3$); la segunda también es una estructura de control Haga Hasta (HH) que en función del valor del switch de transacciones del cliente (SwT) sigue o no

realizando transacciones; la última estructura (HH) se encarga de acumular las estadísticas requeridas para cada cajero en términos del monto y del número de transacciones por cajero.

Finalmente, el arqueo estadístico por cajero y el arqueo total del banco son instrucciones de control de sumas y promedios particulares a cada cajero y al banco en los totales de transacciones diarias.

```
Proc: Evaluador bancario de empleados
| Entero: SwA, c1, c2, c3, m1, m2, m3, ct1, ct2, ct3, st, cc, tc, t1
| Entero: SwT, mtc, t2, t3, Mayor, Mmayor, Menor, Mmenor
| Real: ptc1, ptc2, ptc3, ptb
| // Switch que controla si el banco está abierto
| SwA ← 1
| // Contadores de cliente atendidos por cada cajero
| c1 ← 0, c2 ← 0, c3 ← 0
| // Monto de transacciones realizadas por cajero
| m1 ← 0, m2 ← 0, m3 ← 0
| // Contador de transacciones por cajero
| ct1 ← 0, ct2 ← 0, ct3 ← 0
| st ← 0 // Base para el promedio de totales del banco
| cc ← 0 // Contador de clientes totales atendidos
| tc ← 0 // Total de clientes atendidos en una jornada
|
| HH
| | Lea t1 // Variable que controla el cajero C1 que atiende al cliente
| | Si(t1 = 1)Entonces
| | | SwT ← 1 // Controla las transacciones del cliente
| | | c1 ← c1 + 1 // Cliente atendidos por el cajero uno
| | | HH
| | | | Lea mtc // Monto de la transacción del cliente
| | | | m1 ← m1 + mtc // Monto total del cajero uno
| | | | st ← st + mtc // Suma total de transacciones
| | | | ct1 ← ct1 + 1 // Actualiza el total de transacciones
| | | | Lea SwT
| | | Fin_HH(SwT = 0)
| | F.Si
| | Lea t2 // Variable que controla el cajero C2 que atiende al cliente
| | Si(t2 = 2)Entonces
| | | SwT ← 1
| | | c2 ← c2 + 1 // Cliente atendidos por el cajero dos
| | | HH
| | | | Lea mtc
| | | | m2 ← m2 + mtc
1 2 3 4
```

```
1 2 3 4
|  |  |  |  st ← st + mtc
|  |  |  |  ct2 ← ct2 + 1
|  |  |  |  Lea SwT
|  |  |  Fin_HH(SwT = 0)
|  |  F.Si
|  |  // Variable que controla el cajero C3 que atiende al cliente
|  |  Lea t3
|  |  Si(t3 = 3)Entonces
|  |  |  SwT ← 1
|  |  |  c3 ← c3 + 1
|  |  |  HH
|  |  |  |  Lea mtc
|  |  |  |  m3 ← m3 + mtc
|  |  |  |  st ← st + mtc
|  |  |  |  ct3 ← ct3 + 1
|  |  |  |  Lea SwT
|  |  |  Fin_HH(SwT = 0)
|  |  F.Si
|  |  Lea SwA
|  Fin_HH(SwA = 0)
|  // Cálculo de totales
|  // Total de clientes atendidos en el banco en una jornada
|  tc ← tc + c1 + c2 + c3
|  Escriba "El total de clientes atendidos en una jornada diaria es ", tc
|  // Número total de clientes atendidos por cada cajero
|  Escriba "El número total de cliente atendidos por C1 es ", c1
|  Escriba "El número total de cliente atendidos por C2 es ", c2
|  Escriba "El número total de cliente atendidos por C3 es ", c3
|  // Monto total de transacciones por cada cajero
|  Escriba "El monto total de transacciones realizadas por C1 es ", m1
|  Escriba "El monto total de transacciones realizadas por C2 es ", m2
|  Escriba "El monto total de transacciones realizadas por C3 es ", m3
|  // Promedio de transacciones de cada cajero
|  ptc1 ← m1/ct1, ptc2 ← m2/ct2, ptc3 ← m3/ct3
|  Escriba "El promedio de transacciones realizadas por C1 es ", ptc1
|  Escriba "El promedio de transacciones realizadas por C2 es ", ptc2
|  Escriba "El promedio de transacciones realizadas por C3 es ", ptc3
|  // El promedio de transacciones del banco
|  ptb ← st/(ct1 + ct2 + ct3)
|  Escriba "El promedio diario de transacciones del banco es ", ptb
|  // Ahora se escoge el cajero que realizó el mayor número de
|  // transacciones y su monto correspondiente
|  Mayor ← ct1
1
```

```
1
| Mmayor ← m1
| Si(ct2 > Mayor)Entonces
| | Mayor ← ct2
| | Mmayor ← m2
| F.Si
| Si(ct3 > Mayor)Entonces
| | Mayor ← ct3
| | Mmayor ← m3
| F.Si
| // Ahora se escoge el cajero que realizó el menor número de
| // transacciones y su monto correspondiente
| Menor ← ct1
| Mmenor ← m1
| Si(ct2 < Menor)Entonces
| | Menor ← ct2
| | Mmenor ← m2
| F.Si
| Si(ct3 < Menor)Entonces
| | Menor ← ct3
| | Mmenor ← m3
| F.Si
| Escriba "Cajero con el mayor de transacciones = ", Mayor
| Escriba "Monto del cajero con el mayor número de transacciones = "
| Escriba Mmayor
| Escriba "Cajero con el menor de transacciones = ", Menor
| Escriba "Monto del cajero con el menor número de transacciones = "
| Escriba Mmenor
Fin_proc
```

Ejemplo 3.59 Controlador de entradas de los alumnos en un Sistema Universitario

Un sistema universitario tiene para el registro de entradas de los estudiantes tres torniquetes. Diseñe un algoritmo que cuente el número de estudiantes diarios que pasan por cada torniquete, llevando una cuenta semanal (7 días de la semana) de los alumnos, como también el número de alumnos por programa que pasan al día por cada torniquete de los programas de la División de Ingeniería: Ingeniería Civil ($p1$), Eléctrica ($p2$), Electrónica ($p3$), Industrial ($p4$), Mecánica ($p5$) e Ingeniería de Sistemas ($p6$).

Análisis: El controlador de entradas de estudiantes para un sistema universitario se fundamenta en tres procesos:

i) La inicialización de las variables que representan las entradas diarias de los alumnos por los torniquetes ($t1$, $t2$ y $t3$) y para los seis programas

y los tres torniquetes ((*p*1*t*1, *p*2*t*1, *p*3*t*1, *p*4*t*1, *p*5*t*1, *p*6*t*1), (*p*1*t*2, *p*2*t*2, *p*3*t*2, *p*4*t*2, *p*5*t*2, *p*6*t*2), (*p*1*t*3, *p*2*t*3, *p*3*t*3, *p*4*t*4, *p*5*t*5)). Adicionalmente se requiere el control de los siete días de la semana.

II) El proceso de control semanal de los tres torniquetes se puede realizar o con una estructura de lógica de control Para o con una estructura de control Mientras que (Mq); dentro de las estructuras antes mencionadas se lee el *switch* de apertura de los torniquetes (SwA), y con un controlador Haga Hasta (HH) se distribuyen los estudiantes que pasan diariamente por cada torniquete, leyendo en el interior del HH tanto el número del torniquete (n) como el código del alumno (c).

III) Finalmente, antes de avanzar al siguiente día de la semana se totalizan los acumuladores semanales de los torniquetes ($t1s$, $t2s$, y $t3s$).

```
Proc: Controlador de entradas en un sistema universitario
| Entero: t1, t2, t3, t1s, t2s, t3s, p1t1, p2t1, p3t1, p4t1, p5t1, p6t1, p1t2
| Entero: p2t2, p3t2, p4t2, p5t2, p6t2, p1t3, p2t3, p3t3, p4t3, p5t3, p6t3
| Entero: i, SwA, n, c
| // Contadores de torniquetes (por día y semana)
| t1 ← 0; t2 ← 0; t3 ← 0
| t1s ← 0; t2s ← 0; t3s ← 0
| // Contadores diarios por programa por torniquete
| p1t1 ← 0; p2t1 ← 0; p3t1 ← 0; p4t1 ← 0; p5t1 ← 0; p6t1 ← 0
| p1t2 ← 0; p2t2 ← 0; p3t2 ← 0; p4t2 ← 0; p5t2 ← 0; p6t2 ← 0
| p1t3 ← 0; p2t3 ← 0; p3t3 ← 0; p4t3 ← 0; p5t3 ← 0; p6t3 ← 0
| i ← 1 // Control semanal
| Mq(i ≤ 7)Haga
| | SwA ← 1 // Switch de apertura de los torniquetes
| | HH
| | | Lea n, c
| | | DD(n)Haga
| | | | Opción 1 :
| | | | | // Se aumenta el contador diario de primer torniquete
| | | | | t1 ← t1 + 1
| | | | | DD(c)Haga
| | | | | | Opción 1 :
| | | | | | | p1t1 ← p1t1 + 1
| | | | | | Fin
| | | | | | Opción 2 :
| | | | | | | p2t1 ← p2t1 + 1
| | | | | | Fin
| | | | | | Opción 3 :
| | | | | | | p3t1 ← p3t1 + 1
| | | | | | Fin
1 2 3 4 5 6
```

```
1 2 3 4 5 6
|  |  |  |  |  |  | Opción 4 :
|  |  |  |  |  |  |  p4t1 ← p4t1 + 1
|  |  |  |  |  |  | Fin
|  |  |  |  |  |  | Opción 5 :
|  |  |  |  |  |  |  p5t1 ← p5t1 + 1
|  |  |  |  |  |  | Fin
|  |  |  |  |  |  | Opción 6 :
|  |  |  |  |  |  |  p6t1 ← p6t1 + 1
|  |  |  |  |  |  | Fin
|  |  |  |  |  | Fin_DD
|  |  |  |  | Fin
|  |  |  |  | Opción 2 :
|  |  |  |  |  // Se aumenta el contador diario del segundo torniquete
|  |  |  |  |  t2 ← t2 + 1
|  |  |  |  |  DD(c)Haga
|  |  |  |  |  | Opción 1 :
|  |  |  |  |  |  p1t2 ← p1t2 + 1
|  |  |  |  |  | Fin
|  |  |  |  |  | Opción 2 :
|  |  |  |  |  |  p2t2 ← p2t2 + 1
|  |  |  |  |  | Fin
|  |  |  |  |  | Opción 3 :
|  |  |  |  |  |  p3t2 ← p3t2 + 1
|  |  |  |  |  | Fin
|  |  |  |  |  | Opción 4 :
|  |  |  |  |  |  p4t2 ← p4t2 + 1
|  |  |  |  |  | Fin
|  |  |  |  |  | Opción 5 :
|  |  |  |  |  |  p5t2 ← p5t2 + 1
|  |  |  |  |  | Fin
|  |  |  |  |  | Opción 6 :
|  |  |  |  |  |  p6t2 ← p6t2 + 1
|  |  |  |  |  | Fin
|  |  |  |  |  | Fin_DD
|  |  |  |  | Fin
|  |  |  |  | Sino :
|  |  |  |  |  // Se aumenta el contador diario del tercer torniquete
|  |  |  |  |  t3 ← t3 + 1
|  |  |  |  |  DD(c)Haga
|  |  |  |  |  | Opción 1 :
|  |  |  |  |  |  p1t3 ← p1t3 + 1
|  |  |  |  |  | Fin
1 2 3 4 5 6
```

```
1 2 3 4 5 6
| | | | | | Opción 2 :
| | | | | |   p2t3 ← p2t3 + 1
| | | | | | Fin
| | | | | | Opción 3 :
| | | | | |   p3t3 ← p3t3 + 1
| | | | | | Fin
| | | | | | Opción 4 :
| | | | | |   p4t3 ← p4t3 + 1
| | | | | | Fin
| | | | | | Opción 5 :
| | | | | |   p5t3 ← p5t3 + 1
| | | | | | Fin
| | | | | | Opción 6 :
| | | | | |   p6t3 ← p6t3 + 1
| | | | | | Fin
| | | | | Fin_DD
| | | | Fin
| | | Fin_DD
| | | Lea SwA
| | Fin_HH(SwA = 0)
| | // Acumulador semanal
| | t1s ← t1s + t1; t2s ← t2s + t2; t3s ← t3s + t3;
| | Escriba "Número de estudiantes por torniquete al día ", t1,t2,t3
| | Escriba "Est. por programa en el torniquete 1 "
| | Escriba p1t1,p2t1,p3t1,p4t1,p5t1,p6t1
| | Escriba "Est. por programa en el torniquete 2 "
| | Escriba p1t2,p2t2,p3t2,p4t2,p5t2,p6t2
| | Escriba "Est. por programa en el torniquete 3 "
| | Escriba p1t3,p2t3,p3t3,p4t3,p5t3,p6t3
| | p1t1 ← 0; p2t1 ← p3t1; p4t1 ← 0; p5t1 ← 0; p6t1 ← 0
| | p1t2 ← 0; p2t2 ← 0p3t2; p4t2 ← 0; p5t2 ← 0; p6t2 ← 0
| | p1t3 ← 0; p2t3 ← 0p3t3; p4t3 ← 0; p5t3 ← 0; p6t3 ← 0
| | i ← i + 1
| Fin_Mq
| Escriba "El número de alumnos que ingresaron por semana es "
| Escriba t1s,t2s,t3s
Fin_proc
```

Ejemplo 3.60 Administrador de edificios

Un administrador gestiona k edificios; en cada uno de los edificios hay n apartamentos y cada apartamento paga un costo c de administración; pago que se efectúa si el bit (o switch) de pago viene en 1. Diseñe un algoritmo que utilizando un HH para controlar el número de edificios y un Mq para

controlar los apartamentos de los edificios genere un informe del valor total pagado en cada edificio, el número de apartamentos que pagaron en cada edificio, el número de apartamentos que no pagaron en cada edificio en un determinado mes, y finalmente el valor recaudado por todos los edificios en el mes mencionado.

Análisis: Suponga que el administrador hace la gestión de cuatro edificios, mostrados gráficamente en la tabla 3.5.

El edificio 4 tiene un total de 15 apartamentos; los apartamentos numerados del 1 al 9 del edificio 4 no pagaron la cuota mensual; por lo tanto, el bit de pago (p) está en cero y, consecuentemente, no hay una cuota de administración asociada (c); los apartamentos numerados en el rango $[10, 15]$ del mismo edificio sí pagaron, y hay para cada uno de ellos una cuota de administración asociada que para el total del edificio es $6c$. Finalmente, con base en los recaudos de cada edificio se debe sacar al mes el total de recaudos de cada edificio, que para el caso gráfico presentado son $Recaudos = 20 * c + 4 * c + 1 * c + 6 * c$.

Tabla 3.5 Gestor de edificios

20	$p = 1; c$					15	$p = 1; c$
19						14	
18						13	
17						12	
16						11	
15						10	
14				10	$p = 0$	9	$p = 0$
13				9		8	
12				8		7	
11		7	$p = 1; c$	7		6	
10		6	$p = 0$	6		5	
9		5	$p = 1; c$	5		4	
8		4	$p = 0$	4	$p = 1; c$	3	
7		3	$p = 1; c$	3		2	
6		2	$p = 0$	2	$p = 0$	1	
5		1	$p = 1; c$	1			
4							
3							
2							
1							
Aptos.	Edificio 1	Aptos.	Edificio 2	Aptos.	Edificio 3	Aptos.	Edificio 4
$k = 4$		$k = 3$		$k = 2$		$k = 1$	

Proc: Administrador de edificios
| **Entero:** k, i, $Recaudos$, n, $Suma$, Np, Nop, j, $switchp$, c
| **Lea** k // Número de edificios
| $i \leftarrow k$ // Controla el número de edificios procesados
| $Recaudos \leftarrow 0$ // Acumulador de recaudos de todos los
| // edificios en el mes
1

```
1
  HH
    Lea n // Número de apartamentos del edificio, asociado k
    Suma ← 0 // Suma de los aportes de administración de todos los
    // apartamentos del edificio i
    Np ← 0 // Contador de los apartamentos que pagaron
    Nop ← 0 // Contador de los apartamentos que no pagaron
    j ← 1
    Mq(j ≤ n)Haga
      Lea switchp // switch de pago de los apartamentos
      Si(switchp = 1)Entonces
        Lea c // Lee la cuota de administración
        Suma ← Suma + c
        Np ← Np + 1
      Sino
        Nop ← Nop + 1
      F.Si
      j ← j + 1
    Fin_Mq
    Escriba "El valor recaudado del edificio ", k, " es igual a ", Suma
    Escriba "Apartamentos que pagaron en el edificio ", k
    Escriba " es igual a ", Np
    Escriba "Apartamentos que no pagaron en el edificio ", k
    Escriba " es igual a ", Nop
    // Suma los recaudos del apartamento k
    Recaudos ← Recaudos + Suma
    k ← k − 1
  Fin_HH(k = 0)
  Escriba "El valor recaudado durante el mes de los ", k, " edificios "
  Escriba "es igual a: ", Recaudos
Fin_proc
```

3.9 Algoritmos resueltos

Ejemplo 3.61 Comparación de dos números

Escriba un algoritmo que compare dos números y muestre el mayor.

Análisis: Para saber cuál de los dos números a y b leídos es mayor primero se pregunta si son diferentes; luego se compara si a es mayor, y si no concluir que lo es b; también se muestra el mensaje en caso de ser iguales. En la evaluación de cada condición se indica si es mayor, menor o igual, respectivamente.

$$3, 6 \implies \boxed{\mathbf{A}} \implies \text{El mayor es el 6}$$

Proc: Comparar dos números
| **Entero:** a, b
| **Escriba** "Inserta dos enteros a y b"
| **Lea** a, b
| **Si**$(a \neq b)$**Entonces**
| | **Si**$(a > b)$**Entonces**
| | | **Escriba** a, "es mayor que ", b
| | **Sino**
| | | **Escriba** a, " es menor que ", b
| | **F.Si**
| **Sino**
| | **Escriba** a, "es igual a ", b
| **F.Si**
Fin_proc

Ejemplo 3.62 Mayor de un conjunto de números

Escriba un algoritmo que determine el mayor de tres números. En el desarrollo de este algoritmo se proponen dos soluciones.

Análisis: Se declaran las variables x, y, z como reales. Se forman condicionales compuestos para mayor claridad y un anidamiento simple.

El orden está en si una variable es la mayor, y en caso de que no lo sea, se pregunta por la siguiente variable, y así sucesivamente. Como los números reales resultan ser un conjunto ordenado, si se preguntó por los $n-1$ primeros números y ninguno resultó ser el mayor, entonces el n-ésimo número es el mayor.

Sin embargo, para encontrar el máximo para una cantidad mayor de números resulta ineficiente agregar un sinnúmero de condiciones (Solución 1); en este caso se asigna primeramente a una variable max un número x_1, y si para algún x_k se tiene que $x_k > max$, entonces a max se le asigna x_k, es decir, $max \leftarrow x_k$. De igual manera se hace para encontrar el mínimo; en este caso la condición sería $x_k < min$. Se hace esto para cada número, realizando un máximo de $n-1$ iteraciones. Usando este método el algoritmo queda como en la Solución 2.

Solución 1:
Proc: Mayor de 3 números
| **Entero:** x, y, z, $mayor$
1

```
1
| Escriba "Ingrese los tres números: "
| Lea x, y, z
| Si(x > y ∧ x > z)Entonces
| | Escriba "El mayor de los tres números es: ", x
| Sino
| | Si(y > x ∧ y > z)Entonces
| | | Escriba "El mayor de los tres números es: ", y
| | Sino
| | | Escriba "El mayor de los tres números es: ", z
| | F.Si
| F.Si
Fin_proc
```

Solución 2:

```
Proc: Mayor de 3 números
| Entero: x, y, z, mayor
| Escriba "Ingrese los tres números: "
| Lea x, y, z
| mayor ← x
| Si(y > mayor)Entonces
| | mayor ← y
| F.Si
| Si(z > mayor)Entonces
| | mayor ← z
| F.Si
| Escriba "El mayor de los tres números es: ", mayor
Fin_proc
```

Ejemplo 3.63 Calificaciones de un estudiante

La valoración que representa las calificaciones de un estudiante se calcula de acuerdo con la siguiente tabla:

Calificación numérica	**Valoración**
mayor o igual que 90	Excelente
menor que 90 pero mayor que o igual a 80	Sobresaliente
menor que 80 pero mayor que o igual a 60	Aceptable
menor que 60	Insuficiente

Análisis: Por ejemplo, si un estudiante tiene una calificación de 54, su valoración será "Insuficiente". Análogamente al anterior ejercicio, se pregunta en un orden, en este caso en el orden de la tabla expuesta. Este ejercicio evita condiciones complejas.

```
Proc: Valoración de un estudiante
| Entero: num
| Escriba "Calificacion numérica: "
| Lea num
| Si(num ≥ 90)Entonces
| | Escriba "La calificación corresponde a Excelente"
| Sino
| | Si(num ≥ 80)Entonces
| | | Escriba "La calificación corresponde a Sobresaliente"
| | Sino
| | | Si(num ≥ 60)Entonces
| | | | Escriba "La calificación corresponde a Aceptable"
| | | Sino
| | | | Escriba "La calificación corresponde a Insuficiente"
| | | F.Si
| | F.Si
| F.Si
Fin_proc
```

Ejemplo 3.64 Conversión de Si-Sino a Dependiendo De

Vuelva a escribir la siguiente cadena Si-Sino usando la instrucción Dependiendo De:

```
Proc: Calificaciones
| Caracter: calif
| Escriba "Calificación en letra: "
| Lea calif
| Si(calif = A)Entonces
| | Escriba "La calificación numérica está entre 90 y 100"
| Sino
| | Si(calif = B)Entonces
| | | Escriba "La calificación numérica está entre 80 y 89.9"
| | Sino
| | | Si(calif = C)Entonces
| | | | Escriba "La calificación numérica está entre 70 y 79.9"
| | | Sino
| | | | Si(calif = D)Entonces
| | | | | Escriba "Mala calificación"
| | | | Sino
| | | | | Escriba "Por supuesto que no tuve nada que ver con mi calificación."
| | | | F.Si
| | | F.Si
| | F.Si
| F.Si
Fin_proc
```

Análisis: Se observa que todos los condicionales se deben a los posibles valores que puede tener la variable $calif$. Por lo tanto, es posible convertir la cadena anidada Si-Sino usando la instrucción Dependiendo De.

Solución:

```
Proc: Calificaciones
| Caracter: calif
| Escriba "Calificación en letra: "
| Lea calif
| DD(calif)Haga
| | Opción A :
| | | Escriba "La calificación numérica está entre 90 y 100"
| | Fin
| | Opción B :
| | | Escriba "La calificación numérica está entre 80 y 89.9"
| | Fin
| | Opción C :
| | | Escriba "La calificación numérica está entre 70 y 79.9"
| | Fin
| | Opción D :
| | | Escriba "Mala calificación"
| | Fin
| | Sino :
| | | Escriba "Por supuesto que no tuve nada que ver con mi calificación."
| | Fin
| Fin_DD
Fin_proc
```

Comentarios: Se observa claramente cómo el largo y la complejidad del algoritmo en sí se tornan de menor tamaño, juntando todos los Si-Sino anidados en un solo condicional Dependiendo De.

Ejemplo 3.65 Fecha válida I

Escriba un algoritmo para desplegar los mensajes siguientes:

Introduzca el número del mes:
Introduzca un día del mes:

Haga que su programa acepte y almacene un número en la variable *mes* en respuesta al primer mensaje, y acepte y almacene un número en la variable *dia* en respuesta al segundo mensaje. Si el mes introducido no está entre 1 y 12 inclusive, imprima un mensaje informando al usuario que se ha introducido un mes inválido. Si el día introducido no está entre 1 y 31, imprima un mensaje informando al usuario que se ha introducido un día inválido.

```
Proc: Fechas válidas
| Entero: mes, dia
| Escriba "Introduzca el número del mes: "
| Lea mes
| Escriba "Introduzca un día del mes
| Lea dia
| Si(mes ≥ 1 y mes ≤ 12)Entonces
| | Si(dia ≥ 1 y dia ≤ 31)Entonces
| | | Escriba "La fecha es válida"
| | Sino
| | | Escriba "El día es inválido"
| | F.Si
| Sino
| | Escriba "El mes es inválido"
| F.Si
Fin_proc
```

Ejemplo 3.66 Fecha válida II

En un año que no es bisiesto (febrero tiene 28 días), enero, marzo, mayo, julio, agosto, octubre y diciembre tienen 31 días y todos los demás meses tienen 30. Usando esta información modifique el programa escrito en el ejercicio anterior para desplegar un mensaje cuando se introduce un día inválido para un mes introducido por un usuario. Para este algoritmo ignore los años bisiestos.

```
Proc: Fecha válida
| Entero: mes, dia
| Escriba "Introduzca un mes (use 1 para Ene, etc.): "
| Lea mes
| Escriba "Introduzca un día del mes
| Lea dia
| Si(mes ≥ 1 y mes ≤ 12)Entonces
| | Si(mes = 1 o 3 o 5 o 7 o 8 o 10 o 12)Entonces
| | | Si(dia ≥ 1 y dia ≤ 31)Entonces
| | | | Escriba "La fecha",dia,"/",mes,"es válida"
| | | Sino
| | | | Escriba "El día es inválido"
| | | F.Si
| | Sino
| | | Si(mes = 2)Entonces
| | | | Si(dia ≥ 1 y dia ≤ 28)Entonces
| | | | | Escriba "La fecha ", dia, "/", mes, " es válida"
1 2 3 4 5
```

```
1 2 3 4 5
|  |  |  |  Sino
|  |  |  |  |  Escriba "El día es inválido"
|  |  |  |  F.Si
|  |  |  Sino
|  |  |  |  Si(dia ≥ 1 y dia ≤ 28)Entonces
|  |  |  |  |  Escriba "La fecha ", dia, "/", mes, " es válida"
|  |  |  |  Sino
|  |  |  |  |  Escriba "El día es inválido"
|  |  |  |  F.Si
|  |  |  F.Si
|  |  F.Si
|  Sino
|  |  Escriba "El mes es inválido"
|  F.Si
Fin_proc
```

Ejemplo 3.67 Determinar el cuadrante I

El cuadrante en el que reside una línea trazada desde el origen es determinado por el ángulo que forma la línea con el eje x positivo como sigue:

Ángulo desde el eje x positivo	**Cuadrante**
Entre 0 y 90 grados	I
Entre 90 y 180 grados	II
Entre 180 y 270 grados	III
Entre 270 y 360 grados	IV

Usando esta información escriba un algoritmo que acepte el ángulo de la línea como una entrada del usuario y determine y despliegue el cuadrante apropiado a los datos introducidos. (Nota: Si el ángulo tiene exactamente 0, 90, 180 o 270 grados, la línea no reside en ningún cuadrante sino que se encuentra en un eje).

```
Proc: Cuadrante
|  Real: ang
|  Escriba "Introduzca el ángulo de la línea: "
|  Lea ang
|  Si(ang ≥ 0 y ang ≤ 360)Entonces
|  |  Si(ang = 90 o ang = 180 o ang = 270 o ang = 0 o ang = 360)Entonces
|  |  |  Escriba "La línea está sobre uno de los ejes"
|  |  Sino
|  |  |  Si(ang > 0 y ang < 90)Entonces
|  |  |  |  Escriba "La línea se encuentra en el I cuadrante"
1 2 3 4
```

```
1 2 3 4
|  |  |  Sino
|  |  |  |  Si(ang > 90 y ang < 180)Entonces
|  |  |  |  |  Escriba "La línea se encuentra en el II cuadrante"
|  |  |  |  Sino
|  |  |  |  |  Si(ang > 180 y ang < 270)Entonces
|  |  |  |  |  |  Escriba "La línea se encuentra en el III cuadrante"
|  |  |  |  |  Sino
|  |  |  |  |  |  Escriba "La línea se encuentra en el IV cuadrante"
|  |  |  |  |  F.Si
|  |  |  |  F.Si
|  |  |  F.Si
|  |  F.Si
|  Sino
|  |  Escriba "NO introdujo un ángulo válido"
|  F.Si
Fin_proc
```

Ejemplo 3.68 Determinar el cuadrante II

Modifique el algoritmo anterior de modo que se despliegue un mensaje que identifique un ángulo de cero grados como el eje x positivo, un ángulo de 90 grados como el eje y positivo, un ángulo de 180 grados como el eje x negativo y un ángulo de 270 grados como el eje y negativo.

Análisis: Entre la primera y segunda parte de este ejercicio con una simple combinación del condicional Si y un condicional Dependiendo De no se produce un mayor cambio en la estructura del algoritmo.

```
Proc: Cuadrante
|  Escriba "Introduzca el ángulo de la línea: "
|  Entero: ang
|  Lea ang
|  Si(ang ≥ 0 y ang ≤ 360)Entonces
|  |  Si(ang = 90 o ang = 180 o ang = 270 o ang = 0 o ang = 360)Entonces
|  |  |  DD(ang)Haga
|  |  |  |  Opción 90 :
|  |  |  |  |  Escriba "La línea se encuentra sobre el eje y positivo"
|  |  |  |  Fin
|  |  |  |  Opción 180 :
|  |  |  |  |  Escriba "La línea se encuentra sobre el eje x negativo"
|  |  |  |  Fin
|  |  |  |  Opción 270 :
|  |  |  |  |
1 2 3 4 5
```

```
1 2 3 4 5
|  |  |  |  | Escriba "La línea se encuentra sobre el eje y negativo"
|  |  |  |  Fin
|  |  |  |  Opción 360 :
|  |  |  |  | Escriba "La línea se encuentra sobre el eje x positivo"
|  |  |  |  Fin
|  |  |  |  Opción 0 :
|  |  |  |  | Escriba "La línea se encuentra sobre el eje x positivo"
|  |  |  |  Fin
|  |  |  Fin_DD
|  |  Sino
|  |  | Si(ang > 0 y ang < 90)Entonces
|  |  | | Escriba "La línea se encuentra en el I cuadrante"
|  |  | Sino
|  |  | | Si(ang > 90 y ang < 180)Entonces
|  |  | | | Escriba "La línea se encuentra en el II cuadrante"
|  |  | | Sino
|  |  | | | Si(ang > 180 y ang < 270)Entonces
|  |  | | | | Escriba "La línea se encuentra en el III cuadrante"
|  |  | | | Sino
|  |  | | | | Escriba "La línea se encuentra en el IV cuadrante"
|  |  | | | F.Si
|  |  | | F.Si
|  |  | F.Si
|  |  F.Si
|  Sino
|  | Escriba "NO introdujo un ángulo válido"
|  F.Si
Fin_proc
```

Ejemplo 3.69 Tarifa de registro

Con base en el año del modelo y el peso de un automóvil el estado de Nueva Jersey determina la clase de vehículo y la tarifa de registro que le corresponde usando la siguiente tabla:

Año del modelo	Peso	Clase de peso	Tarifa
1970 o anterior	Menos de 2700 lbs	1	$16.50
	2700 a 3800 lbs	2	25.50
	Más de 3800 lbs	3	46.50
1971 a 1979	Menos de 2700 lbs	1	27.00
	2700 a 3800 lbs	2	30.50
	Más de 3800 lbs	3	52.50
1980 o posterior	Menos de 3500 lbs	7	19.50
	3500 lbs o más	8	52.50

Usando esta información escriba un algoritmo que acepte el año y el peso de un automovíl y determine y despliegue la clase y la tarifa de registro para el mismo.

Entrada 1: Año
Entrada 2: Peso
Salida 1: Clase
Salida 2: Tarifa
Entrada 1 Ejemplo: 2002
Entrada 2 Ejemplo: 3250
Salida 1 Ejemplo: 7
Salida 2 Ejemplo: 19.50

```
Proc: Tarifa de registro
| Entero: anio, peso
| Escriba "Introduzca el año del modelo del automovil: "
| Lea anio
| Escriba "Introduzca el peso del automovil: "
| Lea peso
|
| Si(anio ≤ 1970)Entonces
| | Si(peso < 2700)Entonces
| | | Escriba "Clase de peso: 1"
| | | Escriba "Tarifa de Registro: $16.50"
| | Sino
| | | Si(peso ≥ 2700 y peso ≤ 3800)Entonces
| | | | Escriba "Clase de peso: 2"
| | | | Escriba "Tarifa de Registro: $25.50"
| | | Sino
| | | | Escriba "Clase de peso: 3"
| | | | Escriba "Tarifa de Registro: $46.50"
| | | F.Si
| | F.Si
| Sino
| | Si(anio ≥ 1971 y anio ≤ 1979)Entonces
| | | Si(peso < 2700)Entonces
| | | | Escriba "Clase de peso: 4"
| | | | Escriba "Tarifa de Registro: $27.50"
| | | Sino
| | | | Si(peso ≥ 2700 y peso ≤ 3800)Entonces
| | | | | Escriba "Clase de peso: 5"
| | | | | Escriba "Tarifa de Registro: $30.50"
| | | | Sino
| | | | | Escriba "Clase de peso: 6"
1 2 3 4 5
```

```
1 2 3 4 5
|  |  |  |  | Escriba "Tarifa de Registro: $52.50"
|  |  |  |  F.Si
|  |  |  F.Si
|  |  Sino
|  |  | Si(peso < 3500)Entonces
|  |  |  | Escriba "Clase de peso: 7"
|  |  |  | Escriba "Tarifa de Registro: $19.50"
|  |  |  Sino
|  |  |  | Escriba "Clase de peso: 8"
|  |  |  | Escriba "Tarifa de Registro: $52.50"
|  |  |  F.Si
|  |  F.Si
|  F.Si
Fin_proc
```

Ejemplo 3.70 Juego del 21

En el juego del 21, el valor de las cartas del 2 al 10 es el que tienen impreso, sin importar de qué palo sean. Las cartas de personajes (jota, reina y rey) se cuentan como 10 y el as como 1 u 11, dependiendo de la suma de todas las cartas en una mano. El as se cuenta como 11 solo si el valor total resultante de todas las cartas en una mano no excede de 21, de lo contrario se cuenta como 1. Usando esta información escriba un algoritmo que acepte los valores de tres cartas como entradas (un 1 correspondiente a un as, un 2 a un dos, etc.), calcule el valor total de la mano en forma apropiada y despliegue el valor de las tres cartas con un mensaje impreso.

Entrada 1: Un número del 1 al 13
Entrada 2: Un número del 1 al 13
Salida: Valor de la mano
Entrada 1 Ejemplo: 1
Entrada 2 Ejemplo: 12
Salida Ejemplo: 15

```
Proc: Juego del 21
| Entero: total, carta1, carta2, carta3
| total ← 0
| Escriba "Introduzca la primera carta: "
| Lea carta1
| Si(carta1 ≥ 2 y carta1 ≤ 10)Entonces
| | total ← total + carta1
| Sino
| |
1 2
```

```
1 2
|  |  Si(carta1 ≥ 11 y carta1 ≤ 13)Entonces
|  |  |  total ← total + 10
|  |  Sino
|  |  |  total ← total + 11
|  |  F.Si
|  F.Si
|  Escriba "Introduzca la segunda carta: "
|  Lea carta2
|  Si(carta2 ≥ 2 y carta2 ≤ 10)Entonces
|  |  total ← total + carta2
|  Sino
|  |  Si(carta2 ≥ 11 y carta2 ≤ 13)Entonces
|  |  |  total ← total + 10
|  |  Sino
|  |  |  Si(total + 11 ≤ 21)Entonces
|  |  |  |  total ← total + 11
|  |  |  Sino
|  |  |  |  total ← total + 1
|  |  |  F.Si
|  |  F.Si
|  F.Si
|  Escriba "Introduzca la tercera carta: "
|  Lea carta3
|  Si(carta3 ≥ 2 y carta3 ≤ 10)Entonces
|  |  total ← total + carta3
|  Sino
|  |  Si(carta3 ≥ 11 y carta3 ≤ 13)Entonces
|  |  |  total ← total + 10
|  |  Sino
|  |  |  Si(total + 11 ≤ 21)Entonces
|  |  |  |  total ← total + 11
|  |  |  Sino
|  |  |  |  total ← total + 1
|  |  |  F.Si
|  |  F.Si
|  F.Si
|  Si(total > 21 y carta1 = 1)Entonces
|  |  total ← total − 10
|  F.Si
|  Si(total > 21 y carta2 = 1)Entonces
|  |  total ← total − 10
|  F.Si
1
```

```
1
| Escriba "La suma total de las cartas es: ", total
Fin_proc
```

Ejemplo 3.71 Tipo de ángulo

Un ángulo es considerado agudo si es menor que 90 grados; obtuso si es mayor que 90 grados, y recto si es igual a 90 grados. Usando esta información escriba un algoritmo que acepte un ángulo, en grados, y despliegue el tipo de ángulo correspondiente a los grados introducidos.

Análisis: Se declara la variable *angulo* de tipo Real. Luego se pregunta en orden descendente por el valor del ángulo, es decir, si *angulo* es mayor que 90, es obtuso, si no entonces se pregunta por el siguiente valor, que correspondería a la igualdad a 90, y si no cumple las condiciones anteriores, el ángulo es agudo.

Entrada: Angulo en grados
Salida: Tipo de ángulo
Proc: Tipo de ángulo

```
| Entero: angulo
| Escriba "Ingrese ángulo en grados: "
| Lea angulo
| Si(angulo > 90)Entonces
| | Escriba "El ángulo es obtuso"
| Sino
| | Si(angulo = 90)Entonces
| | | Escriba "El ángulo es recto"
| | Sino
| | | Escriba "El ángulo es agudo"
| | F.Si
| F.Si
Fin_proc
```

Ejemplo 3.72 Tipo de triángulo

Un triángulo es equilátero si todos sus lados son iguales; isósceles, si solo tiene dos lados iguales, y en caso contrario, escaleno. Escriba un algoritmo que dado las longitudes de los lados de un triángulo determine si es equilátero, isósceles o escaleno.

Análisis: Se pregunta si hay igualdad entre todos los lados, por tanto se usa un "y" (AND), luego se pregunta si algún par son iguales, por eso se usa "o" (OR), y en caso contrario se deduce que el triángulo es escaleno.

Entrada: Longitudes de los lados

Salida: Tipo de triángulo (equilátero, isósceles, escaleno)

Entrada Ejemplo: 80, 20, 95
Salida Ejemplo: Escaleno

```
Proc: Tipo de triángulo
| Real: a, b, c
| Escriba "Ingrese las longitudes de los lados del triángulo: "
| Lea a, b, c
| Si(a = b y b = c)Entonces
| | Escriba "El triángulo es equilátero"
| Sino
| | Si(a = b o b = c o a = c)Entonces
| | | Escriba "El triángulo es isósceles"
| | Sino
| | | Escriba "El triángulo es escaleno"
| | F.Si
| F.Si
Fin_proc
```

Ejemplo 3.73 Tipo de año

Todos los años que se dividen exactamente entre 400 o que son divisibles exactamente entre cuatro y no son divisibles exactamente entre 100 son años bisiestos. Por ejemplo, en vista que 1600 es divisible exactamente entre 400, el año 1600 fue bisiesto. Del mismo modo, en vista que 1988 es divisible exactamente entre cuatro pero no entre 100, también fue bisiesto. Usando esta información escriba un algoritmo que acepte el año como una entrada del usuario, determine si el año es bisiesto y despliegue un mensaje apropiado que le indique al usuario si el año es bisiesto o no (Bronson & Borse, 2010).
Análisis: Gracias a las condiciones compuestas en este caso, nos podemos ahorrar varios niveles de anidamiento y hacer un algoritmo que parece largo pero que realmente es corto y sencillo. La variable *anio* representa el año.

Entrada: Año
Salida: Bisiesto o No es bisiesto
Ejemplo: Entrada: 2004
Salida: Es un año bisiesto

```
Proc: Tipo de año
| Entero: anio
| Escriba "Introduzca un año: "
| Lea anio
1
```

```
1
| Si(anio mod 400 = 0 o (anio mod 4 = 0 y anio mod 100 ≠ 0))Entonces
| | Escriba "El año ", anio, " es bisiesto"
| Sino
| | Escriba "El año ", anio, " NO es bisiesto"
| F.Si
Fin_proc
```

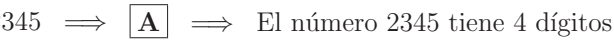
Ejemplo 3.74 Número de cifras

Realice un algoritmo que diga cuántos dígitos tiene un número.

Análisis: Se declaran las variables enteras n, nd y cn; donde n será el número que se va a evaluar, nd el número de dígitos y cn la variable que se usará para realizar las divisiones enteras entre 10.

Se utiliza un condicional Si-Sino anidado en el Sino con un ciclo Mientras que. En el primer condicional se evalúa si el número es 0; si lo es, al número de dígitos se le asigna 1. En el Sino se obtiene el valor absoluto del número y se hacen divisiones sucesivas en el ciclo Mientras que siempre que el valor entero del número sea entero, en cada iteración se incrementa el número de dígitos y al finalizar el ciclo se escribe el resultado.

Entrada: Un número entero.
Salida: El número de dígitos del número proporcionado.




```
Proc: Número de cifras
| Entero: nd, n, cn
| nd ← 0
| Escriba "Por favor digite un número entero"
| Lea n
| Si(n = 0)Entonces
| | nd ← 1
| Sino
| | cn ← abs(n)
| | Mq(cn > 0)Haga
| | | cn ← cn DIV 10
| | | nd ← nd + 1
| | Fin_Mq
| F.Si
| Escriba n, "tiene", nd, "dígitos"
Fin_proc
```

Ejemplo 3.75 Test de número primo

Realice un algoritmo que lea un entero y diga si es primo o no.

Análisis: Se declaran las variables enteras n, j y la booleana sw, donde n será el número que se va a verificar si es primo o no y j y sw servirán de control para el ciclo Mientras que.

Se utiliza un ciclo Mientras que, que se ejecutará siempre que sw sea falso y j menor o igual a la mitad del número n. En el interior del ciclo se evalúa si entre los números del 2 a $n/2$ existe un divisor de n; si es así, se detendrá el ciclo y se escribirá que el número no es primo; de lo contrario, si no se encontró ningún divisor, entonces el ciclo se detendrá porque j llegó a $n/2$, y en este caso se escribirá que el número n es primo.

Por otra parte, vea que si d es un divisor de n, entonces $\frac{n}{d}$ también lo es. A su vez, si d no fuese un divisor de n, entonces $\frac{n}{d}$ tampoco lo sería. Esto quiere decir que podemos agrupar los divisores de n en parejas, es decir,

$$\left(d_1, \frac{n}{d_1}\right), \left(d_2, \frac{n}{d_2}\right), \left(d_3, \frac{n}{d_3}\right), \ldots, \left(d_k, \frac{n}{d_k}\right).$$

Sin embargo, para evitar que se repitan podemos agregar una condición adicional, y es que cada divisor d_k sea menor que su pareja $\frac{n}{d_k}$. Ahora bien, si

$$d_k \leq \frac{n}{d_k} \implies d_k^2 \leq n \implies d_k \leq \sqrt{n}.$$

De lo anterior se puede concluir que si se desea revisar todos los divisores de n o revisar si n es un número primo (es decir, no tiene divisores), basta con revisar hasta los números menores o iguales que $\sqrt{n}$. Si encuentra que a es un divisor, entonces $\frac{n}{a}$ también lo será.

La variable j que se utilizará como contador se declara en 2, puesto que para saber si un número es primo o no interesa saber si tiene un divisor distinto de 1 y de el mismo.

Entrada: Un número natural.
Salida: Diremos si el número proporcionado es primo o no.

$$9 \implies \boxed{\mathbf{A}} \implies \text{El número 9 no es primo}$$

Proc: Número primo
| **Entero:** j, n
| **Booleano:** sw
| $j \leftarrow 2$
| $sw \leftarrow falso$
| **Escriba** "Por favor digite un número entero"
| **Lea** n
| **Mq**($j <= (n/2)$ y $sw = falso$)**Haga**
| | // La condición $j \leq n/2$ puede ser remplazada por $j \leq \sqrt{n}$
| | **Si**(n **mod** $j = 0$)**Entonces**
| | | $sw \leftarrow verdadero$
| | **F.Si**
| | $j \leftarrow j + 1$
| **Fin_Mq**
| **Si**($sw = falso$)**Entonces**
| | **Escriba** n, "es primo"
| **Sino**
| | **Escriba** n, "no es primo"
| **F.Si**
Fin_proc

Ejemplo 3.76 Número perfecto

Realice un algoritmo que diga si un número es perfecto o no, es decir, que la suma de sus divisores propios es igual a el mismo número.
Análisis: Se declaran las variables enteras n, j y *suma*; donde n será el número natural de entrada, j la variable utilizada para controlar el ciclo Mientras que y *suma* la variable en la que se acumularán las sumas de los divisores.

Se realiza un ciclo Mientras que desde 2 hasta la mitad del número (ya que un número no tendrá divisores mayores a su mitad), se usa un condicional en cada iteración para evaluar si j es divisor de n, y si es así, se acumulará en la variable suma. Al finalizar el ciclo se compara n con el valor de suma, y si son iguales se escribe que el número es perfecto, de lo contrario se escribe que el número no es perfecto.

Entrada: Un número natural.
Salida: Diremos si el número proporcionado es perfecto o no.

$$6 \implies \boxed{\mathbf{A}} \implies \text{El número 6 es perfecto}$$

Proc: Número perfecto
|
1

```
1
| Entero: j, suma, n
| j ← 2, suma ← 1
| Escriba "Por favor digite un número entero"
| Lea n
| Mq(j <= (n/2))Haga
| | Si(n mod j = 0)Entonces
| | | suma ← suma + j
| | F.Si
| | j ← j + 1
| Fin_Mq
| Si(suma = n)Entonces
| | Escriba n, "es perfecto"
| Sino
| | Escriba n, "no es perfecto"
| F.Si
Fin_proc
```

Ejemplo 3.77 Números amigos

Realice un algoritmo que diga si dos números son amigos o no. Dos números son amigos cuando la suma de sus divisores es igual al otro, respectivamente.

Análisis: Se declaran las variables enteras $n, m, sum, sum2$ y j; donde n y m son los dos números que se va a verificar si son o no amigos, j servirá para controlar los dos ciclos Mientras que, sum y $sum2$ serán las variables en las que se acumule la suma de los divisores de los números m y n, respectivamente.

Se utilizan dos ciclos Mientras que: uno en el que j va desde 2 hasta la mitad de m y el otro de 2 hasta la mitad de n; en ambos ciclos se pregunta si j es divisor del número; si lo es, se acumula en sum o $sum2$. Al finalizar se evalúa si la suma de los divisores de m (sum) es igual a n y si la suma de los divisores de n (sum) es igual a m; si es así, se escribe que los números son amigos, de lo contrario que no lo son.

Entrada: Dos números enteros.
Salida: Un mensaje que indique si los números son pares o impares.

$$6, 8 \implies \boxed{\mathbf{A}} \implies \text{Los números 6 y 8 no son amigos}$$

```
Proc: Números amigos
| Entero: sum, sum2, j, m, n
| sum ← 1, sum2 ← 1, j ← 2
1
```

```
1
| Escriba "Por favor digite dos números enteros"
| Lea m, n
| Mq(j <= (m/2))Haga
| | Si(m mod j = 0)Entonces
| | | sum ← sum + j
| | F.Si
| | j ← j + 1
| Fin_Mq
| j ← 2
| Mq(j <= (n/2))Haga
| | Si(n mod j = 0)Entonces
| | | sum2 ← sum2 + j
| | F.Si
| | j ← j + 1
| Fin_Mq
| Si(sum = n y sum2 = m)Entonces
| | Escriba m, n, "son amigos"
| Sino
| | Escriba m, n, "no son amigos"
| F.Si
Fin_proc
```

Ejemplo 3.78 Números de Fibonacci

Realice un algoritmo que muestre el n-ésimo término de la serie de Fibonacci. La serie de Fibonnaci se obtiene mediante la siguiente función:

Análisis: Primero se declaran las variables n, sig, ant, $preant$ y j de tipo Entero; la variable n es el número que se pide por pantalla; la variable del control del ciclo Para es j; la variable $preant$ es inicializada en 0, puesto que es el primer término de la serie de Fibonacci; la variable ant es inicializada en 1, como segundo término de la serie, y la variable sig es la que contiene el término que se calcule.

Después se pide digitar el número por pantalla, y se captura su valor. Se verifica si $n = 0$ o $n = 1$; en este caso, el valor de la función es 0. Si $n = 2$, el valor de la función es $0 + 1$. En caso de que $n > 2$, se van calculando los términos en el ciclo Para; el término siguiente en la función es la suma del penúltimo y el último término. Luego de calcular el término se escribe en pantalla y a la variable $preant$ se le asigna el valor de la variable ant, y a esta se le asigna el valor de la variable sig, que corresponde al último término calculado.

$$F(n) = \begin{cases} 0, & \text{si } n = 0 \\ 1, & \text{si } n = 1 \\ F(n-1) + F(n-2), & \text{si } n > 1 \end{cases}$$

$$5 \implies \boxed{\mathbf{A}} \implies 1\ 1\ 2\ 3\ 5$$

```
Proc: Números de Fibonacci
| Entero: n, preant, ant, sig
| Escriba "Digite n"
| Lea n
| preant ← 1
| ant ← 2
| Si(n = 1)Entonces
| | Escriba preant
| Sino
| | Si(n = 2)Entonces
| | | Escriba preant, ant
| | Sino
| | | Escriba preant, ant
| | | Para(j = 3, n, 1)Haga
| | | | sig ← ant + preant
| | | | Escriba sig
| | | | preant ← ant
| | | | ant ← sig
| | | Fin_Para
| | F.Si
| F.Si
Fin_proc
```

Ejemplo 3.79 Divisores de un número

Realice un algoritmo que reciba un número entero y dé como salida sus divisores.

Entrada: Un número natural
Salida: Todos los divisores del número proporcionado por el usuario.

$$28 \implies \boxed{\mathbf{A}} \implies \text{Los divisores de 28 son: 1, 2, 4, 7, 14, 28.}$$

Análisis: Los divisores de un número son aquellos que al dividir el número por uno de ellos, el residuo es cero, por lo tanto se hará uso de la función **mod**. Para obtenerlos basta con llegar a su mitad, y siempre se encontrarán el 1 y él mismo. Como se necesita que la división se haga por lo menos una vez (entre 1) se emplea el ciclo Haga Hasta (HH), que se efectuará hasta que sobrepase la mitad. La variable que recorre de uno en uno los números para saber si son divisores es j. Si al efectuar 'num **mod** j' se obtiene cero, entonces es divisor, por tanto se muestra por pantalla.

```
Proc: Divisores de un número
  Entero: j, n
  j ← 1
  Escriba "Por favor digite un número entero"
  Lea n
  Escriba "Los divisores de ", n, " son: "
  HH
    Si(n mod j = 0)Entonces
      Escriba j
    F.Si
    j ← j + 1
  Fin_HH(j > (n/2))
  Escriba n
Fin_proc
```

Ejemplo 3.80 Cubos de Nicomaco de Gerasa

Nicomaco de Gerasa descubrió la siguiente propiedad de los números naturales: Al sumar el primer impar se obtiene el primer cubo:
$1 = 1^3 = 1$.

Al sumar los dos siguientes impares se obtiene el segundo cubo:
$3 + 5 = 2^3 = 8$.

Al sumar los tres siguientes impares se obtiene el tercer cubo:
$7 + 9 + 11 = 3^3 = 27$.

Al sumar los cuatro siguientes impares se obtiene el cuarto cubo:
$13 + 15 + 17 + 19 = 4^3 = 64$.
etc...

Con esta propiedad calcule y muestre los cubos de los primeros n números naturales.

Análisis: Las variables que se van a usar son las siguientes: n es el número de cubos que se va a calcular; *base* es el valor al que en un momento determinado se le está calculando el cubo; *impar* contendrá el número impar que se va a sumar; i es la variable de control del ciclo Para interior y permite saber cuántos valores se han sumado para una misma base; *resultado* guarda la suma, que al final queda con el cubo correspondiente a la base. El algoritmo debe iniciar pidiendo la cantidad de cubos que se va a calcular. Un ciclo Para externo controla el número de cubos que se debe mostrar. Un ciclo interior debe sumar tantos impares como la base indique, por lo tanto, la suma de los impares se controla con un ciclo Para que va desde 1 hasta *base* con contador i. En su interior se debe actualizar el valor de *impar* de 2 en 2 y, a su vez, se acumula el valor en la variable *resultado*. Vea que *resultado* debe ser iniciado en 0 cada vez que se va a calcular un nuevo cubo.

Por ejemplo, si $n = 4$, la salida debe ser como se muestra a continuación:

$$4 \implies \boxed{\mathbf{A}} \implies \begin{array}{l} 1^3 = 1 \\ 2^3 = 8 \\ 3^3 = 27 \\ 4^3 = 64 \end{array}$$

Proc: Cubos de Nicomaco de Gerasa
Entero: *impar*, *base*, *n*, *i*, *resultado*
impar ← 1
Escriba "Digite el número de cubos que desea ver: "
Lea *n*
Para(*base* = 1, *n*, 1)**Haga**
Escriba *base*, "^3 = "
// Se inicia resultado en 0
resultado ← 0
Para(*i* = 1, *base*, 1)**Haga**
resultado ← *resultado* + *impar*
impar ← *impar* + 2
Fin_Para
Escriba *resultado*
Fin_Para
Fin_proc

Ejemplo 3.81 Potenciación

Dado una base b y un exponente e, determine el valor de b^e.

Análisis: Se usará una variable p que almacenará el resultado; esta se declara en 1; de tal forma que si se hace la instrucción $p \leftarrow p * b$ repetidas veces se obtiene una potencia de b. Esta instrucción se debe controlar mediante un ciclo Para, de tal forma que se haga e veces, y así obtener b^e.

Proc: Potenciación
Entero: *p*, *b*, *e*, *i*
p ← 1
Lea *b*, *e*
Para(*i* = 1, *e*, 1)**Haga**
p ← *p* ∗ *b*
Fin_Para
Escriba "El resultado es", *p*
Fin_proc

Ejemplo 3.82 Factorial de muchos números

Calcule el factorial justo después de leído una serie de números. Se detiene cuando se ingresa un número negativo.

Análisis: Primero ilustremos la situación con un ejemplo. Si el usuario ingresa $3, 4, 5, 1, -1$, entonces la salida debe ser $6, 24, 120, 1$. Se usa un ciclo Haga Hasta de tal manera que por lo menos se ejecute una vez, dado que al menos el usuario debe ingresar un número. Se valida si el número es un entero positivo. Luego se inicializa la variable factorial en 1, puesto que es el módulo de la multiplicación. Se calcula su factorial y se muestra. En caso de no tratarse de un entero positivo, el algoritmo termina.

```
Proc: Factorial de muchos números
| Entero: factorial, num, i
| HH
| | factorial ← 1
| | Escriba "Ingrese número"
| | Lea num
| | Si(num ≥ 0)Entonces
| | | Para(i = 1, num, 1)Haga
| | | | factorial ← factorial * i
| | | Fin_Para
| | | Escriba "Factorial: ", factorial
| | F.Si
| Fin_HH(num < 0)
Fin_proc
```

Ejemplo 3.83 Múltiplos de 3

Escriba un algoritmo que seleccione y despliegue los primeros 20 números enteros que sean divisibles entre 3.

Análisis: Se presentarán 3 soluciones. La primera consiste en ir buscando número por número, y en el momento en que uno de esos sea divisible por 3 incrementa un contador i en 1; cuando haya encontrado 19 números, entra en el Mientras que, debido a que se cumple la condición $i = 19 < 20$. Por último, encuentra el vigésimo múltiplo de 3; allí $i = 20$ y no se cumple que sea menor que 20, por tanto no vuelve a entrar en el ciclo, y finaliza el algoritmo.

```
Proc: Múltiplos de 3
| Entero: i, num
| i ← 0
1
```

```
1
| num ← 1
| Mq(i < 20)Haga
| | Si(num mod 3 = 0)Entonces
| | | Escriba num
| | | i ← i + 1
| | F.Si
| | num ← num + 1
| Fin_Mq
Fin_proc
```

La segunda solución usa el hecho de que un Para puede hacer saltos de 3 en 3, de tal manera que salte de múltiplo en múltiplo y simplemente los muestre en pantalla; se puede observar que el incremento en este caso es 3.

```
Proc: Múltiplos de 3
| Entero: i
| Para(i = 3, 60, 3)Haga
| | Escriba i
| Fin_Para
Fin_proc
```

La tercera solución es la más versátil, pues mostrará 20 números en pantalla, que serán los primeros 20 múltiplos de 3, pues en el Escriba se encuentra la expresión $i * 3$, por tanto, el primer número que se va a mostrar será 3, el siguiente 6, y así sucesivamente hasta mostrar los primeros 20 múltiplos de 3.

```
Proc: Múltiplos de 3
| Entero: i
| Para(i = 1, 20, 1)Haga
| | Escriba i * 3
| Fin_Para
Fin_proc
```

Ejemplo 3.84 Factores primos

Realice un algoritmo que muestre los factores primos de un número n.

$$78 \implies \boxed{\textbf{A}} \implies 2, 3, 13$$

Análisis: El algoritmo utiliza dos variables: el número al que se le hallarán los factores primos n y un índice i que funcionará como divisor. Para hallar los factores primos de un número se divide el número por otro tantas veces como sea posible, es decir, mientras que el residuo sea cero, hecho que se realiza en el Mientras que interno; cuando no se pueda seguir dividiendo por

este número se avanzan dos cantidades, para repetir el proceso hasta que el número sea menor o igual a uno. Al dividir un número, por ejemplo 2, tantas veces como sea posible, este número no será divisible por 4, ni por 8, ni por 16, y así sucesivamente. Por lo tanto, si se entra en el Mientras que, el número con el que se entra es primo. Por consiguiente, en su interior se asigna a una variable booleana *sw* el valor de 1, que permitirá escribir el número si es primo y divide a n.

Además, primero se debe revisar el 2 y luego el resto de los números, de esta forma se evita ir de uno en uno, y se va de dos en dos, lo cual reduce a la mitad el número de cálculos que realizará el algoritmo. Lo anterior se debe a que no es necesario revisar si un número es par, puesto que ya se dividió el número por 2 tantas veces como fue posible, entonces el número no será divisible por un número par.

Por último, observe que se hace necesario el uso de estructuras Mientras que porque no se conoce cuántas veces se hará la división.

```
Proc: Factores primos
| Entero: n, sw, i
| Escriba "Digite un número"
| Lea n
| // Primero se va a dividir el número por 2 como sea posible
| // tantas veces como sea posible
| sw ← 0
| Mq(n mod 2 = 0)Haga
| | sw ← 1
| | n ← n/2
| Fin_Mq
| Si(sw = 1)Entonces
| | Escriba i, ", "
| F.Si
| i ← 3
| // Se procede a revisar los otros números
| Mq(n > 1)Haga
| | Mq(n mod i = 0)Haga
| | | sw ← 1
| | | n ← n/i
| | Fin_Mq
| | i ← i + 2
| | Si(sw = 1)Entonces
| | | Escriba i, ", "
| | F.Si
| Fin_Mq
Fin_proc
```

Ejemplo 3.85 Media y varianza

Realice un algoritmo que lea un número de datos y calcule la media y la varianza de estos.

Media:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} a_i = (a_1 + ... + a_n)/n$$

Varianza:

$$\sigma^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n} = \left(\frac{1}{n}\sum_{i=1}^{n} x_i^2\right) - \overline{x}^2$$

Por ejemplo,

$$1\ 2\ 3 \implies \boxed{\mathbf{A}} \implies \begin{array}{c}\text{Media: 2}\\ \text{Varianza: 0.66}\end{array}$$

Análisis: Se usarán las variables n y j para el número de datos que se van a computar y para controlar el ciclo Para, respectivamente. En la variable x se almacenará el dato ingresado por el usuario; *med* y *var* son la media y la varianza, respectivamente. Se utiliza un ciclo Para que va de 1 hasta el total de datos n que sirve para leer los datos e ir calculando la sumatoria de las fórmulas de la media y la varianza. Al finalizar el ciclo se divide *med* entre n y se tiene la media de los datos; también se realizan los cálculos restantes para obtener la varianza y se imprimen ambas por pantalla.

Proc: Media y varianza
| **Real:** *med*, *var*, x
| **Entero:** n, j
| $med \leftarrow 0, var \leftarrow 0$
| **Escriba** "Por favor, digite el número de datos"
| **Lea** n
| **Para**$(j = 1, n, 1)$**Haga**
| | **Escriba** "Digite el dato número", j
| | **Lea** x
| | $med \leftarrow med + x$
| | $var \leftarrow var + x * x$
| **Fin_Para**
| $med \leftarrow med/n$
| $var \leftarrow ((var/n) - med * med)$
| **Escriba** "La media de los ", n, "datos es:", *med*
| **Escriba** "La varianza de los ", n, "datos es:", *var*
Fin_proc

Ejemplo 3.86 Serie de Taylor de $sin(x)$

Realice un algoritmo que calcule el seno de un ángulo utilizando la serie de Taylor del seno; dicha serie está definida de la siguiente forma:

$$\sin x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1}$$

Análisis: La variable n determina hasta qué término se va a evaluar la serie. En realidad, no se necesitan infinitos términos para evaluar la serie. Con 10 términos que se evalúen se cuenta con una muy buena aproximación del seno de un ángulo. La variable *seno* contendrá la suma de los términos; *pi* es la constante π, la cual se declara en 3.1416. Se le pedirá al usuario que ingrese el ángulo al cual se le desea calcular su seno. El usuario ingresa el ángulo en grados, por lo tanto se hace necesaria una conversión, puesto que x en la serie debe estar en radianes. Recuerde que para pasar de grados a radianes el ángulo debe multiplicarse por $pi/180$.

La variable vf corresponde al exponente de -1 y al exponente de x en la serie. A su vez, hay que calcular el factorial de vf. En la variable $fact$ se almacena el cálculo del factorial de vf utilizando un ciclo Para que se ejecuta en cada iteración del ciclo Para externo y va desde 1 hasta vf; al finalizar este ciclo Para, a la variable *termino* se le asigna $(-1)^j$ para calcular el signo positivo o negativo, multiplicado por $x^{vf}/fact$; como se tiene calculado el término, se adiciona al total de *seno*. Cuando finaliza el ciclo Para externo se escribe el resultado.

$$45 \implies \boxed{\mathbf{A}} \implies \sin 45 = 0.7071$$

Proc: Cálculo de la serie de Taylor de la función $\sin x$
| **Entero:** n, j, $fact$, vf
| **Real:** *seno*, *pi*, x, *termino*
| $n \leftarrow 50$
| $seno \leftarrow 0, pi \leftarrow 3.1416$
| **Escriba** "Escriba el ángulo en grados: "
| **Lea** x
| $x \leftarrow pi * x/180$
| **Para**$(j = 0, n, 1)$**Haga**
| | $fact \leftarrow 1$
| | $vf \leftarrow 2 * j + 1$
| | **Para**$(i = 1, vf, 1)$**Haga**
| | | $fact \leftarrow fact * i$
| | **Fin_Para**
1 2

```
1 2
|  |  termino ← (−1)^j ∗ x^vf/fact
|  |  seno ← seno + termino
|  Fin_Para
|  Escriba "sin(", x, ") = ", seno
Fin_proc
```

Observe que en cada iteración se están haciendo cálculos de más. Vea que se puede usar el término anterior para calcular el siguiente. Para ello note que

$$\frac{(-1)^n}{(2n+1)!}x^{2n+1} = \frac{-1}{(2n+1)(2n)}x^2 \cdot \frac{(-1)^{n-1}}{(2n-1)!}x^{2n-1}$$

Por lo tanto, no es necesario estar calculando el factorial de vf en cada iteración. El primer término de la serie es

$$\frac{(-1)^0}{(2\cdot 0+1)!}x^{2\cdot 0+1} = x. \tag{3.1}$$

A continuación se muestra una nueva solución usando lo anterior.

```
Proc: Cálculo de la serie de Taylor de la función sin x
|  Entero: n, j
|  Real: seno, pi, x, termino
|  n ← 50
|  seno ← 0, pi ← 3.1416
|  Escriba "Escriba el ángulo en grados: "
|  Lea x
|  x ← pi ∗ x/180
|  termino ← x
|  seno ← termino
|  Para(j = 1, n, 1)Haga
|  |  // Se actualiza la variable termino según la expresión (3.1)
|  |  termino ← termino ∗ (−1) ∗ x^2/((2 ∗ j + 1) ∗ (2 ∗ j))
|  |  seno ← seno + termino
|  Fin_Para
|  Escriba "sin(", x, ") = ", seno
Fin_proc
```

Ejemplo 3.87 Serie de Taylor de $\exp(x)$

Realice un algoritmo que calcule la serie exponencial; dicha serie está definida por la serie de Taylor de $\exp x$:

$$e^x = \sum_{j=0}^{\infty} \frac{x^j}{j!}$$

Análisis: De forma similar al problema anterior, primero se va a identificar cómo calcular el término n-ésimo en función del anterior, es decir, del $n-1$. Observe que

$$\frac{x^n}{n!} = \frac{x}{n} \cdot \frac{x^{n-1}}{(n-1)!} \tag{3.2}$$

La variable *termino* contiene el término actual de la serie. Observe que el primer término de la serie es $x^0/0! = 1$.

Por ejemplo, si x es igual a uno, el algoritmo debería mostrar el valor del número de Euler, e, o aproximadamente 2.7182.

$$1 \implies \boxed{\mathbf{A}} \implies e^1 = 2.7182818$$

Proc: Serie de Taylor de la función exponencial
| **Entero:** n, j
| **Real:** exp, x, $termino$
| $n \leftarrow 50$
| $exp \leftarrow 0$
| **Escriba** "Escriba el valor de x: "
| **Lea** x
| $termino \leftarrow 1$
| $exp \leftarrow 1$
| **Para**$(j = 1, n, 1)$**Haga**
| | // Se actualiza la variable *termino* según la expresión (3.1)
| | $termino \leftarrow termino * x/j$
| | $exp \leftarrow exp + termino$
| **Fin_Para**
| **Escriba** "exp(", x, ") = ", exp
Fin_proc

Observe que la estructura del algoritmo es la misma que la del cálculo de la serie de Taylor de la función $\sin(x)$.

Ejemplo 3.88 Pi

El valor del número π se puede calcular con la precisión deseada sabiendo que la serie infinita del matemático alemán Leibnitz se define como

$$1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} \ldots \text{ tiende a } \frac{\pi}{4}$$

Escriba un algoritmo que calcule una aproximación al número π usando un número de términos suministrado por el usuario.

Análisis: La variable n será el número de términos que se generarán de la serie; j será la variable que servirá para controlar el ciclo Para que va desde 1 hasta n; *termino* es la variable en la que se almacena el valor de la serie que se calcula en cada iteración; la variable *signo* será la que indique si el término es positivo o negativo y se va alternando cada vez; *impar* es la variable en la que se almacenan los impares para ir generando la serie; *pi* será la suma de todos los términos.

Se utiliza un ciclo Para en el que en cada iteración se calcula un término de la serie y se van sumando hasta tener n términos, y es entonces cuando finaliza el ciclo; después se multiplica $pi * 4$, ya que la serie genera a $pi/4$, y por último se escribe el resultado de la aproximación.

Ejemplo: $N = 5 \Longrightarrow \frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} = 3.33968$

$$5 \implies \boxed{\mathbf{A}} \implies 3.33968$$

Observe que este problema es similar a los problemas anteriores pero más sencillo, puesto que es más fácil calcular el término nuevo en función del anterior.

Proc: Pi
- **Entero:** *signo*, *impar*, n
- **Real:** *pi*, *termino*
- $signo \leftarrow -1, impar \leftarrow 1$
- $pi \leftarrow 0, termino \leftarrow 0$
- **Escriba** "Escriba el número de términos para la aproximación: "
- **Lea** n
- **Para**$(j = 1, n, 1)$**Haga**
  - $signo \leftarrow signo * -1$
  - $termino \leftarrow signo * (1/impar)$
  - $pi \leftarrow pi + termino$
  - $impar \leftarrow impar + 2$
- **Fin_Para**
- $pi \leftarrow pi * 4$
- **Escriba** "La aproximación de pi es: ", *pi*

Fin_proc

Ejemplo 3.89 Juego de cartas

Realice un algoritmo que simule un juego de cartas entre dos jugadores; gana el que muestre el valor mayor 3 veces consecutivas.

Análisis:

Jugador 1: 3 5 7
Jugador 2: 2 4 6 $\implies \boxed{\mathbf{A}} \implies$ El ganador es el jugador 1

Primero se utilizarán las variables i y j y se declararan en 0. En ellas se almacenará el número de veces que gana cada jugador, respectivamente. Se utilizará un ciclo Mientras que, puesto que no se conoce el número de iteraciones totales que se va a ejecutar por el ciclo. En él se leerán las dos cartas, se pregunta cuál de las dos es mayor y se aumenta el contador del jugador respectivo; posteriormente se reinicia el contador del jugador contrario; el ciclo se acaba cuando uno de los dos contadores llegue a 3. De esta forma, solo se puede ganar cuando se consiga tres victorias consecutivas.

```
Proc: Juego de cartas
| Entero: j, i, a, b
| j ← 0, i ← 0
| Mq(i ≠ 3 y j ≠ 3)Haga
| | Escriba "Digite el valor de carta jugador 1:"
| | Lea a
| | Escriba "Digite el valor de carta jugador 2:"
| | Lea b
| | Si(a > b)Entonces
| | | i ← i + 1
| | | j ← 0
| | F.Si
| | Si(b > a)Entonces
| | | j ← j + 1
| | | i ← 0
| | F.Si
| Fin_Mq
| Si(i = 3)Entonces
| | Escriba "¡EL JUGADOR 1 ES EL GANADOR!"
| Sino
| | Si(j = 3)Entonces
| | | Escriba "¡EL JUGADOR 2 ES EL GANADOR!"
| | F.Si
| F.Si
Fin_proc
```

Ejemplo 3.90 Función real

Haga un algoritmo que reciba un número real y su salida sea

$$f(x) = \begin{cases} \frac{\pi}{2} + 7, & \text{si } x \geq 20 \\ x^2, & \text{si } 0 \leq x < 20 \\ x^8 - x^5, & \text{si } x < 0 \end{cases}$$

Análisis: Para hallar la salida del algoritmo simplemente se debe utilizar

el condicional Si y para cada condición se escribe la salida correcta. Por ejemplo:

$$\begin{array}{rcccc} 21 & \Longrightarrow & & \Longrightarrow & \frac{\pi}{2}+7 \\ 15 & \Longrightarrow & \boxed{\textbf{A}} & \Longrightarrow & 15^2 \\ -21 & \Longrightarrow & & \Longrightarrow & -21^8 - -21^5 \end{array}$$

Proc: Función real
| **Real:** x
| **Escriba** "Por favor, digite un número real"
| **Lea** x
| **Si**($x >= 0$)**Entonces**
| | **Si**($x >= 20$)**Entonces**
| | | **Escriba** "f(x)= pi/2+7*", x
| | **Sino**
| | | **Escriba** "f(x)= ", x, "^2"
| | **F.Si**
| **Sino**
| | **Escriba** "f(x)= ", x, "^8 − ", x, "^ 5"
| **F.Si**
Fin_proc

Ejemplo 3.91 Ver si un dígito está o no

Realice un algoritmo que lea un número y diga si contiene un dígito d.

Análisis: En el algoritmo se emplean las siguientes variables: n para guardar el número que se va a verificar; cn es una copia de n, donde está el dígito que se va a buscar; d es el dígito a buscar; dig guarda cada dígito a contrastar, y una variable booleana; sw, inicializada en falso y que solo cambia su valor en caso que el número sí contenga al dígito.

Por ejemplo,

$$\begin{array}{rcccl} n = 5489,\ d = 9 & \Longrightarrow & \boxed{\textbf{A}} & \Longrightarrow & \text{el dígito 9 está en el número.} \\ n = 5484,\ d = 9 & \Longrightarrow & & \Longrightarrow & \text{el dígito 9 NO está en el número.} \end{array}$$

Para saber si d está contenido se debe ir dividiendo el número entre 10 y sacando el residuo módulo 10 (si se divide entre 10, el residuo será el último dígito). Luego se compara este residuo con el dígito que se está buscando. Dicha división sucesiva se realiza por medio de un Mientras que hasta que se encuentre el dígito, colocando la variable sw en 1, o hasta que el número sea cero. En este último caso quiere decir que el dígito d no está contenido, por tanto, el sw se mantiene en falso. Al final se imprime el resultado.

```
Proc: Ver si un dígito esta o no
| Entero: sw, n, d, cn, dig
| sw ← 0
| Escriba "Por favor, digite el número y el dígito que se va a buscar"
| Lea n, d
| cn ← n
| Mq(n > 0 y sw = 0)Haga
| | dig ← n mod 10
| | Si(dig = d)Entonces
| | | sw ← 1
| | F.Si
| | n ← n/10
| Fin_Mq
| Si(sw = 1)Entonces
| | Escriba "El número", cn, "contiene el dígito ", d
| Sino
| | Escriba "El número", cn, "no contiene el dígito ", d
| F.Si
Fin_proc
```

Ejemplo 3.92 El dígito 2

Haga un algoritmo que lea un grupo de números de cuatro dígitos, cuente los que contienen el número dos y sume los que no lo contienen.

Análisis: Suponga que el usuario ingresa los números 2358, 2587, 3548 y 3597.

$$2358, 2587, 3548, 3597 \implies \boxed{\textbf{A}} \implies \begin{array}{c}\text{Números que contienen al dos: 2}\\ \text{Suma de los que no lo tienen: 7145.}\end{array}$$

La salida es esa, puesto que hay dos números que contienen el dos (2358 y 2587), mientras que los números que no lo contienen (3548 y 3597) suman 7145.

Para el algoritmo se utilizarán las siguientes variables: n para la cantidad de datos que se van a leer, *num* para cada número leído; $d1$, $d2$, $d3$, $d4$ para guardar los dígitos que componen el número leído y dos variables acumulativas; *cont* que llevará el registro de cuántos contienen al dígito 2, y *sum* la suma de los que no lo contienen.

Se utiliza un ciclo Para que va desde uno hasta la cantidad de datos que se leerán. Por cada número leído se evalúa si contiene o no al 2; para esto se separan sus dígitos; como el enunciado dice que solo contiene 4, entonces en $d1$ se guarda la unidad; para esto se aplica la operación **mod** del número

con 10, en $d2$ la decena; para esto es necesario primero dividir el número por 10 para luego obtener el residuo módulo 10; de igual forma se obtiene la centena, asignada a $d3$, pero esta vez dividiéndolo por 100; por último, para sacar la unidad de mil, que se guarda en $d4$, basta con dividir el número entre 1000. Cuando se tiene todos los dígitos se procede a verificarlos, y si por lo menos uno es igual a 2, entonces se incrementa *cont* en 1; si no a la suma que se lleva se le agrega el número.

Proc: Dígito 2
| **Entero:** $cont$, sum, j, $d1$, $d2$, $d3$, $d4$, n, num
| $cont \leftarrow 0, sum \leftarrow 0$
| **Escriba** "Digite el número de datos"
| **Lea** n
| **Para**$(j = 1, n, 1)$**Haga**
| | **Escriba** "Digite el dato número", j
| | **Lea** num
| | $d1 \leftarrow num$ **mod** 10
| | $d2 \leftarrow (num\, DIV\, 10)$ **mod** 10
| | $d3 \leftarrow (num\, DIV\, 100)$ **mod** 10
| | $d4 \leftarrow num\, DIV\, 1000$
| | **Si**$(d1 = 2$ o $d2 = 2$ o $d3 = 2$ o $d4 = 2)$**Entonces**
| | | $cont \leftarrow cont + 1$
| | **Sino**
| | | $sum \leftarrow sum + num$
| | **F.Si**
| **Fin_Para**
| **Escriba** "Los números que contienen el dígito dos son: ",cont
| **Escriba** "La suma de los números que no contienen el dígito dos es: ", sum
Fin_proc

Ejemplo 3.93 Calculadora

Cree un algoritmo que permita sumar, multiplicar y dividir dos números pero que no permita la división por 0 y despliegue un mensaje apropiado cuando se intente dicha división.

Análisis: Se utilizarán las variables $num1$ y $num2$ para almacenar los números digitados por el usuario. La variable *cod* almacenará la opción digitada por el usuario: 1 para sumar, 2 para multiplicar y 3 para dividir. Una estructura Dependiendo De escribirá los resultados correspondientes a la operación seleccionada por el usuario. En el caso de la división, antes de realizarla se verifica si $num2 \neq 0$. En caso de que el usuario no ingrese una opción entre las mostradas, se despliega un mensaje.

```
Proc: Calculadora que no permite la división por 0
| Real: num1, num2
| Entero: cod
| Escriba "Digite el primer número":
| Lea num1
| Escriba "Digite el segundo número":
| Lea num2
| Escriba "1 para Sumar"
| Escriba "2 para Multiplicar"
| Escriba "3 para Dividir"
| Escriba "Seleccione la operación a realizar: "
| Lea cod
| DD(cod)Haga
| | Opción '1' :
| | | Escriba num1, "+", num2, "=", num1 + num2
| | Fin
| | Opción '2' :
| | | Escriba num1, "*", num2, "=", num1 * num2
| | Fin
| | Opción '3' :
| | | Si(num2 ≠ 0)Entonces
| | | | Escriba num1, "/", num2, "=", num1/num2
| | | Sino
| | | | Escriba "La división por 0 no está definida. Inténtelo nuevamente."
| | | F.Si
| | Fin
| | Sino :
| | | Escriba "No seleccionó una opción válida."
| | Fin
| Fin_DD
Fin_proc
```

Ejemplo 3.94 Ecuación cuadrática

Realice un algoritmo que dado los tres coeficientes (a, b, c) muestre el resultado de la ecuación cuadrática de la forma $ax^2 + bx + c$. Asuma que $|a| + |b| > 0$.

Análisis: Las variables a, b y c son los coeficientes de una ecuación cuadrática; las variables $x1$ y $x2$ son las raíces de la misma y dependerán del discriminante de la ecuación, el cual se almacena en la variable dis. Con el discriminante se determina si $x1$ y $x2$ son reales o complejas.

Se le solicita al usuario que ingrese en las variables a, b, c los coeficientes de la ecuación. Se calcula el discriminante y se verifica si es negativo o positivo,

y las raíces serán complejas o reales, respectivamente. Finalmente se escribe el valor de las raíces.

Por ejemplo, si $a = 3, b = 2, c = 1$, entonces el algoritmo debe mostrar $x_1 = -\frac{1}{3} - \frac{\sqrt{2i}}{3}$ y $x_2 = -\frac{1}{3} + \frac{\sqrt{2i}}{3}$.

Proc: Ecuación cuadrática
- **Real:** a, b, c, dis
- **Escriba** "Por favor, digite los coeficientes de la ecuación: a, b y c"
- **Lea** a, b, c
- $dis \leftarrow b^2 - 4 * a * c$
- **Si**$(dis < 0)$**Entonces**
  - **Escriba** "$x1 =$", $-b/2 * a$, " + ", $\sqrt{-dis/2a}$, "i"
  - **Escriba** "$x2 =$", $-b/2 * a$, " - ", $\sqrt{-dis/2a}$, "i"
- **Sino**
  - **Escriba** "$x1 =$", $-b + \sqrt{dis/2a}$
  - **Escriba** "$x2 =$", $-b - \sqrt{dis/2a}$
- **F.Si**

Fin_proc

Ejemplo 3.95 Suma de Gauss

Realice un algoritmo que lea un número entero y muestre la suma desde el 1 hasta el número digitado; demuestre que es igual al resultado de $\frac{n(n+1)}{2}$ (Fórmula de Gauss)

$$\sum_{i=1}^{n} i = 1 + 2 + 3 + ... + n = \frac{n\,(n+1)}{2}$$

Análisis: La variable n se utilizará para almacenar el número que se pide por pantalla; la variable del control del ciclo Para es j; a la variable *suma* se le asigna el cálculo de la sumatoria con la fórmula, y la variable s contiene la suma calculada sumando los términos desde 1 hasta n. Las variables *suma* y s son inicializadas en 0. Por último, se escribe el valor de s y *suma*.

$$3 \implies \boxed{\mathbf{A}} \implies \text{La suma es igual a } 6$$

Proc: Verificación de la Fórmula de Gauss
- **Entero:** s, n
- $s \leftarrow 1$
- **Escriba** "Por favor, digite un número entero"
- **Lea** n

1

```
1
| Si(n = 1)Entonces
| | Escriba 1
| Sino
| | Escriba 1
| | Para(j = 2, n, 1)Haga
| | | s ← s + j
| | | Escriba s
| | Fin_Para
| F.Si
Fin_proc
```

Ejemplo 3.96 Newton-Raphson

Haga un algoritmo que calcule la raíz cuadrada de un número suministrado por el usuario, utilizando el algoritmo de Newton. A es el número que digitó el usuario (es decir, al que se le desea hallar la raíz) y X representa el valor que se le va a mostrar al usuario (es decir, la raíz cuadrada de A), que inicialmente puede ser igual a A. Realice este proceso hasta que $|X_{n+1} - X_n|$ sea menor que una tolerancia *tol* ingresada por el usuario.

$$X_{n+1} = X_n + \frac{1}{2}\left(\frac{A}{X_n} - X_n\right)$$

Análisis: Observe que en este algoritmo se calcula el valor de X_{n+1} en función del valor anterior, es decir, X_n, por lo tanto se requiere una "semilla" o un valor de inicio. Se tomará como $X_0 = A/2$ una primera aproximación al valor de la raíz de A. Ahora bien, este proceso se debe repetir hasta que $|X_{n+1} - X_n| < tol$. La variable xn contendrá la iteración anterior, mientras que la variable $xn1$ contendrá la iteración actual. La variable dif contiene la diferencia entre dos aproximaciones consecutivas.

Por ejemplo, si el usuario ingresa 9, el algoritmo debe mostrar 3.

$$9 \implies \boxed{\mathbf{A}} \implies 3$$

El algoritmo mostrado corresponde a un caso particular del algoritmo de Newton-Raphson. En general, si se tiene una ecuación de la forma $f(x) = 0$, una manera de aproximar la solución de esta en función de una aproximación anterior x_n es

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

```
Proc: Cálculo de la raíz de un número
  Real: a, tol, xn, xn1, dif
  Escriba "Digite un número real: "
  Lea a
  Escriba "Digite la tolerancia: "
  Lea tol
  xn ← a
  xn1 ← xn + 0.5 * (a/xn − xn)
  dif ← |xn − xn1|
  Mq(dif ≥ tol)Haga
    xn ← xn1
    xn1 ← xn1 + 0.5 * (a/xn1 − xn1)
    dif ← |xn − xn1|
  Fin_Mq
  Escriba "La raíz cuadrada de ", a, " es igual a ", xn
Fin_proc
```

Ejemplo 3.97 Calificaciones

Cuatro enteros entre 0 y 100 representan las puntuaciones de un estudiante de un curso de Algoritmia. Escriba un algoritmo para encontrar la media de estas puntuaciones y visualice una tabla de notas de acuerdo con el siguiente cuadro:

Media	**Puntuación**
$90 - 100$	A
$80 - 89$	B
$70 - 79$	C
$60 - 69$	D
$0 - 59$	E

Análisis: Una vez leída una calificación, se debe determinar qué puntuación le corresponde. Como se ha visto anteriormente, la mejor forma de hacerlo es preguntar en orden. Por lo tanto, se pregunta si la calificación es mayor que 89; en caso contrario, si es mayor que 79, y así sucesivamente. Se repite el proceso para las otras tres calificaciones con el uso de un ciclo Para, y en cada Si-Sino se almacena la letra que corresponde en una variable de tipo carácter; al final se escribe la calificación. Lo anterior evita colocar un Escriba en el interior de cada Si-Sino. Por último, se acumula en *prom* la suma de los enteros correspondientes a cada calificación, para luego ser dividido entre 4. Este último resultado corresponde a la media aritmética.

```
Proc: Calificaciones
  Real: prom
  Entero: i, n, entero
  Caracter: cal
  prom ← 0
  Lea n
  Para(i = 1, n, 1)Haga
    Escriba "Por favor, ingrese la nota ", i, ": "
    Lea entero
    Si(entero > 89)Entonces
      cal ← A
    Sino
      Si(entero > 79)Entonces
        cal ← B
      Sino
        Si(entero > 69)Entonces
          cal ← C
        Sino
          Si(entero > 59)Entonces
            cal ← D
          Sino
            cal ← E
          F.Si
        F.Si
      F.Si
    F.Si
    Escriba "Calificación ", i, ": ", cal
    prom ← prom + entero
  Fin_Para
  prom ← prom/n
  Escriba "La media de las calificaciones es: ", prom
Fin_proc
```

Ejemplo 3.98 Fórmula de Herón

El área de un triángulo cuyos lados son a, b, c se puede calcular con la fórmula de Herón:

$$A = \sqrt{s(s-a)(s-b)(s-c)}$$

donde s es el semiperímetro del triángulo y está dado por $s = (a+b+c)/2$. Escriba un algoritmo que lea las longitudes de los tres lados de un triángulo y calcule el área del triángulo.

Análisis: Observe que no se le pide al algoritmo verificar que las longitudes de los lados sean números reales positivos; por lo tanto, el algoritmo corresponde a solamente lectura de datos y asignación.

Por otro lado, si a, b, c son los lados del triángulo, según el teorema del coseno:

$$\cos C = \frac{a^2 + b^2 - c^2}{2ab}$$

donde C es el ángulo formado por los lados a y b. Sin embargo, la bien conocida identidad $\sin^2 C + \cos^2 C = 1$ implica

$$\sin C = \sqrt{1 - \cos^2 C} = \sqrt{\frac{4a^2b^2 - (a^2 + b^2 - c^2)^2}{2ab}}.$$

Por último, recuerde que el área de este triángulo está dada por $A = \frac{1}{2}ab \sin C$. Remplazando y reorganizando términos se llega a la fórmula de Herón.

Por ejemplo, si la longitud de los lados es 3, 4 y 5, entonces el semiperímetro es $s = \dfrac{3+4+5}{2} = 6$. Luego, $s(s-a)(s-b)(s-c) = 6 \cdot (6-3)(6-4)(6-5) = 36$. Finalmente, el área del triángulo es $A = \sqrt{36} = 6$.

Proc: Fórmula de Herón
| **Real:** a, b, c, p, $area$
| **Escriba** "Ingrese los lados del triángulo: "
| **Lea** a, b, c
| $p \leftarrow (a + b + c)/2$
| $area \leftarrow \sqrt{p(p-a)(p-b)(p-c)}$
| **Escriba** "El área del triángulo es ", $area$
Fin_proc

Ejemplo 3.99 Formato de fecha

Escriba un programa que acepte tres números enteros, los cuales corresponderán a: el primero, el día; el segundo, el mes; el tercero, el año de una fecha. Y a continuación muestre en formato '*dia* de *mes* de *año*' para visualizar el mes.

Análisis: Dado que no se está pidiendo verificar si la fecha es válida, se asumirá que el usuario ingresa una fecha válida. A su vez, puesto que el mes se ingresa como un número entero, entonces hay que determinar a qué mes corresponde. Por otro lado, resulta más conveniente usar un Dependiendo De que anidar cláusulas Si-Sino. Por ejemplo, si el usuario ingresa 1, 1, 1999, la salida debe ser "1 de enero de 1999".

Proc: Formato de fecha
| **Entero:** a, b, c
| **Escriba** "Ingrese los números correspondientes a la fecha: "
| **Lea** a, b, c
1

```
1
| DD(b)Haga
| | Opción 1 :
| | | Escriba a, " de enero de ", c
| | Fin
| | Opción 2 :
| | | Escriba a, " de febrero de ", c
| | Fin
| | Opción 3 :
| | | Escriba a, " de marzo de ", c
| | Fin
| | Opción 4 :
| | | Escriba a, " de abril de ", c
| | Fin
| | Opción 5 :
| | | Escriba a, " de mayo de ", c
| | Fin
| | Opción 6 :
| | | Escriba a, " de junio de ", c
| | Fin
| | Opción 7 :
| | | Escriba a, " de julio de ", c
| | Fin
| | Opción 8 :
| | | Escriba a, " de agosto de ", c
| | Fin
| | Opción 9 :
| | | Escriba a, " de septiembre de ", c
| | Fin
| | Opción 10 :
| | | Escriba a, " de octubre de ", c
| | Fin
| | Opción 11 :
| | | Escriba a, " de noviembre de ", c
| | Fin
| | Opción 12 :
| | | Escriba a, " de diciembre de ", c
| | Fin
| Fin_DD
Fin_proc
```

Ejemplo 3.100 Redondeo

Un archivo de datos contiene los cuatro dígitos, A, B, C, D, de un entero positivo N. Se desea redondear N a la centena más próxima y visualizar la

salida. Por ejemplo, si A es 2, B es 3, C es 6 y D es 2, entonces N será 2362 y el resultado redondeado será 2400. Si N es 2300, el resultado será 2300. Diseñe el algoritmo correspondiente.

Análisis: Una vez leídos A, B, C, D, se tiene el número $N = \overline{ABCD}$, y su representación decimal corresponde a $1000A + 100B + 10C + D$. Se quiere aproximar a su centena más cercana, entonces C y D serían 0 y a B se le suma 1 o se deja igual según sea el caso. Pero si B es 9, esta suma no se ve afectada, pues B se vuelve 0 y A se incrementa en 1. Entonces $N1$ y $N2$ son las aproximaciones por encima y por debajo del número N, respectivamente; dependiendo de las restas $N1 - N$ y $N - N2$ se decide qué centena es más próxima a $N = \overline{ABCD}$.

Proc: Redondeo
| **Entero:** A, B, C, D, N, $N1$, $N2$
| **Escriba** "Ingrese los dígitos A,B,C,D: "
| **Lea** A, B, C, D
| $N \leftarrow 1000 * A + 100 * B + 10 * C + D$
| $N1 \leftarrow 1000 * A + (B + 1) * 100$
| $N2 \leftarrow 1000 * A + B * 100$
| **Si**$(N1 - N \leq N - N2)$**Entonces**
| | **Escriba** $N1$
| **Sino**
| | **Escriba** $N2$
| **F.Si**
Fin_proc

Ejemplo 3.101 Ecuaciones lineales

Un sistema de ecuaciones lineales

$$\begin{aligned} ax + by &= c \\ dx + ey &= f \end{aligned}$$

se puede resolver con las siguientes fórmulas:

$$\begin{aligned} x &= \frac{ce - bf}{ae - bd} \\ y &= \frac{af - cd}{ae - bd} \end{aligned}$$

Diseñe un programa que lea los coeficientes a, b, c, d y f (en ese orden) y muestre los valores de x e y.

Análisis: Se deben las variables y se hacen las asignaciones correspondientes de acuerdo con las fórmulas presentadas.

Proc: Ecuaciones lineales
| **Real:** a, b, c, d, e, f, x, y
| **Escriba** "Ingrese los valores de a,b,c,d,e,f: "
| **Lea** a, b, c, d, e, f
| $x \leftarrow (c * e - b * f)/(a * e - b * d)$
| $y \leftarrow (a * f - c * d)/(a * e - b * d)$
| **Escriba** "El valor de x es: ", x
| **Escriba** "El valor de y es: ", y
Fin_proc

Ejemplo 3.102 Fabricante

Cada unidad de disco en un embarque de estos dispositivos tiene estampado un código del 1 al 4, el cual indica el fabricante de la unidad como sigue:

Código	**Fabricante de la unidad de disco**
1	3M Corporation
2	Maxell Corporation
3	Sony Corporation
4	Verbatim Corporation

Escriba un algoritmo que acepte el número de código como una entrada y con base en el valor introducido despliegue el fabricante de la unidad de disco correcto.

Análisis: La mejor forma de resolver este problema es utilizando la primitiva Dependiendo De: simplemente se le pide al usuario que digite el código del fabricante y luego se establecen las opciones del caso; de esta forma, dependiendo del código del fabricante, este escribirá la empresa asociada al código.

Proc: Fabricantes
| **Entero:** *cod*
| **Escriba** "Código del Fabricante: "
| **Lea** *cod*
| **DD**(*cod*)**Haga**
| | **Opción** 1 :
| | | **Escriba** "La unidad de disco fue fabricada por 3M Corporation"
| | **Fin**
| | **Opción** 2 :
| | | **Escriba** "La unidad de disco fue fabricada por Maxell Corporation"
| | **Fin**
| | **Opción** 3 :
| | | **Escriba** "La unidad de disco fue fabricada por Sony Corporation"
| | **Fin**
1 2

```
1 2
|  |  Opción 4 :
|  |  |  Escriba "La unidad de disco fue fabricada por Verbatim Corporation"
|  |  Fin
|  |  Sino  :
|  |  |  Escriba "No digitó un código válido"
|  |  Fin
|  Fin_DD
Fin_proc
```

Ejemplo 3.103 Serie armónica

Calcule la suma de la serie armónica

$$\frac{1}{1}+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{N}$$

donde N es un número que se introduce por teclado.

Análisis: Se utilizarán una variable N para almacenar la cantidad de términos por sumar; la variable i controlará el ciclo y la variable *suma* almacenará el resultado de la serie. Es importante resaltar que la variable *suma* es inicializada en cero porque corresponde a una variable que cumple la función de acumular.

```
Proc: Serie armónica
|  Real: suma
|  Entero: n, i
|  suma ← 0
|  Escriba "Ingrese el numero N: "
|  Lea n
|  Para(i = 1, n, 1)Haga
|  |  suma ← suma + 1/i
|  Fin_Para
|  Escriba "Resultado: ", suma
Fin_proc
```

Ejemplo 3.104 Promedio de datos I

Se llevaron a cabo cuatro experimentos, cada uno consistente en seis resultados de prueba. Escriba un algoritmo que lea los seis datos por experimento y muestre el promedio de los datos por cada experimento.

Análisis: Se utilizarán las variables $dato, prom, suma$ para los datos, el promedio y la suma de los datos, respectivamente. A su vez, dos ciclos anidados, uno para recorrer los experimentos y otro para leer los datos de cada uno.

El ciclo externo controla el número de experimentos y el ciclo interno controla la lectura de los datos de cada experimento. Los datos leídos de un experimento se utilizan para calcular el promedio, dividiendo el acumulador *suma* entre el número de datos leídos, que es 6, y se escribe este promedio. A la variable *suma* se le asigna cero antes de iniciar la lectura de datos de cada experimento, para no acumular valores de los experimentos anteriores que alteren el resultado. La variable i presente en cada Escriba permite enumerar los experimentos de forma que se diferencien cuando se escriben las salidas.

```
Proc: Promedio de datos I
    Entero: i, j
    Real: suma, dato, prom
    Para(i = 1, 4, 1)Haga
        suma ← 0
        Para(j = 1, 6, 1)Haga
            Escriba "Escriba dato ", j, " del experimento ", i
            Lea dato
            suma ← suma + dato
        Fin_Para
        prom ← suma/6
        Escriba "Promedio experimento ", i, ": ", prom
    Fin_Para
Fin_proc
```

Ejemplo 3.105 Promedio de datos II

Modifique el algoritmo anterior de modo que pueda introducirse un número diferente de resultados de prueba para cada experimento. El usuario primero debe introducir el número de pruebas de dicho experimento.

Análisis: Para este ejercicio, el número de iteraciones en el ciclo interior está controlado por información ingresada por el usuario. Por lo tanto, se deben agregar las instrucciones para leer el número de datos de cada experimento; esto se hará en la variable *num*; y en el ciclo interior, el promedio es calculado dividiendo la variable *suma* entre la variable *num*, que indica el número de datos leídos por experimento.

```
Proc: Promedio de datos II
    Entero: i, j, num
    Real: suma, dato, prom
    Para(i = 1, 4, 1)Haga
        Escriba "Ingrese numero de datos del experimento ", i
        Lea num
        suma ← 0
1 2
```

```
1 2
|  |  Para(j = 1, num, 1)Haga
|  |  |  Escriba "Escriba dato del experimento ", i
|  |  |  Lea dato
|  |  |  suma ← suma + dato
|  |  Fin_Para
|  |  prom ← suma/num
|  |  Escriba "Promedio experimento ", i, ": ", prom
|  Fin_Para
Fin_proc
```

Ejemplo 3.106 Menor número triangular mayor que n

Encuentre el número natural N más pequeño tal que la suma de los N primeros números exceda una cantidad introducida por el teclado.

Análisis: En este ejercicio, la variable N determinará el límite de la suma acumulada en la variable *suma*. Se utiliza un ciclo Mientras que, y se van sumando todos los números naturales hasta que el valor de la variable *suma* supere el valor de la variable N. En este momento se conocería el valor de N. Por ejemplo, si la cantidad introducida es 25, entonces la salida debe ser $N = 7$, puesto que $1+2+3+4+5+6+7 = 28 > 25$ y $1+2+3+4+5+6 = 21 < 25$.

```
Proc: Menor número triangular
|  Entero: i, suma, n
|  i ← 1
|  suma ← 1
|  Escriba "Ingrese el número N:"
|  Lea n
|  Mq(suma ≤ n)Haga
|  |  i ← i + 1
|  |  suma ← suma + i
|  Fin_Mq
|  Escriba i
Fin_proc
```

Ejemplo 3.107 Número de Euler

El valor del número de Euler, e, puede aproximarse usando la fórmula

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \frac{1}{5!} + \cdots$$

Esta serie se conoce como la expansión de Taylor de la función $y = e^x$ alrededor de $x = 0$. Usando esta fórmula escriba un algoritmo que aproxime

el valor de e usando un ciclo Para con 50 iteraciones, es decir, se evaluará la serie hasta los primeros 50 términos.

Análisis: Se declara la variable real *euler* en 0, que funcionará como un acumulador de los términos que tienen la forma $k = \frac{1}{k!}$. En el ciclo interno se calcula el factorial y se guarda en la variable $fact$. A la variable $fact$ se le asigna 1 antes de calcular el factorial del término correspondiente, para no acumular resultados de las iteraciones anteriores. Recuerde que cuando una variable acumula el resultado de la suma de varios términos debe asignársele el módulo de la suma: 0. De manera similar, cuando una variable acumula el resultado de una multiplicación de varios términos, se le asigna el módulo de la multiplicación: 1. Por otro lado, en el ciclo interno, la variable de control j debe ir hasta el valor de i, puesto que indica la cantidad de términos que deben multiplicarse para obtener el factorial correspondiente. Luego de calcular el factorial se obtiene el término que corresponde a la iteración i, diviendo 1 entre el valor del factorial y acumulando el resultado en la variable *euler*.

```
Proc: Número de Euler
| Real: euler, termino
| Entero: i, j, fact
| euler ← 0
| termino ← 0
| Para(i = 0, 49, 1)Haga
| | fact ← 1
| | Para(j = 1, i, 1)Haga
| | | fact ← fact * j
| | Fin_Para
| | termino ← 1/fact
| | euler ← euler + termino
| Fin_Para
Fin_proc
```

Como solución alterna, se puede evitar el anidamiento, teniendo en cuenta que el factorial de un número k es: $k! = (k-1)! \cdot k$, siendo $k \in \mathbb{N}$. Se puede observar que la variable $fact$ en cada iteración del ciclo Para incrementa de tal forma que para la iteración k-ésima, $fact$ es igual a $k!$.

```
Proc: Número de Euler
| Real: euler, termino
| Entero: i, fact
| euler ← 1
| termino ← 0
| fact ← 1
| Para(i = 1, 49, 1)Haga
| | fact ← fact * i
| | termino ← 1/fact
1 2
```

```
1 2
| | euler ← euler + termino
| Fin_Para
Fin_proc
```

Ejemplo 3.108 Fahrenheit a Celsius

Escriba un algoritmo que despliegue una tabla de 20 conversiones de temperatura de Fahrenheit a Celsius. La tabla deberá comenzar con un valor Fahrenheit de 20 grados e incrementarse en valores de 4 grados. Recuerde que

$$°\mathrm{C} = \frac{5}{9}(°\mathrm{F} - 32)$$

Análisis: Se utilizan las variables $Celc$ y $Fahr$ para almacenar los valores de salida por cada iteración. En el ciclo Para, la variable $Fahr$ es inicializada en 20, y el incremento es de 4, de acuerdo con el enunciado del algoritmo. El límite del ciclo es noventa y seis, porque al ir incrementando de 4 en 4 los 20 valores de conversiones de temperatura llegan a su límite en este punto.

```
Proc: Fahrenheit a Celsius
| Entero: Fahr
| Real: Celc
| Para(Fahr =20,96,4)Haga
| | Celc ← (5.0/9.0) * (Fahr − 32)
| | Escriba "Grados Fahrenheit: ",Fahr," Grados Celsius: ",Celc
| Fin_Para
Fin_proc
```

En caso de que no se desee calcular el límite del Ciclo Para, se puede realizar un ciclo que inicie en 1, llegue hasta 20 y con incremento de 1. Dentro del ciclo se puede calcular a $Fahr$ como $Fahr = 4(i - 1) + 20$.

```
Proc: Fahrenheit a Celsius
| Entero: i
| Real: Fahr, Celc
| Para(i =1,20,1)Haga
| | Fahr ← 4 * (i − 1) + 20
| | Celc ← (5.0/9.0) * (Fahr − 32)
| | Escriba "Grados Fahrenheit: ",Fahr," Grados Celsius: ",Celc
| Fin_Para
Fin_proc
```

Ejemplo 3.109 Dilatación térmica

La expansión de un puente de acero conforme se calienta hasta una temperatura Celsius final, T_f, desde una temperatura Celsius inicial, T_0, puede aproximarse usando la fórmula

$$\text{Aumento de longitud} = a * L * (T_f - T_0)$$

donde a es el coeficiente de expansión (el cual para el acero es 11.7×10^{-6}) y L es el largo del puente a la temperatura T_0 . Usando esta fórmula escriba un algoritmo que despliegue una tabla de longitudes de expansión para un puente de acero que tiene 7365 metros de largo a 0 grados Celsius conforme la temperatura incrementa a 40 grados en incrementos de 5 grados.

Análisis: Se usará un ciclo Para desde 0 hasta 40 con incremento 5. Se calcula el aumento para cada iteración y se escribe el resultado. Por otro lado, aunque la fórmula original dice que hay que restar T_f y T_0, dado que la T_0 en este problema es igual a 0, siempre la resta va a hacer el valor que tenga la temperatura, T_f, por esto solo se usa una variable denominada T.

Proc: Dilatación térmica
  Real: a, *aumento*
  Entero: L, T
  $a \leftarrow 11.7 * 10^{-6}$
  $L \leftarrow 7365$
  Para$(T = 0, 40, 5)$**Haga**
    aumento $\leftarrow a * L * T$
    Escriba "Temperatura(Celsius): ", T
    Escriba "Aumento de longitud: ", *aumento*
  Fin_Para
Fin_proc

Ejemplo 3.110 Caída libre

Una pelota de golf es soltada desde un avión. La distancia d que cae la pelota en t segundos está dada por la ecuación $d = (\frac{1}{2})gt^2$, donde g es la aceleración debida a la gravedad y es igual a 32 pies/s^2. Usando esta información escriba un algoritmo que despliegue la distancia que cae en cada intervalo de un segundo para diez segundos y la distancia en que cae la pelota de golf al final de cada intervalo. La salida deberá completar la siguiente tabla:

Tiempo	Distancia en el intervalo actual	Distancia total
0	0.0	0.0
1	16.0	16.0
⋮	⋮	⋮
10	304	1600

Análisis: Se utiliza un ciclo Para para calcular la distancia de la pelota de golf. La variable de control t representa la variación del tiempo y es utilizada para calcular la salida del algoritmo. Esta variable t se actualiza en cada iteración.

Proc: Caída libre
  Real: dt, d
  Entero: g, t
  $dt \leftarrow 0$
  $g \leftarrow 32$
  Para$(t = 0, 10, 1)$**Haga**
    $d \leftarrow (1/2) * g * t^2$
    $dt \leftarrow dt + d$
    Escriba "Tiempo: ", t
    Escriba "Distancia en el intervalo: ", d
    Escriba "Distancia Total: ", dt
  Fin_Para
Fin_proc

Ejemplo 3.111 Números de tres cifras especiales

Calcule todos los números de tres cifras tales que la suma de los cubos de las cifras sea igual al valor del número. Es decir, si $\overline{abc} = n$, entonces el algoritmo debe contar este número siempre y cuando $a^3 + b^3 + c^3 = n$.

Análisis: Observe que los números de tres cifras van desde 100 hasta 999, por lo tanto se usará un ciclo Para que recorrerá todos estos números. Para calcular la suma $a^3 + b^3 + c^3$ se hace necesario extraer las tres cifras del número. La cifra de las unidades se puede extraer haciendo n **mod** 10, puesto que un número $n = \overline{abc}$ se puede expresar como $10\overline{ab}+c$, luego n **mod** $10 = c$. Recuerde que p **mod** q es el residuo de la división de p entre q.

Para extraer la segunda cifra se puede hacer uso de la operación /, la cual es una división entera. Si se realiza $n/10$, se obtiene $\overline{ab}$. Luego se realiza **mod** 10, y se extrae b, es decir, $b = (n/10)$ **mod** 10. Finalmente, vea que $c = n/100$.

Salida: Números enteros de tres cifras.

```
Proc: Números de tres cifras especiales
    Entero: n, a, b, c, cont
    Para(n = 100, 999, 1)Haga
        a ← n/100
        b ← (n/10) mod 10
        c ← n mod 10
        cont ← 0
        Si(n = a^3 + b^3 + c^3)Entonces
            cont ← cont + 1
        F.Si
    Fin_Para
    Escriba "Total de números: ", cont
Fin_proc
```

Ejemplo 3.112 Probabilidad exponencial

La probabilidad p de que una llamada telefónica individual dure menos de t minutos puede aproximarse por la función de probabilidad exponencial

$$p = 1 - e^{-t/a}$$

donde a es la duración de la llamada promedio y e es el número de Euler (2.71828). Usando esta ecuación de probabilidad escriba un algoritmo que calcule y despliegue una lista de probabilidades de la duración de una llamada que dure entre 1 y 10 minutos, en incrementos de un minuto.

Análisis: Por ejemplo, suponiendo que la duración promedio de la llamada es 2 minutos, la probabilidad que una llamada dure menos de 1 minuto se calcula como $1 - e^{-1/2} = 0.3297$. El cálculo de las probabilidades especificadas en el algoritmo se realizan utilizando una variable t que represente el tiempo y que además controle el ciclo Para. El cálculo de las probabilidades especificadas en el algoritmo se realizan utilizando una variable t que represente el tiempo y que además controle el ciclo Para.

```
Proc: Probabilidad exponencial
    Real: e, a
    Entero: t
    e ← 2.71828
    Escriba "Introduzca la duración de una llamada promedio: "
    Lea a
    Para(t = 1, 10, 1)Haga
        Escriba "La probabilidad de que una llamada dure menos de ", t,
        Escriba "minutos es de ", 1 − e^(−t/a))
    Fin_Para
Fin_proc
```

Ejemplo 3.113 Probabilidad de Poisson

El índice de llegadas de clientes a un banco concurrido en Nueva York puede estimarse usando la función de probabilidad de Poisson

$$P(x) = \frac{\lambda^x e^{-\lambda}}{x!}$$

Donde x es el número de clientes que llegan por minuto; λ es el número promedio de llegadas por minuto y e el número de Euler (2.71828). Usando la función de probabilidad de Poisson escriba un algoritmo que calcule y despliegue la probabilidad de llegada de 1 a 10 clientes en cualquier minuto teniendo en cuenta que el índice de llegadas promedio es de tres clientes por minuto.

Análisis: Por ejemplo, si el número promedio de clientes que entran en el banco es de tres clientes por minuto, entonces λ es igual a tres. Por tanto, la probabilidad de que un cliente llegue en cualquier minuto es

$$P(x = 1) = \frac{3^1 e^{-3}}{1!} = 0.149561$$

y la probabilidad que lleguen dos clientes en cualquier minuto es

$$P(x = 2) = \frac{3^2 e^{-3}}{2!} = 0.224454$$

En este algoritmo se utilizan un ciclo Para que controlará el número de clientes que entra al banco. La variable $fact$ se incrementa en cada ciclo; observe que no es necesario un ciclo interior para calcular el factorial. Por último, a la variable *prob* se le asigna el resultado de la fórmula expuesta en el algoritmo y se escribe la salida utilizando los valores calculados.

Proc: Probabilidad de Poisson
  Real: e, $prob$
  Entero: lam, $fact$, x
  $e \leftarrow 2.71828$
  $lam \leftarrow 3$
  $fact \leftarrow 1$
  Para$(x = 1, 10, 1)$**Haga**
    $fact \leftarrow fact * i$
    $prob \leftarrow (lam^x e^{-(lam)})/fact$
    Escriba "La probabilidad de que lleguen ", x,
    Escriba " clientes en cualquier minuto es ", $prob$
  Fin_Para
Fin_proc

Ejemplo 3.114 Formato de hora

Escriba un algoritmo que lea la hora de un día en notación de 24 horas y muestre la hora en notación de 12 horas. El programa pedirá al usuario que introduzca exactamente 4 números enteros entre 0 y 9. Así por ejemplo, las nueve en punto se introduce como 0900.

Análisis: Se deben leer los números correspondientes a la hora. En el condicional se hace una conversión teniendo en cuenta que un número expresado en notación decimal como $\overline{ab}$ es igual $10a+b$. Se pregunta si es mayor que 12 o no, lo cual decidirá si es a.m. o p.m. Finalmente, se escribe el resultado en formato de 12 horas. Por ejemplo, si la entrada es 1354, la salida será 1:54 p.m., y esto es porque $13 > 12$.

Proc: Formato de hora
| **Entero:** a, b, c, d
| **Escriba** "Ingrese los números correspondientes a la hora: "
| **Lea** a, b, c, d
| **Si**$(10 * a + b > 12)$**Entonces**
| | **Escriba** $a * 10 + b - 12$, ":", c, d, " p.m."
| **Sino**
| | **Escriba** $a * 10 + b$, ":", c, d, " a.m."
| **F.Si**
Fin_proc

Ejemplo 3.115 Tabla de valores para función real

Realice un algoritmo que muestre una tabla de valores dados: los intervalos, el incremento y las funciones.

Análisis: Para cada uno de los algoritmos se establece una estructura similar, compuesta por un ciclo Para y en la que la variable de control de cada uno es utilizada para calcular el valor de las funciones en los intervalos dados.

1. $y = 3x^5 - 2x^3 + x$ para $5 \leq x \leq 10$ en incrementos de 0.2
   Proc
   | **Entero:** x
   | **Para**$(x = 5, 10, 0.2)$**Haga**
   | | **Escriba** x, $3x^5 - 2x^3 + x$
   | **Fin_Para**
   Fin_proc
2. $y = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24}$ para $1 \leq x \leq 3$ en incrementos de 0.1

Proc
| **Entero:** x
| **Para**$(x = 1, 3, 0.1)$**Haga**
| | **Escriba** x, $1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24}$
| **Fin_Para**
Fin_proc

3. $y = 2e^{0.8t}$ para $4 \leq t \leq 10$ en incrementos de 0.2

Proc
| **Entero:** x
| **Real:** e
| $e \leftarrow 2.7183$
| **Para**$(t = 4, 10, 0.2)$**Haga**
| | **Escriba** t, $2 * e^{0.8*t}$
| **Fin_Para**
Fin_proc

Note que la evaluación de $e^{0.8*t}$ puede realizarse con la serie de Taylor antes descrita.

Ejemplo 3.116 Números romanos

Escriba un algoritmo que acepte un año menor que 4000 escrito en cifras arábigas y visualice el año escrito en números romanos. *Nota:* Recuerde que V=5, X=10, L=50, C=100, D=500, M=1000.

Análisis: Antes de analizar este problema observe los siguientes ejemplos de conversión:

IV=4	XL=40	CM=900
MCM=1900	MCML=1950	MCMLX=1960
MCMXL=1940	MCMLXXXIX=1989	

Un proceso de conversión exitoso sustraerá en orden descendiente las cantidades 1000, 900, 500, 400, ... El lector se preguntará: ¿por qué revisar 900? Se revisa 900 porque en los números romanos esto no se designa como $500 + 400$ sino como 900, es decir, no como $DCCCC$ sino como CD, donde CD es una unidad. Lo anterior quiere decir que los números romanos tienen símbolos tanto para $100, 500$ y 10000 como para 400 y 900. Por lo tanto, una conversión exitosa requiere que se vea a 400 y a 900 como símbolos también o como "bases" de este sistema numérico.

El algoritmo debe, por tanto, sustraer en orden decreciente. Si el número ingresado es mayor que 1000, sustrae mil tantas veces como sea necesario y se agrega 'M' a la salida tantas veces como se restó 1000. El número resultante es un número menor que 1000. Por lo explicado anteriormente, 900 también hace parte del sistema numérico; entonces hay que revisar si el número menor que 1000 es mayor que 900; de serlo, se sustrae 900. De manera similar se revisa para 500, 400, 100, ..., 1.

Proc: Números romanos
Entero: $numero$, r, i
Escriba "Ingrese el año a convertir: "
Lea $numero$
Si$(numero \geq 1000)$**Entonces**
  $r \leftarrow n/1000$
  Para$(i = 1, r, 1)$**Haga**
    Escriba "M"
  Fin_Para
  $numero \leftarrow numero$ **mod** 1000
F.Si
Si$(numero \geq 900)$**Entonces**
  Escriba "CM"
  $numero \leftarrow numero - 900$
F.Si
Si$(numero \geq 500)$**Entonces**
  Escriba "D"
  $numero \leftarrow numero - 500$
F.Si
Si$(numero \geq 400)$**Entonces**
  Escriba "CD"
  $numero \leftarrow numero - 400$
F.Si
Si$(numero \geq 100)$**Entonces**
  $r \leftarrow n/100$
  Para$(i = 1, r, 1)$**Haga**
    Escriba "C"
  Fin_Para
  $numero \leftarrow numero$ **mod** 100
F.Si
Si$(numero \geq 90)$**Entonces**
  Escriba "XC"
  $numero \leftarrow numero - 90$
F.Si
Si$(numero \geq 50)$**Entonces**
  Escriba "L"
  $numero \leftarrow numero - 50$
F.Si
Si$(numero \geq 40)$**Entonces**
  Escriba "XL"
  $numero \leftarrow numero - 40$
F.Si
1

```
1
| Si(numero ≥ 10)Entonces
| | r ← n/10
| | Para(i = 1, r, 1)Haga
| | | Escriba "X"
| | Fin_Para
| | numero ← numero mod 10
| F.Si
| Si(numero ≥ 9)Entonces
| | Escriba "IX"
| | numero ← numero − 9
| F.Si
| Si(numero ≥ 5)Entonces
| | Escriba "V"
| | numero ← numero − 9
| F.Si
| Si(numero = 4)Entonces
| | Escriba "IV"
| | numero ← numero − 4
| F.Si
| Si(numero ≥ 1)Entonces
| | r ← n
| | Para(i = 1, r, 1)Haga
| | | Escriba "I"
| | Fin_Para
| | numero ← numero − r
| F.Si
Fin_proc
```

Ejemplo 3.117 Secuencia de caracteres

Diseñe un algoritmo que produzca la siguiente salida:

```
ZYXWVTSRQPONMLKJIHGFEDCBA
YXWVTSRQPONMLKJIHGFEDCBA
XWVTSRQPONMLKJIHGFEDCBA
WVTSRQPONMLKJIHGFEDCBA
VTSRQPONMLKJIHGFEDCBA
TSRQPONMLKJIHGFEDCBA
⋮
CBA
BA
A
```

Recuerde que el valor ASCII del carácter 'Z' es 90.

Análisis: Una variable de tipo carácter se puede declarar de la forma *caracter* ← *numero*, siendo *numero* el número que corresponde al valor ASCII del carácter. Usando esto, se usarán dos ciclos Para: uno externo que controlara cada línea y uno interno que controlará la impresión de cada carácter. El ciclo Para interno inicia en $90 - i + 1$, pues para cuando $i = 1$ (la primera línea) se quiere que inicie en 90, ya que corresponde con el valor ASCII de la letra Z. Para la segunda línea, el ciclo interno empezará en 89, es decir, Y, y así sucesivamente. Hay 26 letras en el abecedario, entonces el carácter ASCII que corresponde a la letra A sería 65. Por consiguiente, el Para interno cuando $i = 1$ debe ir desde $j = 90$ hasta $j = 65$, que es lo que se desea; luego, para cuando $i = 2$, el Para interno debe ir desde $j = 89$ hasta $j = 65$, de tal manera que se va "eliminando" la primera letra de la linea anterior. El ciclo interior siempre va hasta 65. Finalmente se produce la salida deseada.

```
Proc
    Entero: i, j
    Caracter: a
    // Se realiza la declaración de la variable a y se hace
    // enfásis en que se trata de un carácter
    Para(i = 1, 26, 1)Haga
        Para(j = 90 − i + 1, 65, −1)Haga
            a ← j
            Escriba a
        Fin_Para
        Escriba "Salto de línea"
    Fin_Para
Fin_proc
```

Ejemplo 3.118 Valor máximo de un conjunto de datos

Realice un algoritmo que lea un entero positivo n y a continuación lea n números y despliegue tanto el valor máximo como la posición en el conjunto de números introducido donde ocurre el máximo. A su vez, también realice lo mismo pero para el mínimo valor.

Análisis: Se debe leer el primer número, y este se asignará como el máximo (max) y el mínimo (min). A su vez, $posmin$ y $posmax$ se asignan como 1. Luego se deben leer $n - 1$ números. Cada vez que se lea un número se debe verificar si es mayor que el mayor número encontrado hasta el momento. Si se da lo anterior, se actualiza el valor de max y el valor de $posmax$. De forma similar se hace para el mínimo.

Proc: Valor máximo de un conjunto de datos
| **Entero:** n, i
| **Real:** num, max, $posmax$, min, $posmin$
| **Escriba** "Ingrese el número de números"
| **Lea** n
| **Escriba** "Ingrese el número 1: "
| **Lea** num
| $max \leftarrow num$
| $posmax \leftarrow 1$
| $min \leftarrow num$
| $posmin \leftarrow 1$
| **Para**$(i = 1, num - 1, 1)$**Haga**
| | **Escriba** "Ingrese el numero", $i + 1$, ":"
| | **Lea** num
| | **Si**$(num > max)$**Entonces**
| | | $max \leftarrow num$
| | | $posmax \leftarrow i$
| | **F.Si**
| | **Si**$(num < min)$**Entonces**
| | | $min \leftarrow num$
| | | $posmin \leftarrow i$
| | **F.Si**
| **Fin_Para**
| **Escriba** "Máximo: ", max
| **Escriba** "Mínimo: ", min
| **Escriba** "Posición Máximo: ", $posmax$
| **Escriba** "Posición Mínimo: ", $posmin$
Fin_proc

Ejemplo 3.119 Regalos

Los padres de Yanina le prometieron darle 10 pesos cuando cumpliera 12 años de edad y duplicar esta cantidad en cada cumpleaños subsiguiente hasta que excediera los mil pesos. Escriba un algoritmo para determinar qué edad tendrá la niña cuando se le dé la última cantidad y cuánto fue la cantidad total recibida.

Análisis: Sea a la cantidad inicial. Se sabe que $a = 10$. Para cada año se sabe que la cantidad se duplica. Si n es la edad de Yanina, entonces la cantidad $2^{n-12}a$ determinará la cantidad que se le da en el cumpleaños número n. Por ejemplo, si $n = 12$, a Yanina le dan $2^{12-12}a = 2^0a = a$ pesos en su cumpleaños número 12. Si $n = 13$, se le da $2^{13-12}a = 2^1a = 2a$, el doble del cumpleaños anterior. El problema pide resolver el menor valor de n para el cual

$$a + 2a + 2^2a + \cdots + 2^{n-12}a > 1000.$$

Además, se tiene que determinar el valor del lado izquierdo de esta desigualdad.

Se proponen dos soluciones para la resolución de este problema. En la primera se hará uso de un ciclo Mientras que; este "simulará" el paso de los años. En su interior, la cantidad regalada a Yanina se irá duplicando. La condición que controla el Mientras que permite que no entre cuando el regalo exceda los 1000 pesos. Para duplicar la cantidad de regalo se duplica la variable *regalo*, a su vez se incrementa la edad de Yanina (*edad*) en 1 y se va sumando la cantidad recibida en cada regalo en un acumulador llamado *suma*. La variable *edad* funciona como un contador. Cuando el algoritmo culmine el ciclo Mientras que, el valor de *edad* correspondería al valor de n buscado.

La solución 2 muestra un ciclo Para más general, que va desde $regalo = 10$ hasta $regalo \leq 1000$. Cuando $regalo > 1000$ no se cumplirá esta condición, por tanto no entrará dentro del ciclo; el incremento en este caso es ir duplicando la variable, es decir, $regalo = 2 * regalo$. Dentro del Para se incrementa la edad en 1, y se va acumulando la cantidad total recibida en *suma*, de forma similar a la solución 1.

Solución 1:

```
Proc
| Entero: suma, regalo, edad
| suma ← 10
| regalo ← 10
| edad ← 12
| Mq(regalo ≤ 1000)Haga
| | edad ← edad + 1
| | regalo ← 2 * regalo
| | suma ← suma + regalo
| Fin_Mq
| Escriba "Edad ultimo regalo: ", edad
| Escriba "Cantidad ultimo regalo: ", regalo
| Escriba "Cantidad total regalada: ", suma
Fin_proc
```

Solución 2:

```
Proc
| Entero: e, r, s
| // e = edad, r = regalo, s = suma
| e ← 12
| Para(r = 10, r ≤ 1000, r = r * 2)Haga
| | e ← e + 1
| | s ← s + r
| Fin_Para
1
```

```
1
| Escriba "Edad ultimo regalo: ", e
| Escriba "Cantidad ultimo regalo: ", r
| Escriba "Cantidad total regalada: ", s
Fin_proc
```

Ejemplo 3.120 Patrón Rombo

Escriba un algoritmo que visualice el siguiente dibujo:

```
        *
      * * *
    * * * * *
  * * * * * * *
* * * * * * * * *
  * * * * * * *
    * * * * *
      * * *
        *
```

Análisis: En primera medida, resulta más fácil trabajar con $(n+1)/2$ una vez ingresado el valor de n impar. Por lo tanto, se debe cambiar n por $(n+1)/2$. Por otro lado, observe que primero se deben escribir unos espacios antes de poner un "*", luego esos espacios se van reduciendo y vuelven aumentar a medida que escribimos más líneas. Observe que la función valor absoluto tiene estas características para valores negativos de x; a medida que se aumenta x se disminuye su valor absoluto. Sin embargo, cuando $x > 0$, a medida que se aumenta x se aumenta su valor absoluto.

Antes de continuar se ilustrará la técnica que se va a usar. Por ejemplo, suponga inicialmente que el dibujo tiene en total 5 lineas, entonces, primero se deben escribir 2 espacios y 1 asterisco; luego, 1 espacio y 3 asteriscos; luego, 0 espacios y 5 asteriscos; luego, 1 espacio y 3 asteriscos; finalmente, 2 espacios y 1 asterisco. De lo anterior, un ciclo Para externo debe controlar el salto de linea. Además, debe haber un primer ciclo Para interno, para este ejemplo, que haga primero 2 iteraciones para la primera iteración del ciclo Para externo, 1 iteración para la segunda iteración del ciclo externo, 0 iteraciones para la tercera, 1 para la cuarta y, finalmente, 2 para la quinta. Dado que $n = 5$, entonces $n = (n + 1)/2 = 3$. Entonces, si para el ciclo externo $i = 1$, entonces el contador final del primer ciclo Para debe ser 2, para $i = 2$ el contador debe ser 1, para $i = 3$ debe ser 0, para $i = 4$ debe ser 1 y para $i = 5$ debe ser 2. Pero si a i se le resta n, es decir, 3, queda -2, -1, 0, 1, 2 para cada i que va desde 1 hasta 5, pero se quieren los valores 2, 1, 0, 1, 2, entonces se aplica valor absoluto, y resulta que el número de iteraciones del primer ciclo Para interno es $|i - n|$ o $|n - i|$.

En el caso de los asteriscos, se quiere que se siga la secuencia 1, 3, 5, 3, 1,

para $i = 1, 2, 3, 4, 5$. Sin embargo, la secuencia 1, 3, 5, 3, 1, es análoga a 1, 2, 3, 2, 1, pues basta con multiplicar estos últimos por 2 y restarle 1. Ahora véase que se tiene 2, 1, 0, 1, 2. Si a 3 se le resta cada uno de estos números, se tiene $3-2, 3-1, 3-0, 3-1, 3-2$, lo cual resulta en la secuencia que se desea: 1, 2, 3, 2, 1. Finalmente, en el segundo Para se obtiene que el número de ciclos es $2 * (n - |i - n|) - 1$. Recuérdese que el valor de n inicialmente introducido se cambia por $(n + 1)/2$.

Proc: Patron Rombo
| **Entero:** n, i, j
| **Escriba** "Ingrese numero de filas (cantidad impar): "
| **Lea** n
| $n \leftarrow (n + 1)/2$
| **Para**$(i = 1, 2 * n - 1, 1)$**Haga**
| | **Para**$(j = 1, |i - n|, 1)$**Haga**
| | | **Escriba** " "
| | **Fin_Para**
| | **Para**$(j = 1, 2 * (n - |i - n|) - 1, 1)$**Haga**
| | | **Escriba** "*"
| | **Fin_Para**
| **Fin_Para**
Fin_proc

Ejemplo 3.121 Suma de números enteros

Escriba un programa que calcule la suma de los números enteros comprendidos entre 1 y 50.

Análisis: Se debe usar un ciclo Para con contador i y un acumulador *suma* declarado inicialmente en 0, el cual acumulará la suma de los primeros 50 enteros positivos.

Proc: Suma de los 50 primeros enteros positivos
| **Entero:** *suma*, i
| *suma* $\leftarrow 0$
| **Para**$(i = 1, 50, 1)$**Haga**
| | *suma* $\leftarrow$ *suma* $+ i$
| **Fin_Para**
| **Escriba** "Suma: ", *suma*
Fin_proc

Escriba un algoritmo que visualice las primeras 20 potencias de 2. Suponga que no se dispone de la función potencia, es decir, no se pueden realizar operaciones de la forma a^b, con a, b números reales.

Análisis: Se desea escribir las primeras 20 potencias de 2, por lo tanto se usa un ciclo Para controlado por un contador i, que va desde 1 hasta

20. Internamente se debe tener una variable que se vaya duplicando en cada iteración. De esta forma se crea una nueva potencia de 2 usando que $2^n = 2 \cdot 2^{n-1}$. Dado que $2^0 = 1$, se declarará una variable *pot* (que será la potencia siguiente a mostrar) en 1, y dentro del ciclo Para se debe hacer $pot = 2 * pot$. Observe que primero se debe escribir y luego duplicar, debido a que la primera potencia de 2 es 1 (2^0). Por último, como el ciclo hará 20 iteraciones, por tanto escribirá las primeras 20 potencias de 2.

```
Proc: 20 primeras potencias de 2
| Entero: pot, i
| pot ← 1
| Para(i = 1, 20, 1)Haga
| | Escriba pot
| | pot ← pot * 2
| Fin_Para
Fin_proc
```

Ejemplo 3.122 Incremento en el salario

En una empresa de computadoras, los salarios de los empleados se van a aumentar según su contrato actual:

Contrato	Aumento %
0 a 9 000 pesos	20
9 0001 a 15 000 pesos	10
15 001 a 20 000 pesos	5
Más de 20 000 pesos	0

Escriba un algoritmo que solicite el salario actual del empleado y calcule y visualice el nuevo salario. Se introducen N empleados.

Análisis: Dado que no introduce un número N de empleados, entonces se hará uso de un ciclo Para que va desde 1 hasta N. En su interior se debe preguntar el contrato del empleado. Igual a como se ha hecho anteriormente, se pregunta en cierto orden. Ese orden puede ser de menor a mayor, es decir, si el contrato es menor a 9000, si no, si es menor a 15 000, y así sucesivamente. Otra forma es de mayor a menor, es decir, si el contrato es mayor a 20 000, si no, si es mayor a 15 000, y así sucesivamente. Ahora bien, dado que para mayor de 20 000 no hay aumento, entonces se puede usar una condición compuesta para preguntar si el contrato está entre 15 000 y 20 000 pesos. Por último, una vez se logra entrar a un Si, se asigna el nuevo valor de contrato de acuerdo con la tabla y se muestra dicho valor. Observe que el comando Escriba no es necesario colocarlo dentro de cada anidamiento.

```
Proc: Cálculo de nuevos salarios
|
1
```

```
1
| Entero: N, i, contrato
| Escriba "Ingrese numero de empleados: "
| Lea N
| Para(i = 1, N, 1)Haga
| | Escriba "Ingrese contrato: "
| | Lea contrato
| | Si(contrato > 15000 y contrato < 20000)Entonces
| | | contrato ← contrato * 1.05
| | Sino
| | | Si(contrato > 9000)Entonces
| | | | contrato ← contrato * 1.1
| | | Sino
| | | | contrato ← contrato * 1.2
| | | F.Si
| | F.Si
| | Escriba "Nuevo salario: ", contrato
| Fin_Para
Fin_proc
```

Ejemplo 3.123 Estadística de notas

Haga un algoritmo que reciba las notas de n estudiantes y su salida sea la tabla de distribución estadística para las notas 1, 2, 3, 4 y 5:

$$\begin{matrix}1,2,4,3,5,3\\1,2,3,3,4,5\end{matrix} \Longrightarrow \boxed{\textbf{A}} \Longrightarrow \begin{matrix}2 \text{ Estudiantes sacaron } 1\\2 \text{ Estudiantes sacaron } 2\\4 \text{ Estudiantes sacaron } 3\\2 \text{ Estudiantes sacaron } 4\\2 \text{ Estudiantes sacaron } 5\end{matrix}$$

Análisis: La variable n determinará el número total de estudiantes al que se les va a leer la nota; las variables $c1$, $c2$, $c3$, $c4$, $c5$ determinarán las calificaciones; *nota* guardará temporalmente la nota ingresada por el usuario, y por último se utilizará el apuntador j para el ciclo. Una vez se lee la cantidad de estudiantes se tiene el valor de n. Se usa un ciclo Para, en el que internamente se lee la nota de cada estudiante y mediante un Dependiendo De de acuerdo con la nota, se incrementa alguno de los contadores, $c1$, $c2$, $c3$, $c4$ o $c5$. En caso de ser una nota incorrecta, se muestra un mensaje y se retrocede en uno el valor de j; de esta forma, el ciclo Para terminará solamente si se digitaron n notas correctas. Por último, se deben mostrar las estadísticas.

```
Proc: Distribución estadística de las notas
| Entero: c1, c2, c3, c4, c5, n, j, nota
| c1 ← 0, c2 ← 0, c3 ← 0, c4 ← 0, c5 ← 0
| Escriba "Por favor, digite el número total de estudiantes"
| Lea n
| Para(j = 1, n, 1)Haga
| | Escriba "Digite la nota número", j
| | Lea nota
| | DD(nota)Haga
| | | Opción 1 :
| | | | c1 ← c1 + 1
| | | Fin
| | | Opción 2 :
| | | | c2 ← c2 + 1
| | | Fin
| | | Opción 3 :
| | | | c3 ← c3 + 1
| | | Fin
| | | Opción 4 :
| | | | c4 ← c4 + 1
| | | Fin
| | | Opción 5 :
| | | | c5 ← c5 + 1
| | | Fin
| | | Sino :
| | | | Escriba "Nota incorrecta, reescríbala "
| | | | j ← j − 1
| | | Fin
| | Fin_DD
| Fin_Para
| Escriba c1, "Estudiantes sacaron 1"
| Escriba c2, "Estudiantes sacaron 2"
| Escriba c3, "Estudiantes sacaron 3"
| Escriba c4, "Estudiantes sacaron 4"
| Escriba c5, "Estudiantes sacaron 5"
Fin_proc
```

Ejemplo 3.124 Menor número divisible por todos los números del 1 al 20

2520 es el menor número que es divisible por cada uno de los números del 1 al 10 sin ningún resto. ¿Cuál es el menor número que es divisible por todos los números del 1 al 20?

Análisis: Eventualmente, el menor número corresponderá al mínimo común múltiplo de los primeros 20 números naturales. A continuación se muestran

tres soluciones. La primera presenta el simple hecho de ir recorriendo todos los números naturales, uno por uno, hasta encontrar el primero que sea divisible por todos los números del 1 al 20. La segunda solución calcula el mínimo común múltiplo de los primeros 20 enteros positivos, la cual se extiende a calcular el mínimo común múltiplo de los primeros n enteros positivos.

En el caso de la primera solución no hace falta preguntar si el número es divisible por 1, 2, 3,4, 5, 6, 7, 8, 9, 10, pues lo será si es divisible por 11, 12, 13, 14, 15, 16, 17, 18, 19, 20; por ejemplo, si es divisible por 9, debe serlo por 18, entonces mejor se pregunta por el que excluye al otro, para no preguntar doble, es decir, por 18.

Proc: Solución 1
| **Booleano:** sw
| **Entero:** i
| $sw \leftarrow falso$
| $i \leftarrow 0$
| **Mq**($sw = falso$)**Haga**
| | $i \leftarrow i + 1$
| | // Los puntos suspensivos significa que se pregunta si
| | // el número es divisible por 20, 19, 18, 17, ..., 10 y 11
| | **Si**(i **mod** $20 = 0$ y $\cdots$ y i **mod** $11 = 0$)**Entonces**
| | | $sw \leftarrow verdadero$
| | **F.Si**
| **Fin_Mq**
| **Escriba** "Respuesta: ", i
Fin_proc

Sin embargo, si se desea preguntar si el número es divisible por los primeros n enteros positivos, no se puede usar esta solución; por lo tanto, se usará un ciclo Mientras que para realizar todas estas preguntas. Si alguna resulta ser falsa, es decir, si el número resulta no ser divisible por alguno de los primeros n enteros positivos, entonces el ciclo termina. Si el ciclo termina, el ciclo externo avanza al siguiente número.

Proc
| **Entero:** n, $sw1$, i, $sw2$, j
| **Escriba** "Ingrese el valor de n: "
| **Lea** n
| $sw1 \leftarrow 0, i \leftarrow 0$
| // Este ciclo recorrerá todos los enteros positivos
| // en búsqueda del primer entero positivo que sea divisible
| // por los primeros n enteros positivos
| **Mq**($sw1 = 0$)**Haga**
| | $i \leftarrow i + 1$
1 2

```
1 2
|  |  sw2 ← 1
|  |  // Se inicia en 2, puesto que todos los números son divisibles por 1
|  |  j ← 2
|  |  Mq(sw2 = 1 y j ≤ n)Haga
|  |  |  Si(i mod j ≠ 0)Entonces
|  |  |  |  // Si no es divisible, entonces obliga a terminar
|  |  |  |  // el ciclo interior y dar paso al ciclo externo en búsqueda
|  |  |  |  // de otro número
|  |  |  |  sw2 ← 0
|  |  |  F.Si
|  |  |  j ← j + 1
|  |  Fin_Mq
|  |  Si(sw2 = 1)Entonces
|  |  |  sw1 ← 1
|  |  F.Si
|  Fin_Mq
|  Escriba "Respuesta: ", i
Fin_proc
```

La tercera solución no requiere crear un algoritmo para resolver el problema inicial (los primeros 20 enteros positivos). Esta solución trata de cómo hallar el máximo y mínimo común múltiplo de 20 números, aunque, por lo visto anteriormente, no hace falta, sino hallar el mínimo común múltiplo de los 10 más grandes. Primero se deben descomponer estos números en sus factores primos

$$\begin{aligned}
11 &= 11^1 \\
12 &= 2^2 \times 3^1 \\
13 &= 13^1 \\
14 &= 2^1 \times 7^1 \\
15 &= 3^1 \times 5^1 \\
16 &= 2^4 \\
17 &= 17^1 \\
18 &= 2^1 \times 3^2 \\
19 &= 19^1 \\
20 &= 2^2 \times 5
\end{aligned}$$

El mínimo común múltiplo debe ser múltiplo de 2; en particular debe ser múltiplo de la potencia de 2 más grande encontrada, es decir, debe ser múltiplo de $2^{\max(2,1,4,1,2)} = 2^4$. De la misma forma, debe ser múltiplo de la potencia más grande de 3, es decir, de 3^2. Siguiendo este procedimiento, el mínimo común múltiplo resulta ser

$$2^4 \cdot 3^2 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19 = 232792560.$$

Para extender esta solución, observe que lo que se tiene que hacer es tomar

todos los primos menores que n y encontrar la potencia más alta menor que n e ir acumulando este producto. Para ello se necesitará verificar si un número es primo o no. La variable mcm contendrá el mínimo común múltiplo y será inicializada en 1.

```
Proc
  Entero: n, mcm, i, j, sw, prod
  Escriba "Ingrese el valor de n: "
  Lea n
  mcm ← 1
  Para(i = 2, n, 1)Haga
    // Se revisará si i es primo
    j ← 2, sw ← 0
    Mq(j ≤ √i y sw = 0)Haga
      Si(i mod j = 0)Entonces
        sw ← 1
      F.Si
      j ← j + 1
    Fin_Mq

    // Si sw = 0, entonces i es primo
    Si(sw = 0)Entonces
      prod ← 1
      Mq(prod < n)Haga
        prod ← prod * i
      Fin_Mq
      // Se asigna prod/i porque cuando el ciclo
      // Mientras que termina es porque prod es mayor que n,
      // es decir, prod está en la menor potencia mayor que n
      // y la que se necesita en realidad es la anterior
      // Por ejemplo, si i = 3 y n = 10, entonces cuando prod
      // sale del ciclo Mientras que, prod = 27, pero en realidad
      // se desea prod/i = 9
      mcm ← mcm * prod/i
    F.Si
  Fin_Para
  Escriba "El menor número que es divisible por"
  Escriba "los primeros n enteros positivos es: ", mcm
Fin_proc
```

Por último, existe una forma más eficiente para la solución de este algoritmo: la Criba de Erastótenes, sin embargo, requiere el uso de estructuras más complejas. Dicha solución se mostrará después.

Ejemplo 3.125 Suma de los múltiplos de 3 o 5

Si se listan todos los números naturales menores que 10 que son múltiplos de 3 o 5, se obtiene 3, 5, 6 y 9. La suma de estos múltiplos es 23. Encuentre la suma de todos los múltiplos de 3 o 5 menores que 1000.

Análisis: Se debe usar un acumulador *suma* para ir sumando todos los múltiplos de 3 o 5. Se debe usar un ciclo Para que revisará todos los números menores que 1000, por lo tanto, irá desde 1 hasta 999 con incremento de 1. Por último, se debe agregar una condición para revisar si el número es divisible por 3 o 5.

```
Proc
| Entero: suma, i
| suma ← 0
| Para(i = 1, 999, 1)Haga
| | Si(i mod 3 = 0 o i mod 5 = 0)Entonces
| | | suma ← suma + i
| | F.Si
| Fin_Para
| Escriba "Respuesta: ", suma
Fin_proc
```

Solución: 233168.

Ejemplo 3.126 Número de cifras y la suma de las mismas

Realice un algoritmo que lea por pantalla un número entero y diga el número de dígitos que tiene y la suma de estos.

$$3621 \implies \boxed{\mathbf{A}} \implies \text{Tiene 4 dígitos y la suma de estos es 12}$$

Análisis: La variable n almacenará el número que se va a descomponer. Se hace una copia de n en cn. Si el número es negativo, entonces se lo multiplica por -1 para trabajar con su valor absoluto. La variable sdn contiene la suma de los dígitos de n; nd contiene el número de dígitos de n. Para extraer las cifras del número se hace uso de las funciones **mod** y /. / realiza una división entera, por lo tanto, si se hace /10, se pueden descartar tantas cifras como se desee. Por ejemplo, $12342/10 = 1234$, $1234/10 = 123$, y así sucesivamente. Además, **mod** 10 es el resto que deja un número al ser dividido por 10; esto es equivalente a extraer su última cifra (¿por qué?). Por ejemplo, $12342 \,\mathbf{mod}\, 10 = 2$, $1234 \,\mathbf{mod}\, 10 = 4$, y así sucesivamente. El uso combinado de estas dos funciones extrae cada cifra.

```
Proc: Suma de las cifras de un número
| Entero: sdn, nd, n, cn
| sdn ← 0, nd ← 0
| Escriba "Por favor, digite un número entero"
| Lea n
| cn ← n
| Si(cn < 0)Entonces
| | cn ← −cn
| F.Si
| Si(cn = 0)Entonces
| | // Si el número es 0, entonces n tiene 1 dígito
| | // y la suma de sus cifras es 0
| | sdn ← 0
| | nd ← 1
| Sino
| | Mq(cn > 0)Haga
| | | // Se acumula la suma de las cifras
| | | sdn ← sdn + (cn mod 10)
| | | nd ← nd + 1
| | | // Se descarta la cifra sumada
| | | cn ← cn/10
| | Fin_Mq
| F.Si
| Escriba n, "tiene", nd, "dígitos y la suma de los dígitos de", n, "es:", sdn
Fin_proc
```

Ejemplo 3.127 Figura especial

Visualice en pantalla una figura similar a la siguiente; el usuario debe ingresar el número de líneas que se va a mostrar.

```
*
* *
* * *
* * * *
* * * * *
* * * * * *
* * * * * * *
```

Análisis: La variable n contiene el número de líneas que se debe mostrar. Se requiere de dos ciclos, uno que controle el salto de línea y otro que controle el número de asteriscos que se va a mostrar. Observe que si se está en la segunda línea, entonces el ciclo interno debe mostrar 2 asteriscos. Por lo tanto, si se está en la i-ésima línea, el ciclo interno debe mostrar i asteriscos. Lo anterior quiere decir que si el ciclo externo va con i desde 1 hasta n, entonces el ciclo interior va con j desde 1 hasta i. Por último, en

el interior de cada ciclo Para se usa la instrucción Escriba para mostrar el asterisco.

```
Proc
| Entero: n, i, j
| Escriba "Ingrese el número n: "
| Lea n
| Para(i = 1, n, 1)Haga
| | Para(j = 1, i, 1)Haga
| | | Escriba "*"
| | Fin_Para
| Fin_Para
Fin_proc
```

3.10 Ejercicios propuestos

3.10.1 Condicionales

1. Escriba instrucciones **Si-Sino** apropiadas para cada una de las siguientes condiciones:

   a) Si un ángulo es igual a 90 grados, imprima el mensaje "El ángulo es un ángulo recto"; de lo contrario imprima el mensaje "El ángulo no es un ángulo recto".

   b) Si la temperatura está por encima de 100 grados, despliegue el mensaje "arriba del punto de ebullición del agua"; de lo contrario despliegue el mensaje "abajo del punto de ebullición del agua".

   c) Si el número es positivo, sume al número *sumpos*, sino sume el número a *sumneg*.

   d) Si la pendiente es menor que 0.5, fije la variable $flag$ en 0; de lo contrario fijar $flag$ en 1.

   e) Si la edad es mayor que 18, muestre el mensaje "La persona es mayor de edad"; en caso contrario, muestre "La persona es menor de edad".

   f) Si el ingreso es 'F', muestre el mensaje "Es una mujer"; en caso contrario, "Es un hombre".

   g) Si la diferencia entre *voltios*1 y *voltios*2 es menor que 0.001, fije la variable *aprox* en 0, de lo contrario calcule *aprox* como la cantidad $(voltios1 - voltios2)/2.0$.

   h) Si la frecuencia es superior a 60, despliegue el mensaje "La frecuencia es demasiado alta".

i) Si la diferencia entre $temp1$ y $temp2$ excede 2.4, calcule $error$ como $(temp1 - temp2) * 0.01$.

j) Si la diferencia entre $teorico$ y $experimental$ excede en 5, calcule $error$ como $(teorico - experimental) * 100/teorico$.

k) Si x es mayor que y y z es menor que 20, lea un valor para p.

l) Si la distancia es mayor que 20 y es menor que 35, lea un valor para tiempo.

2. Haga un algoritmo que dada una fecha entre 01/01/1990 al 31/12/2010, determine si es correcta o incorrecta. Tenga en cuenta los años bisiestos.
Entrada: El día, mes y año de una fecha.
Salida: Deberá decir si la fecha ingresada es válida o no.

3. Haga un algoritmo que dado dos números determine si uno es divisor del otro.
Entrada: Dos números enteros.
Salida: Si el número menor es o no divisor del otro.

4. Escriba un algoritmo para calcular el valor de la presión en libras por pulgada cuadrada (psi) de una onda descrita como sigue:

   - Para tiempo, t, igual a o menor que 35 segundos, la presión es $0.46t$ psi, y para tiempo mayor que 35 segundos la presión es $0.19t + 9.45$ psi.

   - El programa deberá pedir el tiempo como entrada y deberá desplegar la presión como salida.

   Entrada: El tiempo
   Salida: Presión en psi.

5. Construya un algoritmo que reciba como datos de entrada tres números enteros, y regrese como dato de salida un mensaje que diga si esos tres números enteros pueden ser las medidas de los lados de un triángulo rectángulo.
Entrada: Tres números enteros.
Ej: 4, 4, 4.
Salida: No pueden ser los lados de un triángulos rectángulo.

6. Diseñe un algoritmo en el que dado un tiempo en minutos, calcule los días, horas y minutos que le corresponden.
Entrada: Tiempo en minutos (número entero).
Salida: Días, horas y minutos a los que equivale.

7. Escriba un algoritmo que le pida al usuario que introduzca dos números. Si el primer número introducido es mayor que el segundo, el programa deberá imprimir el mensaje "El primer número es mayor"; de lo contrario deberá imprimir el mensaje "El primer número es menor". Pruebe su algoritmo introduciendo los números 5 y 8 y luego usando los números 11 y 2. ¿Qué piensa que desplegará su algoritmo si los dos números introducidos son iguales? Pruebe este caso.

8. La tolerancia de componentes críticos en un sistema se determina por la aplicación de acuerdo con la siguiente tabla:

Estado de la especificación	**Tolerancia**
Exploración espacial	Menor que 0.1 %
Grado militar	Mayor que o igual a 0.1 % y menor que 1 %
Grado comercial	Mayor que o igual a 1 % y menor que 10 %
Grado de juguete	Mayor que o igual a 10 %

Usando esta información escriba un algoritmo que acepte la lectura de tolerancia de un componente y determine la especificación que debería asignarse al componente.
Entrada: 0.02
Salida: Exploración espacial

9. Escriba un algoritmo que acepte un número, luego una letra. Si la letra que sigue al número es F, el programa tratará al número introducido como una temperatura en grados Fahrenheit, convertirá el número a los grados Celsius equivalentes y desplegará un mensaje adecuado. Si la letra que sigue al número es C, el programa tratará al número introducido como una temperatura en Celsius, convertirá el número a los grados Fahrenheit equivalentes y desplegará un mensaje adecuado. Si la letra no es f ni c, el algoritmo imprimirá el mensaje que los datos introducidos son incorrectos y terminará. Use una cadena Si-Sino en su algoritmo y use las fórmulas de conversión:

$$Celsius = \frac{5}{9} \times (Fahrenheit - 32.0)$$
$$Fahrenheit = \frac{9}{5} \times (Celsius - 32.0)$$

Entrada: 32.0
Entrada: F
Salida: La temperatura en grados Celsius es 0.

10. Escriba un programa que lea tres enteros y emita un mensaje que indique si están o no en orden numérico.

Entrada: 3
Entrada: 2
Entrada: 5
Salida: Los números no están orden.

Entrada: 3
Entrada: 4
Entrada: 5
Salida: Los números si están orden.

11. Escriba una sentencia Si-Sino que clasifique un entero en una de las siguientes categorías y escriba un mensaje adecuado:

$$x < 0 \qquad 0 \leq x \leq 100 \qquad x > 100$$

Entrada: 5
Salida: El número es menor o igual que 100 y mayor o igual que 0.

12. Escriba un programa que introduzca el número de un mes (1 a 12) y visualice el número de días de ese mes.

    Entrada: 12
    Salida: El mes 12 tiene 31 días

13. La fuerza de atracción entre dos masas, m_1 y m_2, separadas por una distancia d está dada por la fórmula

$$F = \frac{Gm_1m_2}{d^2}$$

    Donde G es la constante de gravitación universal

$$G = 6.673 \times 10^{-8}\,\text{cm}^3/\text{g-seg}^2$$

    Escriba un algoritmo que lea la masa de dos cuerpos y la distancia entre ellos y a continuación obtenga la fuerza gravitacional entre ellos. La salida debe ser en dinas; una dina es igual a 1 g-cm/s^2.

    Entrada: Dos masas en gramos, distancia en centímetros
    Salida: Fuerza de atracción en dinas

    Entrada Ejemplo: 2400, 12000, 0.1
    Salida Ejemplo: 192.18 g-cm/s^2.

14. Escriba un algoritmo que clasifique un entero leído del teclado de acuerdo a los siguientes puntos:

    a) Si es 30 o mayor, o negativo, visualice un mensaje en ese sentido;

    b) si es un numero primo, potencia de 2 o un número compuesto, visualice el mensaje correspondiente;

    c) si es 0 o 1, muestre "cero" o "unidad", según corresponda.

    Entrada: 12
    Salida: El número corresponde a la categoría B.

15. Escriba un algoritmo que reciba como entrada las coordenadas de dos vértices opuestos de un rectángulo, imprima las coordenadas de los otros dos vértices. El usuario ingresará las dos coordenadas (x_1, y_1) y (x_2, y_2) en el orden x_1,y_1,x_2,y_2.
    Entrada: $5, 2, 8, -3$
    Salida: $5, -3, 8, 2$

16. Se sabe que el primer día de un mes es domingo. Haga un algoritmo que reciba como entrada un número entero n ($1 \leq n \leq 31$) correspondiente a un día de ese mes e imprima en qué día de la semana cae ese día n.
Entrada: 1
Salida: domingo
Entrada: 11
Salida: miércoles

17. El costo de inscripción en un concurso de Algoritmia es de 100 000 pesos por inscripción de la universidad más 5 000 pesos por cada alumno participante. Si la universidad se inscribe con al menos 250 participantes, no se cobran los 100 000 pesos de inscripción del colegio. Haga un algoritmo que reciba como entrada el número de estudiantes que un colegio va a inscribir en un evento de olimpiadas y que calcule el monto del pago por la inscripción.
Entrada: 50
Salida: 350000
Entrada: 252
Salida: 1260000

18. Haga un programa que reciba como entrada tres números y diga si hay alguno que sea múltiplo de los otros dos; en caso afirmativo debe decir cuál es.
Entrada: 5, 3, 2
Salida: No
Entrada: 5, 20, 4
Salida: Si, 20

19. Haga un algoritmo que reciba como entrada las coordenadas de dos vértices opuestos de un rectángulo e imprima el área del rectángulo.
Entrada: 5, 2, 8, -3
Salida: 15

20. Dado un número entero n ($1 \leq n \leq 26$), muestre en pantalla un triángulo alfabético como el mostrado en el ejemplo. El valor ANSII que corresponde al carácter 'a' es 96.
Entrada: 3
Salida:
a
bb
ccc

21. Un archivo contiene dos fechas en el formato día (1 a 31), mes (1 a 12) y año (entero de cuatro dígitos), correspondientes a la fecha de nacimiento y la fecha actual, respectivamente. Escriba un programa que calcule y visualice la edad del individuo. Si es la fecha de un bebé (menos de un año de edad), la edad se debe dar en meses y días; en caso contrario, la edad se debe calcular en años.

3.10.2 Ciclos

22. Describa la salida de los siguientes ciclos:

    a)
    Proc
    | **Entero:** i, j
    | **Para**$(i = 1, 5, 1)$**Haga**
    | | **Escriba** i
    | | **Para**$(j = i, 1, -2)$**Haga**
    | | | **Escriba** j
    | | **Fin_Para**
    | **Fin_Para**
    Fin_proc

    b)
    Proc
    | **Entero:** i, j, k
    | **Para**$(i = 3, 1, -1)$**Haga**
    | | **Para**$(j = 1, i, 1)$**Haga**
    | | | **Para**$(k = i, j, -1)$**Haga**
    | | | | **Escriba** i,j,k
    | | | **Fin_Para**
    | | **Fin_Para**
    | **Fin_Para**
    Fin_proc

23. Escriba instrucciones **Para** individuales para los siguientes casos:

    a) Al usar un contador i que tiene valor inicial de 1, un valor final de 20 y un incremento de 1.

    b) Al usar un contador *icuenta* que tiene un valor inicial de 1, un valor final 20 y un incremento de 2.

    c) Al usar un contador j que tiene un valor inicial de 1, un valor final de 100 y un incremento de 5.

    d) Al usar un contador *icuenta* que tiene valor inicial de 20, un valor final de 1 y un incremento de -1.

    e) Al usar un contador *icuenta* que tiene valor inicial de 20, un valor final de 1 y un incremento de -2.

    f) Al usar un contador *cuenta* que tiene valor inicial de 1.0, un valor final de 16.2 y un incremento de 0.2.

    g) Al usar un contador *xcnt* que tiene valor inicial de 20.0, un valor final de 10.0 y un incremento de -

24. Determine el número de veces que se ejecuta cada ciclo en el ejercicio anterior.

25. Determine el valor de la variable *total* después que se ejecuta cada uno de los siguientes ciclos.

a) **Proc**
| **Entero:** $total$, i
| $total \leftarrow 0$
| **Para**$(i = 1, 10, 1)$**Haga**
| | $total \leftarrow total + 1$
| **Fin_Para**
Fin_proc

b) **Proc**
| **Entero:** $total$, $cuenta$
| $total \leftarrow 1$
| **Para**$(cuenta = 1, 10, 1)$**Haga**
| | $total \leftarrow total * 2$
| **Fin_Para**
Fin_proc

c) **Proc**
| **Entero:** $total$, i
| $total \leftarrow 0$
| **Para**$(i = 10, 15, 1)$**Haga**
| | $total \leftarrow total + i$
| **Fin_Para**
Fin_proc

d) **Proc**
| **Entero:** $total$, i
| $total \leftarrow 50$
| **Para**$(i = 1, 10, 1)$**Haga**
| | $total \leftarrow total - 1$
| **Fin_Para**
Fin_proc

e) **Proc**
| **Entero:** $total$, $icnt$
| $total \leftarrow 1$
| **Para**$(icnt = 1, 8, 1)$**Haga**
| | $total \leftarrow total * icnt$
| **Fin_Para**
Fin_proc

f) **Proc**
| **Entero:** $total$, j
| $total \leftarrow 1.0$
| **Para**$(j = 1, 5, 1)$**Haga**
| | $total \leftarrow total/2.0$
| **Fin_Para**
Fin_proc

26. Determine cuál es la salida de este ciclo:

```
Proc
| Entero: i, j
| i ← 0
| Mq(i * i < 10)Haga
| | j ← i
| | Mq(j * j < 100)Haga
| | | Escriba i + j
| | | j ← j * 2
| | Fin_Mq
| | i ← i + 1
| Fin_Mq
| Escriba "Fin"
Fin_proc
```

27. Escriba un algoritmo que calcule y visualice el más grande, el más pequeño y la media de N números. El valor de N se solicitará al principio del algoritmo y los números serán introducidos por el usuario.

    Entrada: 1 2 3 2 3 5
    Salida: Media: 2.66, Mayor: 5, Menor: 1

28. Modifique el algoritmo anterior, de tal forma que muestre solo si hay menor absoluto, el menor, y si hay mayor absoluto, el mayor. Si un número n es menor absoluto, entonces n no se repite.

    Entrada: 1 2 3 2 3 5 5
    Salida: Media: 2.66, Menor absoluto: 1, No hay mayor absoluto.

29. Calcule el factorial de un número leído utilizando las sentencias Mientras que y Para.

    Entrada: 5
    Salida: 120

30. Calcule la suma de una serie de números leídos del teclado. El algoritmo debe indicar qué número interrumpe la lectura.

    Entrada: 1 5 3 5 0
    Salida: 14

31. Cuente el número de enteros negativos introducidos por el usuario. Igual al inciso anterior, el algoritmo debe indicar qué número detiene la lectura.

    Entrada: -1 2 -3 8 9 9 -4 0
    Salida: 3

32. Calcule la suma de los términos de la serie

$$\sum_{i=1}^{n} \frac{1}{2^i}$$

    El usuario debe ingresar el valor de N; tenga en cuenta que entre más grande sea N, el resultado debe acercarse a 1.

33. Haga un algoritmo para hallar de un conjunto de N números qué porcentaje son cero, qué porcentaje son positivos y el porcentaje de números negativos.

    Entrada: -1 2 -3 8 9 9 -4 0 5 5 6 0
    Salida: Positivos: 58.3 %, Ceros: 16.6 %, Negativos: 25 %.

34. Haga un algoritmo para hallar cuántos números se debieron haber leído de un conjunto dado para que la suma de los negativos dé en valor absoluto mayor que 1200.

    Entrada: -16 2 -300 8 9 9 -4000
    Salida: 7

35. Diseñe un algoritmo que reciba como dato un número entero y a partir de este genere un número de un dígito (entre 0 y 9) sumando los dígitos tantas veces como sea necesario.

    Entrada: 3265
    Salida: 7

36. Construya un algoritmo que reciba como datos de entrada n números enteros y regrese como dato de salida el mayor de estos números.

    Entrada: 4 6 2 7 3
    Salida: El número mayor es: 7

37. Diseñe un algoritmo que reciba como dato de entrada un número entero positivo, n, y regrese como dato de salida el valor de la siguiente serie

$$\frac{\pi}{4} = \sum_{i=0}^{n} \frac{(-1)^i}{(2i+1)}$$

    Entrada: 5
    Salida: 0.7440115440

38. Construya un algoritmo que reciba como dato de entrada un número entero positivo, n, y regrese como dato de salida la representación de este número en binario.
    Entrada: 10
    Salida: El número representado en el sistema binario es 1010

39. Construya un algoritmo que reciba como dato de entrada un número entero positivo, n, y regrese como dato de salida la representación de este número en hexadecimal.

    Entrada: 11
    Salida: El número representado en el sistema hexadecimal es B

40. Realice un algoritmo en el que dado n números naturales, determine y escriba qué porcentaje son pares y que porcentaje son primos.

    Entrada: 4 5 6 2 1 -5 9 8
    Salida: Pares: 50 %, Primos: 25 %

41. Realice un algoritmo en el que dado un número, diga si es o no un número de Armstrong.
Un número de n dígitos es un número de Armstrong si la suma de las potencias n-ésimas de los dígitos que lo componen es igual al mismo número.
Entrada: 1634
Salida: Sí, el número es de Armstrong.

42. Para encontrar el máximo común divisor (mcd) de dos números se emplea el algoritmo de Euclides, que se puede describir así: dados los enteros, a y b $(a > b)$, se divide a por b, y se obtiene el cociente q_1 y el resto r_1. Si $r_1 \neq 0$, se divide b por r_1, y se obtiene el cociente q_2 y el resto r_2. Si $r_2 \neq 0$, se divide r_1 por r_2, y se obtiene cocientes y restos sucesivos. El proceso continúa hasta obtener un resto igual a 0. El resto anterior a este es el máximo común divisor de los números iniciales. Diseñar un algoritmo que calcule el máximo común divisor mediante el algoritmo de euclides.

43. Diseñe un algoritmo que encuentre el menor número primo ingresado por el usuario; tenga en cuenta que la lectura de números terminará con el número que el usuario ingresa. Se deben realizar verificaciones respectivas para conocer si el número es o no primo.

44. Escriba un algoritmo el cual permita calcular el coeficiente binomial con una función factorial.

$$\binom{m}{n} = \frac{m!}{n!(m-n)!} \quad m! = \begin{cases} 1 & \text{si } m = 0 \\ 1 \times 2 \times 3 \times \cdots \times m & \text{si } m > 0 \end{cases}$$

45. Escriba una función que permita calcular la serie

$$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$$

46. Realice un algoritmo que calcule la función que corresponda para un x ingresado por el usuario y el número de términos m de la serie que él desee evaluar. En el caso de las funciones trigonométricas, el x se encuentra en radianes.

    a) Función Exponencial

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} \quad \text{para todo } x$$

    b) Función Logaritmo natural

$$\ln(1+x) = \sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{n} x^n \quad \text{para } |x| < 1$$

c) Función Seno

$$\sin x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1} \quad \text{para todo } x$$

d) Función Coseno

$$\cos x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n} \quad \text{para todo } x$$

e) Función Arcoseno

$$\arcsin x = \sum_{n=0}^{\infty} \frac{(2n)!}{4^n (n!)^2 (2n+1)} x^{2n+1} \quad \text{para } |x| < 1$$

f) Función Arcotangente

$$\arctan x = \sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1} x^{2n+1} \quad \text{para } |x| < 1$$

g) Función Seno hiperbólico

$$\sinh x = \sum_{n=0}^{\infty} \frac{1}{(2n+1)!} x^{2n+1} \quad \text{para todo } x$$

h) Función Coseno hiperbólico

$$\cosh x = \sum_{n=0}^{\infty} \frac{1}{(2n)!} x^{2n} \quad \text{para todo } x$$

i) Función Arcoseno hiperbólico

$$\sinh^{-1} x = \sum_{n=0}^{\infty} \frac{(-1)^n (2n)!}{4^n (n!)^2 (2n+1)} x^{2n+1} \quad \text{para } |x| < 1$$

j) Función Tangente hiperbólico

$$\tanh^{-1} x = \sum_{n=0}^{\infty} \frac{1}{2n+1} x^{2n+1} \quad \text{para } |x| < 1$$

k) Serie Geométrica

$$\frac{1}{1-x} = \sum_{n=0}^{\infty} x^n \quad \text{para } |x| < 1$$

47. Un modelo de población mundial, en miles de millones de personas, está dado por la ecuación $Poblacion = 6.0e^{0.02t}$, donde t es el tiempo en años ($t = 0$ representa enero de 2000 y $t = 1$ representa enero de 2001). Usando esta fórmula escriba un algoritmo que despliegue una tabla de población anual para los años de enero de 2005 a enero de 2025.

48. Escriba un programa que calcule y despliegue valores para y cuando

$$y = \frac{xz}{x - z}$$

Su programa deberá calcular y para valores de x que varían entre 1 y 5 y valores de z que varían entre 2 y 6. La variable x deberá controlar el ciclo exterior e incrementarse en pasos de 1 y z deberá incrementarse en pasos de 2. Su algoritmo deberá desplegar también el mensaje *Función Indefinida* cuando sea necesario.

49. Los lenguajes ensambladores para algunos microprocesadores no tienen una operación de multiplicación. Aunque hay algoritmos sofisticados para llevar a cabo la multiplicación en estos casos, un método simple multiplica por adición repetida. En este caso la eficiencia del algoritmo puede incrementarse usando ciclos anidados. Por ejemplo, para multiplicar un número por doce, primero suma el número tres veces y luego suma el resultado cuatro veces. Esto solo requiere siete adiciones en vez de doce. Usando esta información escriba un algoritmo que multiplique 33, 47 y 83 por 1001 usando tres ciclos y luego despliegue el resultado. (*Sugerencia:* $1001 = 7 \times 11 \times 13$).

50. Usando una instrucción Hacer-Hasta escriba un programa para aceptar una calificación. El programa deberá solicitar una calificación en forma continua en tanto se introduzca una calificación inválida. Una calificación es inválida si es menor que 0 o mayor que 100. Después que se ha introducido una calificación válida, su programa deberá desplegar el valor de la calificación introducida. Luego:

    a) Modifique el algoritmo anterior de modo que el usuario sea alertado cuando se ha introducido una calificación inválida. Además, se debe permitir al usuario salir del programa introduciendo el número 999.

    b) Modifique de modo que termine en forma automática después que se han introducido cinco calificaciones inválidas.

51. Escriba un algoritmo que solicite en forma continua que se introduzca una calificación. Si la calificación es menor que 0 o mayor que 100, su algoritmo deberá imprimir un mensaje apropiado que informe al usuario que se ha introducido una calificación inválida; de lo contrario, la calificación deberá sumarse a un total. Cuando se introduzca una calificación de 999, el algoritmo deberá salir del ciclo de repetición y calcular y desplegar el promedio de las calificaciones válidas introducidas.

52. Escriba un algoritmo para invertir los dígitos de un número entero positivo. Por ejemplo, si se introduce el número 8735, el número desplegado deberá ser 5378. (*Sugerencia:* use una instrucción Hacer-Hasta y continuamente quite y despliegue el dígito de las unidades del número). Si la variable *num* en un inicio contiene el número introducido, el dígito de las unidades se obtiene como $num\,MOD\,10$. Después que se despliega un dígito de las unidades, dividir el número entre 10 establece el número para la siguiente iteracción. Por tanto, $8735\,MOD\,10$ es 5 y $8735\,DIV\,10$ es 873. La instrucción Hacer-Hasta deberá continuar en tanto el número restante no sea cero.

53. Dado un número n, y una aproximación para su raíz cuadrada, puede obtenerse una aproximación más cercana a la raíz cuadrada real usando la fórmula

$$\text{aproximación previa} = \frac{\frac{n}{\text{aproximación previa}} + \text{aproximación previa}}{2}$$

Usando esta información escriba un algoritmo que indique al usuario que introduzca un número y una estimación inicial de su raíz cuadrada. Usando estos datos de entrada, su programa deberá calcular una aproximación a la raíz cuadrada que tenga una precisión hasta 0.0001. (*Sugerencia:* detenga el ciclo cuando la diferencia entre dos aproximaciones sea menor que 0.0001).

54. El método de Newton-Raphson puede utilizarse para encontrar las raíces de cualquier ecuación $y(x) = 0$. En este método, la $(i+1)$-ésima aproximación, x_{i+1}, a una raíz de $y(x) = 0$ está dada en términos de la i-ésima aproximación, x_i, por la fórmula

$$x_{i+1} = x_i - \frac{y(x_i)}{y'(x_i)}$$

Por ejemplo, si $y(x) = 3x^2 + 2x - 2$, entonces $y'(x) = 6x + 2$, y las raíces se encuentran haciendo una estimación razonable para una primera aproximación, x_i, y haciendo iteraciones usando la ecuación

$$x_{i+1} = x_i - \frac{3{x_i}^2 + 2x_i - 2}{6x_i + 2}$$

a) Usando el método de Newton-Raphson encuentre las dos raíces de la ecuación $3x^2 + 2x - 2 = 0$. (*Sugerencia:* hay una raíz positiva y una raíz negativa).

b) Extienda el algoritmo escrito para el inciso anterior de modo que encuentre las raíces de cualquier función cuadrática cuando los coeficientes a, b, c de $ax^2 + bx + c$ son introducidos por el usuario.

55. Una serie aritmética se define por

$$a + (a + d) + (a + 2d) + (a + 3d) + \cdots + [(a + (n - 1)d)]$$

donde a es el primer término, d es la "diferencia común" y n es el número de términos que se van a sumar. Usando esta información escriba un algoritmo

que use el ciclo Mientras que para desplegar cada término y para determinar la suma de la serie aritmética si se tiene que a, d y n son introducidos por el usuario. Asegúrese de que su programa despliegue el valor calculado.

Entrada: $a = 1, d = 1, n = 3$
Salida: 6

56. Una serie geométrica se define por

$$a + ar + ar^2 + ar^3 + ? + ar^{n-1}$$

donde a es el primer término, r es la "razón común" y n es el número de términos en la serie. Usando esta información escriba un algoritmo que utilice un ciclo Mientras que para desplegar cada término y para determinar la suma de una serie geométrica si se tiene que a, r y n son introducidos por el usuario. Asegúrese de que su programa despliega el valor que se ha calculado.

Entrada: $a = 1, r = 2, n = 3$
Salida: 7

57. Además del promedio aritmético de un conjunto de números se puede calcular una media aritmética y una media armónica. La media aritmética de un conjunto de n números $x_1, x_2, x_3, \cdots, x_n$ se define como

$$\sqrt[n]{\prod_{i=1}^{n} x_i} = \sqrt[n]{x_1 x_2 \cdots x_n}$$

Y la media armónica como

$$\frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}} = \frac{n}{\frac{1}{x_1} + \frac{1}{x_2} + \cdots + \frac{1}{x_n}}$$

Usando estas fórmulas escriba un algoritmo que continúe aceptando números hasta que se introduzca el numero 999, y luego calcule y despliegue tanto la media aritmética como la armónica de los números introducidos, excepto el 999. (*Sugerencia:* será necesario que su programa cuente en forma correcta el número de valores introducidos.)

58. Escriba un programa que reciba como entrada dos enteros positivos, a y b, con $a \leq b$ e imprima todos los cuadrados perfectos que están entre a y b (incluyendo a y b). Un número es cuadrado perfecto si es el cuadrado de algún número entero, por ejemplo, 9 es cuadrado perfecto ($9 = 3^2$).
Entrada: 8 16
Salida: 9 16

59. Haga un algoritmo que reciba como entrada dos números enteros, a y b, e imprima la suma de todos los números entre a y b (incluyendo a y b). Se garantiza que $a \leq b$.
Entrada: 1, 5
Salida: 15

60. Los siguientes datos se recolectaron en un viaje frecuente en automóvil:

Millas	Galones
22495	Tanque lleno
22841	12.2
23185	11.3
23400	10.5
23772	11.0
24055	12.2
24434	14.7
24804	14.3
25276	15.2

Escriba un algoritmo que acepte un valor de millas y galones y calcule las millas por galón (mpg) logradas para ese segmento del viaje. Las millas por galón se obtienen como la diferencia en millas entre llenadas del tanque divido entre el número de galones de gasolina utilizados desde que se llenó el tanque.

61. Modifique el algoritmo escrito en el inciso anterior para calcular y desplegar adicionalmente las mpg acumuladas después de cada llenada de tanque. Las mpg acumuladas se calculan como la diferencia entre el millas en cada llenada y el millas al principio del viaje dividido entre la suma de los galones usados hasta ese punto en el viaje.

62. Haga un algoritmo que reciba como entrada un entero positivo n ($1 \leq n \leq 20$), a continuación n enteros y que diga de esos n enteros cuántos son impares.
Entrada: 5, 4, 3, 8, 1, 24
Salida: 3

63. Haga un algoritmo que reciba como entrada un número entero positivo y que imprima los factores primos de dicho número.
Entrada: 60
Salida: 2, 3, 5

64. Una máquina comprada por 28 000 pesos se devalúa 4000 pesos por año durante siete años. Escriba un algoritmo que calcule y despliegue una tabla de devaluación para siete años. La tabla deberá tener la forma

Año	**Devaluación**	**Valor al final del año**	**Devaluación acumulada**
1	4000	24000	4000
2	4000	20000	8000
3	4000	16000	12000
4	4000	12000	16000
5	4000	8000	20000
6	4000	4000	24000
7	4000	0	28000

65. Determine el número de billetes y monedas de curso legal equivalentes a cierta cantidad de pesos (Cambio óptimo).
Entrada: 32300
Salida:
Billetes de 20 mil: 1
Billetes de 10 mil: 1
Billetes de 2 mil: 1
Monedas de 200: 1
Monedas de 100: 1

66. Un automóvil viaja a una velocidad promedio de 55 millas por hora durante cuatro horas. Escriba un algoritmo que despliegue la distancia, en millas, que el automóvil ha recorrido después de 1, 2 o varias horas hasta el final del viaje.

3.10.3 Repaso

67. Dado una base k y un número n, diseñe un algoritmo que determine el número de ceros en los que termina $n!$ y el número de cifras cuando es representado en dicha base. Se garantiza la existencia de los símbolos $0, b_1, b_2, \ldots, b_{k-1}$ para los números $0, 1, 2, \ldots, k-1$, respectivamente.

Supongamos que $k = 4$, luego los números que representan a cualquier número son 0123. Por otro lado, supongamos $n = 8$, luego $n! = 40320$, que en base 4 sería 21312000_4, luego el número termina en 3 ceros y tiene 8 cifras.

Entrada: n, k
Salida: Número de cifras de $n!$ y número de ceros en los que termina, en base k.

Entrada Ejemplo: $n = 8, k = 4$
Salida Ejemplo: Tiene 8 cifras y termina en 3 ceros.

68. Modifique el algoritmo anterior de tal forma que no sea necesario evaluar $n!$. Piense en que si $n = 2012$, ¿cuántas cifras se necesitan para calcular inicialmente $n!$?, ¿qué tamaño se requiere en el computador? Escriba su algoritmo en C++, Java o algún lenguaje de programación. Compare el tiempo de ejecución de la versión que calcula $n!$ con la versión que no lo hace. Saque sus conclusiones.

69. La fracción 49/98 es una fracción curiosa; un matemático inexperto en el intento de simplificar creyó incorrectamente que $49/98 = 4/8$, sin embargo, resultó ser correcto, al cancelar los 9's. Se consideran ejemplos triviales fracciones como $30/50 = 3/5$ (en los que se cancelan los ceros). Hay exactamente cuatro ejemplos no triviales de este tipo de fracción, menor que uno en valor, y que contienen dos dígitos en el numerador y el denominador. Si el producto

de estas cuatro fracciones forma una fracción a/b, en la que a y b no tienen factores en común, encuentre el valor de b.
Solución: 100.

3.11 Conclusiones

El aprendizaje y la práctica de las estructuras lógicas básicas para la construcción de algoritmos permiten:

- Organizar las acciones de entrada de datos, procesos aritmético-lógicos o de asignación, y salida de informaciones derivadas del correcto funcionamiento del algoritmo.
- Las estructuras lógicas algorítmicas condicionales simples y compuestas, unidas a las anteriores, permiten hacer algoritmos que necesitan acciones de decisión disyuntas (Sí o No); útiles para la simulación de procesos de árboles de decisión; clasificaciones de grupos, teoría de juegos, simulaciones y procesos estocásticos, entre otros, en los cuales se tenga que tomar una ruta de acción sujeta a una decisión.
- La acción de decisión en la lógica humana, y representada en algoritmos por las condicionales expuestas, se complementa con la estructura también condicional de lógica Dependiendo De; dicha condición sirve cuando se requiere, entre varias acciones, seleccionar una sola, tal que al operar la acción seleccionada de una forma continuada se ejecuten varias estructuras algorítmicas que cumplan con la función de la acción seleccionada y se termine la lógica del algoritmo; estructuras que necesariamente son disyuntas a las acciones que no fueron seleccionadas dentro del algoritmo.
- Las estructuras algorítmicas repetitivas representadas por el Para, el Mientras que y el Haga Hasta, complementadas con las anteriores, apoyan la fabricación de algoritmos que necesitan ciclos, procesos repetitivos, acciones paso a paso, procesos de principio a fin o de fin a principio de un fenómeno, procesos circulares, procesos encapsulados al anidar las estructuras repetitivas o acciones oscilantes.
- Con base en las estructuras algorítmicas expuestas en este capítulo se obtienen diseños algorítmicos útiles que, como piezas de software funcionales, se pueden integrar a nivel de subsistemas de algoritmos iniciales en piezas de software más complejas.
- La potencia y el alcance de las estructuras algorítmicas expuestas solo dependen del ingenio y creatividad con que el diseñador de algoritmos las pueda utilizar en la resolución de un problema de la vida real.

- Es importante resaltar que los casos presentados a nivel de resolución de problemas están estrechamente relacionados con sistemas de información reales con relación a sistemas universitarios, bancarios o de administración de edificios; y fueron desarrollados todos ellos con la integración de las estructuras lógicas básicas para la construcción de algoritmos; se visiona, por lo tanto, su importancia y el horizonte de su alcance en la producción de software. El paso siguiente es complementar la estructuras lógicas algorítmicas básicas con las estructuras de datos y su control; tema que será desarrollado en el siguiente capítulo.

Bibliografía

Bronson, G. J. & Borse, G. J. (2010). *C++ for engineers and scientists*. Boston: Thomson Course Technology.

García, L. (2010). *Todo lo básico que debería saber sobre Programación Orientada a Objetos en Java*. Barranquilla: Ediciones Uninorte.

Rueda, F. (1981). *Curso de Estructuras de Información*, vol. I. Bogotá, D.C.: Publicaciones de la Facultad de Ingeniería, Universidad de los Andes.