

4.2.1 Definición de arreglos	182
4.2.2 Operaciones con arreglos	184
4.3 Arreglos multidimensionales	209
4.3.1 Arreglos bidimensionales	210
4.3.2 Arreglos de más de dos dimensiones	222
Problemas resueltos	229
Estructuras de datos: registros	343
5.1 Registros	343
5.1.1 Definición de registros	344
5.1.2 Acceso a los campos de un registro	346
5.1.3 Diferencias con arreglos	347
5.1.4 Combinaciones entre arreglos y registros	347
Problemas resueltos	354
Problemas supplementarios	401
Arreglos unidimensionales	441
Arreglos bidimensionales	446
Arreglos de más de dos dimensiones	453
Arreglos paralelos	454
Indice	463

Algoritmos, diagramas de flujo y programas

Metodología de la Programación

Osvaldo Cairo

1.1 Problemas y algoritmos

Casi inconscientemente, los humanos efectuamos cotidianamente una serie de pasos, procedimientos o acciones que nos permiten alcanzar un resultado o resolver un problema.

Esta serie de pasos, procedimientos o acciones, comenzamos a aplicarlas muy temprano en la mañana cuando, por ejemplo, decidimos tomar un baño. Posteriormente cuando pensamos en desayunar también seguimos una serie de pasos que nos permiten alcanzar un resultado específico: tomar el desayuno. La historia se repite innumerables veces durante el día. Continuamente seguimos una serie de pasos o conjunto de acciones que nos permiten alcanzar un resultado. Estamos en realidad aplicando un *algoritmo para resolver un problema*.

“Formalmente definimos un algoritmo como un conjunto de pasos, procedimientos o acciones que nos permiten alcanzar un resultado o resolver un problema”.

Muchas veces aplicamos el algoritmo de manera inadvertida, inconsciente o automáticamente. Esto generalmente se produce cuando el problema que tenemos enfrente lo hemos resuelto con anterioridad un gran número de veces.

Supongamos que simplemente tenemos que abrir una puerta. Lo hemos hecho tantas veces que difícilmente nos ponemos a enumerar los pasos para alcanzar este objetivo. Lo hacemos de manera automática. Lo mismo ocurre cuando queremos subirnos a un automóvil, cuando tenemos que lustrar nuestros zapa-

tos, cuando nos calzamos, cuando nos vestimos, cuando tenemos desafortunadamente que cambiar la llanta de un automóvil o, simplemente cuando queremos tomar un vaso con agua.

Por otra parte, existe una gran cantidad de problemas que requieren de un análisis profundo y de un pensamiento flexible y estructurado para su solución. En este libro nos interesa abordar ese tipo de problemas. Invariablemente surgen ciertas preguntas:

- ¿Podemos enseñar a resolver un problema?
- ¿Podemos enseñar a analizar el mismo?
- ¿Podemos enseñar a pensar . . . ?

Lógicamente las respuestas a estas interrogantes son difíciles de obtener. No existen reglas específicas que nos permitan resolver un problema. Sin embargo, creemos que se pueden ofrecer un conjunto de técnicas y herramientas metodológicas que permitan flexibilizar y estructurar el razonamiento utilizado en la solución de un problema. Eso provocará finalmente la construcción de algoritmos eficientes.

Ejemplo 1.1

Construya un algoritmo para preparar “Pechugas de pollo en salsa de elote y chile poblano”

Ingredientes (para 6 personas):

3 pechugas deshuesadas, sin piel y partidas a la mitad.

1 diente de ajo.

4 gramos de pimienta negra.

sal.

6 cucharadas de aceite.

5 chiles poblanos asados y limpios.

$\frac{1}{2}$ taza de leche.

$\frac{1}{4}$ taza de crema ligera.

1 lata de crema de elote.

Algoritmo (Preparación):

- Muela el ajo, la pimienta y un poco de sal y únteselo a las pechugas.
- Caliente el aceite y dore las pechugas.
- Licue los chiles con la leche y la crema, y mézclelos con la crema de elote.
- En una fuente coloque las pechugas y báñelas con la mezcla anterior.
- Cubra el platón con papel aluminio y hornee a 200°C, durante 15 minutos.



Nota: El algoritmo fue probado repetidas veces y siempre arrojó el mismo resultado: un platillo exquisito.

En la figura 1.1 podemos observar las etapas que debemos seguir para la solución de un problema.

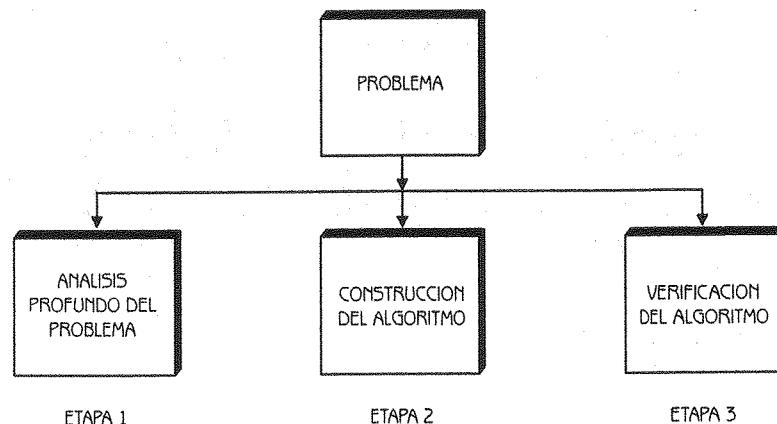


Figura 1.1 Etapas de la Solución de un Problema.



Nota:

Por verificación del algoritmo entendemos el seguimiento del mismo con datos que sean representativos del problema que queremos resolver.

Las características que los algoritmos deben reunir son las siguientes:

Precisión: Los pasos a seguir en el algoritmo deben ser precisados claramente.

Determinismo: El algoritmo, dado un conjunto de datos idénticos de entrada, siempre debe arrojar los mismos resultados.

Finitud: El algoritmo, independientemente de la complejidad del mismo, siempre debe ser de longitud finita.

Por otra parte, un algoritmo consta de tres secciones o módulos principales. En la figura 1.2 podemos observar las secciones que constituyen un algoritmo.

El **módulo 1** representa la operación o acción que permite el ingreso de los datos del problema.

El **módulo 2** representa la operación o conjunto de operaciones secuenciales, cuyo objetivo es obtener la solución al problema.

El **módulo 3** representa una operación o conjunto de operaciones que permiten comunicar al exterior el o los resultados alcanzados.

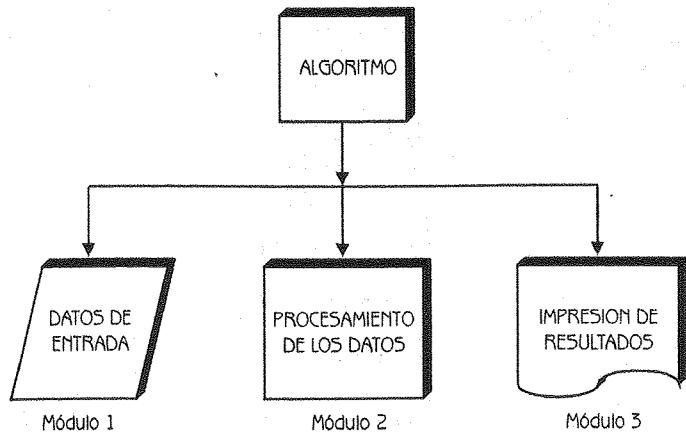


Figura 1.2 Módulos o Secciones de un Algoritmo.

1.2 Diagramas de flujo

Un diagrama de flujo representa la esquematización gráfica de un algoritmo. En realidad muestra gráficamente los pasos o procesos a seguir para alcanzar la solución de un problema. Su correcta construcción es sumamente importante porque a partir del mismo se escribe un programa en algún lenguaje de programación. Si el diagrama de flujo está completo y correcto, el paso del mismo a un lenguaje de programación es relativamente simple y directo.

A continuación en la tabla 1.1 presentamos los símbolos que utilizaremos, y una explicación de los mismos. Estos satisfacen las recomendaciones de la “International Organization for Standardization” (ISO) y la “American National Standards Institute” (ANSI).

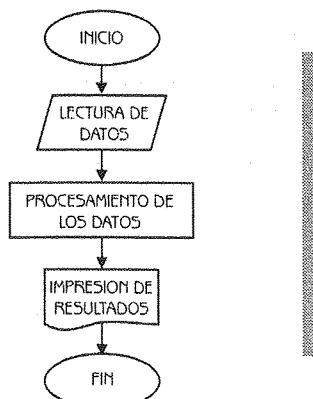
A continuación en la figura 1.3 presentamos las etapas que debemos seguir en la construcción de un diagrama de flujo.

1.2.1 Reglas para la construcción de diagramas de flujo

Debemos recordar que un diagrama de flujo debe ilustrar gráficamente los pasos o procesos a seguir para alcanzar la solución de un problema. Los símbolos presentados, colocados adecuadamente, permiten crear una estructura gráfica flexible que ilustra los pasos a seguir para alcanzar un resultado específico. El diagrama de flujo facilitará más tarde la escritura del programa en algún lenguaje de programación.

Tabla 1.1 Símbolos utilizados en los Diagramas de Flujo

Representación del Símbolo	Explicación del Símbolo
	Símbolo utilizado para marcar el inicio y el fin del diagrama de flujo.
	Símbolo utilizado para introducir los datos de entrada. Expresa lectura.
	Símbolo utilizado para representar un proceso. En su interior se expresan asignaciones, operaciones aritméticas, cambios de valor de celdas en memoria, etc.
	Símbolo utilizado para representar una decisión. En su interior se almacena una condición, y dependiendo del resultado de la evaluación de la misma se sigue por una de las ramas o caminos alternativos. Este símbolo se utiliza en la estructura selectiva si entonces que estudiaremos en el siguiente capítulo, y en las estructuras repetitivas repetir y mientras que analizaremos en el capítulo 3.
	Símbolo utilizado para representar la estructura selectiva doble si entonces/sino. En su interior se almacena una condición. Si el resultado es verdadero se continúa por el camino de la izquierda, y si es falso por el camino de la derecha.
	Símbolo utilizado para representar una decisión múltiple. En su interior se almacena un selector, y dependiendo del valor de dicho selector se sigue por una de las ramas o caminos alternativos. Este símbolo se utiliza en la estructura selectiva si múltiple, que analizaremos en el siguiente capítulo.
	Símbolo utilizado para representar la impresión de un resultado. Expresa escritura.
	Símbolos utilizados para expresar la dirección del flujo del diagrama.
	Símbolo utilizado para expresar conexión dentro de una misma página.
	Símbolo utilizado para expresar conexión entre páginas diferentes.
	Símbolo utilizado para expresar un módulo de un problema. En realidad expresa que para continuar con el flujo normal del diagrama debemos primero resolver el subproblema que enumera en su interior.

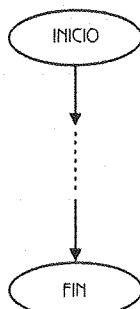
**Nota:**

Se debe observar que estas fases se presentan en la mayoría de los diagramas de flujo, aunque a veces en orden diferente o repitiendo alguna(s) de ellas. También es frecuente tener que realizar *toma de decisiones* y *repetir* una serie de pasos un número determinado o no de veces, pero estos conceptos serán analizados en capítulos posteriores.

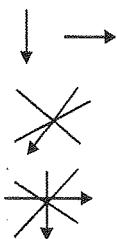
Figura 1.3 Etapas en la Construcción de un Diagrama de Flujo.

A continuación presentamos un conjunto de reglas que permiten la construcción de diagramas de flujo.

1. Todo diagrama de flujo debe tener un *inicio* y un *fin*.



2. Las líneas utilizadas para indicar la dirección del flujo del diagrama deben ser rectas, verticales y horizontales.

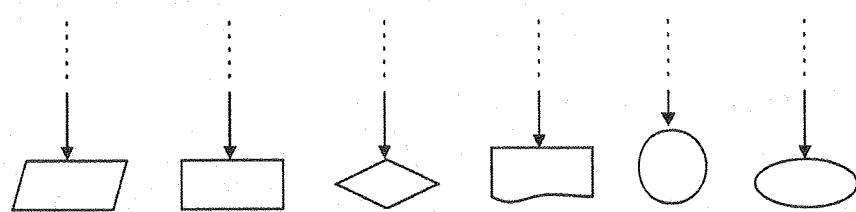


No deben ser inclinadas

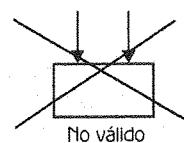
Tampoco debemos cruzarlas

1.2 Diagramas de flujo

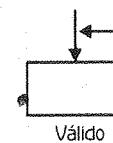
3. Todas las líneas utilizadas para indicar la dirección del flujo del diagrama deben estar conectadas. La conexión puede ser a un símbolo que exprese lectura, proceso, decisión, impresión, conexión o fin de diagrama.



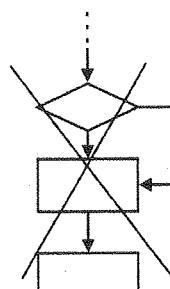
4. El diagrama de flujo debe ser construido de arriba hacia abajo (top-down) y de izquierda a derecha (right to left).
5. La notación utilizada en el diagrama de flujo debe ser independiente del lenguaje de programación. La solución presentada en el diagrama puede escribirse posteriormente y fácilmente en diferentes lenguajes de programación.
6. Es conveniente cuando realizamos una tarea compleja poner comentarios que expresen o ayuden a entender lo que hicimos.
7. Si el diagrama de flujo requiriera más de una hoja para su construcción, debemos utilizar los conectores adecuados y enumerar las páginas convenientemente.
8. No puede llegar más de una línea a un símbolo.



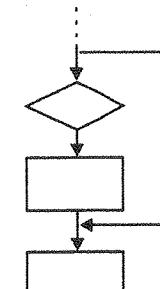
No válido



Válido



No válido



Válido

A continuación nos gustaría mostrar algunos ejemplos con el objeto de que el lector se familiarice con la simbología presentada. Sin embargo, pensamos que antes es conveniente exponer algunos conceptos muy importantes que serán fundamentales para la construcción de los diagramas de flujo. Dejaremos los ejemplos para la sección 1.4.

1.3 Conceptos fundamentales

En esta sección trataremos algunos conceptos que son fundamentales para la construcción de algoritmos, diagramas de flujo y programas. Primero analizaremos los tipos de datos, luego estudiaremos los conceptos de identificador, constantes y variables, más adelante analizaremos las operaciones aritméticas y expresiones lógicas. Por último estudiaremos los bloques de asignación.

1.3.1 Tipos de datos

Los datos a procesar por una computadora pueden clasificarse en:

- Simples
- Estructurados

La principal característica de los datos simples es que ocupan sólo una casilla de memoria (Fig. 1.4a), por lo tanto, una variable simple hace referencia a un único valor a la vez. Dentro de este grupo de datos se encuentran: enteros, reales, caracteres, booleanos, enumerados y subrangos (los dos últimos no existen en algunos lenguajes de programación).

Los datos estructurados se caracterizan por el hecho de que con un nombre identificador de variable estructurada) se hace referencia a un grupo de casillas de memoria (Fig. 1.4b). Es decir, un dato estructurado tiene varios componentes. Cada uno de los componentes puede ser a su vez un dato simple o estructurado. Sin embargo, los componentes básicos (los del nivel más bajo) de cualquier tipo estructurado son datos simples. Dentro de este grupo de datos se encuentran: arreglos, cadena de caracteres, registros y conjuntos.

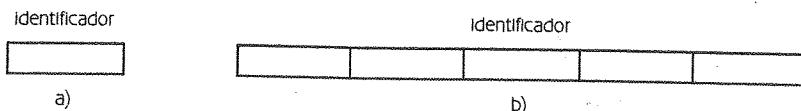


Figura 1.4 Datos simples y estructurados. a) Dato simple. b) Dato estructurado.

A continuación trataremos los datos simples: enteros, reales, caracteres y booleanos; y el dato estructurado: cadena de caracteres. Posteriormente en los capítulos 4 y 5, estudiaremos los datos estructurados arreglos y registros.

1.3 Conceptos fundamentales

Datos numéricos

Dentro de los tipos de datos numéricos encontramos los *enteros* y los *reales*. Los enteros son números que pueden estar precedidos del signo + o -, y que no tienen parte decimal. Por ejemplo:

128 1528 -714 8530 16235 -14780

Los reales son números que pueden estar precedidos del signo + o -, y que tienen una parte decimal. Por ejemplo:

7.5 128.0 -37.865 129.7 16000.50 -15.0

Datos alfanuméricos

Dentro de este tipo de datos encontramos los de tipo carácter (simple) y cadena de caracteres (estructurado). Son datos cuyo contenido pueden ser letras del abecedario (a,b,c,...,z), dígitos (0, 1, 2, ..., 9) o símbolos especiales (#, \$, ^, *, %, /, !, +, -, etc.). Debemos remarcar que aunque este tipo de datos pueden contener números, no pueden ser utilizados para realizar operaciones aritméticas.

Un dato tipo carácter contiene un solo carácter, y se escribe entre apóstrofes. Por ejemplo:

'a' 'B' '\$' '9' '!' '#' 'f'

Un dato tipo cadena de caracteres contiene un conjunto de caracteres, y se escribe entre comillas. La longitud de una cadena depende de los lenguajes de programación, aunque normalmente se acepta una longitud máxima de 255.

"abcde" "\$9#7" "Carlos Gómez" "Rosario" "754-27-22"

Datos lógicos

Dentro de este tipo de datos encontramos los booleanos. Son datos que sólo pueden tomar dos valores: verdadero (true) o falso (false).

1.3.2 Identificadores, constantes y variables

Identificadores

Los datos a procesar por una computadora, ya sean simples o estructurados, deben almacenarse en casillas o celdas de memoria para su posterior utilización. Estas casillas o celdas de memoria (constantes o variables) tienen un *nombre* que permite su identificación.

Llamaremos identificador al nombre que se les da a las casillas de memoria. Un identificador se forma de acuerdo a ciertas reglas (las mismas pueden tener alguna variante dependiendo del lenguaje de programación utilizado):

- El primer carácter que forma un identificador debe ser una letra (a, b, c, ..., z).
- Los demás caracteres pueden ser letras (a,b,c,...,z), dígitos (0,1,2,...,9) o el siguiente símbolo especial: _.
- La longitud del identificador es igual a 7 en la mayoría de los lenguajes de programación.

A continuación, en la figura 1.5, podemos observar ejemplos de identificadores.

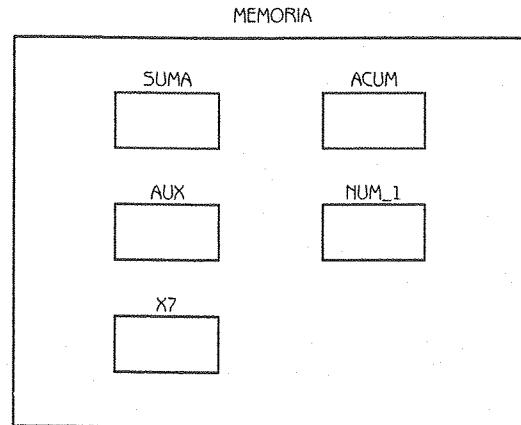


Figura 1.5 Casillas de Memoria con los Nombres de Identificadores.

Constantes

Las *constantes* son datos que no cambian durante la ejecución de un programa. Para nombrar las constantes utilizamos los identificadores que mencionamos anteriormente. Existen tipos de constantes como tipos de datos, por lo tanto, puede haber constantes de tipo entero, real, carácter, cadena de caracteres, etc.

Observe que en la figura 1.6, la constante NUM es de tipo entero, NREAL y NUMREA son de tipo real, y RESU de tipo cadena de caracteres. Estas constantes no cambiarán su valor durante la ejecución del programa. Es muy importante que los nombres de las constantes sean representativas de la función que tienen las mismas en el programa.

Variables

Las *variables* son objetos que pueden cambiar su valor durante la ejecución de un programa. Para nombrar las variables utilizaremos los identificadores que hemos explicado con anterioridad. Al igual que las constantes, pueden existir tipos de variables como tipos de datos.

1.3 Conceptos fundamentales

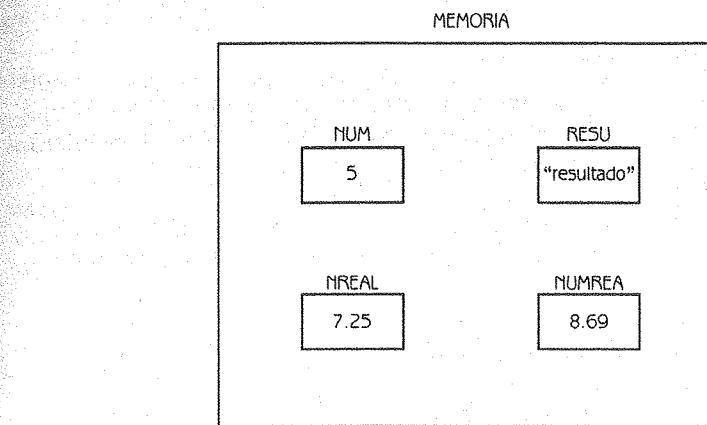


Figura 1.6 Constantes representadas en la Memoria.

En la figura 1.7, la variable I es de tipo entero, tiene un valor inicial de cero y cambiará su valor durante la ejecución del programa. Las variables SUEL y SUMA son de tipo real, están inicializadas con el valor de cero, y al igual que la variable I, seguramente cambiarán su valor durante la ejecución del programa.

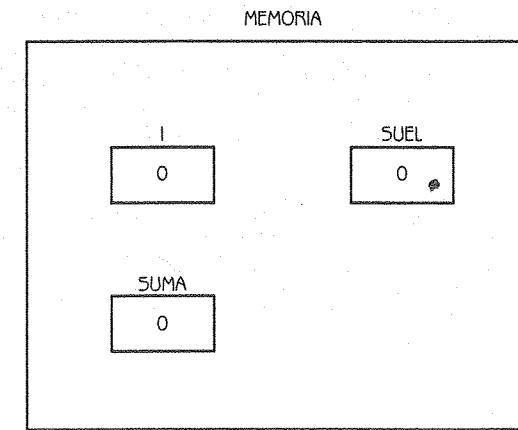


Figura 1.7 Variables representadas en la Memoria.

Debemos remarcar que los nombres de las variables deben ser representativos de la función que cumplen en el programa.

1.3.3 Operaciones aritméticas

Para poder realizar operaciones aritméticas necesitamos de operadores aritméticos. Estos operadores nos permitirán realizar operaciones aritméticas entre operandos: números, constantes o variables. El resultado de una operación aritmética será un número.

A continuación en la tabla 1.2 presentamos los operadores aritméticos, la operación que pueden realizar, un ejemplo de su uso y el resultado de dicho ejemplo.

Tabla 1.2 Operadores Aritméticos			
Operador Aritmético	Operación	Ejemplo	Resultado
**	Potencia	4^3	64
*	Multiplicación	$8.25 \cdot 7$	57.75
/	División	$15/4$	3.75
+	Suma	$125.78 + 62.50$	188.28
-	Resta	$65.30 - 32.33$	32.97
mod	Módulo (residuo)	$15 \text{ mod } 2$	1
div	División entera	$17 \text{ div } 3$	5

Al evaluar expresiones que contienen operadores aritméticos debemos respetar la jerarquía en el orden de aplicación. Es decir, si tenemos en una expresión más de un operador, debemos aplicar primero el operador de mayor jerarquía, resolver esa operación, y así sucesivamente. Es importante señalar que el operador () es un operador asociativo que tiene la prioridad más alta en cualquier lenguaje de programación. En la tabla 1.3 se presenta la jerarquía de los operadores.

Tabla 1.3 Jerarquía de los Operadores Aritméticos		
Operador	Jerarquía	Operación
**	(mayor) ↓	Potencia
*, /, mod, div		Multiplicación, división, módulo, división entera
+, -	(menor)	Suma, resta

Las reglas para resolver una expresión aritmética son las siguientes:

- Si una expresión contiene subexpresiones entre paréntesis, éstas se evalúan primero; respetando claro está la jerarquía de los operadores aritméticos en esta subexpresión. Si las subexpresiones se encuentran anidadas por paréntesis, primero se evalúan las subexpresiones que se encuentran en el último nivel de anidamiento.

1.5 Programas

- Los operadores aritméticos se aplican teniendo en cuenta la jerarquía y de izquierda a derecha.

Ejemplo 1.2

A continuación en este ejemplo presentamos varios casos y la forma de resolver los mismos.

Caso a)

$$\begin{array}{l} 7 + 5 - 6 \\ \hline 1 \\ 12 - 6 \\ \hline 2 \\ 6 \end{array}$$

Caso b)

$$\begin{array}{l} 9 + 7 * 8 - 36 / 5 \\ \hline 1 \\ 9 + 56 - 36 / 5 \\ \hline 2 \end{array}$$

$$\begin{array}{l} 9 + 56 - 7 \cdot 2 \\ \hline 3 \\ 65 - 7 \cdot 2 \\ \hline 4 \\ 57.8 \end{array}$$

Caso c)

$$\begin{array}{l} 7 * 5 ** 3 / 4 \text{ div } 3 \\ \hline 1 \\ 7 * 125 / 4 \text{ div } 3 \\ \hline 2 \end{array}$$

$$\frac{875}{4} \text{ div } 3$$

3

$$\frac{218.75}{4}$$

4

72

$$7 * 8 * (\frac{160 \text{ mod } 3}{3} \text{ ** } 3) \text{ div } 5 * 13 - 28$$

1

$$7 * 8 * (\frac{160 \text{ mod } 27}{27}) \text{ div } 5 * 13 - 28$$

2

$$\frac{7 * 8 * 25}{3} \text{ div } 5 * 13 - 28$$

3

$$\frac{56 * 25}{4} \text{ div } 5 * 13 - 28$$

4

$$\frac{1400}{5} \text{ div } 5 * 13 - 28$$

5

$$\frac{280 * 13}{6} - 28$$

$$\frac{3640}{7} - 28$$

3612

Caso e)

$$15 / 2 * (7 + (68 - 15 * 33 + \frac{(45 \text{ ** } 2 / 16)}{3}) / 15) + 19$$

1

$$15 / 2 * (7 + (68 - 15 * 33 + \frac{(2025 / 16)}{3}) / 15) + 19$$

2

$$15 / 2 * (7 + (68 - 15 * 33 + \frac{126.5625 / 3}{3}) / 15) + 19$$

$$15 / 2 * (7 + (68 - 495 + \frac{126.5625 / 3}{4}) / 15) + 19$$

$$15 / 2 * (7 + (68 - 495 + \frac{42.1875}{5}) / 15) + 19$$

$$15 / 2 * (7 + (-427 + \frac{42.1875}{6}) / 15) + 19$$

$$15 / 2 * (7 + (-384.8125 / 15)) + 19$$

7

$$15 / 2 * (7 + (-25.6541)) + 19$$

8

$$15 / 2 * (-18.6541) + 19$$

9

$$7.5 * (-18.6541) + 19$$

10

$$-139.9062 + 19$$

11

-120.9062

1.3.4 Expresiones lógicas

Las expresiones lógicas o booleanas, llamadas así en honor del matemático George Boole, están constituidas por números, constantes o variables y operadores lógicos o relacionales. El valor que pueden tomar estas expresiones es el de *verdadero* o *falso*. Se utilizan frecuentemente en las estructuras selectivas (dependiendo del resultado de la evaluación se toma por un determinado camino alternativo) y en las estructuras repetitivas (dependiendo del resultado de la evaluación se continúa con el ciclo o se interrumpe al mismo).

Operadores relacionales

Los operadores relacionales son operadores que permiten comparar dos operandos. Los operandos pueden ser números, alfanuméricos, constantes o variables. Las constantes o variables, a su vez, pueden ser de tipo entero, real, carácter o cadena de caracteres. El resultado de una expresión con operadores relacionales es verdadero o falso.

A continuación en la tabla 1.4 presentamos los operadores relacionales, la operación que pueden realizar, un ejemplo de su uso y el resultado de dicho ejemplo.

Tabla 1.4 Operadores Relacionales			
Operador	Operación	Ejemplo	Resultado
=	Igual que	'hola' = 'lola'	FALSO
< >	Diferente a	'a' <> 'b'	VERDADERO
<	Menor que	7 < 15	VERDADERO
>	Mayor que	22 > 11	VERDADERO
<=	Menor o igual que	15 <= 22	VERDADERO
>=	Mayor o igual que	35 >= 20	VERDADERO

Ejemplo 1.3

En este ejemplo presentamos varios casos de expresiones lógicas con operadores relacionales y la forma de resolver las mismas.

Caso a)

$$A = 5$$

$$B = 16$$

$$(A^2) > (B^2)$$

1

$$25 > (B^2)$$

2

$$25 > 32$$

3

FALSO

1.5 Programas

Caso b)

$$X = 6$$

$$B = 7.8$$

$$(X^5 + \underline{B^3} / 4) \leq (X^3 \text{ div } B)$$

1

$$(X^5 + 474.552 / 4) \leq (X^3 \text{ div } B)$$

2

$$(30 + \underline{474.552 / 4}) \leq (X^3 \text{ div } B)$$

3

$$(\underline{30 + 118.638}) \leq (X^3 \text{ div } B)$$

4

$$148.638 \leq (\underline{X^3 \text{ div } B})$$

5

$$148.638 \leq (\underline{216 \text{ div } B})$$

6

$$148.638 \leq 27$$

7

FALSO

Caso c)

$$((1580 \text{ mod } 6^2)^2 \cdot 7) > (7 + 8 \cdot 3^4) > ((15 \cdot 2) = (60 \cdot 2 / 4))$$

1

$$((\underline{1580 \text{ mod } 6^2} \cdot 128) > (7 + 8 \cdot 3^4)) > ((15 \cdot 2) = (60 \cdot 2 / 4))$$

2

$$((2^128) > (7 + 8 \cdot 3^4)) > ((15 \cdot 2) = (60 \cdot 2 / 4))$$

3

$$(256 > (7 + 8 * \boxed{3} ** 4)) > ((15 * 2) = (60 * 2 / 4))$$

4

$$(256 > (7 + 8 * \boxed{81})) > ((15 * 2) = (60 * 2 / 4))$$

5

$$(256 > (\boxed{7} + 648)) > ((15 * 2) = (60 * 2 / 4))$$

6

$$(\boxed{256} > 655) > ((15 * 2) = (60 * 2 / 4))$$

7

$$\text{FALSO} > (\boxed{15 * 2} = (60 * 2 / 4))$$

8

$$\text{FALSO} > (30 = (\boxed{60 * 2} / 4))$$

9

$$\text{FALSO} > (30 = (\boxed{120} / 4))$$

10

$$\text{FALSO} > (\boxed{30} = 30)$$

11

FALSO > VERDADERO

FALSO

Nota:

Cuando se utilizan los operadores de relación con operandos lógicos, falso es menor que verdadero.

Operadores lógicos

Los **operadores lógicos** son operadores que permiten formular condiciones complejas a partir de condiciones simples. Los operadores lógicos son de conjunción (y), disyunción (o) y negación (no). En la tabla 1.5 presentamos el operador lógico, la expresión lógica y significado de dicha expresión, teniendo en cuenta la jerarquía correspondiente.

Tabla 1.5 Operadores Lógicos			
Operador lógico	Jerarquía	Expresión lógica	Significado
NO	(mayor)	No P	NO P No es cierto que P Es FALSO que P
Y		P y Q	P sin embargo Q
O	(menor)	P o Q	P o Q o P o Q o ambas Mínimo P o Q

A continuación en la tabla 1.6 presentamos la tabla de verdad de los operadores lógicos.

Tabla 1.6 Tabla de verdad de los Operadores Lógicos					
P	Q	$\sim P$	$\sim Q$	$P \circ Q$	$P \wedge Q$
VERDADERO	VERDADERO	FALSO	FALSO	VERDADERO	VERDADERO
VERDADERO	FALSO	FALSO	VERDADERO	VERDADERO	FALSO
FALSO	VERDADERO	VERDADERO	FALSO	VERDADERO	FALSO
FALSO	FALSO	VERDADERO	VERDADERO	FALSO	FALSO

Por último en la tabla 1.7 presentamos la jerarquía correspondiente de todos los operadores (aritméticos, relacionales y lógicos).

Tabla 1.7 Jerarquía de los Operadores	
Operadores	Jerarquía
()	(mayor)
**	
; /, div, mod	
=, <, >, <=, >=	
NO	
Y	
O	(menor)



Nota:

El operador () es un operador asociativo que tiene la prioridad más alta en cualquier lenguaje de programación. Por otra parte, debemos señalar que en ciertos lenguajes, las prioridades de los operadores se manejan de forma diferente. Por ejemplo, el operador lógico de negación NO en el lenguaje de programación C tiene la prioridad más alta después del operador asociativo.

Ejemplo 1.4

A continuación en este ejemplo presentamos varios casos y la forma de resolver los mismos.

Caso a)

$$\text{NO } (15 \geq \underline{\underline{7}} \cdot 2) \text{ O } (43 - 8 \cdot 2 \text{ div } 4 \neq 3 \cdot 2 \text{ div } 2)$$

1

$$\text{NO } (\underline{\underline{15}} \geq 49) \text{ O } (43 - \underline{\underline{8}} \cdot 2 \text{ div } 4 \neq 3 \cdot 2 \text{ div } 2)$$

2 3

$$\text{NO FALSO O } (43 - \underline{\underline{16}} \text{ div } 4 \neq 3 \cdot 2 \text{ div } 2)$$

4

$$\text{NO FALSO O } (43 - 4 \neq \underline{\underline{3}} \cdot 2 \text{ div } 2)$$

5

$$\text{NO FALSO O } (43 - 4 \neq \underline{\underline{6}} \text{ div } 2)$$

6

$$\text{NO FALSO O } (43 - 4 \neq \underline{\underline{3}})$$

7

$$\text{NO FALSO O } (\underline{\underline{39}} \neq 3)$$

8

$$\text{NO FALSO O VERDADERO}$$

9

$$\text{VERDADERO O VERDADERO}$$

10

VERDADERO

Caso b)

$$(15 \geq 7 \cdot \underline{\underline{3}} \cdot 2 \text{ Y } 8 > 3 \text{ Y } 15 > 6) \text{ O NO } (7 \cdot 3 < 5 + 12 \cdot 2 \text{ div } 3 \cdot 2)$$

1

1.5 Programas

$$(15 \geq \underline{\underline{7}} \cdot 9 \text{ Y } 8 > 3 \text{ Y } 15 > 6) \text{ O NO } (7 \cdot 3 < 5 + 12 \cdot 2 \text{ div } 3 \cdot 2)$$

2

$$(\underline{\underline{15}} \geq 63 \text{ Y } 8 > \underline{\underline{3}} \text{ Y } 15 > 6) \text{ O NO } (7 \cdot 3 < 5 + 12 \cdot 2 \text{ div } 3 \cdot 2)$$

3 4 5

$$(\text{FALSO Y VERDADERO Y VERDADERO}) \text{ O NO } (7 \cdot 3 < 5 + 12 \cdot 2 \text{ div } 3 \cdot 2)$$

6

$$(\text{FALSO Y VERDADERO}) \text{ O NO } (7 \cdot 3 < 5 + 12 \cdot 2 \text{ div } 3 \cdot 2)$$

7

$$\text{FALSO O NO } (7 \cdot 3 < 5 + \underline{\underline{12}} \cdot 2 \text{ div } \underline{\underline{3}} \cdot 2)$$

9 10 8

$$\text{FALSO O NO } (21 < 5 + \underline{\underline{24}} \text{ div } 9)$$

11

$$\text{FALSO O NO } (21 < 5 + \underline{\underline{2}})$$

12

$$\text{FALSO O NO } (21 < \underline{\underline{7}})$$

13

$$\text{FALSO O NO FALSO}$$

14

$$\text{FALSO O VERDADERO}$$

15

VERDADERO

Caso c)

$$\text{NO } ((\underline{\underline{7}} \cdot 3 \text{ div } 2 \cdot 4) > (15 / 2 \cdot 6 \geq 15 \cdot 2 / 17 = 15))$$

1

$$\text{NO } ((\underline{\underline{21}} \text{ div } 2 \cdot 4) > (15 / 2 \cdot 6 \geq 15 \cdot 2 / 17 = 15))$$

2

NO (10 * 4) > (15 / 2 * 6 >= 15 * 2 / 17 = 15))

3

NO (40 > (15 / 2 * 6 >= 15 * 2 / 17 = 15))

4

NO (40 > (7.5 * 6 >= 15 * 2 / 17 = 15))

5

6

NO (40 > (45 >= 30 / 17 = 15))

7

NO (40 > (45 >= 1.75 = 15))

8

NO (40 > (VERDADERO = 15))

9

Error

Nota:

No se puede realizar la comparación entre un valor lógico y un numérico, utilizando un operador relacional.

1.3.5 Bloque de asignación

Un bloque de asignación se utiliza para asignar valores o expresiones a una variable. La asignación es una operación destructiva. Esto significa que si la variable tenía asignado un valor, éste se destruye, conservando ahora el nuevo valor. El formato de la asignación es el siguiente:

Variable \leftarrow expresión o valor

Donde: expresión puede ser aritmética o lógica, o una constante o variable.

Observemos a continuación el siguiente ejemplo:

Ejemplo 1.5

Supongamos que las variables I, ACUM y J son de tipo entero, REA y SUM de tipo real, CAR de tipo carácter y BAND de tipo booleano. Consideraremos también que tenemos que realizar las siguientes asignaciones:

1. I \leftarrow 0
2. I \leftarrow I + 1
3. ACUM \leftarrow 0
4. J \leftarrow 5 ** 2 div 3
5. CAR \leftarrow 'a'
6. ACUM \leftarrow J div I
7. REA \leftarrow ACUM / 3
8. BAND \leftarrow (8 > 5) y (15 < 2 ** 3)
9. SUM \leftarrow ACUM * 5 / J ** 2
10. I \leftarrow I * 3
11. REA \leftarrow REA / 5
12. BAND \leftarrow BAND o (I = J)
13. I \leftarrow REA
14. CAR \leftarrow J

En la tabla 1.8 podemos observar los valores que van tomando las variables en memoria.

Tabla 1.8 Memoria							
Número de Asignación	I	J	ACUM	REA	SUM	CAR	BAND
1	X						
2	X						
3			X				
4		8					
5						X	
6			8				
7				X			
8				2.66			
9					0.625		X
10	X						
11				0.532			
12							X
13	Error						
14						Error	

**Nota:**

Observe el lector que en la asignación número 13 a la variable *I* se le asigna una variable de tipo real, por lo que se produce un error. Lo mismo ocurre en la asignación número 14, a la variable tipo carácter *CAR* se le asigna una variable de tipo entero, por lo que también se produce un error.

1.4 Construcción de diagramas de flujo

Hasta el momento el lector ha estudiado algunos conceptos que le permiten construir algunos diagramas de flujo. Reforzaremos estos conceptos con ejemplos seleccionados. Es nuestro interés que el lector comience a desarrollar habilidad y una capacidad de razonamiento estructurada y flexible que le permita, en la medida que practique, obtener la solución a los problemas planteados.

Ejemplo 1.6

Construya un diagrama de flujo tal que dado los datos A, B, C y D que representan números enteros, escriba los mismos en orden inverso.

Datos: A, B, C, D (variables de tipo entero).

Consideraciones:

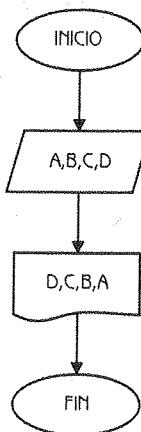
- Para el *inicio* y *fin* del diagrama de flujo se utiliza el símbolo:



- Para *lectura* se utiliza el símbolo:



- Para *escritura* se utiliza el símbolo:

**1.4 Construcción de diagramas de flujo**

{Se leen los datos}

{Se escriben los datos en orden inverso}

Diagrama de Flujo 1.1

Observe el lector que si se ingresan los datos: 7, 28, 150 y 35, la impresión produce lo siguiente: 35, 150, 28, 7.

Ejemplo 1.7

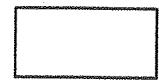
Construya un diagrama de flujo tal que dado los datos enteros A y B, escriba el resultado de la siguiente expresión:

$$\frac{(A + B)^2}{3}$$

Datos: A, B (variables de tipo entero).

Recuerde lo siguiente:

- Para realizar un *proceso* se utiliza el símbolo:



Variable \leftarrow expresión o valor

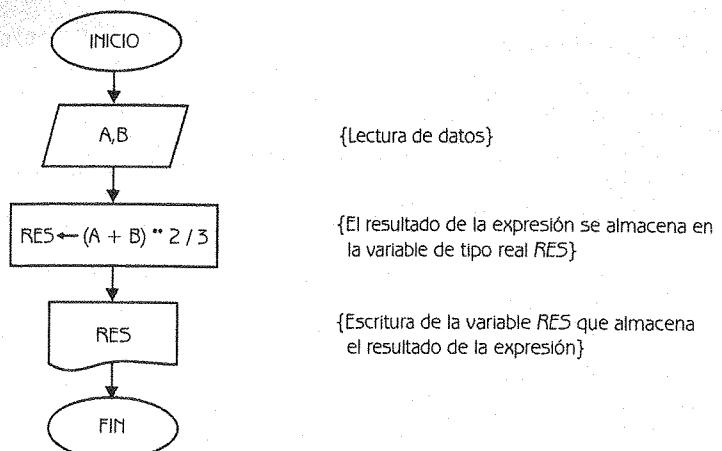


Diagrama de Flujo 1.2

Explicación de las variables

A, B: Variables de tipo entero.

RES: Variable de tipo real. Almacena el resultado de la expresión.

En la tabla 1.9 el lector podrá observar los datos que se ingresan y el resultado obtenido, para 5 corridas diferentes.

NUMERO DE CORRIDA	DATOS		RESULTADO
	A	B	RES
1	5	6	40.33
2	7	10	96.33
3	0	3	3.00
4	12	2	65.33
5	14	-5	27.00

: Expresa valores que se imprimen

Ejemplo 1.8

Dada la matrícula y 5 calificaciones de un alumno obtenidas a lo largo del semestre, construya un diagrama de flujo que imprima la matrícula del alumno y el promedio de sus calificaciones.

Datos: MAT, CAL1, CAL2, CAL3, CAL4, CAL5

Donde:

MAT es una variable de tipo entero que representa la matrícula del alumno.

CAL1, CAL2, CAL3, CAL4 y CAL5 son variables de tipo real que representan las 5 calificaciones del alumno.

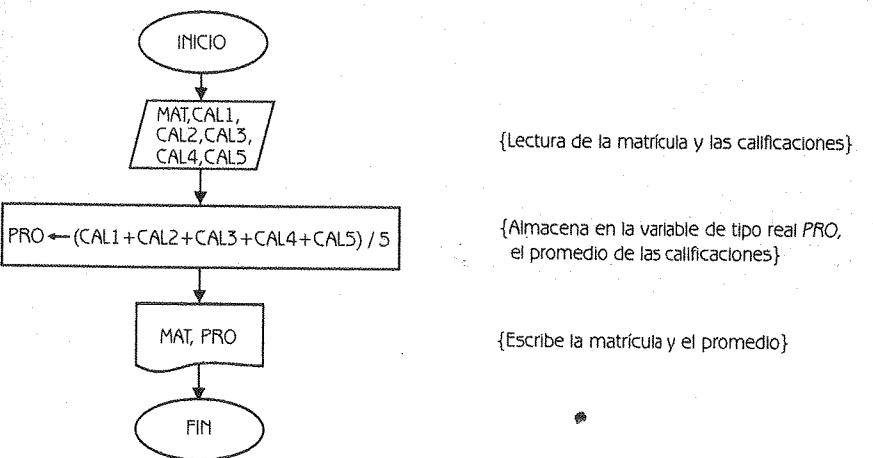


Diagrama de flujo 1.3

Explicación de las variables

MAT: Variable de tipo entero.

CAL1, CAL2, CAL3, CAL4, CAL5: Variables de tipo real.

PRO: Variable de tipo real. Almacena el promedio de las calificaciones del alumno.

En la tabla 1.10 el lector podrá observar los datos y resultados para 5 corridas diferentes.

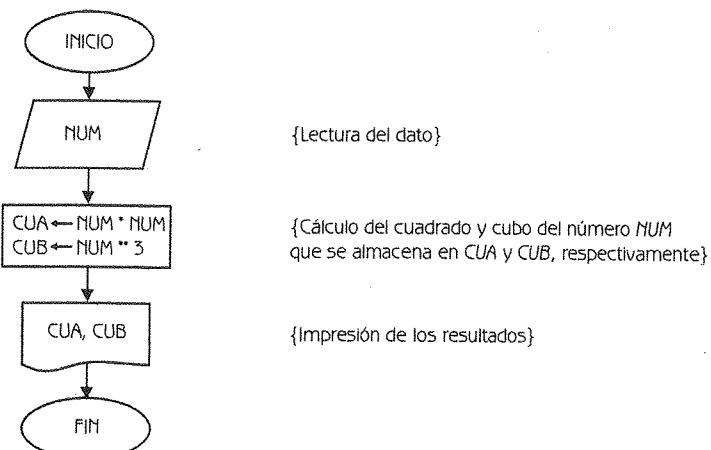
NUMERO DE CORRIDA	DATOS						RESULTADOS	
	MAT	CAL1	CAL2	CAL3	CAL4	CAL5	MAT	PRO
1	16500	8	8.5	9	7	6	16500	7.7
2	16650	9	8	9	7	9	16650	8.4
3	17225	9	10	10	8	9	17225	9.2
4	17240	8.5	9	7.5	6	6.5	17240	7.5
5	18240	7.3	6.8	9.5	8	8.5	18240	8.02

: Expresa valores que se imprimen

Ejemplo 1.9

Escriba un diagrama de flujo que permita calcular e imprimir el cuadrado y el cubo de un número entero positivo NUM.

Dato: NUM (variable de tipo entero).



1.4 Construcción de diagramas de flujo

Explicación de las variables

NUM: Variable de tipo entero.

CUA: Variable de tipo real. Almacena el cuadrado del número que se ingresa.

CUB: Variable de tipo real. Almacena el cubo del número que se ingresa.

En la tabla 1.11 podemos observar el seguimiento del algoritmo para diferentes corridas.

NUMERO DE CORRIDA	DATO	RESULTADOS		
		NUM	CUA	CUB
1	7	49	343	
2	15	225	3375	
3	8	64	512	
4	12	144	1728	
5	30	900	27000	

: Expresa valores que se imprimen

Ejemplo 1.10

Construya un diagrama de flujo tal que dado como datos la base y la altura de un rectángulo, calcule el perímetro y la superficie del mismo.

Datos: BASE, ALTU

Donde:

BASE es una variable de tipo real que representa la base de un rectángulo.

ALTU es una variable de tipo real que indica la altura del rectángulo.

Recuerde que:

- La superficie de un rectángulo se calcula aplicando la siguiente fórmula:

$$\text{Superficie} = \text{base} * \text{altura}$$

Fórmula 1.1

- El perímetro se calcula como:

$$\text{Perímetro} = 2 * (\text{base} + \text{altura})$$

Fórmula 1.2

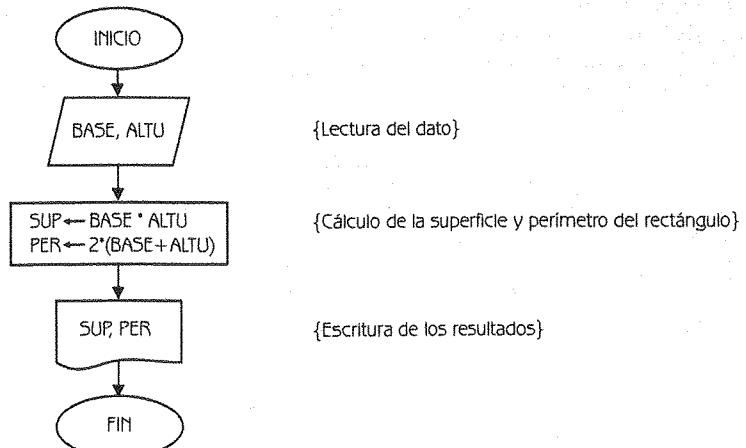


Diagrama de Flujo 1.5

Explicación de las variables

BASE, ALTU: Variables de tipo real.

SUP: Variable de tipo real. Almacena la superficie del rectángulo.

PER: Variable de tipo real. Almacena el perímetro del rectángulo.

NUMERO DE CORRIDA	DATOS		RESULTADOS	
	BASE	ALTU	SUP	PER
1	8.5	6.2	52.70	29.40
2	7.9	15.3	120.87	46.40
3	15.18	22.0	333.96	74.36
4	12.63	7.9	99.77	41.06
5	39.40	68.5	2698.90	215.80

: Expresa valores que se imprimen

1.5 Programas

Un *programa*, concepto desarrollado por Von Neumann en 1946, es un conjunto de instrucciones que sigue la computadora para alcanzar un resultado específico. El programa se escribe en un lenguaje de programación a partir de un diagrama de flujo diseñado con anterioridad.

Un *lenguaje de programación*, por otra parte, está constituido por un conjunto de reglas sintácticas (especifica la formación de instrucciones válidas) y semánticas (especifica el significado de estas instrucciones), que hacen posible escribir un programa.

Cientos de lenguajes de programación se han desarrollado hasta la fecha. Ya en 1969, se habían enumerado más de 120 que habían sido utilizados ampliamente. Actualmente a los lenguajes de programación podemos clasificarlos teniendo en cuenta el *tipo de problemas* que son capaces de resolver de manera natural. Así tenemos, por ejemplo, para programación estructurada: PASCAL, C, BASIC, FORTRAN, COBOL; estos tres últimos en su versión estructurada. Para programación orientada a objetos: C++, SMALLTALK y JAVA. Para programación simbólica: LISP, y para programación lógica: PROLOG. Cabe señalar que sólo hicimos mención de algunos lenguajes y tipos de lenguajes.

En este libro nos enfocaremos sobre el tipo de lenguajes de programación estructurada. En este enfoque los programas se diseñan de arriba hacia abajo (top-down) jerárquicamente, usando sólo un conjunto restringido de estructuras de control en cada nivel, instrucciones secuenciales, estructuras selectivas y estructuras repetitivas. Cuando esto se hace en forma adecuada el programa resulta muy fácil de entender, depurar y modificar.

Cuando tenemos que resolver un problema de tipo algorítmico, entendiendo por esto aquellos que tienen una solución determinística, primero desarrollamos el *algoritmo*, que proporciona una solución muy general. Posteriormente construimos el *diagrama de flujo*, que esquematiza gráfica y detalladamente la solución del problema, y a partir de éste, escribimos el *programa* en algún lenguaje de programación estructurado.

Pensamos que la tarea intelectual, la que requiere de un pensamiento profundo, de una capacidad de razonamiento flexible y crítica, es la de la construcción del diagrama de flujo, que representa la solución detallada del problema. La escritura del programa puede ser muy simple, conociendo las reglas sintácticas y semánticas que constituyen el lenguaje de programación.

El lenguaje que utilizaremos en este libro para la construcción de los programas es un *lenguaje algorítmico de pseudo-código*, independiente de cualquier lenguaje de programación. Esta característica consideramos que es muy importante, ya que permite al lector comprender las estructuras de datos y los algoritmos asociados a ellas sin relacionarlos a un lenguaje de programación en particular. Se considera que una vez que el lector domine correctamente estos conceptos, muy fácilmente podrá transportar los programas realizados a un lenguaje de programación, tal como PASCAL o C.

A continuación en la tabla 1.13 podemos observar los símbolos que utilizamos para la construcción del diagrama de flujo, la instrucción adecuada en el lenguaje algorítmico y un ejemplo de su uso. Los símbolos y las instrucciones que presentaremos son las que usaremos en una etapa inicial. Posteriormente, conforme avancemos en los temas de los siguientes capítulos, iremos presentando los símbolos restantes y las instrucciones correspondientes que permiten describirlos.

Tabla 1.13		
Representación del símbolo	Instrucción	Ejemplo
	Leer...	Leer A, B, C Leer MAT, CAL
	Hacer...	Hacer A ← A + SUE Hacer A ← A + SUE Hacer SUE ← SUE * 1.15 + 200 Hacer SUE ← SUE * 1.15 + 200
	Escribir...	Escribir A, B Escribir "Sueldo:", SUE

Ejemplo 1.11

Consideremos el diagrama de flujo 1.6, el mismo es idéntico al diagrama de flujo 1.1 construido para resolver el problema del ejemplo 1.6. A la derecha del diagrama podemos observar las instrucciones escritas en lenguaje algorítmico.

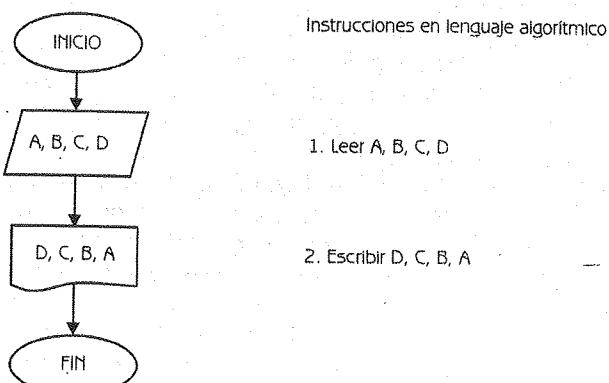


Diagrama de Flujo 1.6

1.5 Programas

Todo programa tiene un *nombre* que lo define. Utilizaremos las reglas presentadas para la construcción de un *identificador*, para construir el nombre del programa. Es muy importante que los *nombres de los programas* sean representativos de la función que cumplen los mismos. Posterior al nombre, debajo, es muy recomendable escribir un párrafo como comentario de lo que realiza el programa. También es necesario definir las constantes y variables que utilizaremos en el desarrollo de la solución.

A continuación presentamos el programa escrito en lenguaje algorítmico, del diagrama de flujo 1.6.

Programa 1.1

INVIERTE_DATOS

{El programa dado un conjunto de datos de entrada invierte el orden de los mismos cuando los imprime}

{A, B, C y D son variables de tipo entero}

1. Leer A, B, C, D
2. Escribir D, C, B, A

Observe el lector que escribir un programa es muy sencillo, conociendo las instrucciones correspondientes. La tarea intelectual, creativa, radica en la construcción del diagrama de flujo.

Ejemplo 1.12

Consideremos el diagrama de flujo 1.7, el mismo es idéntico al diagrama de flujo 1.2, construido para resolver el problema del ejemplo 1.7. A la derecha presentamos las instrucciones en lenguaje algorítmico.

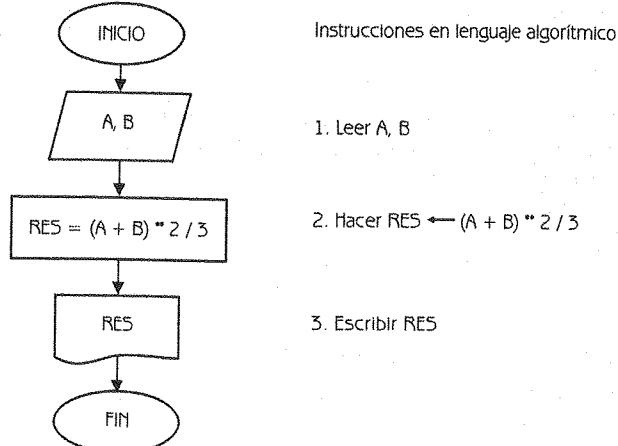


Diagrama de Flujo 1.7

A continuación presentamos el programa en lenguaje algorítmico.

Programa 1.2**CALCULA**

{El programa, dados como datos los enteros A y B, calcula el resultado de una expresión}

{A y B son variables de tipo entero. RES es una variable de tipo real}

1. Leer A, B
2. Hacer RES $\leftarrow (A + B) * 2 / 3$
3. Escribir RES

1.5 Programas**Ejemplo 1.13**

En este ejemplo presentamos la solución en lenguaje algorítmico del diagrama de flujo 1.3.

Programa 1.3**PROMEDIO_CALIFICACION**

{El algoritmo, dadas cinco calificaciones de un alumno, calcula su promedio}

{MAT es una variable de tipo entero. CAL1, CAL2, CAL3, CAL4, CAL5 y PRO son variables de tipo real}

1. Leer MAT, CAL1, CAL2, CAL3, CAL4, CAL5
2. Hacer PRO $\leftarrow (CAL1 + CAL2 + CAL3 + CAL4 + CAL5) / 5$
3. Escribir MAT, PRO

Ejemplo 1.14

A continuación presentamos la solución en lenguaje algorítmico del diagrama de flujo 1.4.

Programa 1.4**CUADRADO_CUBO**

{El programa, dado como dato un número entero positivo, calcula el cuadrado y el cubo de dicho número}

{NUM es una variable de tipo entero. CUA y CUB son variables de tipo real}

1. Leer NUM
2. Hacer CUA $\leftarrow NUM * NUM$ y CUB $\leftarrow NUM ^ 3$
3. Escribir CUA y CUB

Ejemplo 1.15

En este ejemplo presentamos la solución en lenguaje algorítmico del diagrama de flujo 1.5.

Programa 1.5**PERIMETRO_SUPERFICIE_RECTANGULO**

{El programa, dado como datos la base y la altura de un rectángulo, calcula su perímetro y superficie}

{BASE, ALTU, SUP y PER son variables de tipo real}

1. Leer BASE, ALTU
2. Hacer SUP \leftarrow BASE * ALTU y PER \leftarrow 2 * (BASE + ALTU)
3. Escribir SUP y PER

A continuación presentamos una serie de *problemas resueltos*, diseñados expresamente como elementos de ayuda para el análisis y la retención de los conceptos. Además se utilizan en muchos de ellos, tablas que permiten hacer el seguimiento de la solución planteada en estos algoritmos.

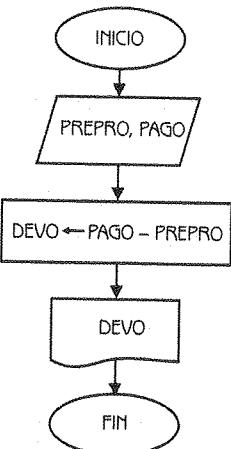
Problemas resueltos**Problema 1.1**

Construya un diagrama de flujo tal que dado el costo de un artículo vendido y la cantidad de dinero entregada por el cliente, calcule e imprima el cambio que se debe entregar al mismo.

Datos: PREPRO, PAGO

Donde:

- PREPRO es una variable de tipo real que representa el precio del producto.
 PAGO es una variable de tipo real que representa el pago que realiza el cliente.



Nota:
 Asumimos que el pago del cliente es mayor al precio del producto.

Diagrama de Flujo 1.8

Explicación de las variables

DEVO: Variable de tipo real. Almacena el cambio que se debe entregar al cliente.

A continuación en la siguiente tabla, presentamos el seguimiento del algoritmo para diferentes corridas.

NUMERO DE CORRIDA	DATOS		RESULTADO
	PREPRO	PAGO	
1	86.25	100	13.75
2	4.86	50	45.14
3	21.73	50	28.27
4	1.68	5	3.32
5	49.20	100	50.80

: Expresa valores que se imprimen

Programa 1.6

VUELTO_DE_UN_PAGO

{El programa, dado el costo de un producto y la cantidad de dinero entregada por el cliente, calcula el vuelto que hay que entregarle al mismo}

{PREPRO, PAGO y DEVO son variables de tipo real}

1. Leer PREPRO y PAGO
2. Hacer DEVO \leftarrow PAGO - PREPRO
3. Escribir DEVO

Problema 1.2

Construya un diagrama de flujo tal que dadas la base y la altura de un triángulo, calcule e imprima su superficie.

Datos: BASE, ALTU

Donde:

BASE es una variable de tipo real que indica la base del triángulo.

ALTU es una variable de tipo real que representa la altura del triángulo.

Consideraciones:

La superficie de un triángulo se calcula aplicando la siguiente fórmula:

$$\frac{\text{BASE} \cdot \text{ALTU}}{2}$$

Fórmula 1.3

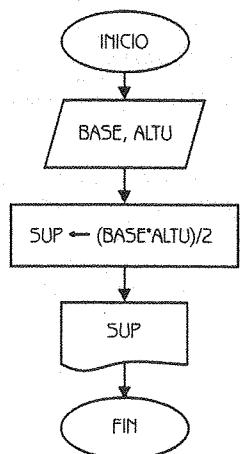


Diagrama de Flujo 1.9

Explicación de las variables

BASE y ALTU: Variables de tipo real.

SUP: Variable de tipo real. Almacena la superficie del triángulo.

A continuación en la siguiente tabla, presentamos el seguimiento del algoritmo para diferentes corridas.

NUMERO DE CORRIDA	DATOS		RESULTADO
	BASE	ALTU	
1	8.50	7.20	30.60
2	9.30	12.50	58.12
3	120.60	85.90	5179.77
4	4.33	1.98	4.28
5	11.60	7.40	42.92

: Expresa valores que se imprimen

A continuación presentamos el diagrama de flujo escrito en lenguaje algorítmico.

Programa 1.7**SUPERFICIE_TRIANGULO**

{El programa, dados como datos la base y la altura de un triángulo, calcula su superficie}

{BASE, ALTU y SUP son variables de tipo real}

1. Leer BASE y ALTU
2. Hacer $SUP \leftarrow (BASE * ALTU) / 2$
3. Escribir SUP

Problema 1.3

Escriba un diagrama de flujo tal que dado como datos el nombre de un dinosaurio, su peso y su longitud, expresados estos dos últimos en libras y pies respectivamente; escriba el nombre del dinosaurio, su peso expresado en kilogramos y su longitud expresada en metros.

Datos: NOM, PES, LON

Donde:

NOM es una variable de tipo cadena de caracteres que indica el nombre del dinosaurio.

PES es una variable de tipo real que representa el peso del dinosaurio en libras.

LON es una variable de tipo real que expresa la longitud del dinosaurio en pies.

Consideraciones:

- 1 tonelada equivale a 1000 kilogramos.
- 1 pie equivale a 0.3047 metros.

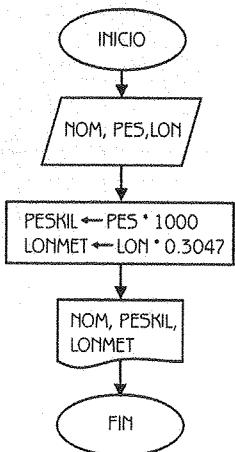
Problemas resueltos

Diagrama de Flujo 1.10

Explicación de las variables

NOM: Variable de tipo cadena de caracteres.

PES y LON: Variables de tipo real.

PESKIL: Variable de tipo real. Almacena el peso del dinosaurio en kilogramos.

LONMET: Variable de tipo real. Almacena la longitud del dinosaurio en metros.

A continuación, en la tabla 1.16 podemos observar el seguimiento del algoritmo para diferentes corridas.

NUMERO DE CORRIDA	DATOS			RESULTADOS	
	NOM	PES	LON	PESKIL	LONMET
1	PLATEOSAURUS	5	30	5000	9.14
2	DIPLOJOCUS	15	90	15000	27.42
3	BRACHIOSAURUS	50	80	50000	24.37
4	BRONTOSAURUS	25	70	25000	21.32
5	TYRANNOSAURUS	8	30	8000	9.14

: Expresa valores que se imprimen

Programa 1.8**DINOSAURIOS**

{El programa, dado el nombre de un dinosaurio, su peso expresado en toneladas y su longitud expresada en libras; escribe el nombre del dinosaurio, su peso y longitud expresadas en kilogramos y metros, respectivamente}

{NOM es una variable de tipo cadena de caracteres. PES, ALT, PESKIL y LONMET son variables de tipo real}

1. Leer NOM, PES y LON
2. Hacer PESKIL \leftarrow PES * 1000 y LONMET \leftarrow LON * 0.3047
3. Escribir NOM, PESKIL y LONMET

Problema 1.4

Construya un diagrama de flujo que resuelva el problema que tienen en una gasolinera. Los surtidores de la misma registran lo que "surten" en galones, pero el precio de la gasolina está fijado en litros. El diagrama de flujo debe calcular e imprimir lo que hay que cobrarle al cliente.

Dato: GAL (variable de tipo real que representa los galones de gasolina que le surtieron a un cliente).

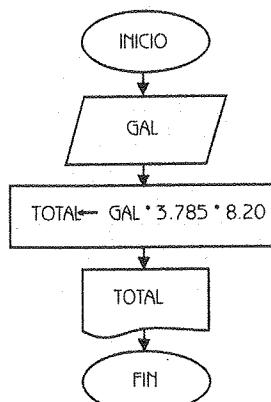


Diagrama de Flujo 1.11

Consideraciones:

- Cada galón tiene 3.785 litros.
- El precio del litro es \$8.20.

Explicación de las variables

GAL: Variable de tipo real.

TOTAL: Variable de tipo real. Almacena el total de lo que debe pagar el cliente.

A continuación, en la tabla 1.17 el lector podrá observar el seguimiento del algoritmo para diferentes corridas.

NUMERO DE CORRIDA	DATOS	RESULTADO
	GAL	TOTAL
1	10.38	322.16
2	15.90	493.49
3	8.40	260.71
4	9.68	299.81
5	19.90	617.64

: Expresa valores que se imprimen

A continuación presentamos el diagrama de flujo, en lenguaje algorítmico.

Programa 1.9**GASOLINERA**

{El programa, dado como dato los galones surtidos a un cliente en una gasolinera, calcula lo que el mismo debe pagar}

{GAL y TOTAL son variables de tipo real}

1. Leer GAL
2. Hacer TOTAL \leftarrow GAL * 3.785 * 8.20
3. Escribir TOTAL

Problema 1.5

Construya un diagrama de flujo tal que dado como datos el radio y la altura de un cilindro, calcule e imprima el área y su volumen.

Datos: RADIO, ALTU

Donde:

RADIO es una variable de tipo real que representa el radio de un cilindro.

ALTU es una variable de tipo real que representa la altura del cilindro.

Consideraciones:

- El volumen de un cilindro lo calculamos aplicando la siguiente fórmula:

$$\text{Volumen} = \pi * \text{radio}^2 * \text{altura}$$

Fórmula 1.4

Donde: $\pi = 3.141592$.

- La superficie del cilindro la calculamos como:

$$\text{Area} = 2 * \pi * \text{radio} * \text{altura}$$

Fórmula 1.5

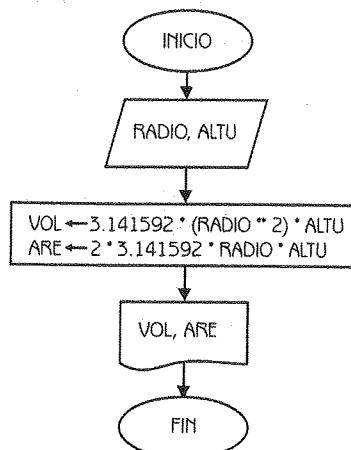


Diagrama de Flujo 1.12

Explicación de las variables

RADIO y ALTU: Variables de tipo real.

VOL: Variable de tipo real. Almacena el volumen del cilindro.

ARE: Variable de tipo real. Almacena el área del cilindro.

En la tabla 1.18, podemos observar el seguimiento del algoritmo para diferentes corridas.

Tabla 1.18

NUMERO DE CORRIDA	DATOS		RESULTADOS	
	RADIO	ALTU	VOL	ARE
1	45.22	11.60	74519.33	3295.86
2	17.30	8.45	7945.09	918.51
3	69.30	72.40	1092332.40	31524.75
4	125.30	117.40	5790552.70	92427.01
5	85.90	237.20	5498583.10	128022.89

: Expresa valores que se imprimen

Programa 1.10

VOLUMEN_AREA_CILINDRO

{El programa dado como datos el radio y la altura de un cilindro, calcula su área y su volumen.}

{RADIO, ALTU, VOL y ARE son variables de tipo real}

1. Leer RADIO y ALTU
2. Hacer VOL ← 3.141592 * (RADIO ^ 2) * ALTU y
ARE ← 2 * 3.141592 * RADIO * ALTU
3. Escribir VOL y ARE

Problema 1.6

Construya un diagrama de flujo que calcule e imprima el número de segundos que hay en un determinado número de días.

Datos: DIAS (variable de tipo entero que expresa el número de días).

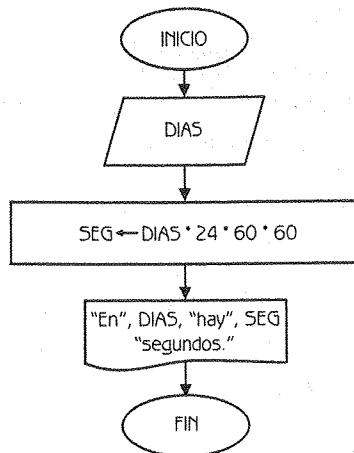


Diagrama de Flujo 1.13

Explicación de las variables

DIAS: Variable de tipo entero.

SEG: Variable de tipo real. Almacena la cantidad de segundos que hay en un número determinado de días.

Tabla 1.19		
NUMERO DE CORRIDA	DATO	RESULTADO
	DIAS	SEG
1	7	604800
2	15	1296000
3	116	10022400
4	28	2419200
5	3	259200

: Expresa valores que se imprimen

A continuación presentamos al diagrama de flujo 1.13, en lenguaje algorítmico.

Programa 1.11**SEGUNDOS_EN_DIAS**

{El programa, dado un número determinado de días, calcula cuántos segundos tienen éstos}

{DIAS es una variable de tipo entero. SEG es una variable de tipo real}

1. Leer DIAS
2. Hacer SEG ← DIAS * 24 * 60 * 60
3. Escribir "En ", DIAS, "hay ", SEG, "segundos."

Problema 1.7

Construya un diagrama de flujo tal que dados los tres lados de un triángulo, pueda determinar su área. Esta la calculamos aplicando la siguiente fórmula:

$$\text{Área} = \sqrt{s \cdot (s - l_1) \cdot (s - l_2) \cdot (s - l_3)}$$

$$s = (l_1 + l_2 + l_3) / 2$$

Fórmula 1.6

Datos: L1, L2, L3 (variables de tipo real que representan los tres lados).

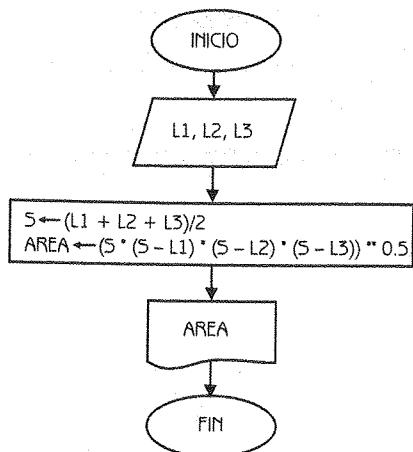


Diagrama de Flujo 1.14

Explicación de las variables

L1, L2, L3: Variables de tipo real.

S: Variable de tipo real. Se utiliza como una variable auxiliar para el cálculo del área.

AREA: Variable de tipo real. Almacena el área del triángulo.

En la tabla 1.20 podemos observar el seguimiento del algoritmo para diferentes corridas.

NUMERO DE CORRIDA	DATOS			CALCULO AUXILIAR	RESULTADO
	L1	L2	L3	S	AREA
1	7.5	7.5	7.5	11.25	24.3569
2	6.1	4.8	3.4	7.15	8.1358
3	10.0	10.0	4.5	12.25	21.9250
4	2.0	16.0	15.8	16.90	15.7889
5	17.6	-17.6	25.0	30.10	154.8739

: Expresa valores que se imprimen.

Programa 1.12**AREA_TRIANGULO**

{El programa, dados los tres lados de un triángulo, calcula su área}

{L1, L2, L3, S y AREA son variables de tipo real.}

1. Leer L1, L2, L3
2. Hacer $S \leftarrow (L1 + L2 + L3) / 2$ y
 $AREA \leftarrow (S * (S - L1) * (S - L2) * (S - L3)) ** 0.5$
3. Escribir AREA

Problema 1.8

Construya un diagrama de flujo que calcule la distancia entre dos puntos, dado como datos las coordenadas de los puntos P1 y P2.

Datos: X1, Y1, X2, Y2

Donde:

X1 y Y1 son variables de tipo real que representan las coordenadas del punto P1 en el eje de las X y Y, respectivamente.

X2 y Y2 son variables de tipo real que indican las coordenadas del punto P2 en el eje de las X y Y.

Consideraciones:

- Para calcular la distancia “D” entre dos puntos dados P1 y P2 aplicamos la siguiente fórmula:

$$D = \sqrt{(X_1 - X_2)^2 + (Y_1 - Y_2)^2}$$

Fórmula 1.7

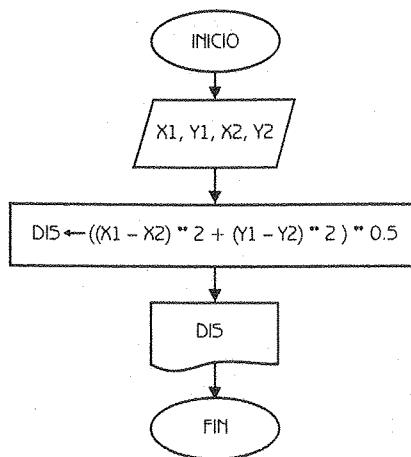


Diagrama de Flujo 1.15

Explicación de las variables

X1, Y1, X2, Y2: Variables de tipo real.

DIS: Variable de tipo real. Almacena la distancia entre dos puntos P1 y P2.

En la tabla 1.21 mostramos el seguimiento del algoritmo.

Tabla 1.21

NUMERO DE CORRIDA	DATOS				RESULTADO
	X1	Y1	X2	Y2	
1	3.17	4.78	4.99	7.88	3.59
2	7.15	21.60	1.93	4.38	17.99
3	12.17	10.40	10.40	29.30	18.98
4	39.40	78.90	68.30	187.20	112.08
5	88.70	118.30	295.30	18.40	229.48

: Expresa valores que se imprimen

Programa 1.13**DISTANCIA_ENTRE_DOS_PUNTOS**

{El programa, dadas las coordenadas de dos puntos P1 y P2, calcula la distancia entre estos puntos}

{X1, Y1, X2, Y2 y DIS son variables de tipo real}

1. Leer X1, Y1, X2, Y2
2. Hacer $DIS \leftarrow ((X1 - X2)^2 + (Y1 - Y2)^2)^{0.5}$
3. Escribir DIS

Estructuras algorítmicas selectivas

Metodología de la programación

Osvaldo Cairo

2.1 Introducción

Las estructuras lógicas selectivas se encuentran en la solución algorítmica de casi todo tipo de problemas. Las utilizamos cuando en el desarrollo de la solución de un problema debemos *tomar una decisión*, para establecer un proceso o señalar un camino alternativo a seguir.

Esta toma de decisión (expresada en el diagrama de flujo con un rombo) se basa en la evaluación de una o más condiciones que nos señalarán como alternativa o consecuencia, la *rama* a seguir.

Hay situaciones en las que la toma de decisiones se realiza en cascada. Es decir se toma una decisión, se marca la rama correspondiente a seguir, se vuelve a tomar otra decisión y así sucesivamente. Por lo que para alcanzar la solución de un problema o subproblema debemos aplicar prácticamente un árbol de decisión.

Las estructuras algorítmicas selectivas que se utilizan para la toma de decisiones lógicas las podemos clasificar de la siguiente forma:

1. SI ENTONCES (Estructura selectiva simple)
2. SI ENTONCES / SINO (Estructura selectiva doble)
3. SI MULTIPLE (Estructura selectiva múltiple)

Cabe señalar que cuando a las estructuras selectivas las aplicamos en cascada, podemos utilizar una combinación de las estructuras señaladas anteriormente en la clasificación.