1. Conceptos básicos

1.1 Introducción

El desarrollo del texto –que arranca de cero, suponiendo que el aprendiz no sabe programar– está basado en el paradigma de programación estructurada: las instrucciones se ejecutan en forma secuencial, una tras otra, y se utilizan estructuras de control condicionales no repetitivas y repetitivas (ciclos), así como rutinas. Se hace énfasis en el uso disciplinado de un lenguaje algorítmico y en la realización de pruebas de escritorio, ilustradas a través de entornos de memoria y de dispositivos estándares de entrada y salida, como el teclado y la pantalla.

Antes de empezar a programar computadores, es necesario adquirir algunos conceptos fundamentales para dicha tarea. Con el fin de no enfrentar al mismo tiempo el aprendizaje de tantos temas nuevos, primero se hace el trabajo en un lenguaje algorítmico que se va definiendo poco a poco, de tal manera que el conocimiento se adquiera en forma gradual y firme. En este libro, a excepción de los anexos, sólo se trabaja con lenguaje algorítmico, es decir, únicamente se emplean papel y lápiz; el aprendiz está concentrado en un solo asunto. Sin embargo, continuamente se hace referencia al computador (o máquina) porque el fin es prepararse para programarlo. El lenguaje algorítmico no depende de ninguna máquina en especial.

1.2 Concepto de algoritmo

Aunque la palabra *algoritmo* no es de las primeras que aprendemos, ejecutamos algoritmos durante toda la vida.

Un algoritmo es una secuencia de pasos o instrucciones realizadas en cierto orden, con un fin determinado.

Ejemplo 1.1 Retirar dinero de un cajero automático

Retirar dinero de un cajero automático es una tarea cotidiana de los seres humanos. Para llevarla a cabo, *grosso modo* se realizan las siguientes acciones:

– Insertar la tarjeta
– Seleccionar el idioma
– Ingresar la clave
– Seleccionar la operación por realizar
– Ingresar los datos de acuerdo con la operación seleccionada
– Obtener el resultado de la operación

Este conjunto de acciones hechas con un fin determinado, en dicho orden, es lo que se llama algoritmo.

Un algoritmo es una secuencia de pasos o instrucciones, realizados en un orden determinado, con un fin claramente definido.

Un lenguaje algorítmico permite plasmar de manera clara, por medio de sus herramientas, los pasos o instrucciones que componen un algoritmo. En general, un lenguaje algorítmico es bastante laxo con respecto a un lenguaje de computador porque lo que se quiere con él es expresar las ideas de un modo parecido a como lo hace un lenguaje que entiende el computador, pero no tan formal. Sin embargo, en este libro el lenguaje algorítmico que se utiliza, aunque sencillo, es bastante formal, con el propósito de que constituya una verdadera preparación para iniciar el aprendizaje de un lenguaje de computador. Las personas con algún grado de ilustración en programación de computadores notarán que el lenguaje algorítmico usado en este libro contempla muchos temas que por lo regular se cubren sólo cuando se está trabajando con un lenguaje de computador.

No existe un solo lenguaje algorítmico. Eso sería como pensar que todos los seres humanos se expresan igual. Lo que sí es importante en este primer curso de programación de computadores es que el estudiante sea disciplinado en el uso del lenguaje que está trabajando en su clase. Los frutos de esa disciplina se verán cuando haga sus prácticas con un lenguaje de computador. La utilización del lenguaje algorítmico será un paso intermedio en la verdadera comunicación del estudiante con el computador.

Si esta parte se cubre satisfactoriamente, el trabajo de programar un computador se va a facilitar de manera significativa.

Los seres humanos ejecutan algoritmos permanentemente. Lo que pasa es que no son conscientes de ello. Por ejemplo, hacer una carrera profesional es un algoritmo, ¿verdad? Cada acción que se ejecuta para alcanzar la meta de

obtener un título profesional se llamará instrucción en el lenguaje algorítmico y tendrá una sintaxis determinada, esto es, habrá que expresarla de un modo claro y preciso, de acuerdo con el lenguaje algorítmico.

1.3 Instrucciones básicas

Para empezar, todo algoritmo tendrá la siguiente estructura:

```
"nombre_del_algoritmo"
empiece
    Cuerpo
    del
    algoritmo
termine
```

donde empiece y termine son delimitadores del cuerpo del algoritmo. Ellos indican dónde empieza y dónde termina, pero no implican la ejecución de acción alguna. Sin embargo, su uso es obligatorio.

Los delimitadores empiece y termine son las dos primeras palabras reservadas de este lenguaje. Una **palabra reservada** es aquella que tiene un significado especial dentro del lenguaje y, por tanto, su uso es restringido.

El cuerpo del algoritmo, como se verá más adelante, está conformado por instrucciones ejecutables y no ejecutables, encargadas en general de la reserva de memoria, obtención o entrada de los datos, procesamiento de esos datos y presentación de resultados.

Cuerpo de un algoritmo

La entrada, el procesamiento y la presentación de resultados no se llevan a cabo en ese orden, necesariamente. Puede darse, por ejemplo, entrada, procesamiento, entrada, procesamiento, y luego sí la presentación de resultados.

Es recomendable que los algoritmos se escriban totalmente en letra minúscula, no sólo por presentación sino porque algunos lenguajes de computador así lo exigen.

El nombre del algoritmo puede estar compuesto por caracteres alfanuméricos y por el carácter de subrayado (_), y debe comenzar con una letra. Los caracteres alfabéticos con tilde o diéresis no son válidos.

Instrucción pare

Indica la terminación normal de un algoritmo y libera a la máquina de la tarea que está ejecutando. Todo algoritmo debe tener por lo menos una instrucción pare.

La palabra pare es reservada.

Palabras reservadas
empiece
termine
pare

La tabla de palabras reservadas se irá llenando a medida que éstas vayan apareciendo en el lenguaje algorítmico.

1.4 Variables

Las variables son lugares de memoria en los que se pueden almacenar datos. Se llaman así precisamente porque su contenido puede variar. Toda variable tiene tipo, nombre y contenido. El tipo puede ser entero, real, carácter o cadena de caracteres, dependiendo de lo que se quiera almacenar en ellas. Su nombre debe contener sólo caracteres alfanuméricos y el carácter de subrayado (_), y tiene que empezar con carácter alfabético; los caracteres alfabéticos con tilde o diéresis no son válidos. Su contenido está ligado al tipo.

Variables de tipo caracter o cadena de caracteres: Si la variable es de tipo caracter, su contenido es un carácter, y si es de tipo cadena de caracteres, su contenido es uno o más caracteres.

Variables numéricas: Son de tipo entero o real y pueden ser operandos en expresiones aritméticas o matemáticas.

Tipo: De qué naturaleza es el dato que se puede almacenar.
Nombre: Cómo hacer referencia a un espacio de memoria específico.
Contenido: Qué dato hay almacenado.

Los nombres de las variables deben ser significativos, de modo que den una idea de su contenido. Por ejemplo, si se quiere guardar un promedio, la variable debería llamarse promedio o prom, en lugar de m o x. Esto es una ayuda para quien construye o modifica el algoritmo (un ser humano), no para la máquina porque esto no la afecta.

1.5 Declaración de variables

Para poder utilizar una variable ésta debe existir, es decir, debe haber un espacio en la memoria asignado a ella. Esto se logra en la declaración de la variable, la cual debe ir al principio del algoritmo, antes de cualquier instrucción ejecutable. Su sintaxis es:

```
tipo lista_de_variables
```

donde el tipo de variable puede ser entero, real, caracter o cadena_de. Las variables de la lista se separan con el carácter coma (,).

Estas instrucciones son llamadas declarativas y son no ejecutables, porque no afectan el contenido de las variables ni implican la realización de otra acción diferente de la de reserva de memoria. Además, sólo tienen efecto una vez, cuando se inicia la ejecución del algoritmo. Esto significa que no puede haber instrucciones declarativas en otro punto que no sea exactamente el principio del algoritmo. Una instrucción ejecutable es la que realiza una acción, como hacer una asignación, leer, escribir o invocar una rutina, y su efecto se refleja en el contenido de las variables o en la pantalla.

Una palabra reservada no se puede usar como nombre de variable.

Ejemplo 1.2 Declaración de variables

```
"declaracion_de_variables"
empiece
entero a, b, c
```

```
real m
caracter x, p
cadena_de 8 t
cadena_de 5 k, h
pare
termine
```

Este algoritmo no hace nada, aparte de asignar espacio a las variables. La siguiente tabla contiene el significado de cada una de esas instrucciones declarativas.

Declaración	Significado
entero a, b, c	Se solicita espacio en memoria para guardar tres números enteros en las variables llamadas a, b y c, respectivamente.
real m	Se solicita espacio en memoria para guardar un número real en la variable llamada m.
caracter x, p	Se solicita espacio en memoria para guardar dos caracteres en las variables llamadas x y p, respectivamente. Un carácter puede ser numérico, alfabético, gráfico, de puntuación, etc.
cadena_de 8 t	Se solicita espacio en memoria para guardar una cadena de caracteres, esto es, una palabra o una frase. El valor constante 8 determina el número máximo de caracteres que puede almacenarse en esa variable. Así que la variable podría contener la palabra *servir* pero no la palabra *justiciero*.

Al ejecutar este algoritmo, se genera lo que desde ahora se llamará el **entorno de memoria**, que puede visualizarse como una tabla que contiene toda la información de las variables que se declaran en un algoritmo. Esa información consiste en el nombre de la variable, el tipo y el contenido. El entorno existe mientras el algoritmo se esté ejecutando. Una vez que termina, se libera la memoria reservada.

Para el algoritmo llamado declaracion_de_variables se generaría el siguiente entorno de memoria:

Variable	Tipo	Contenido
a	entero	?
b	entero	?
c	entero	?
m	real	?
x	caracter	?

Variable	Tipo	Contenido
p	caracter	?
t	cadena_de 8	?
k	cadena_de 5	?
h	cadena_de 5	?

El carácter ? indica que a la variable no se le ha definido valor. De aquí en adelante, se dirá que el contenido de la variable es desconocido.

En un mismo algoritmo, una variable no puede ser de más de un tipo.

Palabras reservadas	
empiece	entero
termine	real
pare	caracter
cadena_de	

1.6 Instrucción de asignación

La instrucción de asignación permite cambiar el contenido de una variable y se hace a través del operador de asignación ←. En la parte izquierda debe aparecer siempre la variable cuyo contenido se desea modificar y en la parte derecha el nuevo valor, una expresión que puede ser tan sencilla como una constante o una variable simple.

1.6.1 Expresión aritmética

Una expresión aritmética es aquella en la que se hacen operaciones con números. Una expresión se compone de operadores (suma (+), resta (–), multiplicación (*) y división (/)), y operandos (constantes o variables).

1.6.2 Orden de evaluación de una expresión aritmética

Para evaluar una expresión aritmética hay que saber en qué orden se ejecutan las operaciones. Los operadores tienen una jerarquía y pueden estar más de

una vez en una expresión. Los paréntesis ( ) son los operadores de agrupación y tienen la más alta jerarquía o mayor prioridad. En una expresión, el orden de evaluación es:

1. Lo que está entre paréntesis.
2. Menos unario: indica que se invierte el signo de la variable, constante o expresión a la que precede. Ejemplo: –a, –11, –(b*c).
3. Multiplicación y división.
4. Suma y resta.
5. La asignación. Debido a que tiene la más baja prioridad, primero se evalúa toda la expresión del lado derecho y luego se asigna el valor a la variable que está en la izquierda de la expresión.

– Si dos operadores tienen la misma prioridad, entonces primero se realiza el operador que está a la izquierda y luego el que está a la derecha.
– Si dos operadores son de diferente jerarquía, se ejecuta el de mayor jerarquía y se deja pendiente el operador de menor jerarquía, hasta que se encuentre otro operador de la misma jerarquía o ya no haya operaciones pendientes por realizar.

Algunos casos:

– resultado ← c + 2 * d. Primero se hace la multiplicación y luego la suma.
– resultado ← 10 + 4 *3 / 2 – 5. Primero se hace el producto (12), luego la división (6), después la suma (16) y, finalmente, la resta (11).
– resultado ← (12 + 30) / (20 – 6) * 2. Primero se hacen las operaciones que están entre paréntesis (42 y 14, respectivamente). Luego se hace la división (42 / 14) y luego el producto (3 * 2). El resultado es 6.

Constantes: Son valores que, como su mismo nombre lo indica, no varían. Las constantes pueden ser:

– **Numéricas**: Corresponden a los elementos que conforman los conjuntos de números, es decir, complejos, reales (racionales e irracionales) o enteros (incluye los naturales). Se pueden hacer operaciones.
– **Caracteres**: Los del alfabeto, los signos de puntuación, los signos de operación, los dígitos (0, 1,…, 9). En el último caso, aunque son numéricos, no se pueden hacer operaciones con esas constantes. Para utilizarlas en una condición o asignarlas, se escriben entre comillas sencillas. Por ejemplo: '3', 'c', ':', '<'.

– **Cadenas de caracteres**: Se escriben entre comillas dobles, como por ejemplo "Colombia", "promedio", "c".

En algunos lenguajes se declaran las constantes; en este lenguaje algorítmico, no.

Ejemplo 1.3 Variables caracter o cadena de caracteres

Supóngase ahora que el tipo de la variable no es numérico, es decir, la variable es de tipo caracter o de tipo cadena de caracteres. En estos casos hay que tener en cuenta lo siguiente:

Instrucción de asignación	Tipo de la variable	Observaciones sobre la instrucción
var ← 'x'	caracter	Es válida e implica el cambio del contenido de la variable var por el carácter x.
var ← '8'	caracter	Es válida e implica el cambio del contenido de la variable var por el carácter 8.
var ← x	caracter	Es válida, siempre y cuando la variable x exista y sea de tipo carácter. Implica el cambio del contenido de la variable var por el contenido de la variable x.
var ← "Colombia hermosa"	cadena_de 20	Es válida e implica el cambio del contenido de la variable var por la cadena de caracteres Colombia hermosa.
var ← "3"	cadena_de 10	Es válida e implica el cambio del contenido de la variable var por la cadena de caracteres 3.
var ← b	cadena_de 12	Es válida, siempre y cuando la variable b exista y sea de tipo cadena de caracteres con longitud $\leq$ 12. Implica el cambio del contenido de la variable var por el contenido de la variable b.

Aunque la variable de tipo caracter o cadena de caracteres contenga un carácter numérico, sólo con las variables numéricas (enteras o reales) se pueden realizar operaciones aritméticas o matemáticas.

Ejemplo 1.4 Variables numéricas

Supóngase que se tiene una variable llamada a y que se quiere que su contenido sea 5. Para hacerlo, se daría la instrucción de asignación

```
a ← 5
```

El contenido anterior de la variable a (definido o no) se cambia por el valor 5. Obsérvese que la dirección de la flecha que representa el operador de asignación indica el sentido en el que se hace la asignación. ¿Tendría sentido la operación 5 ← a? ¿Por qué?

Instrucciones como

```
a → 5
5 → a
```

no son válidas porque el operador → no existe.

Ejemplo 1.5 Copiar el contenido de una variable en otra

Si se desea que el contenido de la variable numérica a sea el mismo de la variable numérica d, se daría la instrucción de asignación

```
a ← d
```

que significa hacer una copia de la variable d en la variable a. El contenido de la variable a (definido o no) se cambia por el contenido de la variable d (definido o no). Supóngase que se tiene el siguiente entorno de memoria en la ejecución de cierto algoritmo:

Variable	Tipo	Contenido
a	entero	?
d	entero	?

```
...
d ← 8
a ← d
...
```

Los puntos suspensivos (…) indican que hay otras instrucciones.

La ejecución de las instrucciones tendría el siguiente efecto en el entorno de memoria:

Variable	Tipo	Contenido
a	entero	~~?~~ 8
d	entero	~~?~~ 8

Obsérvese que la instrucción de asignación a ← d (que copia el contenido de la variable d en la variable a) no tiene ningún efecto sobre la variable d. El caracter ~~?~~ indica que la variable deja de tener ese valor, en este caso desconocido.

Si se tuviera este entorno

Variable	Tipo	Contenido
a	entero	?
d	entero	?

El efecto de la ejecución de las instrucciones

```
...
a ← 8
d ← d + a
...
```

sería el cambio del contenido de la variable a (desconocido) por 8. Tomar una copia del contenido de las variables d (desconocido) y a (8), sumarlas y asignar el resultado a la variable d. El entorno quedaría así:

Variable	Tipo	Contenido
a	entero	~~?~~ 8
d	entero	~~?~~ ~~?~~

Porque un contenido "desconocido", operado con cualquier otra cosa, da desconocido. En algunos lenguajes, esto puede producir un error.

Cuando se tienen instrucciones como

```
d ← d + a
```

se debe verificar que la variable d se haya inicializado. Inicializar una variable significa definir su contenido, lo cual hasta este momento sólo se puede hacer con una instrucción de asignación.

También es posible que el valor que se quiera asignar a una variable sea el resultado de evaluar una expresión aritmética o matemática. Por ejemplo, se quiere que el nuevo valor de la variable a sea el resultado de sumar los números 7 y 20.

La suma de 7 y 20 se expresa como 7 + 20. En este caso, se requiere poder hacer uso del operador de suma.

1.7 Operadores aritméticos

El conjunto de operadores disponible en el lenguaje algorítmico que se está definiendo es el siguiente:

Suma	+	Resta	–
Multiplicación	*	División	/

La parte izquierda de una instrucción de asignación *siempre* es una variable y la *parte derecha* puede ser una constante, una variable o una expresión, donde las dos primeras son las expresiones más simples que existen.

Ejemplo 1.6 Sumar los números 7 y 20

Por el enunciado del problema, el resultado se debería guardar en una variable de tipo entero porque la suma de dos enteros da un entero, pero también se podría declarar real (¿por qué?).

```
"suma_de_7_y_20"
empiece
entero suma
suma ← 7 + 20
pare
termine
```

Al ejecutar este algoritmo, se genera el siguiente entorno de memoria:

Variable	Tipo	Contenido
suma	entero	?

Luego, al ejecutar la instrucción de asignación, ocurre lo siguiente:

Se suman los valores 7 y 20 y luego se busca en el entorno de memoria una variable llamada suma; si existe, se modifica su contenido con el valor 27 y el entorno quedaría así:

Variable	Tipo	Contenido
suma	entero	? 27

Después se ejecuta la instrucción pare, que indica la terminación normal del algoritmo y que conlleva la liberación de la memoria reservada en la declaración de variables. Que la memoria reservada se libere significa que ya no se tendrá acceso a esa área de memoria, a pesar de que los datos pueden continuar existiendo ahí porque el sistema operativo no ha asignado de nuevo ese espacio de memoria.

Si la variable suma no existe, se hace imposible continuar con la ejecución del algoritmo y ésta se termina de manera anormal; esto se conoce comúnmente como "aborto". Un algoritmo diseñado en forma correcta no debe abortar.

Aunque la expresión 20 + 7 es válida, no tiene sentido poner a la máquina a computar algo que ya uno sabe cuánto da. Es decir, ¿cuál es la razón para no asignar de una vez el valor 27?

Obsérvese que el algoritmo suma_de_7_y_20 sólo sirve para sumar los números 7 y 20. ¿Cuántos algoritmos parecidos pero diferentes habría que hacer para disponer siempre de uno que sume dos números enteros cualesquiera? ¿Sería posible? ¿Tendría sentido?

Otra versión del algoritmo anterior sería la siguiente:

```
"suma_de_7_y_20"
empiece
entero suma, num1, num2
num1 ← 7
num2 ← 20
suma ← num1 + num2
pare
termine
```

En num1 y num2 se almacenan los sumandos 7 y 20, respectivamente, y luego se almacena el resultado en la variable suma.

El efecto es el mismo. Al ejecutarlo, se genera el siguiente entorno de memoria:

Variable	Tipo	Contenido
suma	entero	?
num1	entero	?
num2	entero	?

Al ejecutar la primera instrucción de asignación, num1 ← 7, se busca en el entorno una variable llamada num1, y si existe, su contenido se modifica con el valor constante 7.

Variable	Tipo	Contenido
suma	entero	?
num1	entero	~~?~~ 7
num2	entero	?

Después, al ejecutar la segunda instrucción de asignación, num2 ← 20, se busca en el entorno una variable llamada num2, y si existe, su contenido se modifica con el valor constante 20.

Variable	Tipo	Contenido
suma	entero	?
num1	entero	~~?~~ 7
num2	entero	~~?~~ 20

Posteriormente, se ejecuta la tercera instrucción de asignación, suma ← num1 + num2, que consiste en buscar las variables num1 y num2 en el entorno de memoria; si existen, se toma una copia de su contenido y se suman, y luego se busca en el entorno de memoria una variable llamada suma; si existe, se modifica su contenido con el valor 27 y el entorno quedaría finalmente así:

Variable	Tipo	Contenido
suma	entero	~~?~~ 27
num1	entero	~~?~~ 7
num2	entero	~~?~~ 20

Luego se ejecuta la instrucción pare, que indica la terminación normal del algoritmo y que conlleva la liberación de la memoria reservada en la declaración de variables.

Cualquier búsqueda de una variable en el entorno de memoria que resulte infructuosa provocará la terminación anormal de la ejecución del algoritmo. Lo mismo ocurre cuando se hace referencia a una instrucción o a un operador que no existe en el lenguaje.

1.8 Operaciones entre enteros y reales

Cuando se realizan operaciones numéricas hay que tener en cuenta la operación, así como el tipo de los operandos (entero o real). Por ejemplo, cualquier persona que sepa dividir diría que 1/2 es igual a 0.5, lo cual puede no ser cierto en la máquina. Al hacer esta operación en la máquina, el resultado depende del tipo de los operandos y de la variable en la que se guarda.

Cuando se hace una operación básica en la máquina y por lo menos uno de los operandos es de tipo real, el resultado es de tipo real. Esto es, si los operandos involucrados en una operación son todos enteros, el resultado será un entero.

El operando no necesariamente es una variable, podría ser una constante. En este último caso, la regla se aplica de la misma forma.

Supóngase que se tienen las siguientes instrucciones de asignación:

	Operandos		Resultado	
Instrucción	**n1**	**n2**	**res**	**Observaciones**
res ← n1 + n2	entero	entero	entero	En el caso de la división, cuando los dos operandos son enteros, lo que ocurre es que se trunca la parte decimal del resultado. Así, 1/2 es igual a 0 y no a 0.5.
res ← n1 – n2	entero	real	real	
res ← n1 * n2	real	entero	real	
res ← n1 / n2	real	real	real	

Una constante numérica es entera (no tiene parte decimal) si pertenece al conjunto de los números enteros. En caso contrario es real, así su parte decimal sea cero. En este lenguaje algorítmico, la parte decimal se indica con punto (.) y no con coma (,).

Supóngase que las variables op1 y op2 son enteras y que op3 es real.

	Operandos			Resultado		
Operación	**op1**	**op2**	**op3**	**res** de tipo real	**res** de tipo entero	**Observaciones**
res ← op1 + op2 – op3	4	7	13.8	–2.8	–2	Al menos uno de los operandos es real, por lo que el resultado es real.
res ← op1/op2 + 101	6	4		102.0	102	Como op1 y op2 son enteras, op1/op2 da 1 y no 1.5.
res ← op3 * 2 – op1/2.0	5	5	2	1.5	1	Aunque op1 es entero, op1/2.0 da 2.5 porque 2.0 es una constante real.
res ← op1 + op2/3	17	14		21.0	21	Como op2 es una variable entera y 3 es una constante entera, op2/3 da un entero. Es decir, 14/3 es igual a 4, no a 4.67.

1.9 Comentarios

Los comentarios son anotaciones que se incluyen en el algoritmo con el fin de explicar, hacer aclaraciones o incluir datos que ayuden a su documentación (como autor, fecha y lenguaje). Pueden ir en cualquier parte del algoritmo, se ponen precedidos de dos asteriscos y son considerados como tales hasta el final de la línea donde se inician. Si la línea no alcanza para escribir todo lo que se quiere, se pasa a la siguiente y se procede igual.

Los comentarios se utilizan también para incluir la información que se acostumbra poner en la portada de un trabajo y para elaborar el directorio de variables del algoritmo. El directorio de variables no es otra cosa que la descripción breve del uso que se le da a cada variable dentro del algoritmo.

Los estudiantes a veces confunden los comentarios con la instrucción escriba, creen que son equivalentes. La diferencia entre los comentarios y la instrucción escriba es que sólo con esta última se establece comunicación con el usuario. Es decir, cuando hay una instrucción escriba en un algoritmo, su efecto se ve en el dispositivo de salida (la pantalla, en este caso). No hay acción alguna asociada a los comentarios porque en la ejecución, simplemente, no se toman en cuenta. Además, a los comentarios tiene acceso sólo quien construye o modifica el código del algoritmo. El usuario, normalmente, sólo puede ejecutar el algoritmo y no le interesa el código sino solucionar su problema.

Ejemplo 1.7 Calcular el perímetro de un triángulo cuyos lados miden 17.5, 23.4, 30.0. Deje todos los datos y el resultado en memoria

Una posible solución

```
"perim_un_tri"
empiece                    ** Delimitador del inicio del cuerpo del algoritmo.
real per, lado1, lado2, lado3   ** Declaración de variables. Se reserva memoria para cuatro
                           ** variables de tipo real.
lado1 ← 17.5               ** Se da valor inicial a la variable real lado1.
lado2 ← 23.4
lado3 ← 30.0
per ← lado1 + lado2 + lado3
pare                       ** Se detiene la ejecución del algoritmo y se liberan los recursos
                           ** que estaba utilizando el algoritmo.
termine                    ** Delimitador del final del cuerpo del algoritmo. No implica
                           ** acción alguna.
```

Ejercicios 1.1 Construir una solución en lenguaje algorítmico para cada uno de los siguientes problemas.

1.1.1 Hallar el resultado de las operaciones de asignación especificadas, dado el estado de las siguientes variables enteras y teniendo en cuenta el resultado de la operación anterior.

y	b	a	p	z	n	x	q
10	14	2	?	?	?	?	?
						18	

Operación	Resultado
x ← y + 8.5	18
x ← y/4 + 3	
x ← x + y + 1	
z ← b*(a – p)	
p ← x + x/2	
q ← (x + x)/2	
z ← y + 11.34	
q ← n + x + z – y	

1.1.2 Marque con una X los nombres de variables que son válidos:

	Variable		Variable
X	a100		h_251
	salario		2h
	+z		hk+z
	peso		mnl
	lea		pare
	348		_mn21

1.1.3 Escribir un algoritmo que calcule el área de cinco trapecios y el área promedio. Ejecutar el algoritmo e ilustrar el efecto en el entorno de memoria.

1.1.4 Ejecutar los siguientes algoritmos y decir qué hace cada uno. Si se encuentra algún error, explicar claramente en qué consiste y cómo podría solucionarse. Si el algoritmo está correcto, ilustrar el efecto en el entorno de memoria.

```
"ejemplito1"
entero a, b, c
a ← 10
b ← 5
c ← a + b
pare
termine
```

```
"ejemplito2"
empiece
entero m, n
n ← m + 14
pare
termine
```

```
"ejemplito3"
empiece
real n1, n2, n3, def
n1 ← 3.5
n2 ← 2.5                    ** Datos de entrada: las tres calificaciones.
n3 ← 4.0                    ** Datos de salida: la calificación definitiva.
def ← n1 + n2 + n3/3
pare
termine
```

```
"ejemplito4"
empiece
real n1, n2, n3, def
n1 ← 3.5
n2 ← 2.5                    ** Datos de entrada: las tres calificaciones.
n3 ← 4.0                    ** Datos de salida: la calificación definitiva.
def ← (n1 + n2 + n3)/3
pare
termine
```

¿Son iguales los resultados de los algoritmos ejemplito3 y ejemplito4? ¿Por qué?

1.1.5 Definir en propias palabras:

– Algoritmo

– Palabra reservada

– Variable

1.1.6 Supóngase que en cierto algoritmo se tienen la siguiente declaración de variables y las siguientes asignaciones:

entero num1, num2, num3
real valor1, valor2, valor3
num1 ← 20
num2 ← 4
num3 ← 24
valor1 ← 17.4
valor2 ← 8.6
valor3 ← 16.4

En forma independiente, evaluar cada una de las operaciones especificadas en la tabla, con base en la información dada anteriormente. Llenar sólo los espacios de las variables involucradas en la operación y expresar los resultados reales con una sola cifra decimal.

Ejemplo. Para la operación num3 ← num1 + num2 hay que llenar los espacios correspondientes a las variables num3, num1 y num2 con los valores respectivos, después de ejecutada la asignación. En las siguientes asignaciones no se deben tener en cuenta los cambios ocurridos en las variables, hay que trabajar con los valores definidos al principio.

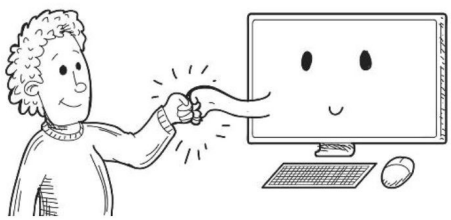
	Variables enteras			Variables reales		
Operación	**num1**	**num2**	**num3**	**valor1**	**valor2**	**valor3**
num3 ← num1 + num2	20	4	24			
num3 ← valor1 + valor2						
num3 ← num1 / num2						
num3 ← num3 + 8.7						
num3 ← num3 + valor2						
num3 ← num1 * valor2						
num3 ← valor2						
valor3 ← valor1 - valor2						
valor3 ← valor1 * num1						
valor3 ← num3 / num1						
valor3 ← valor3 / num2						
num3 ← valor3						
valor3 ← num1 + valor1						

1.10 Instrucciones de entrada o salida

Volviendo al algoritmo que suma dos números determinados, lo ideal sería contar con uno que permitiera sumar dos números enteros cualesquiera; de lo contrario, programar computadores no pasaría de ser una tarea interesante.

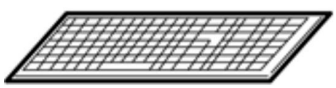
Para poder hacer esto, tiene que haber interacción entre el usuario y el computador para que sea posible especificar los números por sumar. Obsérvese que hasta este momento esa interacción no existe, ni siquiera se había mencionado.

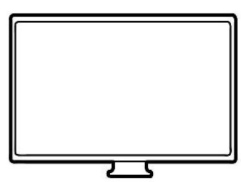
El usuario es quien hace los requerimientos y se beneficia del algoritmo.



Para que exista interacción entre el usuario y el computador, es necesario que haya instrucciones que permitan que éste "hable" con la máquina y que ésta le "responda" y viceversa.

Las instrucciones que permiten la interacción entre el usuario y la máquina son las llamadas instrucciones de entrada o salida.

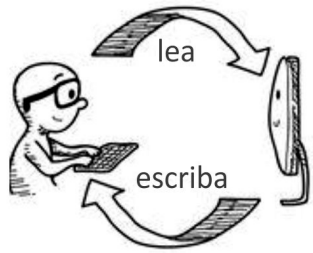




Estas instrucciones son las de lectura o escritura y están estrechamente relacionadas con los dispositivos de entrada o salida. En el presente libro se trabajará con el teclado como dispositivo de entrada y con la pantalla como dispositivo de salida.

Cuando se necesita que haya comunicación entre el usuario y la máquina, a través del usuario, se ejecuta una instrucción de lectura; el usuario tiene la

palabra (teclado). Si la comunicación se hace a través de la máquina, se ejecuta una instrucción de salida; la máquina "tiene la palabra" (pantalla).



Instrucciones de entrada y salida.

1.10.1 Instrucción de entrada lea

Consiste en almacenar en una variable el valor suministrado por el usuario mediante el dispositivo de entrada, es decir, el teclado. Su sintaxis es lea variable.

Ejemplo 1.8 Instrucción lea

El efecto de la instrucción lea m es el cambio del contenido de la variable m por el valor especificado por el usuario. En otras palabras, una lectura implica una asignación. Así que, aparte de la asignación, la instrucción de lectura es la otra posibilidad que existe para cambiar el contenido de una variable.

Aunque se suele decir "se lee la variable", lo que realmente pasa es que "se lee *en* la variable".

¿Por qué no es válida ni tiene sentido la instrucción lea 20?

1.10.2 Instrucción de salida escriba

Consiste en desplegar en el dispositivo de salida (la pantalla) los mensajes, el contenido de las variables o las constantes que aparecen en la lista. Su sintaxis puede ser una de las tres siguientes:

– escriba lista_de_variables

  Las variables de la lista deben separarse con el carácter coma (,).

Ejemplo 1.9 Escribir el contenido de variables

```
escriba a, b
escriba h
```

El efecto de estas instrucciones es escribir en la pantalla el contenido de las variables que aparecen en la lista (a y b, en el primer caso, o h, en el segundo). Si no existen, la ejecución del algoritmo se terminará anormalmente, y si existen, pero no se han inicializado, se mostrarán los valores desconocidos que contengan, que puede ser cero o espacio (si es de tipo caracter), en el mejor de los casos.

– escriba "mensaje"

Ejemplo 1.10 Escribir un mensaje

```
escriba "Colombia es un país hermoso."
```

El efecto de esta instrucción es escribir en la pantalla el texto que aparece entre comillas, tal como está. El mensaje tiene que estar entre comillas, o de lo contrario se terminará anormalmente la ejecución del algoritmo.

```
Colombia es un país hermoso.
```

– escriba variable1, "mensaje1", variable2, "mensaje2"...

Con esta instrucción se escribiría en la pantalla el contenido de la variable llamada variable1: el texto que aparece entre comillas, tal como está (mensaje1); el contenido de la variable llamada variable2: el texto que aparece entre comillas, tal como está (mensaje2),…

Ejemplo 1.11 Escribir variables y mensajes

```
escriba a, " + ", b, " = ", c
```

El efecto de esta instrucción es escribir en la pantalla el contenido de la variable llamada a: el texto que aparece entre comillas, tal como está (+); el contenido de la variable llamada b: el texto que aparece entre comillas, tal como está, (=) y el contenido de la variable llamada c.
Así, si las variables a, b y c contienen los valores 2, 3 y 5, respectivamente, en la pantalla aparecería lo siguiente:

```
2 + 3 = 5
```

Ejemplo 1.12 Escribir variables y mensajes

```
escriba "De los", ne, "estudiantes de la Escuela," , est_bog, "son bogotanos."
```

Si, por ejemplo, ne vale 4500 y est_bog vale 2700, en la pantalla aparecería lo siguiente:

```
De los 4500 estudiantes de la Escuela, 2700 son bogotanos.
```

¿Son diferentes las instrucciones escriba a y escriba "a"?

Son totalmente diferentes. La instrucción escriba a escribe en el dispositivo de salida (la pantalla en este libro) el contenido de la variable a, mientras que la instrucción escriba "a" escribe la **letra a** porque ésta aparece entre comillas.

Ahora que ya hay interacción entre el usuario y la máquina, se puede volver al problema de sumar dos números cualesquiera.

Ejemplo 1.13 Sumar dos números

Como no se especifica qué clase de números se sumarán, se puede suponer que son enteros o que son reales. En el segundo caso, se tendría un mayor alcance (¿por qué?).

¿Qué variables se necesitan?

Se necesitan dos variables para guardar los sumandos y una para guardar el resultado. En el algoritmo solución se supondrá que los números son enteros.

Una posible solución

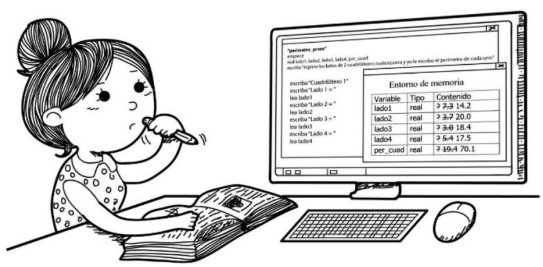
```
"suma_dos_numeros_enteros"
empiece
entero num1, num2, suma
escriba "Yo sumo dos números enteros cualesquiera."
escriba "Por favor, ingrese el primer número."
lea num1
escriba "Por favor, ingrese el segundo número."
lea num2
suma ← num1 + num2
escriba num1, "+", num2, " = ", suma
escriba "Tarea finalizada."
pare
termine
```

Al diseñar los mensajes para la instrucción escriba, no se debe olvidar que la comunicación es entre seres humanos y que, por tanto, se deben escribir en el lenguaje natural de la persona. Suele pasar que por pensar en la máquina los mensajes son secos, carecen de calidez. Tampoco es conveniente incluir términos propios del lenguaje que son innecesarios y pueden confundir al usuario. Por ejemplo:
En lugar de la instrucción escriba "Este algoritmo suma dos números." sería mejor algo como escriba "Suma de dos números."

Obsérvese que en este algoritmo no se puede decir que el contenido final de la variable suma sea desconocido porque, si bien no se sabe qué valores ingresará el usuario en el momento de la ejecución, a la variable suma sí se le está definiendo un valor: el de la suma de las variables num1 y num2, cuyos contenidos son definidos previamente por el usuario.

1.11 Prueba de escritorio

En este libro, la ejecución de un algoritmo consistirá en hacer las veces de máquina, es decir, llevar a cabo todas las tareas que después tendrá que hacer el computador. A este trabajo se le llamará prueba de escritorio, la cual se hace con papel y lápiz y en una forma similar a la que se muestra a continuación. Esas pruebas se hacen con valores cualesquiera, de acuerdo con el tipo de las variables, y su efecto se refleja en la pantalla y en el entorno de memoria.



Prueba de escritorio.

Ejemplo 1.14 Prueba de escritorio del algoritmo suma_dos_numeros_enteros y su reflejo en el entorno de memoria y en la pantalla

<table>
<tr><th>Instrucción</th><th>Efecto</th></tr>
<tr><td>entero num1, num2, suma</td><td>Se reserva espacio para tres variables enteras llamadas num1, num2 y suma, respectivamente.

<table><tr><th>Variable</th><th>Tipo</th><th>Contenido</th></tr><tr><td>num1</td><td>entero</td><td>?</td></tr><tr><td>num2</td><td>entero</td><td>?</td></tr><tr><td>suma</td><td>entero</td><td>?</td></tr></table></td></tr>
<tr><td>escriba “Yo sumo dos números enteros cualesquiera.”</td><td>Aparece el mensaje en la pantalla.

Yo sumo dos números enteros cualesquiera.</td></tr>
<tr><td>escriba “Por favor, ingrese el primer número”</td><td>Aparece el mensaje en la pantalla.

Por favor, ingrese el primer número</td></tr>
<tr><td>lea num1</td><td><table><tr><th>Variable</th><th>Tipo</th><th>Contenido</th></tr><tr><td>num1</td><td>entero</td><td>~~?~~ 13</td></tr><tr><td>num2</td><td>entero</td><td>?</td></tr><tr><td>suma</td><td>entero</td><td>?</td></tr></table></td></tr>
<tr><td>escriba “Por favor, ingrese el segundo número”</td><td>Aparece el mensaje en la pantalla.

Por favor, ingrese el segundo número</td></tr>
<tr><td>lea num2</td><td><table><tr><th>Variable</th><th>Tipo</th><th>Contenido</th></tr><tr><td>num1</td><td>entero</td><td>~~?~~ 13</td></tr><tr><td>num2</td><td>entero</td><td>~~?~~ –4</td></tr><tr><td>suma</td><td>entero</td><td>?</td></tr></table></td></tr>
</table>

<table>
<tr><th>Instrucción</th><th>Efecto</th></tr>
<tr><td>suma ← num1 + num2</td><td>
<table>
<tr><th>Variable</th><th>Tipo</th><th>Contenido</th></tr>
<tr><td>num1</td><td>entero</td><td>? 13</td></tr>
<tr><td>num2</td><td>entero</td><td>? -4</td></tr>
<tr><td>suma</td><td>entero</td><td>? 9</td></tr>
</table>
</td></tr>
<tr><td>escriba num1, " + ", num2, " = ", suma</td><td>Aparece el mensaje en la pantalla.
13 + –4 = 9</td></tr>
<tr><td>escriba "Tarea finalizada."</td><td>Aparece el mensaje en la pantalla.
Tarea finalizada.</td></tr>
<tr><td>pare</td><td>El espacio de memoria reservado para las variables num1, num2 y suma queda libre.</td></tr>
</table>

¿Qué pasa si el usuario ingresa un valor real en lugar de uno entero? Simplemente, se toma la parte entera. Por ejemplo, si el usuario ingresa 13.57, el valor que se almacena es 13.

¿Cómo sería la versión de esta solución para números reales? ¿Qué pasaría en este caso si el usuario ingresara un número entero?

Si ahora se quisiera restringir el algoritmo de suma de dos enteros a enteros positivos, ¿cómo se haría?

Con las herramientas que se tienen hasta ahora en el lenguaje algorítmico, no es posible hacer tal restricción. Es necesario tener la forma de condicionar la realización de la suma. Si los números son positivos… Primero, para establecer relación entre los operandos se necesitan los operadores relacionales, y segundo, para exigir que la condición "ser positivo" sea satisfecha por ambos operandos, se requieren los conectores lógicos.

Ejemplo 1.15 Calcular y escribir el área de un círculo, definido por el usuario, y el perímetro de su circunferencia. Hay que escribir mensajes claros al usuario, de tal manera que sepa qué datos debe ingresar y por qué

Una posible solución

```
"area_cir_cualquiera"
empiece
real radio, area, per_cir, pi
** El siguiente mensaje se verá en la pantalla.
escriba "Si me da el radio de un círculo, calculo su área y la longitud de la circun-
ferencia."
pi ← 3.1415    ** Se inicializa la variable pi con el valor aproximado de la constante π.
escriba "radio = "
lea r                ** El usuario debe dar el valor del radio.
area ← pi*r*r
per_cir ← 2*pi*r
** Resultados.
escriba "El área de un círculo de radio", r, "es", area, "unidades cuadradas
y el perímetro de su circunferencia es ", per_cir, "unidades de longitud."
escriba "Fin."
pare
termine
```

Ejemplo 1.16 Perímetro de dos cuadriláteros

Calcular y escribir el perímetro de dos cuadriláteros cualesquiera definidos por el usuario. Hágale una prueba de escritorio e ilustre el efecto en el entorno de memoria correspondiente y en la pantalla.

Una posible solución

```
"perimetro_prom"
empiece
real lado1, lado2, lado3, lado4, per_cuad
escriba "Ingrese los lados de dos cuadriláteros cualesquiera y yo le escribo el
perímetro de cada uno."
escriba "Cuadrilátero 1"
escriba "Lado 1 = "
lea lado1
escriba "Lado 2 = "
lea lado2
escriba "Lado 3 = "
lea lado3
escriba "Lado 4 = "
lea lado4
```

Entorno de memoria

Variable	Tipo	Contenido
lado1	real	~~?~~ ~~7.3~~ 14.2
lado2	real	~~?~~ ~~3.7~~ 20.0
lado3	real	~~?~~ ~~3.0~~ 18.4
lado4	real	~~?~~ ~~5.4~~ 17.5
per_cuad	real	~~?~~ ~~19.4~~ 70.1

```
per_cuad ← lado1 + lado2 + lado2 + lado4
escriba "El perímetro del cuadrilátero de lados", lado1, ", ", lado2, ", ", lado3, "y",
lado4, " es ", per_cuad, "."
escriba "Cuadrilátero 2"
escriba "Lado 1 = "
lea lado1
escriba "Lado 2 = "
lea lado2
escriba "Lado 3 = "
lea lado3
escriba "Lado 4 = "
lea lado4
per_cuad ← lado1 + lado2 + lado2 + lado4
escriba "El perímetro del cuadrilátero de lados", lado1, ", ", lado2, ", ", lado3, "y",
lado4, " es ", per_cuad, "."
escriba "Fin."
pare
termine
```

Pantalla, prueba de escritorio

```
Ingrese los lados de dos cuadriláteros cualesquiera y yo le escribo el perímetro de cada uno.
Cuadrilátero 1
Lado 1 = 7.3
Lado 2 = 3.7
Lado 3 = 3.0
Lado 4 = 5.4
El perímetro del cuadrilátero de lados 7.3, 3.7, 3.0 y 5.4 es 19.4.
Cuadrilátero 2
Lado 1 = 14.2
Lado 2 = 20.0
Lado 3 = 18.4
Lado 4 = 17.5
El perímetro del cuadrilátero de lados 14.2, 20.0, 18.4 y 17.5 es 70.1.
Fin.
```

Ejemplo 1.17 Pedir dos números enteros al usuario y luego intercambiar el contenido de las variables en las que quedaron guardados. Tener en cuenta que no se trata de escribir las variables al revés sino que en la memoria los contenidos deben quedar modificados

Si los contenidos de las variables fueran:

Variable 1: 7
Variable 2: –14

El resultado sería:

Variable 1: –14
Variable 2: 7

Una posible solución

```
"intercambia"
empiece
entero valor1, valor2, aux
escriba "Ingrese dos números enteros, yo los intercambio y los escribo de nuevo."
escriba "Primer número"
lea valor1
escriba "Segundo número"
lea valor2
** Intercambia el contenido de las dos variables utilizando una tercera variable.
aux ← valor1
valor1 ← valor2
valor2 ← aux
escriba "Después de intercambiar los números, número 1 = ", valor1, "número 2
= ", valor2, "."
escriba "Fin."
pare
termine
```

– ¿Necesariamente se requiere una tercera variable para hacer el intercambio?
– Si se hicieran las siguientes asignaciones, sin utilizar una tercera variable, ¿quedarían intercambiados los contenidos de las variables? ¿Por qué?

```
valor1 ← valor2
valor2 ← valor1
```

Ejemplo 1.18 Pendiente y punto de corte de una recta

Pedir al usuario dos puntos de una recta y calcular su pendiente (m) y el valor de la ordenada (b) en el punto de corte con el eje y. Adicionalmente, escribir la ecuación de la recta en la forma $y = mx + b$, por ejemplo, $y = -3.5x + 5$. ¿La recta dada por el usuario es vertical u horizontal?

Una posible solución

```
"ecuac_recta_dos_puntos_1"
empiece
real x1, y1, x2, y2, m, b

escriba "Ingrese dos puntos de una recta y yo calcularé su pendiente (m) y el valor
de la ordenada (b) en el punto de corte con el eje y. Adicionalmente escribiré la
ecuación de la recta en la forma y = mx + b, por ejemplo, y = –3,5x + 5."
** Entrada de datos.
escriba "Coordenadas del punto 1 (abscisa, ordenada): "
escriba "Abscisa = "
lea x1
escriba "Ordenada = "
lea y1
escriba "Coordenadas del punto 2:"
escriba "Abscisa = "
lea x2
escriba "Ordenada = "
lea y2
** Procesamiento y escritura de resultados. Cálculo de la pendiente.
m ← (y2 - y1)/(x2 - x1)
** La forma general de la ecuación de la recta es y = mx + b. Como se conocen la pendiente y un punto de ella,
** entonces b (ordenada del punto de corte con el eje y) es igual, por ejemplo, a y1 – mx1.
b ← y1 – m* x 1
escriba "La ecuación de la recta que pasa por los dos puntos (", x1, "," , y1, ") y (",
x2, "," , y2, ") es y=", m, "x + ", b, ", donde ", m, "es el valor de la pendiente y ",
b, "es el valor de la ordenada en el punto de corte con el eje y."
escriba "Fin."
pare
termine
```

Palabras reservadas	
empiece	entero
termine	real
pare	caracter
cadena_de	lea
escriba	

Ejemplo 1.19 Suma de dos números enteros cualesquiera

Una posible solución

```
"suma_dos_num"
empiece
** Tema: Uso de las instrucciones de lectura y escritura.
** Lenguaje: Algorítmico.
** Problema: Sumar dos números enteros cualesquiera.
entero s1, s2, suma
** Directorio de variables.
** s1: primer sumando.
** s2: segundo sumando.
** suma: suma de s1 y s2.
escriba "Yo sumo dos números enteros cualesquiera."
** Se solicitan los sumandos.
escriba "Primer número "
lea s1
escriba "Segundo número "
lea s2
suma ← s1 + s2                    ** Se efectúa la suma.
escriba s1, " + ", s2, " = ", suma  ** Se escribe el resultado.
escriba "Fin."
pare
termine
```

Al ejecutar el algoritmo, el usuario verá sólo los mensajes que corresponden a las instrucciones escriba. Los comentarios no se reflejan en el dispositivo de salida; no van dirigidos al usuario.

En el siguiente ejemplo no se ejecuta la instrucción de escritura porque es un comentario.

```
"instrucción_sin_efecto"
empiece
** escriba "Colombia es un país hermoso."
pare
termine
```

Ejercicios 1.2 Pruebas de escritorio y solución de problemas

1.2.1 Realizar una prueba de escritorio a cada uno de los siguientes algoritmos sin modificarlos. Si se encuentra algún error, explicar claramente en qué consiste. Ilustrar el efecto en el entorno de memoria y en la pantalla correspondientes.

1.2.1.1

```
"suma"
empiece
entero a, b, suma
escriba "Suma de dos números enteros cualesquiera."
suma ← a + b
escriba a," + ", b, " = ", suma
escriba "Suma realizada."
pare
termine
```

1.2.1.2

```
"area_rectan"
empiece
escriba "Declaración de variables"
real base, altura, area
escriba "Cálculo del área de un rectángulo cualquiera."
escriba "Base = "
lea base
escriba "Altura = "
lea altura
area ← base * altura
escriba "El área del rectángulo cuya base es ", base, " y cuya altura es ", altura, " es ", area
escriba "Fin - área rectángulo."
pare
termine
```

1.2.2 Construir una solución en lenguaje algorítmico para cada uno de los siguientes problemas:

1.2.2.1 Calcular y escribir el perímetro de un cuadrilátero cuyos lados miden 14, 22, 8 y 17. Deje todos los datos y el resultado en memoria. Hágale una prueba de escritorio e ilustre el efecto en el entorno de memoria correspondiente.

1.2.2.2 Solicitar al usuario las dimensiones de una pieza rectangular de vidrio y las de la pieza cuadrada que quiere extraer de la primera, tantas veces como sea posible. Averiguar cuántas piezas cuadradas puede sacar. Si no es posible, explíquele al usuario el porqué. Ilustre gráficamente dos situaciones que pueden presentarse: una en la que es posible extraer piezas cuadradas y otra en la que no.

1.2.2.3 Calcular y escribir el área de un círculo, definido por el usuario, y el perímetro de su circunferencia. No olvide escribir mensajes claros al usuario, de tal manera que sepa qué datos debe ingresar.

1.2.2.4 Calcular y escribir las coordenadas de dos puntos que pertenezcan a una recta ingresada por el usuario en la forma y = mx + b. ¿Cuáles son los datos de entrada que se necesitan? Escribir también la recta que definió el usuario. Hay que escribir mensajes claros al usuario, de modo que sepa qué datos debe ingresar o a qué se refiere lo que está viendo en el dispositivo de salida (la pantalla, en este caso). Dejar todos los datos y el resultado en memoria.

1.2.2.5 Calcular y escribir la pendiente de una recta, dados dos puntos que pertenecen a ella. No olvide escribir mensajes claros al usuario, de tal manera que sepa qué datos debe ingresar y por qué. Deje todos los datos y el resultado en memoria. Hágale una prueba de escritorio e ilustre el efecto en el entorno de memoria correspondiente.
Para pensar. Si tuviera que hacer un algoritmo para averiguar si un punto pertenece a una recta, ¿qué datos de entrada necesitaría? ¿Por qué? ¿Cómo solucionaría el problema? ¿Cuáles serían los posibles resultados de su algoritmo?

1.2.2.6 Calcular y escribir el área promedio de tres trapecios cualesquiera. Los datos de los trapecios se solicitarán al usuario.

1.2.2.7 Calcular y escribir el volumen de tres cilindros cualesquiera. Calcular y escribir también el volumen promedio.

1.2.2.8 Calcular y escribir la edad promedio de seis personas. En el resultado hay que escribir las edades.

1.2.2.9 Calcular y escribir la edad promedio de seis personas. En el resultado no hay que escribir las edades. ¿Cuál es la diferencia con el algoritmo anterior? ¿Qué podría aprovechar?

1.2.2.10 Calcular y escribir la calificación obtenida en la asignatura que el usuario especifique. El usuario también suministrará las calificaciones; las dos primeras valen el 60 % (30 y 30) y la tercera el 40 %.