



En complementos de aprendizaje:
Presentaciones - Videos - Ejercicios

Aprenda Practicando

2ª Edición

Introducción a la Programación

Algoritmos y su implementación en

VB.NET, C#, Java y C++

Felipe Ramírez

- Aprenda los elementos y técnicas de la lógica de programación
- Aprenda el proceso de análisis, abstracción y documentación de casos reales de negocios
- Desarrolle algoritmos, diagramas de flujo, miniespecificaciones y pruebas de escritorio
- Implemente algoritmos en Visual Basic.NET, C#, C++ Java y Raptor
- Aprenda los elementos de la programación orientada a objetos, usando Visual Basic

 **Alfaomega**

A continuación se describen los tipos de datos generales que soportan los lenguajes VB.NET, C#, C++ y Java, en las versiones tratadas en este libro; aunque los lenguajes poseen más tipos de datos que los relacionados, no se mencionan más que los que utilizaremos en los ejercicios del libro. Le dimos énfasis a los datos de tipo numérico, dado que los datos más complejos, como las fechas y las cadenas, tienen diversas formas de implementación que dificultarían el aprendizaje de los algoritmos, que es nuestro tema de interés.

Integer	4 bytes	-2.147.483.648 a 2.147.483.647
String	Clase (2 bytes por caracter, aprox)	Cadenas de caracteres.

C#

Estos son los tipos de datos básicos en C#

Tipo	Consumo en Bytes	Dominio de tipo
bool		True o False
byte	1 byte	0 a 255 (sin signo).
decimal	16 bytes	0 a +/- 79.228.162.514.264.337.593.543.950.335 sin separador decimal; 0 a +/-7,9228162514264337593543950335 con 28 posiciones a la derecha del signo decimal; el número más pequeño distinto de cero es +/-0,00000000000000000000000000000001 (+/-1E-28).
int	4 bytes	-2.147.483.648 a 2.147.483.647
String	Clase (2 bytes por caracter, aprox)	Cadenas de caracteres.

C++

Estos son los tipos de datos en C++

Tipo	Consumo en Bytes	Dominio de tipo
bool		True o False
short	2 bytes	-32,767 a 32,767
double	8 bytes	De 10^{-308} a 10^{308}

Contenido

Categorías de operadores	76
Operadores aritméticos.....	76
Operadores de asignación	78
Operadores comparativos.....	78
Operadores lógicos.....	79
Reglas de precedencia	80
Precedencia implícita.....	81
Precedencia posicional	82
Precedencia explícita	82
<i>Aplicación de operadores y sus reglas de precedencia</i>	<i>83</i>
<i>Representación de expresiones complejas operadores y elementos de lógica</i>	
<i>simbólica</i>	<i>85</i>
Mapa mental del capítulo	88
Terminología	89
Preguntas	89
Examen rápido	91

Lo que no es verdad es mentira.

Mentira $\leftarrow (\neg \text{Verdad})$

Los *operadores* son los elementos del lenguaje que nos permitirán asignar, calcular y comparar expresiones, dando lugar a lo que conocemos como procesamiento.

Todos los lenguajes poseen operadores, siendo las clasificaciones más importantes las siguientes: aritméticos, de asignación, de comparación y lógicos.

Categorías de operadores

Operadores aritméticos

Los operadores aritméticos forman parte de la educación básica de cualquier persona, por lo cual difícilmente podríamos agregar algo que no sepa con respecto a ellos y lo que representan. Si en su vida ya ha sumado, restado, multiplicado y dividido, en este libro no tenemos nada nuevo que explicarle.

No obstante lo obvio, para no ser omisos definiremos a los *operadores aritméticos* como los símbolos que producen, a partir de dos expresiones numéricas conocidas,

una nueva expresión numérica, como resultado de la aplicación de un cálculo aritmético entre aquellas.

La mayoría de los lenguajes disponen de los siguientes operadores aritméticos:

Operador	Nombre	Función y sintaxis
+	Suma	Suma, expresiones numéricas. Resultado = Expresión1 + Expresión2
-	Resta	Representa la diferencia entre dos números o especifica la condición negativa de uno. Resultado = Número1 - Número2
*	Multiplicación	Multiplifica dos números. Resultado = Número1 * Número2
/	División	Divide un número entre otro. Resultado = Número1 / Número2
^	Exponencial	Sirve para elevar un número a una determinada potencia; un número puede ser negativo si el exponente es un entero. Resultado = Número1 ^ Número2

La forma en que se resuelven este tipo de operadores es la siguiente:

$$\text{Resultado} = \text{Expresión1} \text{ Operador } \text{Expresión2}$$

Expresión1 y *Expresión2* serán siempre datos numéricos. *Resultado* será siempre numérico.

Se debe tomar en cuenta que *Resultado* puede no ser del mismo tipo que las expresiones, por ejemplo, la suma de dos enteros (2,000,000,000 y 2,000,000,000) puede no ser entero.

Operadores de asignación

El *operador de asignación* es el que permite asignar un valor a una variable. El operador clásico de asignación es el signo de igual (=).

Variable = *Valor*

Donde *Variable* es una variable y *Valor*, una expresión válida para el tipo de dato de *Variable*.

Desde el punto de vista de la lógica simbólica, la asignación se representa con el símbolo “←”, que indica que una expresión pasa a una variable. Nuestra sintaxis original quedaría de la siguiente forma.

Variable ← *Valor*

Operadores comparativos

Los *operadores comparativos* son los que permiten comparar expresiones, una en relación a la otra, proporcionando un valor de falso (False), verdadero (True), o nulo (Null), dependiendo si la comparación es una verdad o no.

La tabla que sigue describe los operadores comparativos clásicos:

Operador	Verdadero si	Falso si
< (Menor que)	<i>expresión1</i> < <i>expresión2</i>	<i>expresión1</i> >= <i>expresión2</i>
<= (Menor o igual que)	<i>expresión1</i> <= <i>expresión2</i>	<i>expresión1</i> > <i>expresión2</i>
> (Mayor que)	<i>expresión1</i> > <i>expresión2</i>	<i>expresión1</i> <= <i>expresión2</i>
>= (Mayor o igual que)	<i>expresión1</i> >= <i>expresión2</i>	<i>expresión1</i> < <i>expresión2</i>
= (Igual a)	<i>expresión1</i> = <i>expresión2</i>	<i>expresión1</i> <> <i>expresión2</i>
<> (Distinto de)	<i>expresión1</i> <> <i>expresión2</i>	<i>expresión1</i> = <i>expresión2</i>

El resultado de la comparación será Null si cualquiera de las expresiones comparadas es Null.

La forma en que se resuelven este tipo de operadores es la siguiente:

Resultado = *Expresión1* Operador *Expresión2*

Expresión1 y *Expresión2* pueden ser cualquier tipo de expresiones comparables entre sí. *Resultado* será siempre lógico (True, False), o nulo (Null).

Algunos ejemplos del uso de estos operadores son:

"A" = "B"	retorna False
1 > 0.5	retorna True
2 >= 2	retorna True
"S" < "s"	retorna True

En el caso de los datos de tipo cadena, se comparan en base al código que tienen asignado en la página de caracteres. Tome en cuenta que las letras mayúsculas, para efectos de comparación, tienen un valor en código ASCII menor a las minúsculas, y por tanto, son menores. Siendo así, la "A" es menor que la "Z", pero "z" es mayor a "A".

Operadores lógicos

Los *operadores lógicos* son aquellos que sirven para unir o negar condiciones, produciendo un valor lógico. Los operadores lógicos básicos son los siguientes.

Nombre	Operador	Comportamiento	Exp. Lógica simbólica
Negación	Not	Niega el resultado de una condición. Revierte el valor; si la condición que afecta es True producirá False, y viceversa.	¬
Disyunción	And	Cuando de entre dos condiciones, las dos deben ser True para que en su conjunto la expresión sea True.	∧
Conjunción	Or	Cuando de entre dos condiciones, al menos una debe ser True para que en su conjunto la expresión sea True.	∨

La forma en que se resuelven este tipo de operadores es la siguiente:

Resultado = [*Expresión1*] Operador *Expresión2*

Expresión1 y *Expresión2* son expresiones de tipo lógico. En el caso de Not, *Expresión1* no se debe poner, ya que el operador actúa sobre una sola expresión, que deberá ser lógica. *Resultado* será siempre lógico (True, False).

Cuando una expresión lógica se compone por una sola comparación, se dice que es una *expresión lógica simple*, debido a que sólo se resolverá en un tiempo (se resuelve la comparación y es todo); si involucra dos o más comparaciones, o la utilización de un operador lógico, se dice que es una *expresión lógica compuesta*, debido a que la expresión lógica deberá resolverse en dos o más tiempos (se resuelve la comparación, y luego se resuelve dentro del contexto en que se encuentra). Una expresión lógica compuesta siempre se compone de expresiones lógicas simples, afectadas por operadores lógicos que las obligan a resolverse como una sola expresión lógica.

Ejemplos:

\neg True	Es False, porque es lo contrario a True
\neg False	Es True, porque es lo contrario a False
\neg "A" = "B"	Es True, porque "A" = "B" es False, pero negado es True
"A" = "B" \wedge 1 > 0.5	Es False, porque no todas las condiciones son True.
"A" = "B" \vee 1 > 0.5	Es True, porque al menos una condición es True.

En caso de que tenga más de dos condiciones conjuntas, entra en operación lo que se conoce como *preferencia*, que consiste en determinar el orden en que las condiciones u operaciones se han de resolver.

Reglas de precedencia

Podemos definir a la *precedencia* como la característica de una expresión compuesta de resolverse con anterioridad a otras.

Existen los siguientes tipos de precedencia: implícita, posicional y explícita.

Precedencia implícita

La *precedencia implícita* es aquella inherente a los operadores y la categoría a la que pertenecen.

Precedencia implícita por categoría. La *precedencia implícita por categoría* se presenta cuando hay expresiones que contienen operadores de más de una categoría (aritméticos, comparativos y lógicos). Por regla general se resuelven antes las expresiones que involucran operadores aritméticos, a continuación se resuelven las expresiones que involucran operadores de comparación y por último se resuelven las expresiones que involucran operadores lógicos.

Ejemplo:

$8 > 9 \wedge 4 + 3 > 5$	Se resolverían primero las expresiones que involucran expresiones aritméticas.
$8 > 9 \wedge 7 > 5$	Luego se resolverían las expresiones que involucran expresiones comparativas.
$\text{False} \wedge \text{True}$	Y finalmente se resolvería la expresión que involucra operadores lógicos.
False	Expresión resuelta.

Precedencia implícita por operador. La *precedencia implícita por operador* es la que tiene un operador respecto a los operadores de su misma categoría.

En el caso de los operadores de comparación, todos tienen la misma precedencia implícita por operador.

Los operadores aritméticos se evalúan en el siguiente orden de prioridad.

Aritméticos
Exponenciación (^)
Negatividad (-)
Multiplicación y división (*, /)
Adición y sustracción (+, -)

Los operadores lógicos se evalúan en el siguiente orden de prioridad.

Lógicos
Not
And
Or

Ejemplo:

Imagine que quiere comprar dos productos, uno de 200 pesos y otro de 500 pesos, a los cuales habrá que aplicarles el 15% de impuestos. Vea cómo se resolvería la siguiente expresión compuesta.

$200 + 500 * 1 + 0.15$	Se resolverían primero las expresiones que involucran la multiplicación.
$200 + 500 + 0.15$	Luego se resolverán las sumas, en orden de izquierda a derecha.
$700 + 0.15$	
700.15	Expresión resuelta.

El ejemplo anterior demuestra cómo la precedencia de los operadores nos puede llevar a errores de cálculo difíciles de detectar. Lo que realmente queríamos era sumar el precio de los productos y multiplicarlos por 1.15, cosa que no sucedió.

Precedencia posicional

La *precedencia posicional* se presenta cuando se tienen varias expresiones que involucran operadores de misma precedencia implícita, y consisten en que se resolverán las expresiones de izquierda a derecha.

El ejemplo anterior ilustró la precedencia posicional, dado que al encontrarse puras sumas (operadores de igual precedencia) se resolvieron en orden de izquierda a derecha.

Precedencia explícita

La *precedencia explícita* es aquella que se provoca mediante el uso de paréntesis.

Aquello que se encierra en paréntesis en una expresión es obligado a resolverse sin respetar otras reglas de precedencia respecto a lo que está afuera de los paréntesis, sin embargo, lo que está entre paréntesis sigue sujeto a las reglas de precedencia implícitas y posicionales.

Veamos cómo podríamos resolver el ejemplo que ofreció resultados incorrectos.

Ejemplos:

Imagine que quiere comprar dos productos, uno de 200 pesos y otro de 500 pesos, a los cuales habrá que aplicarles el 15% de impuestos. Vea cómo se resolvería la siguiente expresión compuesta.

$(200 + 500) * (1 + 0.15)$	Se resolverían primero las expresiones que se encuentren entre paréntesis. Como son sumas, se resuelven atendiendo precedencia posicional.
$(700) * (1 + 0.15)$	
$(700) * (1.15)$	Finalmente queda la multiplicación que realmente pretendemos.
805	Expresión resuelta.

No olvide que por más larga que parezca una expresión, todos los operadores actúan sobre dos expresiones, y de dos en dos, hasta dejar un solo valor final.



Ejercicio 05.01

Aplicación de operadores y sus reglas de precedencia

Resuelva las expresiones, una por una, en orden de precedencia. Subraye en cada línea la expresión que ha de resolverse. Vea el ejemplo que se propone.

$$23 + \underline{25 * 25} + 2 / 10$$

$$23 + 625 + \underline{2 / 10}$$

$$\underline{23 + 625} + 0.2$$

$$\underline{648 + 0.2}$$

$$648.2$$

$$1.- \underline{48 + 25 * 10 / 4} + 5$$
