

Sesión Sincrónica Nº 3




Fundamentos de Programación (PRY2201)

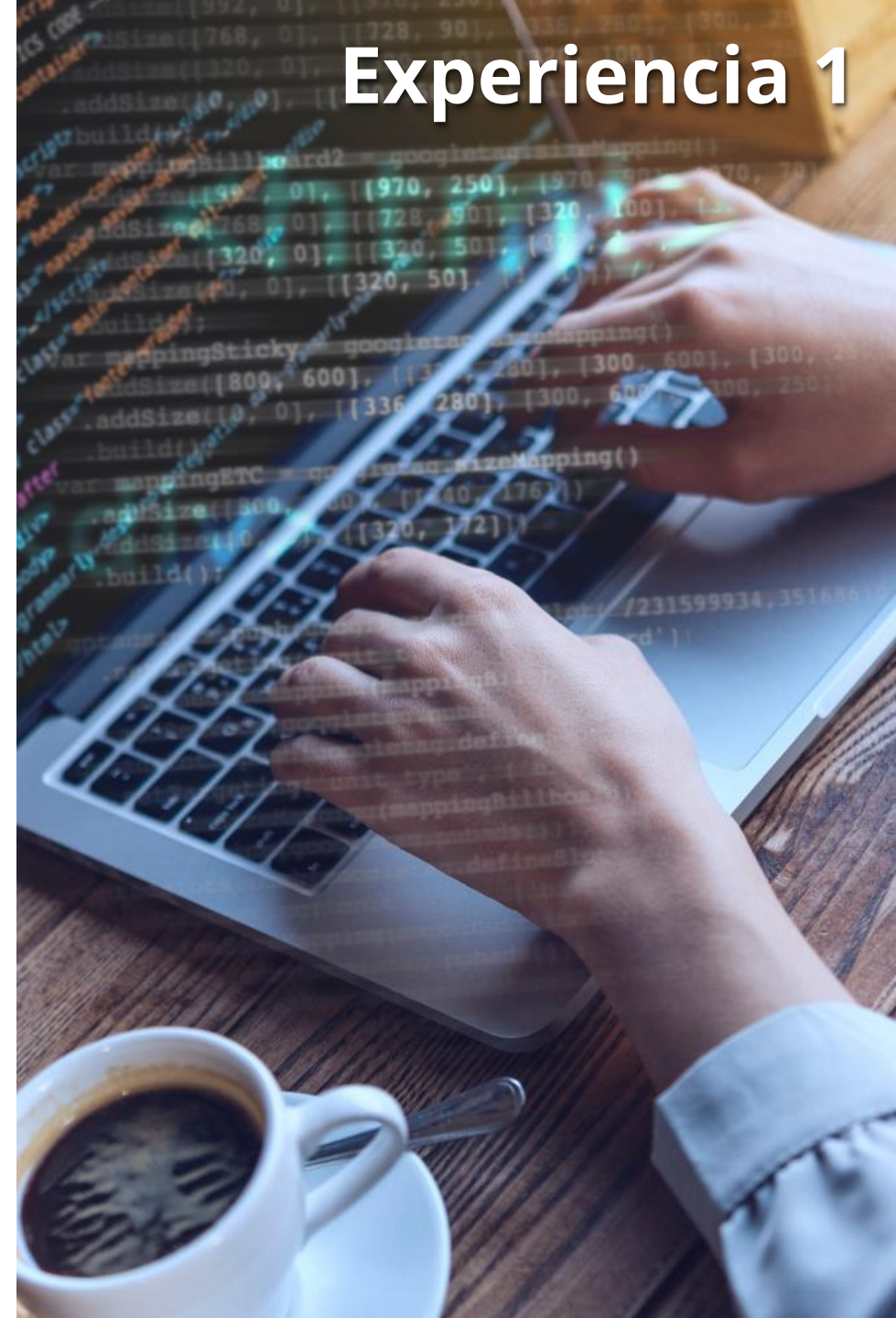
Profesor: Miguel Puebla

Experiencia 1: Creando Mi Primer Algoritmo

RA1. Utiliza estrategias de abstracción para la construcción de algoritmo, aplicando pseudocódigo con el objetivo de dar solución a problemáticas planteadas.

Semana 3: Problemas algorítmicos con expresiones aritméticas y lógicas

-  **IL1.** Identifica estrategias de abstracción y tipos de algoritmos aplicables en la construcción de algoritmos.
-  **IL2.** Aplica Pseudocódigo en la implementación de algoritmos, brindando una solución a las problemáticas planteadas.
-  **IL3.** Utiliza Variables, Expresiones Aritméticas y Lógicas en la Resolución de Problemas Algorítmicos, garantizando la exactitud y coherencia de sus soluciones.



Semana 3



Conocimientos Generales

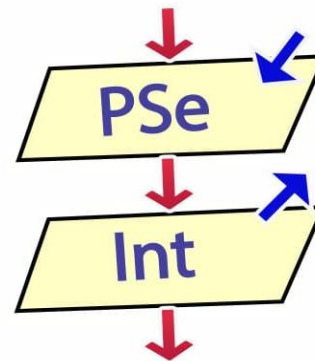
Operadores
aritméticos

Operadores
lógicos

Expresiones
aritméticas

Comentarios

Convenciones



Apache
NetBeans IDE



Operadores aritméticos

Los **operadores aritméticos** permiten realizar operaciones matemáticas básicas en programación, como **suma**, **resta**, **multiplicación** y **división**. Son esenciales para el desarrollo de cálculos y expresiones en algoritmos. Su uso varía ligeramente según el lenguaje, pero su función es común en todos los entornos de programación.

Operador	Descripción	Equivalente en PSeInt	Equivalente en Java	Características principales
+	Suma	+	+	Suma dos valores numéricos.
-	Resta	-	-	Resta el segundo valor del primero.
*	Multiplicación	*	*	Multiplica dos valores.
/	División	/	/	Divide el numerador entre el denominador.
%	Módulo (residuo división)	%	%	Retorna el residuo de una división.
++	Incremento (aumenta en 1)	No aplica	++	Aumenta el valor de una variable en una unidad.
--	Decremento (disminuye en 1)	No aplica	--	Disminuye el valor de una variable en una unidad.

Nota: En **PSeInt** no existen operadores de incremento o decremento como ++ o --, por lo que se usa:

`x <- x + 1` o `x <- x - 1`.

Operadores aritméticos

Ejemplo PSeInt

```
1  Proceso CalculadoraBasica
2      // Declarar variables
3      Definir num1, num2, suma, resta, multiplicacion, division Como Real;
4
5      // Solicitar entrada del usuario
6      Escribir "Ingrese el primer número: ";
7      Leer num1;
8      Escribir "Ingrese el segundo número: ";
9      Leer num2;
10
11     // Realizar operaciones aritméticas
12     suma ← num1 + num2;
13     resta ← num1 - num2;
14     multiplicacion ← num1 * num2;
15
16     // Verificar si el divisor es cero antes de realizar la división
17     Si num2 ≠ 0 Entonces
18         division ← num1 / num2;
19     Sino
20         Escribir "División por cero no permitida";
21     FinSi
22
23     // Mostrar resultados
24     Escribir "La suma es: ", suma;
25     Escribir "La resta es: ", resta;
26     Escribir "La multiplicación es: ", multiplicacion;
27     Escribir "La división es: ", division;
28 FinProceso
```


Operadores aritméticos



Ejemplo Java

```
public class OperadoresAritmeticos1 {  
    public static void main(String[] args) {  
        // Crear un objeto Scanner para leer la entrada del usuario  
        Scanner scanner = new Scanner(System.in);  
  
        // Declarar variables  
        double num1, num2;  
        double suma, resta, multiplicacion, division;  
  
        // Solicitar entrada del usuario  
        System.out.print("Ingrese el primer número: ");  
        num1 = scanner.nextDouble();  
  
        System.out.print("Ingrese el segundo número: ");  
        num2 = scanner.nextDouble();  
  
        // Realizar operaciones aritméticas  
        suma = num1 + num2;  
        resta = num1 - num2;  
        multiplicacion = num1 * num2;  
  
        // Verificar si el divisor es cero antes de realizar la división  
        if (num2 != 0) {  
            division = num1 / num2;  
        } else {  
            System.out.println("División por cero no permitida.");  
            division = Double.NaN; // No es un número (Not a Number)  
        }  
  
        // Mostrar resultados  
        System.out.println("La suma es: " + suma);  
        System.out.println("La resta es: " + resta);  
        System.out.println("La multiplicación es: " + multiplicacion);  
        if (!Double.isNaN(division)) {  
            System.out.println("La división es: " + division);  
        }  
  
        // Cerrar el scanner  
        scanner.close();  
    }  
}
```


Operadores lógicos



Los **operadores lógicos** se utilizan para combinar o evaluar expresiones **booleanas**, devolviendo resultados **verdaderos** o **falsos**. Son fundamentales para controlar el flujo de los algoritmos y tomar decisiones mediante **condiciones**. Su sintaxis puede variar según el lenguaje, pero su propósito lógico es universal en programación.

Operador	Descripción	Equivalente en PSeInt	Equivalente en Java	Características principales
&&	Conjunción lógica (AND)	Y	&&	Retorna verdadero si ambas condiciones son verdaderas.
	Disyunción lógica (OR)	O		Retorna verdadero si al menos una condición es verdadera.
!	Negación lógica (NOT)	NO	!	Invierte el valor lógico de una condición.

Ejemplo en PSeInt:

SI (edad > 17) Y (nacionalidad = "mexicana") Entonces ...

Ejemplo en Java:

FORMA 1: `if (edad > 17 && nacionalidad.equals("mexicana")) { ... }`

FORMA 2: `if (edad > 17 && nacionalidad == "mexicana") { ... }`

Operadores lógicos



Ejemplo PSeInt

```
1  Proceso VerificacionOferta
2      // Declarar variables
3      Definir edad, membresia, formulario, asistencia Como Entero;
4      Definir tieneMembresia, completoFormulario, esElegible Como Logico;
5      // Solicitar entrada del usuario
6      Escribir "Ingrese su edad: ";
7      Leer edad;
8
9      Escribir "?Tiene membres?a? (1 = Verdadero / 2 = Falso): ";
10     Leer membresia;
11
12     SI membresia = 1 Entonces
13         tieneMembresia ← Verdadero;
14     SiNo
15         tieneMembresia ← Falso;
16     FinSi
17
18     Escribir "?Ha completado un formulario de registro? (1 = Verdadero / 2 = Falso): ";
19     Leer formulario;
20
21     SI formulario = 1 Entonces
22         completoFormulario ← Verdadero;
23     SiNo
24         completoFormulario ← Falso;
25     FinSi
26
27     // Verificar si el usuario es elegible para la oferta
28     esElegible ← (edad ≥ 18) Y tieneMembresia Y completoFormulario;
29
30     // Mostrar resultado
31     Si esElegible Entonces
32         Escribir "?Felicidades! Usted es elegible para la oferta especial.";
33     Sino
34         Escribir "Lo sentimos, usted no es elegible para la oferta especial.";
35     FinSi;
36 FinProceso
```


Operadores lógicos



Ejemplo Java

```
import java.util.Scanner;

public class OperadoresLogicos{
    public static void main(String[] args) {
        // Crear un objeto Scanner para leer la entrada del usuario
        Scanner scanner = new Scanner(System.in);

        // Declarar variables
        int edad, membresia, formulario, asistencia;
        boolean tieneMembresia, completoFormulario, esElegible;

        // Solicitar entrada del usuario
        System.out.print("Ingrese su edad: ");
        edad = scanner.nextInt();

        System.out.print("¿Tiene membresía? (1 = Verdadero / 2 = Falso): ");
        membresia = scanner.nextInt();
        tieneMembresia = (membresia == 1);

        System.out.print("¿Ha completado un formulario de registro? (1 = Verdadero / 2 = Falso): ");
        formulario = scanner.nextInt();
        completoFormulario = (formulario == 1);

        // Verificar si el usuario es elegible para la oferta
        esElegible = (edad >= 18) && tieneMembresia && completoFormulario ;

        // Mostrar resultado
        if (esElegible) {
            System.out.println("¡Felicidades! Usted es elegible para la oferta especial.");
        } else {
            System.out.println("Lo sentimos, usted no es elegible para la oferta especial.");
        }

        // Cerrar el scanner
        scanner.close();
    }
}
```




Expresiones aritméticas

Las expresiones aritméticas combinan **variables**, **constantes** y **operadores** para realizar cálculos numéricos. Son esenciales para resolver problemas matemáticos en programación. Estas expresiones siguen un orden de prioridad de **operadores**, como en matemáticas, y se utilizan en asignaciones, condiciones y cualquier parte donde se necesiten valores calculados.

Componente	Descripción	Ejemplo en PSeInt	Ejemplo en Java	Componente
Constantes	Valores numéricos fijos	10, 3.14	10, 3.14	Constantes
Variables	Almacenan datos para operar	total, precio	total, precio	Variables
Operadores	Símbolos que realizan operaciones	+, -, *, /, %	+, -, *, /, %	Operadores
Paréntesis	Alteran el orden de evaluación	(a + b) * c	(a + b) * c	Paréntesis
Resultado	Valor obtenido de la operación	total <- (precio + impuesto) * 2	total = (precio + impuesto) * 2;	Resultado

Ejemplo en PSeInt:

```
total <- (precio + impuesto) * cantidad;
```

Ejemplo en Java:

```
total = (precio + impuesto) * cantidad;
```


Expresiones aritméticas

Ejemplo PSeInt

```
1  Proceso OperacionesAritmeticasExtendido
2      Definir a, b, c Como Real;
3      Definir suma, multiplicacion, division, modulo, resultado Como Real;
4
5      // Asignación de valores
6      a ← 8;
7      b ← 3;
8      c ← 2;
9
10     // Suma de a + b
11     suma ← a + b;
12     Escribir "Paso 1 - Suma (a + b): ", suma;
13
14     // Multiplicación de la suma por c
15     multiplicacion ← suma * c;
16     Escribir "Paso 2 - Multiplicación (suma * c): ", multiplicacion;
17
18     // División de a entre b
19     division ← a / b;
20     Escribir "Paso 3 - División (a / b): ", division;
21
22     // Módulo de a entre c (residuo)
23     modulo ← a % c;
24     Escribir "Paso 4 - Módulo (a % c): ", modulo;
25
26     // Resultado final: (a + b) * c - (a / b) + (a % c)
27     resultado ← multiplicacion - division + modulo;
28     Escribir "Resultado final: ", resultado;
29 FinProceso
30
```


Expresiones aritméticas

Ejemplo Java

```
public class OperacionesAritmeticas {  
    public static void main(String[] args) {  
        double a = 8;  
        double b = 3;  
        double c = 2;  
  
        // Paso 1 - Suma  
        double suma = a + b;  
        System.out.println("Paso 1 - Suma (a + b): " + suma);  
  
        // Paso 2 - Multiplicación  
        double multiplicacion = suma * c;  
        System.out.println("Paso 2 - Multiplicación (suma * c): " + multiplicacion);  
  
        // Paso 3 - División  
        double division = a / b;  
        System.out.println("Paso 3 - División (a / b): " + division);  
  
        // Paso 4 - Módulo  
        double modulo = a % c;  
        System.out.println("Paso 4 - Módulo (a % c): " + modulo);  
  
        // Resultado final  
        double resultado = multiplicacion - division + modulo;  
        System.out.println("Resultado final: " + resultado);  
    }  
}
```


Comentarios

Los **comentarios** son líneas de texto dentro del código que no se ejecutan. Se usan para explicar, **documentar** o **desactivar** partes del código durante pruebas. Son muy útiles para que otros (¡o tú mismo!) entiendan lo que hace un programa.

Tipo de comentario	Sintaxis en PSeInt	Sintaxis en Java
Una línea	// o :	//
Múltiples líneas	No tiene bloque; usar // o : en cada línea	/* ... */

Ejemplo en PSeInt:

```
// Esto es un comentario
: También es un comentario
// Comentario línea 1
// Comentario línea 2
```

Ejemplo en Java:

```
// Esto es un comentario

/*
  Esto es un comentario
  de múltiples líneas
*/
```

Convenciones

$$\frac{x^3 + ay^2 + bz}{abx^2} = ?$$

Las **convenciones** relacionadas con todos los temas que hemos trabajado: **operadores aritméticos, operadores lógicos, expresiones aritméticas y comentarios**, tanto en PSeInt como en Java. Las convenciones son importantes porque permiten escribir un código claro, ordenado y fácil de entender por otros programadores (¡y por uno mismo después de un tiempo!).

Tema	Convenciones recomendadas
Operadores aritméticos	<ul style="list-style-type: none">- Usar espacios entre los operadores y los operandos para mejorar la legibilidad: a + b en lugar de a+b.- Evitar expresiones muy largas en una sola línea.
Operadores lógicos	<ul style="list-style-type: none">- Combinar expresiones lógicas con paréntesis para mayor claridad.- Escribir condiciones legibles, por ejemplo: if (edad >= 18 && tieneMembresia) en lugar de una condición demasiado anidada.
Expresiones aritméticas	<ul style="list-style-type: none">- Descomponer cálculos complejos en pasos intermedios usando variables auxiliares.- Usar nombres de variables descriptivos (subtotal, descuento, totalFinal) en lugar de letras sueltas (x, y).
Comentarios	<ul style="list-style-type: none">- Usar comentarios breves y claros.- Comentar por qué se hace algo, no qué hace (eso ya lo muestra el código).- No abusar de los comentarios; deben aportar valor.
Nombres de variables	<ul style="list-style-type: none">- En PSeInt: usar minúsculas y nombres con palabras completas: totalVenta, esMayor.- En Java: usar camelCase: totalVenta, esElegible. Evitar abreviaciones innecesarias.
Formato del código	<ul style="list-style-type: none">- Indentar correctamente (muy importante en PSeInt para lectura y en Java para estructura).- Separar visualmente las secciones del código (por ejemplo, dejando líneas en blanco entre bloques).

Convenciones

$$\frac{x^3 + ay^2 + bz}{abx^2} = ?$$

Ejemplo de buena convención (Java):

```
double precioUnitario = 10.5;
double cantidad = 3;
double subtotal = precioUnitario * cantidad;
double descuento = subtotal * 0.1;
double totalFinal = subtotal - descuento;

// Se aplica un 10% de descuento
System.out.println("Total a pagar: " + totalFinal);
```

Ejemplo de buena convención (PSeInt):

Definir precioUnitario, cantidad, subtotal, descuento, totalFinal **Como Real**;

```
precioUnitario <- 10.5;
cantidad <- 3;
subtotal <- precioUnitario * cantidad;
descuento <- subtotal * 0.1;
totalFinal <- subtotal - descuento;
```

Escribir "Total a pagar: ", totalFinal; // Se aplica un 10% de descuento



Apache NetBeans es un entorno de desarrollo integrado (IDE) gratuito y de código abierto, ideal para programar en Java y otros lenguajes. Ofrece herramientas visuales, autocompletado de código y depuración. Es utilizado tanto por principiantes como por desarrolladores profesionales gracias a su interfaz intuitiva y características integradas.

Ventajas

1. Instalación sencilla y lista para usar.
2. Soporte completo para Java y JavaFX.
3. Interfaz gráfica amigable.
4. Autocompletado de código inteligente.
5. Depurador integrado y eficaz.

Desventajas

1. Puede ser lento en equipos con pocos recursos.
2. Consume bastante memoria RAM.
3. Menos plugins que otros IDEs como IntelliJ.
4. Interfaz algo anticuada en versiones anteriores.
5. A veces no detecta bien errores de compilación menores.



Ejercicios



