UNL FICH Facultad de Ingeniería y Ciencias Hídricas

Universidad Nacional del Litoral
Facultad de Ingeniería y Ciencias Hídricas
Departamento de Informática

Ingeniería Informática

FUNDAMENTOS DE PROGRAMACIÓN

UNIDAD 2

Algoritmos Computacionales

2018

UNIDAD 2

Algoritmos Computacionales

Introducción

Los problemas y algoritmos desarrollados en la unidad anterior, reflejan situaciones de la vida diaria. En la mayoría de los casos fueron planteados para que los resuelva o ejecute una persona.
Pero el objetivo es desarrollar algoritmos que pueden ser interpretados por una computadora. Para ello es necesario utilizar un lenguaje que interprete una computadora y que permita una descripción precisa de cada una de las acciones a emplear en la solución del problema.

Esta unidad propone describir la formalización necesaria para el desarrollo de Algoritmos Computacionales empleando un pseudolenguaje similar a los empleados en la confección de programas.
De aquí en más, cuando se mencione al ejecutante de un algoritmo, se estará haciendo referencia a una computadora.

En el ejemplo que se propone a continuación se desarrolla un algoritmo completo de acuerdo a la formalización que se propone.

Ejemplo

Problema**:**

Plantear un algoritmo computacional que calcule la hipotenusa de un triángulo rectángulo. Se conoce como información de entrada las longitudes de
los catetos.

Análisis del Problema:

Datos: Longitudes de los catetos.
Resultado a informar: Hipotenusa.
Relaciones entre datos y resultados: Teorema de Pitágoras.

Algoritmo:

```
Algoritmo Hipotenusa
    Leer A, B
    H ← RC(A^2+B^2)
    Escribir 'Hipotenusa:', H
FinAlgoritmo
```

En el ejemplo **A**, **B** y **H** constituyen identificadores de variables; **2** e **'Hipotenusa:'** son constantes; **Leer**, **Escribir** y ← son las acciones primitivas de la lectura,

escritura y asignación respectivamente; **RC(A^2 + B^2)** es una expresión numérica y **RC(...)** es la función cuadrada. En esta unidad se desarrollarán todos estos elementos que conforman un lenguaje algorítmico formal que llamado pseudocódigo.

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

```
Algoritmo nombre
    acción 1
    acción 2
    ...
    ...
    ...
FinAlgoritmo
```

Elementos de un algoritmo computacional

Constante

Se define constante como a un valor que no puede alterarse en el transcurso de un algoritmo.

Ejemplos de constantes:

```
123          'López'    Falso            3.1459
```

Esta información se expresa mediante un valor intrínseco y único que no puede alterarse. Se utilizarán con frecuencia estos datos para asignarlos a variables o construir expresiones.

Variable

Una variable es una posición de memoria capaz de almacenar un único valor por vez. A medida que se ejecuten las acciones que describe el algoritmo esa variable podrá representar a nuevos valores (cada vez que toma un nuevo valor, éste reemplaza al viejo, el cual se pierde). En un algoritmo, se referencia a una variable a través de nombres o identificadores.

En el ejemplo inicial del cálculo de la hipotenusa se observa que **2** e **'Hipotenusa ='** constituyen constantes; mientras que **A**, **B** y **H** son variables.

Nombres o identificadores

Para proponer el nombre o identificador de algún elemento del algoritmo -como las variables- el diseñador tiene amplia libertad y solo debe respetar tres reglas simples:

1. Utilizar sólo letras y/o dígitos y/o guión bajo, comenzando siempre con una letra.
2. No utilizar palabras claves para acciones primitivas que emplea el pseudocódigo: LEER, ESCRIBIR, MIENTRAS, HACER, SEGUN, etc., o para las funciones internas: RC, SEN, TRUNC, LN, etc., o las palabras que corresponden a las constantes lógicas VERDAD y FALSO
3. No hacer distinción entre mayúsculas y minúsculas. Esto implica que

VENTA, venta y Venta, constituyen en el algoritmo el mismo nombre.

Esta sintaxis y sus restricciones no representan inconvenientes para proponer nombres de cualquier elemento del algoritmo: variables, algoritmos, procedimientos, funciones, archivos; pues se dispone de un sinnúmero de combinaciones diferentes de letras y dígitos. La finalidad de las restricciones es evitar ambigüedad en la interpretación (recordar la caracterísitica de “presición” que debe tener un algoritmo).

Ejemplos de identificadores válidos:

```
Venta       x12    resultado  SUMA2 M3M
```

Se mencionó que el diseñador del algoritmo tiene total libertad para proponer nombres a sus elementos, aunque como consejo, es conveniente proponer identificadores que tengan alguna relación con lo que el elemento representa.

En nuestro ejemplo inicial del cálculo de la hipotenusa de un triángulo rectángulo, se podría haber empleado los nombres de variables **CATETO1**, **CATETO2**, **HIPOT**, en lugar **A**, **B** y **H** para mayor claridad. Si bien en este caso con identificar con **H** a la hipotenusa, la relación puede parecer obvia, a medida que los ejemplos se tornan más complejos y en el algoritmo intervienen más variables, esto se hace cada vez más importante.

Expresión

Se define como expresión a un conjunto de operandos (ej: variables, constantes, resultados de funciones) ligados correctamente por operadores (ej: **+**, **-**, ***, **/**, etc) cuya evaluación arroja un resultado.

En el ejemplo inicial **RC(A^2 + B^2)** es una expresión numérica (derivada de la relación de Pitágoras) que permite calcular el valor de la hipotenusa.

Ejemplos de expresiones:

```
2+a-x*5          45    A<B         TRUNC(R)+1
```

Tipos de Información

Se puede clasificar la información que puede manejar una computadora en un algoritmo en pseudocódigo en los siguientes tipos:

- NUMÉRICO
- CARÁCTER (también denominado TEXTO o CADENA)
- LÓGICO

Esta clasificación nos define los tipos primitivos de la información. Cada tipo se guarda de forma diferente en la memoria, y permite realizar diferentes operaciones (por ejemplo, se pueden multiplicar números, pero no textos o caracteres). Se estudiará a cada uno de ellos y su modo de empleo.

Tipo Numérico

Constantes Numéricas
Los valores o constantes de tipo numérico se decimal y pueden estar precedidos por los signos '+ o '-'. La ausencia de signo implica un número positivo. Se ueden subdividir en reales y enteros, o admitir clasificaciones más detalladas, de acuerdo al lenguaje de programación empleado. Para la formalización propuesta mediante el empleo de un pseudocódigo universal, no se harán distinciones de esta clase y simplemente se hablará de tipo numérico.
Un detalle importante: los números reales deben separar su parte entera de la fracción decimal con un punto en lugar de la coma. Se reserva la coma como separador. No se utilizará ningún símbolo para separación de miles.

Ejemplos de constantes numéricas:

```
0   14    -12500    +1.9452    -567.91    1.3E+2    -0.94E-3
```

En los 2 últimos ejemplos se indican constantes numéricas con notación científica. Donde la cifra a la derecha de la E indica el exponente de la potencia de base 10. Es decir: 1.3 E+2 = $1.3*10^2$ = 130

Variables Numéricas
Una posición de memoria, que contenga cualquier valor de tipo numérico se denomina Variable Numérica. Las variables se identifican en un algoritmo a través de nombres o identificadores.

Expresiones Numéricas
Las expresiones numéricas se plantean en general con constantes numéricas, variables numéricas y funciones; y los operadores algebraicos como nexo:

Operadores algebraicos

- **+** suma
- **-** resta
- ***** multiplicación
- **/** división
- **^** potenciación
- **%** resto de la división entera (también puede usarse la palabra clave **MOD**)

Es de mencionar que ciertos lenguajes de programación no disponen de todos los operadores antes descritos, así como también es posible hallar otros operadores algebraicos adicionales a los presentados.
La jerarquía de estos operadores es idéntica a la planteada por el álgebra de números y sólo puede ser alterada a través de la intercalación de niveles de paréntesis.

Ejemplos de expresiones numéricas

```
2+a*102-800/C    1-(2*TOT-30 )(1+P)
AREA*(Y+1.34 /(X2-T ))    X mod 2
```

Obsérvese que el operador de radicación no existe; pero esto no es un problema

porque esta operación puede plantearse fácilmente a través de la potenciación de exponente fraccionario (aplicar la raíz n-ésima equivale a elevar a la potencia 1/n), o para el caso particular de la raíz cuadrada, la función predefinida **RC**.

Funciones Predefinidas
Se asume que el ejecutante del algoritmo, conoce y puede resolver ciertas funciones numéricas. A estas funciones se las llama funciones predefinidas y tienen la propiedad de devolver un valor o resultado, al ser aplicadas sobre un argumento que se indica entre paréntesis.

Algunas de las funciones predefinidas que incluye el pseudocódigo:

- RC(...) Raíz cuadrada
- ABS(...) Valor absoluto
- LN(...) Logaritmo natural
- EXP(...) Función exponencial
- SEN(...) Seno de un ángulo en radianes
- COS(...) Coseno de un ángulo en radianes
- ATAN(...) Arco Tangente del argumento
- TRUNC(...) Parte entera del argumento
- REDON(...) Entero más cercano
- AZAR(....) Entero aleatorio no-negativo

Con estos nuevos elementos, se puede ampliar el uso de expresiones numéricas.

Ejemplos de expresiones numéricas

```
TRUNC(2/3)-ABS(X)-2*(X-1)
SEN(X)+1-TAN(C/2)
(-b+RC(b*b-4*a*c))/(2*a)
```

Los lenguajes de programación suelen disponer de un número mucho mayor de funciones predefinidas.

Tipo Caracter

Constantes Tipo Caracter
Se incluyen aquí a todos los caracteres y símbolos del código ASCII (Código Standard Americano para Intercambio de Información), los cuales pueden obtenerse de las teclas o combinaciones de teclas de su computadora; es decir, las letras del alfabeto en minúsculas y las letras mayúsculas, los signos de puntuación, los operadores aritméticos, paréntesis, el espacio en blanco, caracteres especiales, etc. También se incluyen dentro de este tipo a las cadenas de caracteres, como los apellidos, nombres, direcciones y también cadenas de caracteres numéricos, y aún la cadena vacía (compueta por cero caracteres).

Ejemplos de constantes tipo carácter

```
'Luis Rodríguez'  'Z'   '25 de Mayo'    ''(cadena vacía)
 '3124'     'Resultado='            '123/890-12'
```

Importante: El conjunto de caracteres ASCII, es un conjunto ordenado, y por tanto existe una relación de precedencia u orden entre sus elementos. Cada elemento del conjunto tiene un número de orden en lo que se conoce como tabla o código ASCII, y ese número es el que determina la relación de orden.

El código ASCII es un standard mundial que permite compatibilizar la información que manejan las computadoras. La siguiente tabla muestra algunos de los caracteres del código ASCII y su correspondiente número de orden. En total son 128 caracteres en su versión original, 256 en su versión extendida (la más usual). A continuación se lista la parte más utilizada de dicha tabla:

032	(espacio)	048	0	064	@	080	P	096	`	112	p
033	!	049	1	065	A	081	Q	097	a	113	q
034	"	050	2	066	B	082	R	098	b	114	r
035	#	051	3	067	C	083	S	099	c	115	s
036	$	052	4	068	D	084	T	100	d	116	t
037	%	053	5	069	E	085	U	101	e	117	u
038	&	054	6	070	F	086	V	102	f	118	v
039	'	055	7	071	G	087	W	103	g	119	w
040	(	056	8	072	H	088	X	104	h	120	x
041	)	057	9	073	I	089	Y	105	i	121	y
042	*	058	:	074	J	090	Z	106	j	122	z
043	+	059	;	075	K	091	[	107	k	123	{
044	,	060	<	076	L	092	\	108	l	124	|
045	-	061	=	077	M	093	]	109	m	125	}
046	.	062	>	078	N	094	^	110	n	126	~
047	/	063	?	079	O	095	_	111	o		

Si se tiene en cuenta la relación de orden dada por el código ASCII, se puede afirmar que son verdaderas las expresiones siguientes:

- La letra 'a' es mayor que la letra 'B'.
  (‘a’ tienen número de orden 97 que es mayor al 66 de la ‘B’)
- El caracter '5' es menor que la letra 'A'.
  (el carácter ‘5’ tiene código 53 y el de la letra ‘A’ es 65)
- La cadena de caracteres 'Mario' es menor que ‘Roberto’.
  (se encuentra antes alfabéticamente)

Se observa que a las constantes tipo caracter o las cadenas de caracteres se indican entre apóstrofos o comillas. Esto es para evitar confundir estos datos con identificadores de otros elementos del algoritmo. Por ejemplo, **'A'** es un dato tipo caracter y **A** un identificador de un elemento del algoritmo.

Variables Tipo Caracter
Una posición de memoria, que contenga cualquier dato de tipo caracter o cadena de caracteres se denomina Variable Tipo Caracter.

Expresiones Tipo Caracter
En el pseudolenguaje que se empleará en esta primer parte de Fundamentos de Programación no se utilizarán funciones para el manejo de caracteres y cadenas de caracteres, ni operadores cuyos resultados sean cadenas de caracteres. Por

esto, las expresiones de este tipo quedan reducidas a simples constantes o variables.

Tipo Lógico

Constantes de Tipo Lógico
Dentro de este tipo se incluye a solo dos constantes o valores posibles: **VERDADERO** y **FALSO**. Esto implica tener en cuenta una nueva limitación al proponer identificadores: no pueden utilizarse los nombres **VERDADERO** y **FALSO** para evitar ambigüedades.

Variables Lógicas
Una posición de memoria, que contenga cualquier dato de tipo lógico es una Variable Lógica.

Expresiones Lógicas
Aquí cobran mucha importancia una serie de operadores que nos permiten plantear expresiones de tipo lógico. Las expresiones más simples son las relacionales que utilizan los operadores relacionales matemáticos para comparar operandos de igual tipo.

Operadores relacionales

- > Mayor que
- < Menor que
- = Igual que
- <= Menor o igual que
- >= Mayor o igual que
- <> Distinto que

Ejemplos de expresiones lógicas relacionales

```
23<45        A+2>=100          'ANA'<'LUIS'
(2+B)>=Trunc(2+B)       'a'<>LETRA
```

Las expresiones lógicas simples mostradas en el ejemplo se conocen como expresiones relacionales, pues permiten comparar o relacionar a 2 operandos del mismo tipo.
Recordar que para el tipo Cadena de Caracteres, la comparación se hace en orden lexicográfico (letra a letra según su orden en la tabla ASCII).

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

Operadores lógicos

- ∧ Y Conjunción
- ∨ Ó Disyunción
- ~ NO Negación

Para cada operador lógico, en papel se puede utilizar tanto el símbolo matemático (primera columna) como la palabra clave (segunda columna). Sin embargo, dado que los símbolos de Conjunción y Disyunción no están fácilmente accesibles en los

teclados, en la PC solemos utilizar versiones alternativas (en este caso las palabras clave **Y**, **O**, **NO**)

Dichos operadores son binarios (es decir, se usan entre dos operandos) en el caso de los dos primeros y unario (sólo un operando a su derecha) en el caso del último, y se definen de la siguiente forma:

Conjunción (verdadero cuando tanto A como B son ambos verdaderos):

a	b	a **Y** b
Verdadero	Verdadero	Verdadero
Verdadero	Falso	Falso
Falso	Verdadero	Falso
Falso	Falso	Falso

Disyunción (verdadero cuando al menos uno, A ó B, es verdaderos):

a	b	a **Ó** b
Verdadero	Verdadero	Verdadero
Verdadero	Falso	Verdadero
Falso	Verdadero	Verdadero
Falso	Falso	Falso

Negación (invierte el valor lógico):

a	**NO** a
Falso	Verdadero
Verdadero	Falso

Se observa entonces el valor de verdad de las expresiones lógicas propuestas en el ejemplo siguiente:

Expresión	Resultado
`(7<10) Y ('a'<'c')`	VERDADERO
`('A'<'a') Ó (12>19)`	VERDADERO
`( 37<18 )`	FALSO
`( SEN(x)<=1 )`	VERDADERO
`NO ( SEN(x)<=1 )`	FALSO
`('MARIA'<'MARTA')`	VERDADERO
`('MARIA'<'MARTA') Y (1>2)`	FALSO

Acciones Algorítmicas

Primitivas

En el lenguaje algorítmico (pseudocódigo), las primitivas se identifican con palabras claves o reservadas, y el símbolo de asignación. Se han empleado en el ejemplo

inicial las primitivas: **LEER**, **ESCRIBIR** y ← (asignar).

La ejecución de las acciones primitivas de un algoritmo suele ser diferente según el caso, lo cual lleva a plantear la siguiente clasificación:

- Secuenciales
- Estructuras de Control
  - Condicionales
  - Repetitivas

En la próxima unidad se desarrollarán las estructuras algorítmicas de control que implican el uso de primitivas de estructura condicional y de estructura repetitiva.

Se plantean a continuación, algunas acciones de estructura secuencial.

Acciones Primitivas de Estructura Secuencial

Asignación
Esta acción permite a una variable representar o memorizar cierto valor. Para describirla se utilizará la notación siguiente:

```
V ← E
```

Donde **V** es el nombre de la variable a la cual el ejecutante debe asignar el valor de la expresión **E**. El símbolo ← puede leerse como *"toma el valor"* (es decir: **V** toma el valor **E**).

Los tipos de **V** y **E** deben ser coincidentes, en caso contrario será causa de error.

Ejemplos de Asignaciones Numéricas:

```
A ← 43          ( A toma el valor 43 )
X ← A           ( X toma el valor contenido en A )
NUM ← 3*X/A+2   ( NUM toma el valor del resultado de hacer 3*X/A+2 )
```

Importante: Nótese que para poder realizar una asignación aritmética se debe evaluar primero la expresión de la derecha, por lo tanto es perfectamente válida la acción: N←N+1, que puede leerse como: tomar el valor actual de N, sumarle 1, y asignar ese resultado a la variable N. Por ejemplo, si antes de la acción, N contenía el valor 8, luego de dicha acción contendrá 9. Obsérvese que esta acción algorítmica no tiene nada que ver con los conceptos asimilados en matemáticas, donde N=N+1 es una expresión incompatible.

Del mismo modo se definen las asignaciones tipo caracter y tipo lógica.

Ejemplos de Asignaciones tipo caracter:

```
LETRA ← 'A'       ( LETRA toma el valor 'A' )
X3 ← A            ( X toma el valor contenido en A )
NOMBRE ← 'Jorge'  ( NOMBRE toma el valor de 'Jorge' )
```

Ejemplos de Asignaciones Lógicas:

```
M ← FALSO                ( M toma el valor FALSO )
CIERTO ← 34 <= 78        ( CIERTO toma el valor VERDADERO )
G ← (A<2) Y (C=10)       (G toma el valor lógico resultante de la expresión (A<2) Y (C=10) )
```

Entrada

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, nos lleva a definir una acción algorítmica primitiva de Lectura o Entrada. Se usará para ello la palabra clave **LEER** que permitirá al ejecutante identificar esta acción, seguida de la variable o lista de variables, que representan en el algoritmo la los valores que deben ser ingresados.

En un programa real, la entrada puede provenir de diferentes fuentes: información ingresada por el usuario mediante el teclado, mediante una selección con el ratón, leída desde un archivo en disco, recibida a través de la red, etc.

Ejemplos de entrada de datos

```
LEER Dato
LEER Nombre, Apellido, DNI
```

Esta acción tiene el mismo efecto que una asignación (en el sentido en que asigna/reemplaza el valor de una variable), solo que esta última utiliza valores del ambiente del algoritmo; en cambio la lectura asigna valores desde el exterior.

También esta acción contribuye a hacer a los algoritmos de uso general, pues permite incorporar información nueva para producir nuevos resultados. Sin esta acción, la ejecución de un algoritmo producirá siempre la misma respuesta.

Las acciones de lectura y asignación permiten **definir variables** en un algoritmo; y se las denomina **acciones destructivas**, dado que si las variables estaban previamente definidas (en la asignación, sólo para la que está a la izquierda del símbolo), se pierden (destruyen) los valores previos al reemplazarlos por los nuevos. En el ejemplo inicial para calcular la hipotenusa la expresión utilizada era: H ← RC(A2+B2); en ese caso se accede a los valores A y B en forma no destructiva (se usan sus valores para el cálculo sin modificarlos).

Salida

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

Ejemplos de salida:

```
ESCRIBIR Dat, Nombre
```

```
ESCRIBIR 23
ESCRIBIR 'Dato=', X
ESCRIBIR 'Resultado=', 3*X+(X-1)/2
```

Se destacan 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. En cambio, si se escriben resultados a través de variables el ejecutante realizará un acceso no destructivo a dichas variables, pues solo necesita conocer su contenido, para ejecutar la escritura, pero no modificarlo. 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 (ó I/O en inglés).

Representaciones Gráficas

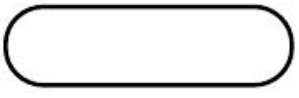
Las acciones descritas antes corresponden a un lenguaje algorítmico denominado pseudocódigo. El pseudocódigo es una de las formas que se puede emplear para representar algoritmos. Además, se diseñarán algoritmos en forma gráfica a través de los llamados diagramas de flujo.

En un diagrama de flujo, las estructuras de las primitivas del pseudocódigo se representan 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.

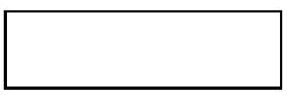
Por ejemplo, para las acciones de lectura y escritura se empleará un paralelogramo con una pequeña flecha que apunta hacia adentro o hacia afuera del bloque, respectivamente. Para la acción de asignación, se utilizará un rectángulo.

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 se describan primitivas de estructura condicional y repetitiva.

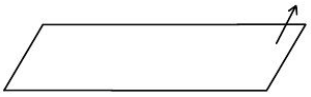
Se observan ahora, algunas de las formas geométricas que identifican acciones en un diagrama de flujo. En las próximas unidades se incorporarán nuevas acciones con su simbología correspondiente:



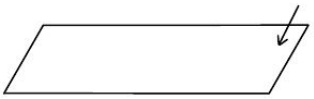
Inicio o fin de algoritmo



Asignación



Escritura o salida de información



Lectura o entrada de datos

Veamos un ejemplo de aplicación de Diagramas de Flujo en la resolución de un problema.

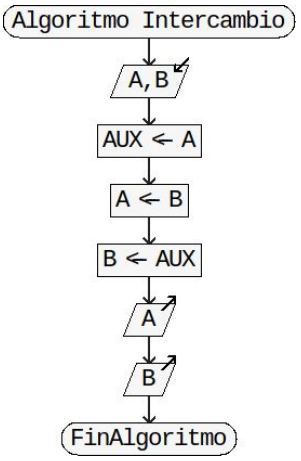
Problema:

Intercambiar los valores de 2 variables numéricas que se leen como datos.

Pseudocódigo

```
Algoritmo Intercambio

    Leer A, B

    AUX ← A
    A ← B
    B ← AUX

    Escribir A
    Escribir B

FinAlgoritmo
```

Diagrama de flujo



Prueba de escritorio o seguimiento de algoritmos

En las etapas de resolución de problemas se propuso la etapa de prueba luego de escribir la codificación (algoritmo). Probar un algoritmo es ejecutar cada una de las acciones incluidas en él; pero ¿cómo efectuar una prueba en un algoritmo escrito en pseudocódigo o mediante un diagrama de flujo?

La prueba de escritorio o seguimiento de un algoritmo puede efectuarse de la siguiente manera:

- Proponga un conjunto de datos. Estos datos deben coincidir en cantidad y tipo con las variables que aparecen en las acciones de lectura.
- Construya una tabla y coloque, encabezando cada columna, cada una de las variables que aparezcan en el algoritmo.
- Escriba SALIDA en el encabezado de la última columna de la tabla. Aquí se anotarán los resultados que produzca el algoritmo como consecuencia de la acción de Escribir .
- Comience a ejecutar las acciones del algoritmo. Cuando encuentre una asignación de una variable coloque el valor o dato a asignar en la columna correspondiente a esa variable.

- Si lee una variable, tome el dato de prueba propuesto para esa variable y colóquelo en la columna de esa variable.
- Si vuelve a asignar o a leer una variable ya creada, continúe anotando en la columna correspondiente.
- Al terminar de ejecutar las acciones, los resultados o salida del algoritmo deben aparecer en la columna de SALIDA.

Ejemplo

El siguiente algoritmo calcula el promedio de 3 números utilizando una sola variable para leer los datos de entrada.

```
Algoritmo Promedio
1   Sum ← 0
2   Leer X
3   Sum ← Sum+X
4   Leer X
5   Sum ← Sum+X
6   Leer X
7   Sum ← Sum+X
8   Prom ← Sum/3
9   Escribir 'Promedio=', Prom
FinAlgoritmo
```

Se realizará la prueba o seguimiento para los datos 15, 40 y 35. Cada acción del algoritmo se halla numerada para poder seguir en detalle la modificación de cada variable. Cada fila de la tabla se corresponde con una acción algorítmica.

	Sum	**X**	**Prom**	Salida
1	0			
2		15		
3	15			
4		40		
5	55			
6		35		
7	90			
8			30	
9				Promedio=30

Obsérvese en la fila 1, se ha asignado 0 a la variable sum. En la fila 2 la acción es Leer X y como el primer dato propuesto para la prueba es 15, se coloca 15 en la columna de la x. En el paso 3 se debe sumar sum+x, para lo cual se observa en la tabla que el valor actual de sum es cero y el de x es 15 con lo cual la operación arroja 15; luego, se asigna ese resultado a sum. Es decir que en el paso 3 la variable sum cambió tomando un nuevo valor (15) y perdiendo el anterior(0).
Así sucesivamente hasta llegar a la acción de salida en el paso 9, donde se refleja

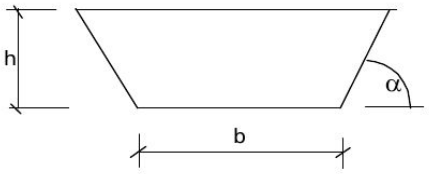
el resultado de dicha acción en la última columna de la derecha de la tabla.

Documentación

La documentación de programas es fundamental para diseñadores y usuarios. En pseudocódigo se podrán documentar los algoritmos internamente. Esto es, incluir dentro del algoritmo comentarios de ciertas acciones o grupos de acciones para permitir al diseñador o al equipo de diseño releer el algoritmo con facilidad. Esto evita a veces, tener que recordar por qué se definió tal variable, o por qué se utilizó tal expresión, etc.

Para documentar internamente un algoritmo en pseudocódigo se empleará la doble barra (//) y a continuación el texto o la frase explicativa. Al ejecutar el algoritmo, este texto a la derecha de la // debe ser ignorado pues no constituye una acción algorítmica. Obsérvese el ejemplo:

Ejemplo: Calcular el área del trapecio de la figura conociendo como datos la base menor b, el ángulo y la altura h.



Algoritmo:

```
Algoritmo Trapecio
    Leer b_menor, alfa, h
    // x será la diferencia entre
    // la base mayor y la base menor b
    x ← 2*(h/TAN(alfa))
    b_mayor ← b+x
    // cálculo del área del trapecio
    AreaT ← (b_mayor+b_menor)*h/2
    Escribir 'Area trapecio=', AreaT
FinAlgoritmo
```

Utilice comentarios con frecuencia, le evitará pérdidas de tiempo a la hora de recordar pasos, o depurar algoritmos.

Síntesis

- Para plantear algoritmos computacionales y dar solución a problemas diversos se requiere de un lenguaje algorítmico formal. Se empleará un pseudolenguaje de estructura y sintaxis similar al de los lenguajes de programación de computadoras: pseudocódigo.

- Esa formalización es necesaria para utilizar -más tarde- lenguajes de programación y construir programas que una computadora pueda interpretar.

- Para codificar un algoritmo computacional empleando pseudocódigo se debe comenzar con la palabra Algoritmo <Nombre>. A continuación se plantean las acciones que conforman el algoritmo y se finaliza con la palabra FinAlgoritmo.

- Los elementos del pseudocódigo presentes en un algoritmo son las constantes, variables, identificadores, expresiones y primitivas del lenguaje.

- Las variables son lugares de memoria que permiten almacenar información. Se las suele identificar con nombres (identificadores). Una de las formas de colocar un dato en una variable es a través de la asignación. Otra forma es con la acción de lectura. Por ejemplo: X←4.5; Leer z

- La información que maneja un algoritmo puede ser de naturaleza diferente, por ello se definen 3 tipos de datos: numérico, caracter y lógico. Para cada tipo se plantean operaciones diferentes y no se pueden mezclar en una misma expresión datos de distinto tipo.

- Dentro del tipo numérico se pueden emplear los operadores algebraicos de la matemática básica y se dispone de una serie de funciones predefinidas similares a las que Ud. puede hallar en una calculadora científica: SEN, COS, TRUNC, LN, etc.

- Las expresiones permiten combinar variables, constantes y operadores para realizar cálculos y comparaciones necesarios para resolver diversos casos.

- El tipo lógico tiene 2 constantes: VERDADERO y FALSO. En una expresión lógica se pueden usar operadores relacionales y operadores lógicos.

- El tipo caracter se basa en el conjunto de caracteres del código ASCII. Cada caracter tiene un número de orden y gracias a ese orden es posible establecer relaciones de orden: menor que, igual que, mayor que.

- Las acciones primitivas son órdenes perfectamente definidas y preestablecidas para que el ejecutante (la computadora) las interprete y las lleve a cabo.

- Las acciones primitivas de estructura secuencial, se denominan así porque se ejecutan una tras otra secuencialmente. Son ellas: la lectura o ingreso de datos, la asignación, la escritura o salida de información.

- Estas acciones se identifican unívocamente a través de palabras claves. Estas palabras son reservadas y no pueden emplearse como identificadores de otros elementos del algoritmo.

- Los algoritmos computacionales también pueden representarse gráficamente a través de diagramas de flujo. En estos diagramas los bloques o formas gráficas representan acciones. El diseño en 2

dimensiones permite seguir con más claridad la lógica propuesta.

- Es posible probar algoritmos mediante un seguimiento del mismo. Esto es, proponer una lista de datos que coincida con la cantidad y tipo de variables a leer. Luego ejecute cada acción y anote en una tabla de variables como estas se van modificando. Cada acción de escribir con el estado actual de las variables me permite determinar la salida del algoritmo.

- La documentación interna de algoritmos puede hacerse en pseudocódigo a través de la doble barra (//). Utilice estos comentarios para describir la lógica aplicada a ciertas acciones, o para recordar por qué efectuó cierto cálculo, etc. Estos comentarios son útiles a la hora depurar (corregir) el código o interpretar la lógica del algoritmo más ágilmente.

Por último: Los conceptos tratados hasta aquí, contribuyen a formalizar la metodología del diseño de algoritmos computacionales a través de ciertas reglas formales, eliminando acciones ambiguas o carentes de precisión. Esto no quiere decir, que planteado un problema, el algoritmo que lo resuelva responda a una metodología única. 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. Solo se trata de establecer pautas claras y precisas para que el ejecutante (luego, la computadora) las interprete y pueda procesar la secuencia de acciones que conforman un algoritmo computacional.

Actividades

Como se dijo anteriormente, haremos uso de un pseudo-lenguaje con las especificaciones presentadas anteriormente para que el ejecutante pueda ser una computadora Si bien se trata de un lenguaje muy limitado, podremos representar algoritmos sencillos de forma detallada y a la vez similar en muchos aspectos a un posible al lenguaje de programación real.

El alumno podrá utilizar el software PSeint (http://pseint.sourceforge.net), que efectivamente interpreta cada orden escrita según este pseudolenguaje, y permite entonces realizar una simulación de la ejecución de un algoritmo para mostrarnos su resultado. Se recomienda el uso de la herramienta, tanto para validar que las soluciones planteadas arrojen los resultados correctos, como también para explorar alternativas o experimentar con el funcionamiento de los diferentes elementos y estructuras de los algoritmos computacionales que iremos aprendiendo durante de este curso.