ALGORITMIA Y PROGRAMACIÓN

I. EXPRESIONES

Se entiende como expresiones, sentencias que involucran el uso de constantes, variables y operadores. Las expresiones se clasifican de acuerdo al tipo de datos y operadores que emplean, lo cual implica tambien el tipo de resultado que se obtiene en su desarrollo o evaluación.

OBSERVACIÓN:

- Se definen como ***constantes*** datos que no cambian su valor, por ejemplo un número (3 ó 5.34).
- Se definen como ***variables*** datos que pueden cambiar su valor, lo que generaría un cambio en el valor de la expresión que los involucra, a menuda se representan con letras.

Expresiones aritméticas

Una expresion aritmética es aquella que incluye variables numéricas (enteras o reales), constantes numéricas y operadores aritméticos.

Operadores aritméticos

+	Operador de suma
-	Operador de resta
*	Operador de multiplicación
/	Operador de división

Algunos lenguajes pueden emplear otros símbolos para representar las diferentes operaciones o pueden inlcuir nuevos operadores como el de modulo o residuo.

Prioridad: Un aspecto importante para revisar cuando se estudian los operadores aritméticos en cada lenguaje es la prioridad. La prioridad corresponde a las reglas que indican cuál operación debe realizarse primero en una expresión aritmética; aunque estas reglas pueden variar de un lenguaje a otro, en general la mayor prioridad la tienen los operadores de multiplicación y división, después seguirían los de suma y resta.

Cuando se encuentran dos o más operadores del mismo nivel de prioridad, las operaciones se resuelven de izquierda a derecha.

El uso de parentesis permite saltar las reglas de prioridad para obligar a que se desarrolle una operación primero, dado que las operaciones incluidas en parentesis tendrían la mayor prioridad en la expresión.

Ejemplo:

4 + 8 * 2 – 6 / 2	Para resolver esta expresión se realizaría primero la multiplicación 8*2 (=16), luego la división 6/2 (=3), luego la suma 4 + 16 (=20) y finalmente la resta 20 – 3, dando como resultado 17.
(4 + 8) * (2 – 6) / 2	En este caso se ha alterado la priodidad gracias al uso de parentesis, aquí se realiza primero la suma 4+ 8 (=12), luego la resta 2-6 (=-4), luego la multiplicación 12 * -4 (=-48) y finalmente la división -48 / 2 dando como resultado -24.

Como se puede observar en el ejemplo anterior, el uso de parentesis cambia completamente el resultado de una expresión, así que debe tenerse mucho cuidado al momento de escribir expresiones aritméticas para verificar que las prioridades harán que se tenga el resultado que realmente se espera.

Para pensar….

¿ La prioridad que emplean los lenguajes de programación, es la misma que emplean las calculadoras?

OBSERVE QUE: En los lenguajes de programación, se tiene muy en cuenta los tipos de datos empleados en las operaciones, aunque puede variar, en términos generales si se operan números enteros (sin parte decimal) el resultado será un número entero; pero si se operan números reales (que incluyen parte decimal) o una combinación entre reales y enteros, el resultado será un numero real.

Considerando lo anterior, el resultado de operaciones como la división puede variar, asi:

10 / 4 dará como resultado **2**, ya que los dos números son enteros
10.0 / 4.0 dará como resultado **2.5,** ya que los números son reales.

Expresiones Lógicas

Las expresiones lógicas pueden emplear datos de diferentes tipos (lógicos o numéricos), emplean operadores lógicos o relacionales y obtienen como resultado un valor lógico (verdadero o falso).

Operadores relacionales

Estos operadores son los que permiten establecer algún tipo de relación entre dos operandos, las relaciones comunes son las siguientes:

Relación	Operador
Igual	==
Diferente	!=
Mayor	>
Menor	<
Mayor o igual	>=
Menor o igual	<=

Los operandos (datos]) que se pueden emplear, en general corresponden a números, aunque algunos lenguajes permiten comparar otro tipo de datos como caracteres o cadenas.

El resultado de una opreación que involucra operadores relacionales será de tipo lógico (verdadero o falso)

Ejemplo:

3 > 4	tendrá como resultado Falso
5 < 20	tendrá como resultado verdadero
10 >= 10	Tendrá resultado Verdadero
3 < 4 < 10	**no es una expresión válida porque en algún momento quedaría algo como verdadero < 10, lo cual no puede resolverse.**

Operadores lógicos

Los operadores lógicos permiten enlazar expresiones lógicas a traves de las conexiones lógicas que se emplean comunmente. Los operadores lógicos comúnmente empleados son:

Conexión lógica	Operador	Símbolo en C#
Conjunción	AND	&&
Disyunción	OR	||
Negacion	NOT	!

Estos operadores trabajan siguiendo la lógica de las tablas de verdad que se presentan a continuación:

AND	Verdadero	Falso
Verdadero	Verdadero	Falso
Falso	Falso	Falso

OBSERVE QUE: para la conjunción (AND) el único caso en que puede ser verdadero, es que las dos partes de la expresión sean verdaderas, cualquier otra combinación sería falsa.

OR	Verdadero	Falso
Verdadero	Verdadero	Verdadero
Falso	Verdadero	Falso

OBSERVE QUE: para la disyunción (OR) el único caso en que puede ser falso, es que las dos partes de la expresión sean falsas, cualquier otra combinación sería verdadera.

El operador de Negación trabaja sobre un único operando, cambiando su valor de verdad (o valor lógico)

NOT	
Verdadero	Falso
Falso	Verdadero

Ahora bien, regresando al tema de las expresiones lógicas, estas podrían involucrar solamente operadores lógicos, operadores relacionales o combinación de ambos; dependiendo de lo que se quiera presentar.

Ejemplo:

Si se quiere verificar que la variable nota se encuentre en el rango [0.0, 5.0] la expresión correcta sería:

(Nota >= 0.0) && (nota <=5.0)

En este caso, se resuelven primero las expresiones en parentesis, de la siguiente forma.

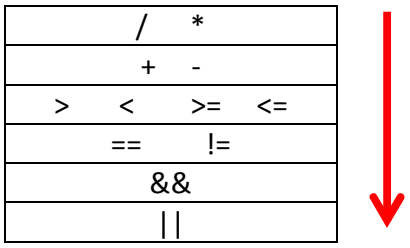
Si la variable Nota obtiene el valor de 2.6 tendríamos:

(2.6 >= 0.0) && (2.6 <=5.0)
Verdadero && verdadero
Verdadero

Si la variable Nota obtiene el valor de 6.2 tendríamos:

(6.2 >= 0.0) && (6.2 <=5.0)
Verdadero && Falso
Falso

Al combinar operadores lógicos y aritméticos, las prioridades establecidas varían, quedando de la siguiente forma:



Como siempre, los paréntesis permiten saltar los niveles de prioridad.

Ejemplo:

Para el siguiente caso:

3 + 4 < 10 && 56 – 3 > 100 || 4 * 6 -2 <=22

La solución, atendiendo las prioridades establecidas, quedaría de la siguiente forma:

1. Se resuelven divisiones y multiplicaciones:
   3 + 4 < 10 && 56 – 3 > 100 ||24 -2 <=22
2. Se resuelven sumas y restas (de izquierda a derecha):
   7 < 10 && 53 > 100 ||22 <=22
3. Se resuelven los operadores relacionales:
   Verdadero && Falso || Verdadero
4. Se resuelven las operaciones de conjunción (AND, &&)
   Falso || verdadero
5. Se resuelven las operaciones disynción (OR, ||)
   Verdadero

La solución de la expresión es: Verdadero.

Operaciones matemáticas

Algunas operaciones matemáticas comunes, no tienen representación en los lenguajes de programación como un operador, en lugar de ello debe emplearse un método que ya está incorporado en el lenguaje. En el caso de C#, muchas de estas operaciones se incorporan en la clase Math, entre las más comunes se encuentran las siguientes:

Nombre del método	**Función**
Cos(double x)	Retorna el coseno del ángulo especificado.
Exp (double x)	Retorna e elevado a la potencia especificada.
Log (double x)	Devuelve el logaritmo natural (en base e) de un número especificado.
Log10(double x)	Devuelve el logaritmo en base 10 de un número especificado.
Pow(double x, double y)	Retorna un número (x) elevado a la potencia especificada (y)
Round (decimal x)	Redondea un número decimal al entero más próximo.
Sin (double x)	Retorna el seno del ángulo especificado.
Sqrt(double x)	Retorna la raíz cuadrada del número indicado.
Tan(double x)	Retorna la tangente del ángulo especificado.

Para emplear cualquiera de estas funciones se debe anteponer *Math.* al nombre del método correspondiente, esto le especifica al lenguaje que el método que se desea usar esta en una clase llamada Math.

Ejemplo:

```
X = Math.sqrt(64);
```

Esta expresión dejaría en x el valor de 8 (raíz cuadrada de 64)

Estos métodos pueden ser involucrados dentro de una expresión.

Ejemplo:

```
58 – Math.sqrt(64) < 100
```

En este caso, se resuelve primero el método, luego la resta y finalmente la comparación; lo que daría como resultado Verdadero

Para pensar….

Algunos símbolos que comunmente empleamos para un propósito tienen un significado totalmente diferente en el lenguaje de programación, por ejemplo, el símbolo de circunflejo (^) corresponde a un operador particular en C# y no es el exponente como comúnmente lo usamos en otros campos.