



## ANÁLISIS Y DESARROLLO DE SOFTWARE

# LÓGICA PROPOSICIONAL

# OPERADORES Y EXPRESIONES ARITMÉTICAS

```
margin-top: -3px;
}
.label-default {
  background-color: #777;
}
.label {
  display: inline-block;
  width: 6em;
  height: 3em;
  padding: 2em 0 0 5%;
  font-size: 700;
  font-weight: 1;
  font-height: 150%;
  line-height: center;
  color: nowrap;
  text-align: baseline;
  white-space: nowrap;
  vertical-radius: 10px;
  border: 1px solid #777;
}
#size {
  font-size: 1em;
  font-family: sans-serif;
  border: 1px solid #777;
  width: 100px;
  height: 30px;
  margin-bottom: 10px;
}
label {
  font-size: 1em;
  font-family: sans-serif;
  border: 1px solid #777;
  width: 100px;
  height: 30px;
  margin-bottom: 10px;
}
```

### 3. Operadores y expresiones aritméticas.

#### 3.1. Los operadores aritméticos.

La mayoría de los programas de computador requieren realizar cálculos u operaciones que involucran operadores aritméticos, por esta razón, como programadores es necesario conocer cada uno de ellos y la manera como el computador los interpreta para calcular los resultados de una determinada expresión o fórmula.

¿Cuál es el resultado de la siguiente expresión?

$$3 + 5 \times 2 = \underline{\hspace{2cm}} ?$$

Explicación

$$5 \times 2 = 10$$

$$3 + 10 = 13$$

la respuesta correcta es **13**, pues bien, además de conocer los diferentes operadores aritméticos, también es importantísimo conocer los niveles de prioridad de cada uno de ellos.

En el caso de la expresión  **$3 + 5 \times 2$** , primero se realiza la multiplicación  **$5 \times 2$**  cuyo resultado es **10** y posteriormente se realiza la operación  **$3 + 10$** , dando como resultado final **13**.

#### 3.2. Reglas de prioridad en los operadores aritméticos.

Cuando dos operadores tienen el mismo nivel de prioridad, dentro de una expresión se evalúan de izquierda a derecha.

En LPP el signo igual (=), se representa mediante una flecha dirigida hacia la variable que recibe el valor, esta flecha está conformada por los caracteres menor que (<) y menos (-) así: <-

Prioridad	Operador	Significado	Ejemplo	
1	$\wedge$	Exponenciación	$4 \wedge 2 = 16$	$3 \wedge 3 = 27$
2	*	Multiplicación	$2 * 4 = 8$	$7 * 5 = 35$
	/	División	$5 / 2 = 2.5$	$6 / 3 = 2$
3	DIV	División entera	$5 \text{ DIV } 2 = 2$	$7 \text{ DIV } 4 = 1$
	MOD	Residuo de la División	$5 \text{ MOD } 2 = 1$	$8 \text{ MOD } 4 = 0$
4	+	Suma	$3 + 4 = 7$	$2 + 9 = 11$
	-	Resta	$8 - 5 = 3$	$7 - 6 = 1$

ejemplo, para representar la siguiente expresión:  $X = 3 + 5$   
En LPP sería:  $X <- 3 + 5$

### Programa de Ejemplo:

Ahora, después de conocer los operadores aritméticos y sus reglas de prioridad, puedes encontrar el resultado de la siguiente expresión:

$$2 + 2 \wedge 2 * 2 + 2 \text{ MOD } 2 = \underline{\hspace{2cm}} ?$$

$$2 + 2 \wedge 2 * 2 + 2 \text{ MOD } 2 = \underline{\hspace{2cm}} ?$$

$$2 + 2 \wedge 2 + 2 = \underline{\hspace{2cm}} ?$$

$$2 + 8 + 2 \text{ MOD } 2 = \underline{\hspace{2cm}} ?$$

$$2 + 8 + 0 = 10$$

### 3.3. Ejemplos de expresiones aritméticas.

Cuando el programador desea determinar un orden específico de ejecución en una expresión aritmética, puede emplear los paréntesis para agrupar, de esta manera, las operaciones que se encuentren dentro del paréntesis serán las primeras en ejecutarse. Retomando el ejemplo de la expresión:

$$3 + 5 \times 2 = 13 \quad \text{pero} \quad (3 + 5) \times 2 = 16$$

$$5 \times 2 = 10 \quad (3 + 5) = 8$$

$$3 + 10 = 13 \quad 8 \times 2 = 16$$

### 3.4. Ejemplo, manejando expresiones en un programa.

**Programa No. 3** Manejando expresiones: a un programador le solicitan realizar una aplicación que calcule la nota promedio de un alumno a partir de las 2 notas que tiene en una asignatura.

Durante el análisis, el programador toma un caso de prueba para descubrir cuál es el procedimiento que debe llevar a cabo. En el caso de prueba toma como la primera nota el valor de 4 y como segunda nota el valor de 3.

Nota 1	Nota 2	NotaPromedio
4	3	3.5

Durante el análisis, el programador identifica que debe sumar las dos notas y el resultado lo debe dividir entre dos:  $4 + 3 = 7$  luego  $7 / 2 = 3.5$

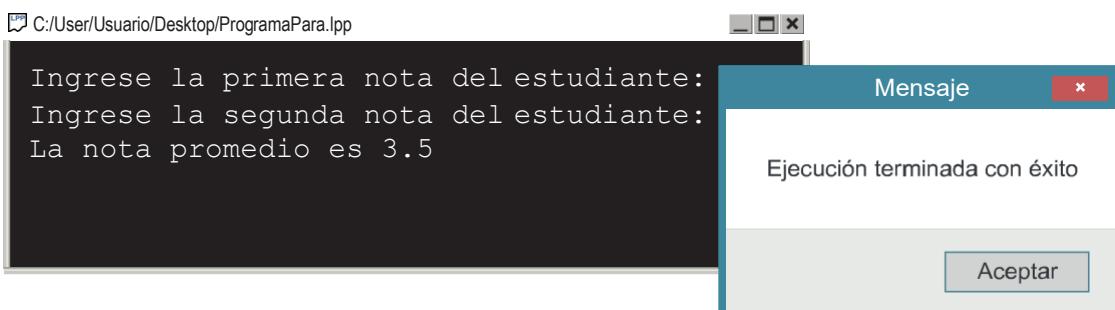
A partir de este análisis, el programador desarrolla la siguiente aplicación en LPP:

```

Real nota1, nota2, notaPromedio
Inicio
escriba "Ingrese la primera nota del estudiante:"
lea nota1
escriba "Ingrese la primera nota del estudiante:"
lea nota2
notaPromedio <-nota1 + nota2 / 2
escriba "la nota promedio es;" notaPromedio
Fin

```

Al ejecutar la aplicación ingresando los datos de prueba, el programador obtiene el siguiente resultado:



Después de buscar el error, se da cuenta que este se encuentra en la siguiente línea:

**Real** nota1, nota2, notaPromedio

```

Inicio
escriba "Ingrese la primera nota del estudiante:"
lea nota1
escriba "Ingrese la segunda nota del estudiante:"
lea nota2
notaPromedio <- (nota1 + nota2) / 2
escriba "La nota promedio es;" notaPromedio
Fin
```

Se debe recordar las reglas de prioridad y concluir que la primera operación que se está ejecutando es la división, por lo tanto, el programador realiza un ajuste al programa para definir el orden deseado de ejecución de los operadores aritméticos mediante el uso de paréntesis.

$$3 / 2 = 1.5 \text{ y}$$

$$4 + 1.5 = 5.5$$

Al ejecutar nuevamente la aplicación, obtiene el resultado esperado:

