

Universidad Autónoma de Entre Ríos
Facultad de Ciencia y Tecnología
Sede: Oro Verde



FUNDAMENTOS DE PROGRAMACIÓN

RESUMEN DE CONTENIDOS N° 2 **Formalización de Algoritmos**

RESUMEN DE CONTENIDOS N° 2

Formalización de Algoritmos

ALGORITMOS COMPUTACIONALES

Los problemas desarrollados hasta el momento, se resolvieron empleando instrucciones coloquiales y se basaban en situaciones de la vida diaria. Los algoritmos así desarrollados son fáciles de entender para cualquier persona o ejecutante.

Pero nuestro objetivo es llegar a desarrollar algoritmos que puedan ser interpretados por una computadora. Para ello, es necesaria una descripción clara y carente de ambigüedades de cada una de las acciones que llevan a la solución del problema, por medio de órdenes comprensibles para la computadora.

En este capítulo nos ocuparemos de describir la formalización necesaria para el desarrollo de Algoritmos Computacionales.

De aquí en más cuando mencionemos al ejecutante de un algoritmo, nos estaremos refiriendo concretamente a una computadora.

En el siguiente ejemplo planteamos un algoritmo completo de acuerdo a la formalización que propondremos posteriormente.

Ejemplo: Plantear un algoritmo computacional que calcule la hipotenusa de un triángulo rectángulo. Se conocen las longitudes de los catetos.

Análisis del problema:

- Datos: Longitudes de los catetos
- Resultados: Hipotenusa
- Relación: Teorema de Pitágoras

Algoritmo:

PROCESO: Hipotenusa

```
cin>> A, B;  
H = sqrt(pow(A,2) + pow(B, 2));  
cout<< "Hipotenusa = ";  
cout<< H;
```

FINPROCESO

En este ejemplo A, B y H constituyen **variables**; 2 e "Hipotenusa = " son **constantes**; **CIN>>**, **COUT<<** y **=** son las acciones primitivas de lectura, escritura y asignación respectivamente; **sqrt(pow(A,2) + pow(B, 2))** es una **expresión numérica**; **pow()** y **sqrt()** son funciones de potencia y raíz cuadrada.

El signo ; (punto y coma) es el separador de acciones. En esta unidad iremos desarrollando todos estos elementos, que conforman un lenguaje algorítmico que llamaremos **PSEUDOCODIGO orientado a C++**.

La forma general de un algoritmo escrito en pseudocódigo es la siguiente:

```
PROCESO <nombre del proceso>
  acción 1;
  acción 2; acción 3;
  :
  acción n-1;
  acción n
FINPROCESO
```

Donde las acciones van separadas por el signo punto y coma (;) y pueden ir varias acciones en una misma línea.

CONSTANTES, VARIABLES Y EXPRESIONES



Definimos **constante**, como un elemento cuyo valor no puede alterarse en el transcurso de la ejecución de un algoritmo. Por ejemplo: 123, 'López', 3.14459

Pero además en un algoritmo existen otros elementos cuyo valor cambia durante la ejecución del mismo, llamados **variables**. En el ejemplo inicial A, B y H son variables -correspondientes a los catetos y a la hipotenusa del triángulo, respectivamente- que adquieren un valor en el momento de la ejecución del algoritmo.



Una **variable** es un elemento o lugar asociado a un valor que puede variar conforme se ejecuta el algoritmo, se representa por un nombre que identifica a una posición de memoria donde puede asignarse o almacenarse un único valor por vez.

Para proponer el nombre de una variable observaremos tres reglas simples:

- 1) Utilizar sólo letras y/o dígitos, comenzando siempre con una letra.
- 2) No utilizar las palabras claves o reservadas en el pseudocódigo, como: **CIN**, **COUT**, **PROCESO**, **FINPROCESO**, etc.; o para las funciones internas: **SQRT**, **POW**, **SIN**, **LOG**, etc.
- 3) No hacer distinción entre mayúsculas y minúsculas. Esto implica que VENTA, venta y Venta, constituyen en el algoritmo la misma variable.

Esta sintaxis y sus restricciones no presentan inconvenientes para proponer nombres de variables, pues disponemos de un sinnúmero de combinaciones diferentes de letras y dígitos.

Por ejemplo: NOMBRE, DIRECCION, Res1, MONTO.

Ejercicio: En los siguientes ejemplos de variables, indique si son válidas; caso contrario especifique la causa:

- a) venta b) x12 c) (to
tal) d) resultado e) *PRODU f) promedio

El diseñador del algoritmo tiene total libertad para proponer nombres a sus variables, aunque como consejo, es conveniente proponer identificadores que tengan alguna relación con el

dato que representan. En nuestro ejemplo inicial, podríamos haber empleado CATETO1, CATETO2, HIPOT, en lugar de A, B, H.

Por último:



Definimos **expresión** a un conjunto de operandos ligados por operadores, que describen una operación o cálculo arrojando un único resultado. En nuestro primer ejemplo $\text{sqrt}(\text{pow}(A,2) + \text{pow}(B,2))$ es una expresión que permite obtener el valor de la hipotenusa.

Algunos ejemplos de expresiones: 1) $2+a-x*5$, 2) $A < B$, 3) $\text{round}(R) + 1$

TIPOS DE DATOS

Un algoritmo computacional puede emplear los siguientes tipos de datos:

- Numérico
- Caracter
- Lógico

Esta clasificación nos define los tipos primitivos de datos. Estudiaremos a cada uno de ellos y su modo de empleo.

Tipo Numérico

Es el conjunto formado por los valores numéricos que pueden incluir un punto decimal y pueden estar precedidos por los signos '+' o '-'. La ausencia de signo implica un valor positivo.

Se clasifican a su vez en tipo entero o tipo real, pero no haremos distinciones en su empleo. Los datos de tipo numérico pueden representarse como constantes, variables y /o expresiones.

Constantes numéricas

Una constante numérica es un valor formado por algún elemento del conjunto numérico. Veamos algunos ejemplos de constantes numéricas que puedan emplearse en un algoritmo computacional:

12 -25000 +1.2345 -2345 -23576.998 0.45E+02

En el último ejemplo se indica una constante numérica con notación científica, donde E+02 indica la potencia de base diez, es decir:

$$0.45E+02 = 0.45 * 10^2 = 45$$

Variables numéricas

Cualquier identificador o variable que represente a un dato numérico se denomina variable numérica. Si una variable se define en un algoritmo como numérica, no podrá adoptar valores de tipo no numérico.

Expresiones numéricas

Una expresión numérica es aquella que combina operandos numéricos (constantes numéricas, variables numéricas u otras expresiones numéricas) con los operadores algebraicos.

Los operadores algebraicos y su función son:

Operador	Función
+	Suma
-	Resta
*	Producto
/	Cociente

La jerarquía de estos operadores es la misma que plantea el álgebra de los números, y podrá ser alterada con la intercalación de niveles de paréntesis.

Ejemplos: $2 + a * \text{pow}(10, 2) - 800 / C$
 $1000 - (2 * \text{TOT} - 30) / P$
 $4 * (\text{VENTA} / - 25)$

donde a, C, TOT, P, VENTA son todas variables numéricas. En cada caso, al evaluar la expresión se obtiene un único resultado.



Observación: Pondremos como regla para la escritura de pseudocódigo, que la potencia no existe como operador. Para ello, se utiliza una función numérica denominada pow.

Supondremos además, que el ejecutante de nuestro algoritmo (la computadora) conoce y puede resolver ciertas funciones numéricas. A estas funciones las llamaremos "*funciones internas*" y, tienen la propiedad de devolver un valor o resultado al ser aplicadas sobre un argumento indicado entre paréntesis.

Algunas funciones son:

Función	Significado
sqrt()	Raíz cuadrada
abs()	Valor absoluto
pow()	Potenciación
sin()	Seno
cos()	Coseno
tan()	Tangente
round()	Redondeo

Con esto podemos ampliar el uso de las expresiones numéricas antes mencionadas.

Ejemplos: $\text{round}(2/3) - \text{abs}(A) * 3$
 $\sin(X) + 1 - \tan(C/2)$

$$12/\sqrt{\tan(3.1416/4)}$$

Tipo Caracter

El tipo caracter incluye al conjunto formado por todos los caracteres o símbolos de código ASCII (Código Estándar Americano para el Intercambio de Información). La mayoría de ellos se generan en el teclado de una computadora, es decir, las letras de nuestro alfabeto en mayúscula, en minúscula, los dígitos, los operadores relacionales '<', '>', '=', el espacio en blanco ' ', los operadores aritméticos '+', '-', '*', '/', los caracteres de puntuación y otros especiales '!', '@', '%', '(', etc.

Este conjunto de 256 caracteres es un conjunto ordenado, y por lo tanto, existe una relación de precedencia u orden entre sus elementos (de menor a mayor) determinado por su número de código. El orden lo establece el código ASCII (ver la tabla con algunos de los códigos).

Tener en cuenta que en el orden de precedencia del código ASCII, los siguientes grupos se hallan propuestos de menor a mayor:

- El espacio en blanco
- Los dígitos '0', '1', '2',.....'9'.
- Las letras mayúsculas 'A', 'B',.....'Z'.
- Las letras minúsculas 'a', 'b',.....'z'.

Algunos caracteres del código ASCII con su número de orden correspondiente

32	' '	48	'0'	64	'@'	80	'P'	96	'`'	112	'p'
33	'!'	49	'1'	65	'A'	81	'Q'	97	'a'	113	'q'
34	'"'	50	'2'	66	'B'	82	'R'	98	'b'	114	'r'
35	'#'	51	'3'	67	'C'	83	'S'	99	'c'	115	's'
36	'\$'	52	'4'	68	'D'	84	'T'	100	'd'	116	't'
37	'%'	53	'5'	69	'E'	85	'U'	101	'e'	117	'u'
38	'&'	54	'6'	70	'F'	86	'V'	102	'f'	118	'v'
39	'"'	55	'7'	71	'G'	87	'W'	103	'g'	119	'w'
40	'('	56	'8'	72	'H'	88	'X'	104	'h'	120	'x'
41	')'	57	'9'	73	'I'	89	'Y'	105	'i'	121	'y'
42	'*'	58	':'	74	'J'	90	'Z'	106	'j'	122	'z'
43	'+'	59	','	75	'K'	91	'['	107	'k'	123	'{'
44	':'	60	'<'	76	'L'	92	'\'	108	'l'	124	' '
45	'-'	61	'='	77	'M'	93	']'	109	'm'	125	'}'
46	'.'	62	'>'	78	'N'	94	'^'	110	'n'	126	'~'
47	'/'	63	'?'	79	'O'	95	'_'	111	'o'	127	'\''

Constantes caracter

Una constante tipo caracter es una constante cuyo valor pertenece al conjunto mencionado anteriormente delimitada por apóstrofes (' '), si se trata de un único caracter, o delimitada por comillas (" "), cuando se trate de una cadena de caracteres.

De acuerdo a la relación de orden establecida, podemos afirmar que son verdaderas las siguientes proposiciones:

La letra 'a' es mayor que la letra 'B'.
El caracter '5' es menor que la letra 'A'.

Dentro de este tipo incluiremos a las cadenas de caracteres secuencia finita de caracteres encerrados entre apóstrofos y por lo tanto, tienen una longitud que está dada por la cantidad de caracteres que la forman.

Por ejemplo, los nombres, apellidos y direcciones, constituyen información formada con sucesiones o cadenas finitas de los caracteres mencionados anteriormente.

Ejemplos:

"Jorge Fernández"	"Resultado="	"San Martín 2377 - Dpto. 5"
"043-234558"	"HIPOTENUSA"	"*****"

Aquí también es válido establecer una relación de orden basada en la precedencia entre caracteres ya señalada. Para establecer dicho orden entre dos cadenas, se compara carácter por carácter hasta encontrar una desigualdad.

Entonces son válidas las proposiciones:

"Mario"	es mayor que	"Mariana"
"043-553456"	es menor que	"TE:043-553456"
"mesa"	es menor que	"mesas"

Observemos que en nuestra formalización algorítmica, los datos de tipo carácter o las cadenas de caracteres se indican siempre entre apóstrofos. Esto es para evitar confusiones con otros elementos algorítmicos que desarrollaremos más adelante.

Variables carácter

Una variable o identificador que represente a un carácter o a una cadena de caracteres es una variable de tipo carácter.

Tipo Lógico

El tipo lógico nos permite expresar un valor de verdad en un algoritmo a través de constantes, variables y/o expresiones.

Constantes Lógicas

Las constantes lógicas son sólo 2 y están representadas por los valores Verdadero y Falso. Representaremos a las **constantes lógicas** con los símbolos V y F. El subrayado es para evitar ambigüedades con las variables V y F. Esta información se produce como resultado de cualquier expresión lógica.

Variables Lógicas

Una variable lógica será una variable que representa a alguno de los dos valores lógicos: V o F.

Expresiones lógicas

Las expresiones lógicas más sencillas se plantean con la combinación de operandos del mismo tipo y los operadores relacionales matemáticos.

Operador	Significado
<	Menor que
>	Mayor que
==	Igual que
>=	Mayor o igual que
<=	Menor o igual que
<>	Distinto

Ejemplos:

<u>Expresión Lógica</u>	<u>Resultado</u>
45 > 12	<u>V</u>
56 <= 30	<u>F</u>
"Andrea" > "Mariana"	<u>F</u>
"Andrea" > "Ana"	<u>V</u>
60 < "23"	expresión no válida

Las expresiones lógicas simples mostradas en el ejemplo, también son conocidas como expresiones relacionales pues permiten comparar o relacionar a dos operandos del mismo tipo. Justamente, la última expresión del ejemplo anterior no es válida pues compara a una constante numérica con una constante de tipo carácter.

La algorítmica computacional, permite también formar expresiones lógicas más complejas, a través de operandos lógicos y los operadores que emplea la lógica proposicional:

Conector Lógico	Significado
^	Conjunción
∨	Disyunción
~	Negación

Observemos entonces el valor de verdad de las siguientes expresiones lógicas:

<u>Expresión Lógica</u>	<u>Resultado</u>
(7 < 10) ^ ('a' < 'c')	<u>V</u>
("Ana" < "Angel") ∨ (12 > 19)	<u>V</u>
~ (37 > 18)	<u>F</u>
(VENTA > X) ∨ (sin(X) <= 1)	<u>V</u>

PRIMITIVAS

En la unidad anterior definimos a primitiva como la acción algorítmica cuyo enunciado es suficiente para que el ejecutante pueda realizarla sin ningún tipo de información adicional.

En un lenguaje algorítmico como lo son los lenguajes de programación, las primitivas se identifican con palabras claves o reservadas. Nosotros empleamos bloques gráficos en el diagrama de flujo para representar acciones primitivas. En algorítmica computacional las acciones primitivas se clasifican en:

- Secuenciales
- Condicionales
- Repetitivas

Desarrollaremos a continuación algunas acciones primitivas fundamentales de tipo Secuencial.

Asignación

Esta acción permite a un identificador o variable, representar o memorizar cierto valor. Para describirla utilizaremos la notación siguiente:

$$V = E$$

Donde V es la variable a la cual el ejecutante debe asignar el valor de E. El símbolo '=' puede leerse 'toma el valor'.

Según sean los tipos de V y E una asignación puede ser:

- aritmética
- caracter
- lógica

Asignación aritmética

La asignación **V= E** es aritmética si V es una variable numérica y E es una constante numérica, una variable numérica o cualquier expresión de tipo numérico.

Ejemplos:

A = 43; (A toma el valor 43)

X = A; (X toma el valor contenido en la variable A)

NUM = 3*X+2; (NUM toma el valor que resulta de evaluar la expresión 3*X+2)

Notemos que para poder realizar una asignación aritmética debemos evaluar la expresión de la derecha y luego ese resultado se almacena en la variable que figura a la izquierda. Por lo tanto es perfectamente válida la acción: $N = N+1$; que equivale a: tomar el valor actual de N, sumarle 1 y asignarle ese resultado a la variable N. Por ejemplo, si antes de la acción N contenía el valor 8, luego de dicha acción contendrá 9. Observemos que esta acción algorítmica no tiene nada que ver con los conceptos asimilados en matemática, donde $N = N+1$ es una expresión incompatible.

Asignación tipo caracter

La asignación **V= E** es una asignación de tipo caracter si V es una variable tipo caracter y E es un constante caracter o una variable tipo caracter

Ejemplos:

A = '7';	(A toma el valor '7')
X3 = A;	(X3 toma el valor contenido almacenado en la variable A)
APELLIDO = "López";	(APELLIDO toma el valor 'López')

Asignación lógica

La asignación **V = E** es una asignación lógica si V es una variable de tipo lógico y E es una constante lógica (Verdadero o Falso), una variable lógica o una expresión lógica.

Ejemplos:

M = E ;	(M toma el valor E)
TRUE = 34 <= 78;	(TRUE toma el valor lógico V)
G = (A < 2) ^ (C = 10)	(G toma el valor lógico resultante de la expresión (A<2) ^ (C=10))

Entrada y Salida

Todo algoritmo tiene por objetivo principal producir resultados, pudiendo o no incorporar información del medio externo (datos), al ambiente o sistema que observa.

Esta incorporación de valores desde el exterior al ambiente del algoritmo, nos lleva a definir una acción primitiva de lectura o entrada. Usaremos para ello la palabra clave **cin>>** que permitirá al ejecutante identificar esta acción.

El formato de la acción de lectura es:

cin>> V cin>> lista-variables
--

donde **V** es una variable o
donde **lista-variables** es una lista de variables separadas por el signo coma (,).

De esta manera, la acción **cin>>** nos permite ingresar, desde el medio externo, uno o más valores los cuales son asignados a la variable V o a las variables que figuran a continuación de **cin>>**.

Ejemplos:

```
cin>>DAT;  
cin>>NOMBRE, APELLIDO, DNI;
```

Esta acción tiene el mismo efecto que una asignación, sólo que ésta utiliza valores del ambiente del algoritmo; en cambio la lectura asigna valores desde el medio exterior. Esta acción contribuye a hacer a los algoritmos de uso general, pues permite incorporar datos nuevos para producir nuevos resultados. Sin esta acción, la ejecución de un algoritmo producirá siempre la misma respuesta.

La acción primitiva que permite a un algoritmo comunicar resultados o **salida** de información al medio exterior, la describiremos con la palabra clave **cout<<**; y a continuación una variable, una constante, o una lista de variables y/o constantes.

Su formato general es:

cout<< C; cout<< V; cout<< lista-variables-constantes;

donde **C** representa una constante, **V**, una variable del ambiente y **lista-variable-constante** es una secuencia de variables y/o constantes.

En esta acción el valor almacenado en una variable o el de una constante es comunicado al mundo exterior.

Ejemplos:

```
cout<<APELLIDO;  
cout<<DAT, NOMBRE;  
cout<<"23";  
cout<<"Total = ", X;
```

Destaquemos algunas diferencias entre las acciones de lectura y escritura. La lectura se realiza solamente a través de variables y, por lo tanto, si se lee una variable que ya fue definida en el algoritmo, implicará un acceso destructivo, esto es, la variable perderá su valor para tomar el del nuevo dato que se ingresa. En cambio, si se escriben resultados a través de variables, el ejecutante realizará un acceso no destructivo a dichas variables, pues sólo necesita conocer su contenido para ejecutar la escritura. Aquí las variables conservan sus valores después de la acción.

Las acciones de lectura y escritura son conocidas como acciones de entrada/salida o abreviadamente E/S.

MODELO PARA LA RESOLUCION DE PROBLEMAS

Propondremos ahora para este curso, un modelo para la presentación del planteo y diseño de un algoritmo, que permita resolver un problema dado.

Para ello, y habiendo abordado las etapas de Definición del problema y del Análisis del problema - vistas en la unidad anterior - nos centraremos en la etapa de Elección y creación del método, en la cual debemos considerar tres elementos:

- Estrategia y refinamiento de la misma
- Ambiente
- Algoritmo

Hemos desarrollado en la unidad anterior el objetivo de la **estrategia**. En el **ambiente** se definen y describen las variables que se emplearán, indicando por cada una su nombre, su tipo, su clase (por el momento solamente: simples) y su significado. Y por último, se define el **algoritmo** escrito en un lenguaje algorítmico. En el curso emplearemos como lenguajes algorítmicos el PSEUDOCODIGO y los DIAGRAMAS DE FLUJO.

En estos primeros capítulos, en la resolución de problemas no incluiremos la estrategia y su posterior refinamiento.

Veamos en un ejemplo como aplicamos el modelo de resolución de problemas.

Problema: intercambiar los valores de dos variables numéricas que se leen como datos.

Ambiente:

Variable	Numérica	Carácter	Lógica	Clase	Significado
A	X			Simple	Dato
B	X			Simple	Dato
Aux	X			Simple	Auxiliar

Algoritmo: (en pseudocódigo)

PROCESO Intercambio

```

cin>> A, B;
AUX= A;
A= B;
B= AUX;
cout<< A, B;

```

FINPROCESO

DIAGRAMAS DE FLUJO

Las acciones antes descriptas corresponden a un lenguaje algorítmico que llamaremos pseudocódigo. Además, diseñaremos algoritmos en forma gráfica a través de los llamados **diagramas de flujo**.

En un diagrama de flujo, las estructuras de las primitivas del pseudocódigo se identifican con una forma geométrica identificatoria o bloque. Estos bloques se unen con flechas que nos indican la secuencia u orden en que deben ejecutarse las instrucciones, es decir el flujo o recorrido que ha de seguirse en el diagrama.



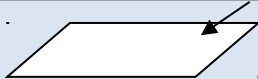
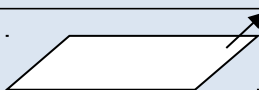
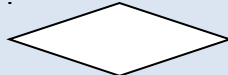


- Para las acciones de lectura y escritura emplearemos un paralelogramo con una pequeña flecha que apunta hacia adentro o hacia afuera del bloque respectivamente.
- Para la acción de asignación usaremos un rectángulo.

Cada bloque se corresponde con una única acción.

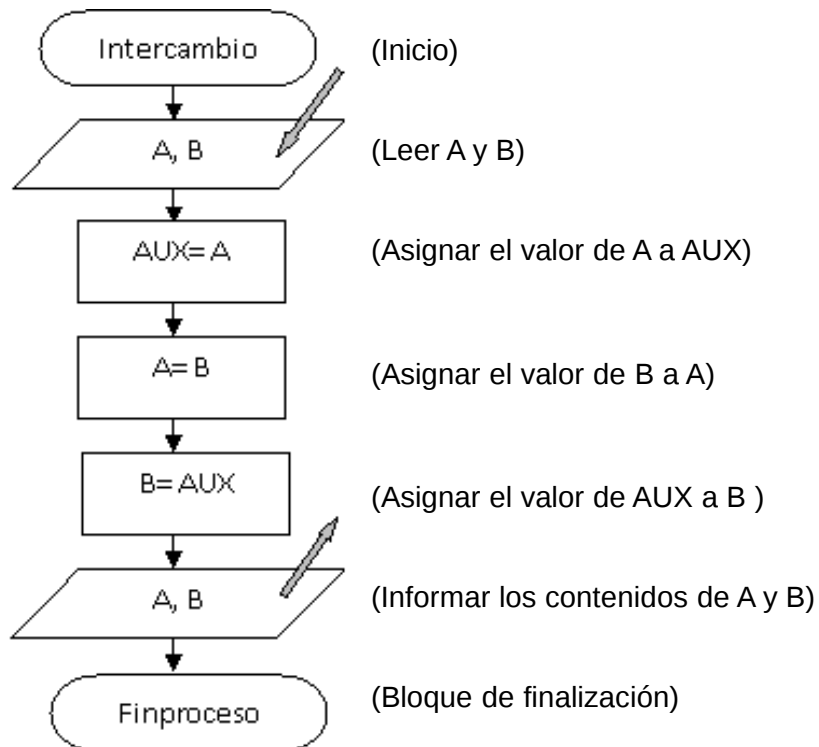
Una de las ventajas del empleo de diagramas de flujo, es la visualización plana de las acciones que forman el algoritmo, permitiendo seguir fácilmente su lógica.

Estas ventajas se apreciarán más adelante, cuando desarrollemos primitivas de estructura condicional y repetitiva.

Veamos ahora algunas de las formas geométricas que usaremos como bloques que identifican acciones en un diagrama de flujo:

Bloque	Significado o Uso
	Inicio o fin de proceso
	Asignación
	Lectura o ingreso de datos
	Escritura o salida
	Expresión lógica con una salida por V y otra por F
	Subalgoritmo
	Conectores

El Algoritmo del ejemplo anterior, si se utiliza diagrama de flujo, quedaría plasmado de la siguiente manera:



CONCLUSIÓN

Los conceptos tratados hasta aquí, contribuyen a formalizar la metodología de la definición de algoritmos. Quien los diseñe dispone ahora de reglas claras y formales, eliminando acciones ambiguas o carentes de precisión.

Esto no quiere decir que, planteado un problema, el algoritmo que describe la solución del mismo sea único. Al contrario, el diseño de los algoritmos requiere una gran dosis de creatividad y, es común, hallar varios caminos para la obtención de un resultado. Sólo se trata de establecer pautas claras y precisas para que distintos ejecutantes que interpreten dichas pautas, puedan realizar la secuencia de acciones que conforman el algoritmo.