TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE NUEVO LAREDO

"Con la Ciencia por la Humanidad"

Introducción a la Ingeniería en Sistemas Computacionales y al Diseño Orientado a Objetos

Curso propedéutico

Instructor:

Bruno López Takeyas

bruno.lt@nlaredo.tecnm.mx

5.- Expresiones aritméticas

- **5.1 Operadores aritméticos**
- **5.2 Representación de expresiones aritméticas en forma algorítmica**
- **5.3 Reglas de prioridad de los operadores aritméticos**
- **5.4 Representación de expresiones matemáticas en forma algorítmica**
- **5.5 Evaluación de expresiones aritméticas**
- **5.6 Contadores**
- **5.7 Acumuladores**

Una parte importante del procesamiento de datos de parte de un sistema computacional es la ejecución de cálculos matemáticos representados mediante expresiones aritméticas. Una expresión aritmética es la representación algorítmica que puede contener variables, constantes, operadores aritméticos, paréntesis o nombres de funciones, cuyo resultado es un valor numérico.

Operadores aritméticos

Operación aritmética	*Símbolo del operador*
Suma	`+`
Resta	`-`
Multiplicación	`*`
División	`/`
Exponenciación	`**`, `^`, ↑
División entera	`DIV`
Módulo (residuo)	`MOD`

Operadores aritméticos.

Representación de expresiones aritméticas en forma algorítmica

Cuando se desea que una computadora realice cálculos aritméticos, éstos deben plantearse mediante expresiones aritméticas que cumplan con ciertas reglas de representación. La siguiente tabla muestra algunos ejemplos de representaciones algorítmicas de expresiones aritméticas (Fig. 6.2).

Operación matemática	*Representación algorítmica*
5 + 4	5 + 4
3 – 6	3 – 6
3 x 2	3 * 2
$\frac{10}{5}$	10 / 5
2^3	2 ** 3 ó 2^3 ó 2↑3

Fig. 6.2. Ejemplos de representación de expresiones aritméticas.

Representación de expresiones aritméticas en forma algorítmica

El resultado de una expresión aritmética debe almacenarse en una variable que corresponda con el tipo de dato del resultado esperado. La variable que recibe dicho valor siempre debe colocarse en el extremo izquierdo de la expresión y los operandos y operadores aritméticos a la derecha del símbolo de asignación (=) (Fig. 6.3).

Expresión algorítmica	*Resultado*
x = 12.3 / 4.2	x = 2.928571428571429
A = 4 / 2	A = 2
R = 5 ^ 2	R = 25
W = 3.2 + 4	W = 7.2
Q = 12 * 3	Q = 36

Fig.6.3. Ejemplos de resultados de expresiones aritméticas

Suma (+)

Esta operación realiza la adición de operandos y se representa con el operador +. Cuando se realiza una suma, el tipo de dato del resultado depende de los operandos involucrados. Por ejemplo, si se suman dos datos de tipo entero, entonces el resultado es de tipo entero. Cuando se suman dos datos de tipo real, entonces el resultado también es de tipo real; sin embargo, este operador permite realizar sumas entre un dato entero y un dato real, lo cual provoca que el resultado sea de tipo real (Fig. 6.4).

Expresión algorítmica de la suma	*Resultado*	*Tipo de dato del resultado*
X = 2 + 3	X = 5	Entero
Y = 3.2 + 5.7	Y = 8.9	Real
W = 3 + 2.1	W = 5.1	Real

Fig.6.4. Ejemplos de sumas.

Resta (-)

Esta operación realiza la sustracción de operandos y se representa con el operador -. Cuando se realiza una resta, el tipo de dato del resultado depende de los operandos involucrados. Por ejemplo, si se restan dos datos de tipo entero, entonces el resultado es de tipo entero. Cuando se restan dos datos de tipo real, entonces el resultado también es de tipo real; sin embargo, este operador permite realizar restas entre un dato entero y un dato real, lo cual provoca que el resultado sea de tipo real (Fig. 6.5).

Expresión algorítmica de la resta	*Resultado*	*Tipo de dato del resultado*
X = 2 – 3	X = -1	Entero
Y = 3.2 - 5.7	Y = -2.5	Real
W = 3 - 2.1	W = 0.9	Real

Fig.6.5. Ejemplos de restas.

Multiplicación (*)

Esta operación realiza el producto de operandos y se representa con el operador *. Cuando se realiza el cálculo de un producto, el tipo de dato del resultado depende de los operandos involucrados. Por ejemplo, si se multiplican dos datos de tipo entero, entonces el resultado es de tipo entero. Cuando se multiplican dos datos de tipo real, entonces el resultado también es de tipo real; sin embargo, este operador permite realizar multiplicaciones entre un dato entero y un dato real, lo cual provoca que el resultado sea de tipo real (Fig. 6.6).

Expresión algorítmica de la multiplicación	*Resultado*	*Tipo de dato del resultado*
X = 2 * 3	X = 6	Entero
Y = 3.2 * 5.7	Y = 18.24	Real
W = 3 * 2.1	W = 6.3	Real

Fig.6.6. Ejemplos de multiplicaciones.

División (/)

La división es una operación aritmética que cuando se representa algorítmicamente debe tener especial atención, ya que en ella, el resultado puede verse afectado por los tipos de datos de los operandos involucrados. Por eso, se analiza la división con sus cuatro modalidades algorítmicas:

- ***División entera***
- ***División real***
- ***Cálculo del cociente (DIV)***
- ***Cálculo del residuo (MOD)***

En cualquiera de los casos debe asegurarse que el denominador o divisor no sea igual que cero, ya que las divisiones por cero no son computables.

División entera

Se conoce como división entera a la operación de la división que involucra únicamente operandos de tipo entero y cuyo resultado también es de tipo entero. Sin embargo, debe tomarse en cuenta que en algunas ocasiones, aunque los operandos sean enteros, el resultado es un dato numérico real (Fig. 6.7).

Expresión algorítmica de la división	*Resultado*	*Tipo de dato del resultado*
X = 6 / 2	X = 3	Entero
Y = 10 / 2	Y = 5	Entero
W = 3 / 2	W = 1	Entero

Fig.6.7. Ejemplos de divisiones enteras.

División entera (*cont.*)

Obsérvese el último ejemplo de la Fig. 6.7, donde el resultado obtenido mediante el operador de la división (/) es de tipo entero (a pesar de que 3 / 2 = 1.5). Esto se debe a que, en este caso, el operador de la división identifica que ambos operandos son de tipo entero y por lo tanto obtiene automáticamente un resultado de tipo entero.

Expresión algorítmica de la división	***Resultado***	***Tipo de dato del resultado***
X = 6 / 2	X = 3	Entero
Y = 10 / 2	Y = 5	Entero
W = 3 / 2	W = 1	Entero

Fig.6.7. Ejemplos de divisiones enteras.

División entera (*cont.*)

Cuando se desea obtener un resultado de tipo real aunque los operandos sean de tipo entero, entonces se debe prototipar real alguno de los operandos de la expresión (*cast* en inglés). Esto permite que el operando se considere internamente como si fuese de tipo real y se logra anteponiendo al operando correspondiente el tipo de dato deseado entre paréntesis (Fig. 6.8).

Expresión algorítmica de la división	***Resultado***	***Tipo de dato del resultado***
X = 3 / 2	X = 1	Entero
Y = (real) 3 / (real) 2	Y = 1.5	Real

Fig.6.8. Ejemplos de divisiones enteras con y sin prototipar.

División real

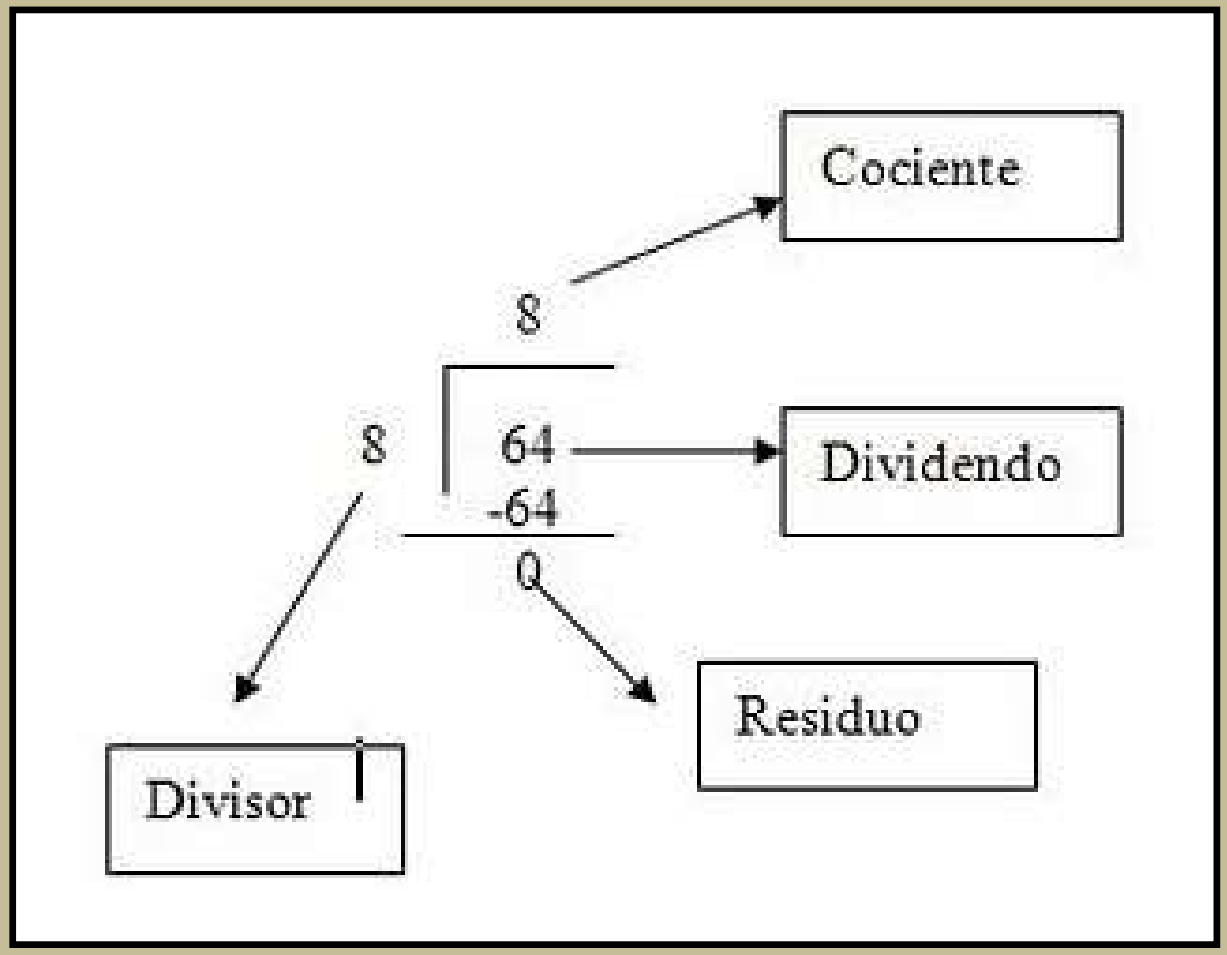
Se conoce como división real a la operación de la división que involucra operandos de tipo real y cuyo resultado también es de tipo real. En estos casos, basta que uno de los operandos sea de tipo real para que el resultado también sea real (Fig. 6.9).

Expresión algorítmica de la división	*Resultado*	*Tipo de dato del resultado*
X = 3.4 / 2.3	X = 1.47826086956521 7	Real
Y = 10.0 / 5.0	Y = 2.0	Real
W = 10.0 / 5	W = 2.0	Real
Z = 3.0 / 2.0	Z = 1.5	Real

Fig.6.9. Ejemplos de divisiones reales.

Cálculo del cociente (DIV)

El operador DIV funciona igual que la división entera y calcula el cociente de una división. Se le llama cociente al resultado entero de una división (Fig. 6.10).



El operador DIV

El operador DIV puede utilizarse para evitar confusiones con la división entera; sin embargo, tiene la restricción de que solamente permite operadores de tipo entero; es decir, ocurre un error cuando se intenta ejecutar el operador DIV con un operando de tipo real (Fig. 6.11).

Expresión algorítmica del operador DIV	*Resultado*	*Tipo de dato del resultado*
X = 6 DIV 2	X = 3	Entero
Y = 10 DIV 2	Y = 5	Entero
W = 3 DIV 2	W = 1	Entero

Fig.6.11. Ejemplos de cálculo del cociente con el operador DIV.

Cálculo del residuo (MOD)

El operador MOD se utiliza para calcular el residuo de una división entre operandos de tipo entero. Esto significa que este operador no permite que alguno de los operandos sea de tipo real (Fig. 6.12).

Expresión algorítmica del operador MOD	*Resultado*	*Tipo de dato del resultado*
X = 6 MOD 2	X = 0	Entero
Y = 11 MOD 2	Y = 1	Entero
W = 7 MOD 4	W = 3	Entero

Fig.6.12. Ejemplos de cálculo del residuo con el operador MOD.

Exponenciación (^, ** o ↑)

Esta operación sirve para obtener el resultado de elevar un número a una potencia y se representa con cualquiera de los siguientes operadores ^, ** o ↑ (Fig. 6.13).

Expresión algorítmica del operador de exponenciación	*Resultado*	*Tipo de dato del resultado*
X = 2 ^3	X = 8	Entero
Y = 3 ** 3	Y = 27	Entero
W = 4 ↑ 2	W = 16	Entero

Fig.6.13. Ejemplos de cálculo de exponenciaciones.

Reglas de prioridad de los operadores aritméticos

Una computadora realiza las operaciones aritméticas de dos en dos operandos, sin embargo, algunas expresiones pueden tener más operandos y operadores, en estos casos, es necesario aplicar ciertas reglas para determinar el orden de aplicación de los operadores. Estas reglas se conocen como las reglas de prioridad o jerarquía de las operaciones.

Cuando una expresión aritmética tiene paréntesis, entonces su contenido se evalúa primero; luego se evalúan las exponenciaciones, multiplicaciones y divisiones, operadores DIV y MOD y por último las sumas y restas. Si una expresión tiene varios operadores con la misma jerarquía, entonces se evalúan de izquierda a derecha (Fig- 6.14).

Reglas de prioridad de los operadores aritméticos

Orden de evaluación	*Operador*
1	( )
2	^, ** o ↑
3	*, /, DIV, MOD
5	+, -

Fig. 6.14. Jerarquía de los operadores aritméticos.

Representación de expresiones matemáticas en forma algorítmica

Muchos sistemas computacionales ofrecen resultados que son producto de la evaluación de expresiones matemáticas, las cuales deben representarse en forma algorítmica cumpliendo con ciertas reglas, tomando como referencia los operadores aritméticos y algunas funciones matemáticas.

Las expresiones matemáticas no sólo pueden contener operadores aritméticos, sino también pueden tener algunas funciones matemáticas (trigonométricas, logarítmicas o de cálculos específicos). La Fig. 6.15 muestra algunas de las funciones matemáticas internas más comunes.

Funciones matemáticas

Función	Descripción	Tipo de dato del argumento	Tipo de dato del resultado
sqrt(x)	Raíz cuadrada de x	Entero o real	Real
sqr(x)	Elevar al cuadrado x	Entero o real	Entero o real
sin(x)	Seno de x	Entero o real	Real
cos(x)	Coseno de x	Entero o real	Real
tan(x)	Tangente de x	Entero o real	Real
arcsin(x)	Arco seno de x	Entero o real	Real
arccos(x)	Arco coseno de x	Entero o real	Real
arctan(x)	Arco tangente de x	Entero o real	Real
log10(x)	Logaritmo base 10 de x	Entero o real	Real
ln(x)	Logaritmo neperiano de x	Entero o real	Real
abs(x)	Valor absoluto de x	Entero o real	Entero o real
round(x)	Redondeo de x	Real	Entero
trunc(x)	Truncamiento de x	Real	Entero

Fig.6.15. Funciones matemáticas más comunes.

Conversión de funciones matemáticas a expresiones algorítmicas

Para representar una función matemática mediante una expresión algorítmica es necesario utilizar tanto los operadores aritméticos como las funciones matemáticas internas. Algunas reglas generales son:

No se permite el uso de subíndices: Cuando esto se presente, entonces el subíndice se escribe del mismo tamaño que la variable.
Los símbolos de las funciones matemáticas deben sustituirse por las funciones matemáticas internas correspondientes (utilizando la notación algorítmica).
Debe respetarse la jerarquía de los operadores y agrupar con paréntesis cuando sea necesario.

La Fig. 6.17 muestra algunos ejemplos de funciones matemáticas con sus correspondientes expresiones algorítmicas. Entre ellas se destacan cálculo de cuadrados, raíz cuadrada, funciones trigonométricas, etc.

Función matemática	Expresión algorítmica
$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$	`d = sqrt((x2-x1)^2+(y2-y1)^2)`
$m = \frac{(y_2 - y_1)}{(x_2 - x_1)}$	`m = (y2-y1)/(x2-x1)`
$V = \frac{4\pi r^3}{3}$	`V = 4 * Pi * r ^3 / 3`
$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$	`x1 = (-b+sqrt(b^2-4*a*c))/(2*a)` `x2 = (-b-sqrt(b^2-4*a*c))/(2*a)`
$V_R = \sqrt{V_1^2 + V_2^2 - 2 \cdot V_1 \cdot V_2 \cdot Cos\,\beta}$	`VR=sqrt( V1^2 + V2^2 - (2 * V1 * V2 * cos(beta)))`
$\alpha = Cos^{-1} \frac{b^2 + c^2 - a^2}{2bc}$	`Alfa = arccos( sqr(b) + sqr(c) - sqr(a) /(2*b*c))`

Fig. 6.17. Funciones matemáticas y su representación algorítmica.

Evaluación de expresiones aritméticas

La evaluación de una expresión aritmética representada algorítmicamente para obtener su resultado debe hacerse respetando la jerarquía de los operadores (Fig. 6. 14). Las expresiones aritméticas algorítmicas no se evalúan según el orden de aparición de los operadores, sino guiados por las reglas de prioridad. Por ejemplo, si se evalúa la expresión x = 3 + 4 * 2, primero se obtiene el resultado de 4 * 2 (8) y luego se le suma el valor de 3, esto produce el valor x = 11, ya que el operador de la multiplicación tiene prioridad más alta que la suma.

Otro ejemplo, si se evalúa la expresión y = 7 + 2 – 3 * 4 / 2, primero se evalúa -3 * 4 (-12), luego ese resultado se divide por 2 (-6), después se le suma 7 (1) y por último se suma 2, dando como resultado y = 3.

Evaluación de expresiones aritméticas (*cont.*)

Cuando se evalúan expresiones aritméticas representadas algorítmicamente también deben tomarse en cuenta los tipos de datos de los operandos involucrados; por ejemplo, al evaluar la expresión a = 3 + 9 / 2, primero se obtiene el resultado de 9 / 2 (debido a que el operador de la multiplicación tiene mayor prioridad que la suma) y se obtiene el valor de 4 (recuerde que al dividir operandos enteros, entonces el resultado es entero); posteriormente se le suma el valor de 3, dando como resultado a = 7; en cambio, si la expresión fuese a = 3 + 9.0 / 2, entonces el resultado obtenido sería a = 7.5.

Contadores

En numerosas ocasiones se requiere que un sistema computacional cuente en una variable una cantidad de objetos o procesos; a este tipo de variables se les llama contadores. Típicamente los contadores utilizan los números naturales para llevar a cabo su conteo, esto significa que son declarados de tipo entero. Dicho conteo puede realizarse tanto en forma ascendente como en forma descendente.

Es sumamente importante que se asigne el valor inicial del contador antes de usarlo por primera vez.

Incrementos

Los incrementos son contadores ascendentes y se representan mediante expresiones de tipo x = x + 1, esto significa que al valor actual de la variable x se le suma 1 y el resultado se almacena en la misma variable x. Por ejemplo, si la variable x = 3, al evaluar x = x + 1, se suma 1 al valor de 3 y el resultado (4) se almacena en la misma variable, por lo que ahora x = 4.

Regularmente los incrementos se inicializan a cero la primera vez.

En algunos lenguajes de programación, los incrementos se pueden representar de diferentes formas (Fig. 6.18).

Incrementos

Incremento	*Expresión algorítmica*	*Representación en algunos lenguajes de programación*

i = i + 1	`i = i + 1`	`i = i + 1`
		`i++`
		`i+=1`

y = y + 1	`y = y + 1`	`y = y + 1`
		`y++`
		`y+=1`

Edad = Edad + 1	`Edad = Edad + 1`	`Edad = Edad + 1`
		`Edad++`
		`Edad+=1`

Fig. 6.18. Incrementos.

Decrementos

Los decrementos son contadores descendentes y se representan mediante expresiones de tipo x = x - 1, esto significa que al valor actual de la variable x se le resta 1 y el resultado se almacena en la misma variable x. Por ejemplo, si la variable x = 3, al evaluar x = x - 1, se resta 1 al valor de 3 y el resultado (2) se almacena en la misma variable, por lo que ahora x = 2.

En algunos lenguajes de programación, los decrementos se pueden representar de diferentes formas (Fig. 6.19).

Decrementos

Decremento	*Expresión algorítmica*	*Representación en algunos lenguajes de programación*

i = i - 1	`i = i - 1`	`i = i - 1`
		`i--`
		`i-=1`

y = y - 1	`y = y - 1`	`y = y - 1`
		`y--`
		`y-=1`

Edad = Edad - 1	`Edad = Edad - 1`	`Edad = Edad - 1`
		`Edad--`
		`Edad-=1`

Fig. 6.19. Decrementos.

Acumuladores

Los acumuladores son muy semejantes a los contadores, pero con las siguientes diferencias:
Un contador solamente puede tener operadores de suma o resta y el acumulador puede tener cualquier operador aritmético.
Un contador solamente puede aumentar o disminuir su valor de uno en uno, mientras que el acumulador puede modificar su valor de n en n.

La Fig. 6.20 muestra algunos ejemplos de acumuladores.

Acumuladores

Acumulador	*Expresión algorítmica*	*Representación en algunos lenguajes de programación*
i = i + 2	`i = i + 2`	`i = i + 2` `i+=2`
y = y – 3	`y = y - 3`	`y = y - 3` `y-=3`
a = a * 4	`a = a * 4`	`a = a * 4` `a*=4`
w = w / 2	`w = w / 2`	`w = w / 2` `w/=2`

Fig. 6.20. Acumuladores.

Prácticas

- Descargue del sitio web:

`https://nlaredo.tecnm.mx/takeyas/LibroISC`

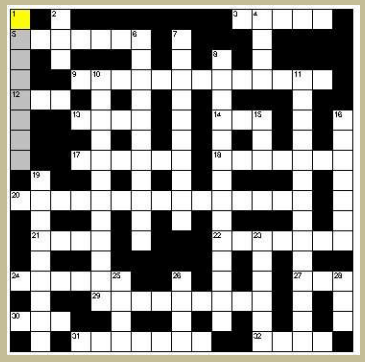
- *Práctica 5.1.- Notación algorítmica de expresiones y prioridad de los operadores aritméticos*

Tarea

Resuelva en el sitio web

https://nlaredo.tecnm.mx/takeyas/LibroISC

- ❑ **Crucigrama 5.1**



Fuentes de información:

- López Takeyas, Bruno. (2019) “Introducción a la Ingeniería en Sistemas Computacionales y al diseño orientado a objetos”. Editorial Pearson.

- https://nlaredo.tecnm.mx/takeyas/LibroISC/