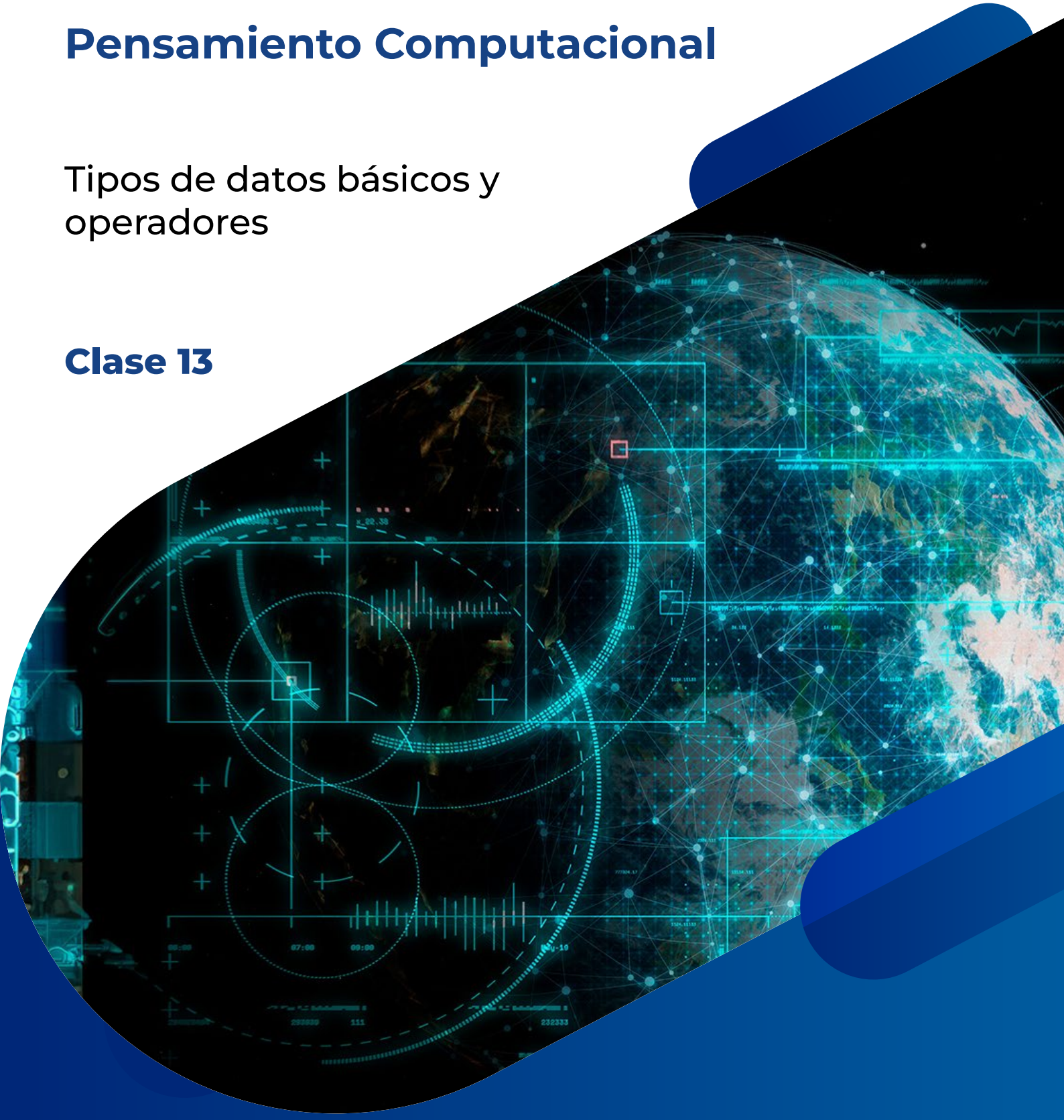
Pensamiento Computacional

Tipos de datos básicos y operadores

Clase 13



Ingeniería en ciberseguridad

CLASE 13

13.1 Tipos de datos básicos y operadores

PSeInt: un programa para crear algoritmos

Para ayudarnos a crear programas en pseudocódigo, vamos a utilizar el programa **PSeInt**.

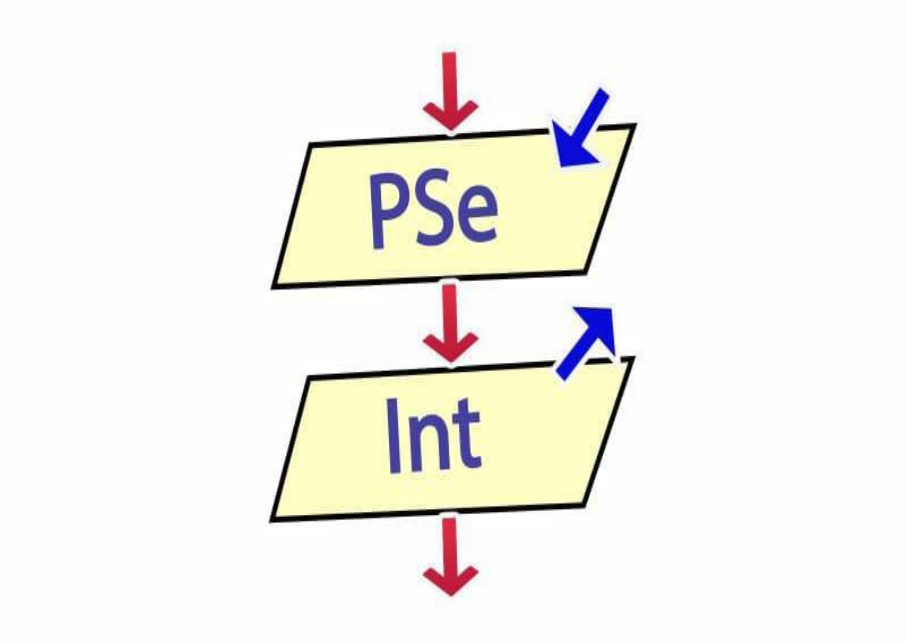
- Su uso se recomienda para estudiantes que inician sus estudios en programación, gracias a su sintaxis simple para declarar, leer, imprimir y operar con variables.
- En **PSeInt** se pueden generar diagramas de flujo a partir de pseudocódigo y viceversa.

Instrucciones para instalar PSeInt en tu computadora

1. Descarga **PSeInt** desde: Instalación de PSeInt en Windows.
2. Instala el programa según tu sistema operativo.
3. Ejecuta el instalador.

Figura 1

PSeInt logo



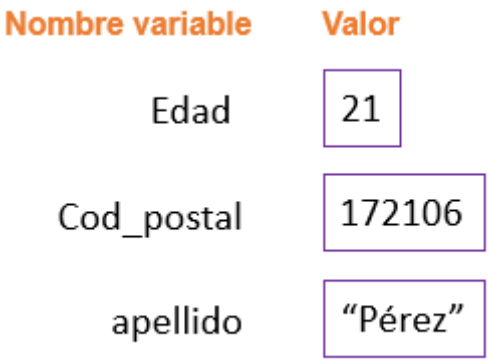
Fuente: Codersfree (s. f)

PSeInt: Variables y constantes

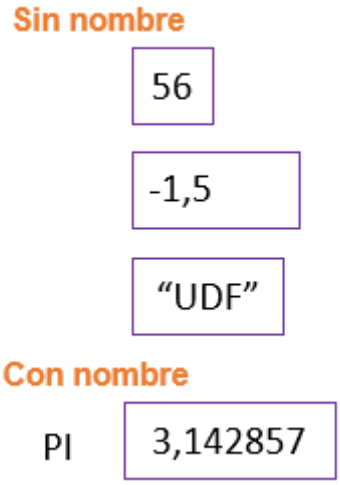
Variable: es un contenedor que representa un espacio de memoria que almacena un valor. Este valor puede ser de tipo texto, número, entre otros. El valor puede cambiar durante la ejecución del programa y se identifica por un nombre y su tipo de variable.

Reglas para definir variables:

- Debe comenzar siempre con una letra.
- Se deben usar nombres significativos y no demasiado extensos.
- No se permiten espacios en blanco en el nombre de una variable.
- El único carácter especial aceptado es el guion bajo (_) .



Constante: A diferencia de las variables, una constante no cambia su valor durante la ejecución de un programa. Un ejemplo sería el número π, el número de Euler, entre otros.



Operadores Básicos

- Suma +

- Resta -
- Multiplicación *
- División /
- Potenciación ^

Tipos de instrucciones

- **Instrucciones de declaración**: Permiten reservar áreas de memoria para definir variables. Definir.
- **Instrucciones de entrada/salida**: Leer, escribir.
- **Instrucciones aritméticas**: Sumar, restar, multiplicar, dividir.
- **Instrucciones de bifurcación**: Si... entonces... sino... entonces...

13.2 Estructura de un programa, entrada y salida de datos

Vamos a desarrollar paso a paso un programa que permite calcular el área de un círculo:

1. Inicio - Fin

Inicio: Marca el comienzo de un bloque de instrucciones.

Fin: Marca el final de un bloque de instrucciones.

- Todos los algoritmos poseen un inicio y un fin.
- En PSeint el **inicio se** lo **reemplaza** por el **nombre del algoritmo**.

Sintaxis:

Algoritmo Nombre_representativo

Código

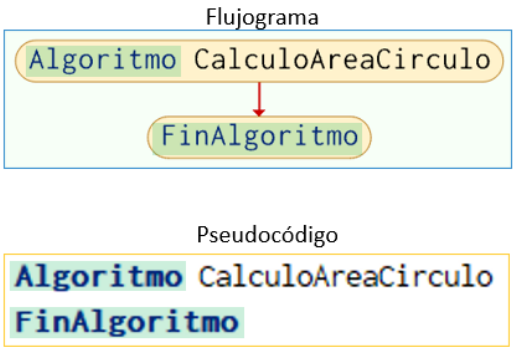
FinAlgoritmo

En el ejemplo puedes observar que el inicio posee la sintaxis anterior **Algoritmo CalculoAreaCirculo.**

De momento no tenemos código solo la instrucción **FinAlgoritmo** que marca el fin del flujo.

Figura 2

Inicio y Fin en PseInt



Fuente: Autor

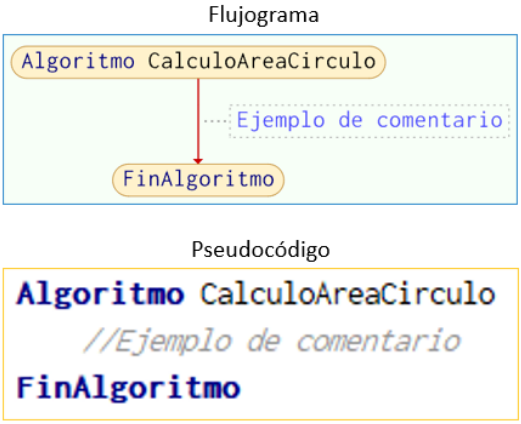
2. Comentarios en PseInt

En programación, es una buena práctica comentar nuestros programas, ya que los comentarios nos ayudan a identificar segmentos específicos de código dentro del programa. Comentar el código también facilita el trabajo de otros programadores que utilizarán el código que hemos creado.

Recuerda comentar las partes más relevantes de tu código. Evita comentar línea por línea y usa un lenguaje claro y preciso para evitar dudas o confusiones. En la imagen siguiente, puedes observar cómo se visualizan los comentarios en un diagrama de flujo y pseudocódigo.

Figura 3

Comentarios en PSeInt



Fuente: Autor

3. Declaración de variables

En la mayoría de los lenguajes de programación, es importante declarar el tipo de dato de las variables.

Declarar los tipos de datos correctos mejora la eficiencia en la ejecución de nuestros programas. Recuerda que al usar variables estamos utilizando la memoria del computador; si declaramos el tipo de dato correcto, solo utilizaremos la memoria necesaria.

Sintaxis:

Definir nombre_variable Como tipo_de_dato

Continuando con el ejemplo puedes ver:

1. Dos variables **radio y área.**
2. Ambas variables son del tipo de dato **real.** El tipo de dato real se utiliza para los números decimales positivos o negativos.

Figura 4

Declaración de variables en PSeInt

Fuente: Autor

4. Declaración de constantes

En PSeInt no existe una gran diferencia al declarar variables o constantes.

La diferencia radica en el procesamiento de ellas dentro del código del programa.

Recuerda que el valor de una constante no puede modificarse a lo largo del programa.

No todos los programas requieren constantes, pero sí deben tener variables.

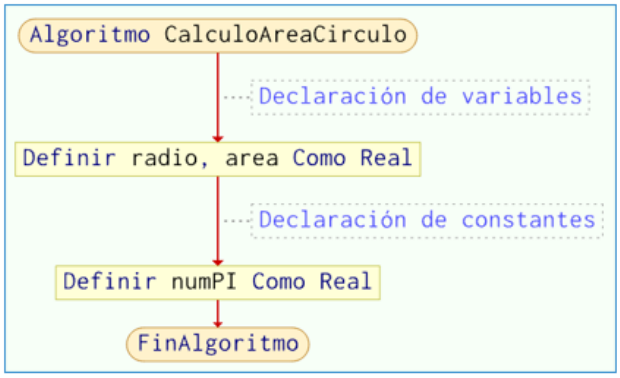
En el ejemplo se han declarado las dos variables y una **constante numPI** de tipo de dato **Real**

La diferencia estará que numPI almacena el valor de PI = 3.14159265359. Este valor es constante, nunca cambia.

Figura 5

Declaración de constantes en PSeInt

Flujograma



Pseudocódigo

```
Algoritmo CalculoAreaCirculo
    //Declaración de variables
    Definir radio, area Como Real
    //Declaración de constantes
    Definir numPI Como Real
FinAlgoritmo
```

Fuente: Autor

5. Asignación

Es el proceso de almacenar una expresión (valor) dentro de una variable.

Sintaxis: **variable = expresión**

En el ejemplo: podemos ver cómo se asigna valores a las tres variables.

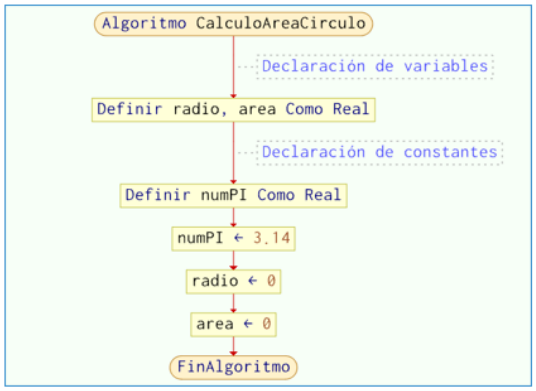
Figura 6

Lectura de variables en PSeInt

Pseudocódigo

```
Algoritmo CalculoAreaCirculo
    //Declaración de variables
    Definir radio, area Como Real
    //Declaración de constantes
    Definir numPI Como Real

    numPI = 3.14
    radio = 0
    area = 0

FinAlgoritmo
```

Flujograma



Fuente: Autor

6. Salida de datos en pantalla

Sintaxis:

Escribir 'texto a imprimir' o **Escribir nombre_variable**

En nuestro ejemplo aumentemos la instrucción:

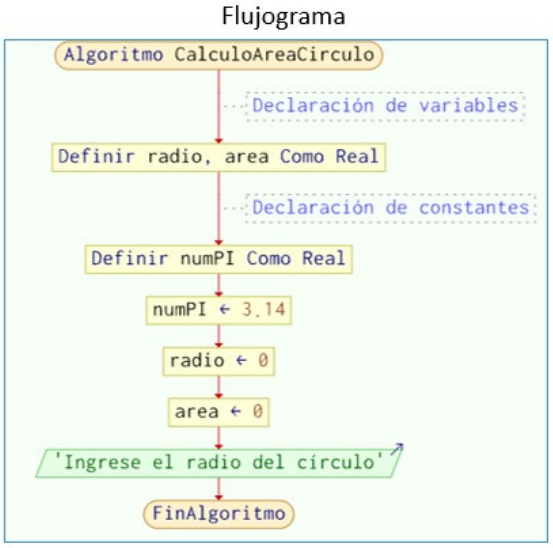
Escribir "Ingrese el radio del círculo"

Pseudocódigo

```
Algoritmo CalculoAreaCirculo
    //Declaración de variables
    Definir radio, area Como Real
    //Declaración de constantes
    Definir numPI Como Real

    numPI = 3.14
    radio = 0
    area = 0

    Escribir "Ingrese el radio del círculo"
FinAlgoritmo
```

Figura 7

Salida de datos en PSeInt



Nota. Fuente: Autor

7. Entrada de datos

Sintaxis:

Leer variable

Para guardar los datos que ingresamos por teclado necesitamos de la instrucción **Leer**.

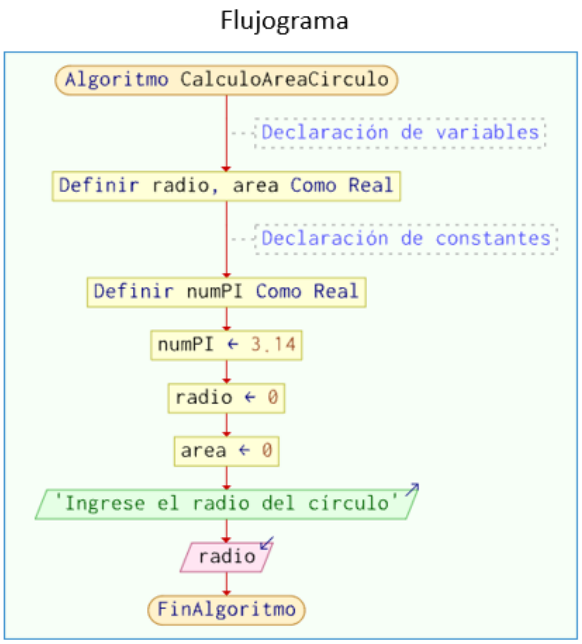
Ahora vamos a aumentar la línea Leer radio.

Pseudocódigo

```
Algoritmo CalculoAreaCirculo
    //Declaración de variables
    Definir radio, area Como Real
    //Declaración de constantes
    Definir numPI Como Real

    numPI = 3.14
    radio = 0
    area = 0

    Escribir "Ingrese el radio del circulo"
    Leer radio
FinAlgoritmo
```

Figura 8

Entrada de datos en PSeInt



Nota. Fuente: Autor

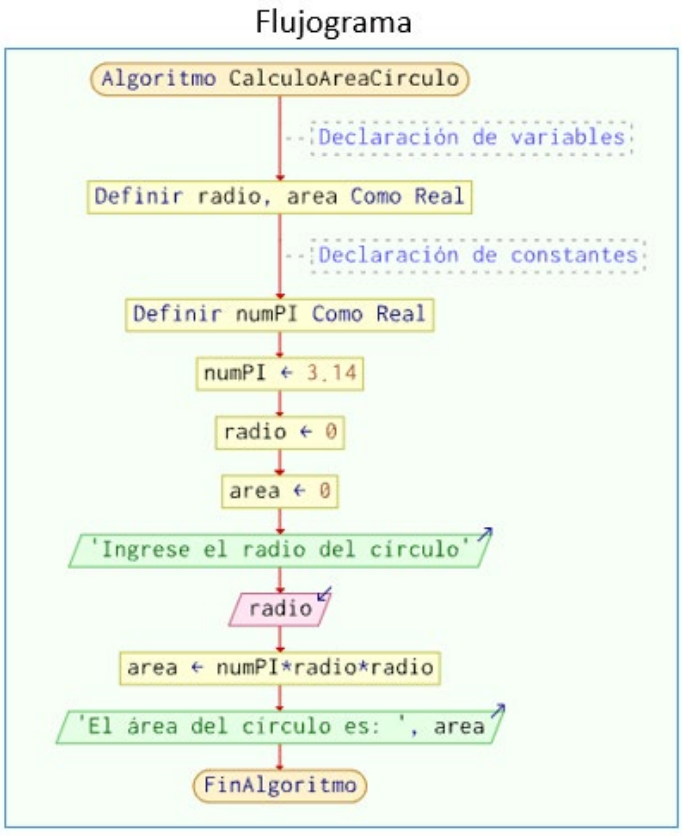
8. Código completo

Pseudocódigo

```
Algoritmo CalculoAreaCirculo
    //Declaración de variables
    Definir radio, area Como Real
    //Declaración de constantes
    Definir numPI Como Real

    numPI = 3.14
    radio = 0
    area = 0

    Escribir "Ingrese el radio del círculo"
    Leer radio

    area = numPI * radio * radio
    Escribir "El área del círculo es: ", area
FinAlgoritmo
```

Figura 9

Salida de datos en PSeInt



Fuente: Autor

Ejecución del Ejemplo

Pseudocódigo

```
Algoritmo CalculoAreaCirculo
    //Declaración de variables
    Definir radio, area Como Real
    //Declaración de constantes
    Definir numPI Como Real

    numPI = 3.14
    radio = 0
    area = 0

    Escribir "Ingrese el radio del círculo"
    Leer radio

    area = numPI * radio * radio
    Escribir "El área del círculo es: ", area
FinAlgoritmo
```

Figura 10

Salida de datos en PSeInt

Salida

```
PSeInt - Ejecutando proceso CALCULOAREACIRCULO
*** Ejecución Iniciada. ***
Ingrese el radio del círculo
> 10
El área del círculo es: 314
*** Ejecución Finalizada. ***
```

Fuente: Autor

13.3 Expresiones booleanas

Las expresiones booleanas y las funciones embebidas son conceptos clave en la programación y en el pensamiento computacional. Las expresiones booleanas permiten realizar evaluaciones lógicas y tomar decisiones dentro de un programa, mientras que las funciones embebidas facilitan operaciones comunes sin necesidad de escribir código adicional. En este tema, exploraremos cómo se utilizan las expresiones booleanas y las funciones embebidas en pseudocódigo, proporcionando una base sólida para comprender estos conceptos (Martínez & Gutiérrez, 2019).

Las expresiones booleanas son evaluaciones que resultan en un valor de verdadero o falso (True o False). Estas expresiones se utilizan comúnmente en estructuras de control de flujo, como condicio-

nales (si, sino) y bucles (mientras, para). Las operaciones booleanas básicas incluyen **AND**, **OR** y **NOT**, que permiten combinar y manipular valores lógicos (Gómez & Morales, 2018).

Operadores Booleanos Básicos

AND (Y lógico): La expresión es verdadera solo si ambas subexpresiones son verdaderas.

Ejemplo:

```
a ← 10
b ← 20
si (a > 5 AND b < 30) entonces
    imprimir("Ambas condiciones son verdaderas.")
fin si
```

OR (O lógico): La expresión es verdadera si al menos una de las subexpresiones es verdadera.

Ejemplo:

```
a ← 10
b ← 40
si (a > 5 OR b < 30) entonces
    imprimir("Al menos una condición es verdadera.")
fin si
```

NOT (Negación lógica): Invierte el valor de una expresión booleana.

Ejemplo:

```
esVerdadero ← True
si (NOT esVerdadero) entonces
    imprimir("La condición es falsa.")
```

sino

imprimir(“La condición es verdadera.”)

fin si

Expresiones booleanas en condicionales

Las expresiones booleanas son fundamentales para tomar decisiones en los programas a través de estructuras condicionales.

Ejemplo:

edad ← 18

si (edad >= 18) entonces

    imprimir(“Eres mayor de edad.”)

sino

    imprimir(“Eres menor de edad.”)

fin si

13.4 Funciones Embebidas

Las funciones embebidas, también conocidas como funciones integradas o *built-in*, son funciones que están disponibles de manera predeterminada en un lenguaje de programación. Estas funciones permiten realizar tareas comunes de manera eficiente sin necesidad de escribir código adicional. Entre las funciones embebidas más comunes se encuentran aquellas destinadas a operaciones matemáticas, manipulación de cadenas y conversiones de tipo de datos (Rodríguez & Pérez, 2020).

Ejemplos de Funciones Embebidas

Función longitud: Calcula la longitud de una cadena o lista.

Función concatenar: Une dos o más cadenas en una sola.

Función convertir_a_entero: Convierte una cadena de caracteres en un número entero.

REFERENCIAS

Codersfree. (s. f.). *Introducción a la programación con PSeInt: Conceptos básicos.* https://codersfree.com/posts/introduccion-a-la-programacion-con-pseint-conceptos-basicos

De Roer, D. D., & De Roer, D. D. (2022, 12 junio). Variables y constantes en pseudocódigo. *Disco Duro de Roer*. https://www.discoduroderoer.es/variables-y-constantes-en-pseudocodigo/

Del Amo Blanco, I. (2020, 11 noviembre). Sumas: Simples y llevando + Ejemplos y ejercicios | Smartick. *Smartick*. https://www.smartick.es/blog/matematicas/sumas-y-restas/sumas-con-llevadas/

González, J. D. M. (2020, 26 abril). Final y constantes. https://www.programarya.com/Cursos/Java/Sistema-de-Tipos/Final-y-Constantes

López, A., & Pérez, D. (2020). *Programación avanzada con pseudocódigo*. Ediciones Académicas.

Maluenda, R. (2023, 19 octubre). Qué es un algoritmo informático: Características, tipos y ejemplos. *Profile Software Services*. https://profile.es/blog/que-es-un-algoritmo-informatico/

Rodríguez, M., & Gutiérrez, P. (2019). *Fundamentos de programación con pseudocódigo*. Ediciones Académicas.

Variables en pseudocódigo. (s. f.). *Abrirllave.com*. https://www.abrirllave.com/pseudocodigo/variables.php

GLOSARIO

Dato variable: Un dato variable es un valor almacenado en la memoria de un programa que puede cambiar durante su ejecución. Las variables son fundamentales en la programación, ya que permiten que los programas sean dinámicos y gestionen datos que varían a lo largo del tiempo o según diferentes condiciones. Cada variable está asociada a un nombre y a un espacio en la memoria, y su valor puede ser modificado tantas veces como sea necesario durante la ejecución del programa (López & Pérez, 2020).

Dato constante: Un dato constante es un valor que, una vez definido, no puede cambiar durante la ejecución de un programa. Las constantes se utilizan para representar valores fijos que no deben ser alterados, como el valor de pi (π) o la velocidad de la luz. Definir datos como constantes es una buena práctica en programación, ya que mejora la claridad del código y previene errores, al garantizar que ciertos valores críticos permanezcan inalterados (Rodríguez & Gutiérrez, 2019).

PUCE Virtual

La excelencia no se improvisa

síguenos

ICYS - 12 - 241