DESAFIO DE PROGRAMACION DESDE CERO

KEVIN JAVIER VELASQUE GUANOTASIG

Resumen

Este es un documento de Word de ejemplo que puede ser usado como plantilla para dar formato a su tesis. El resumen debe contar con **350 palabras o menos**.

Tabla de Contenidos

[Capítulo 1 Introducción e información general 1](#_Toc410628920)

[Título 2 1](#_Toc410628921)

[Título 2 1](#_Toc410628922)

[Título 3. 1](#_Toc410628923)

[Título 3. 1](#_Toc410628924)

[Capítulo 2 Figuras y tablas 2](#_Toc410628925)

[Título 2 2](#_Toc410628926)

[Título 3. 2](#_Toc410628927)

[Título 3. 2](#_Toc410628928)

[Capítulo 4 Resultados . 5](#_Toc410628929)

[Lista de Referencias 6](#_Toc410628930)

[Apéndice 7](#_Toc410628931)

[Vita 8](#_Toc410628932)

Lista de tablas

# INTRODUCCION A LA PROGRQAMACION

## Dia 0: Introducción a la Programación y PSeInt

### ¿Qué es la Programación?

La programación es el proceso de crear un conjunto de instrucciones que una computadora puede entender y ejecutar. Estas instrucciones se escriben en un lenguaje de programación, que puede variar en complejidad y propósito.

### ¿Por qué Aprender a Programar?

Aprender a programar es una habilidad valiosa por varias razones:

Resolución de Problemas: La programación enseña a descomponer problemas complejos en partes más pequeñas y manejables.

* **Creatividad:** Permite crear desde aplicaciones y sitios web hasta juegos y herramientas de automatización.
* **Carrera Profesional:** Hay una alta demanda de programadores en diversas industrias.
* **Pensamiento Lógico:** Mejora la capacidad de pensar de manera estructurada y lógica.

### Conceptos Básicos de la Programación

* **Algoritmos:** Un algoritmo es una secuencia de pasos para resolver un problema o realizar una tarea.
* **Lenguajes de Programación:** Hay muchos lenguajes de programación, cada uno con sus propias sintaxis y usos. Ejemplos populares incluyen Python, JavaScript, y en este caso, PSeInt para aprender los fundamentos.
* **Variables:** Son espacios de almacenamiento que pueden contener datos y cuyo valor puede cambiar durante la ejecución del programa.
* **Condicionales:** Permiten tomar decisiones en el código basadas en ciertas condiciones (e.g., if, else).
* **Bucles:** Permiten repetir una secuencia de instrucciones múltiples veces (e.g., for, while).
* **Funciones:** Bloques de código reutilizables que realizan una tarea específica.

### ¿Qué es PSeInt?

PSeInt es una herramienta educativa diseñada para ayudar a los principiantes a aprender los fundamentos de la programación. Permite escribir algoritmos en pseudocódigo, una representación simple y cercana al lenguaje natural de los programas.

Ventajas de Usar PSeInt

* **Simplicidad:** La sintaxis es fácil de entender y similar al español.
* **Visualización:** Permite ver la ejecución del algoritmo paso a paso.
* **Enfoque en la Lógica:** Ayuda a los principiantes a centrarse en la lógica de programación sin preocuparse por las complejidades de los lenguajes de programación más avanzados.

### Objetivo del Desafío

El objetivo de este desafío es comprender los fundamentos de la programación utilizando PSeInt. A lo largo de una semana, trabajaremos en varios conceptos clave, como variables, operadores, estructuras de control, bucles, funciones y estructuras de datos. Cada día incluirá una tarea práctica para aplicar lo aprendido, culminando en un proyecto final que integrará todos los conceptos.

¡Prepárate para comenzar tu viaje en el mundo de la programación y desarrollar habilidades que te abrirán muchas puertas en el futuro!

## Dia 1: Introducción a la Programación y PSeInt

### Objetivo

Familiarizarse con PSeInt, instalar el software y crear el primer programa en pseudocódigo: "Hola, Mundo".

### Instalación de PSeInt

* Descargar PSeInt: Ve al sitio oficial de PSeInt.
* Descarga la versión correspondiente a tu sistema operativo (Windows, macOS o Linux).
* Instalar PSeInt:
* Sigue las instrucciones de instalación específicas para tu sistema operativo.
* Una vez instalado, abre el programa.

### Exploración de la Interfaz de PSeInt

* **Ventana Principal:**

1. Menú de Opciones: En la parte superior, encontrarás opciones como "Archivo", "Editar", "Ver", "Ejecutar", entre otros.
2. Área de Edición: Aquí es donde escribirás tus algoritmos en pseudocódigo.
3. Consola de Resultados: En la parte inferior, verás los resultados de la ejecución de tus algoritmos.

* **Primera Ejecución:**

1. Familiarízate con las herramientas y opciones disponibles. Puedes explorar los menús y ver las diferentes configuraciones.

### Crear el Primer Programa - "Hola, Mundo"

**Nuevo Algoritmo:**

* En el menú "Archivo", selecciona "Nuevo".
* Esto abrirá un nuevo archivo en el área de edición.

**Escribir el Algoritmo:**

* Escribe el siguiente pseudocódigo en el área de edición:

Escribir "¡Hola, Mundo!"

**Guardar el Algoritmo:**

* En el menú "Archivo", selecciona "Guardar" o "Guardar Como".
* Asigna un nombre al archivo, por ejemplo, HolaMundo.pse.

**Ejecutar el Algoritmo:**

* En el menú "Ejecutar", selecciona "Ejecutar".
* Observa el resultado en la consola de resultados, debería mostrar ¡Hola, Mundo!.

### Actividades Adicionales

**Personaliza tu "Hola, Mundo":** Modifica el mensaje para que salude con tu nombre.

**Experimenta con Errores:** Intenta cometer un error deliberado (como omitir la palabra "Escribir" o el FinAlgoritmo) y ejecuta el programa para ver cómo PSeInt te ayuda a identificar y corregir errores.

### Reflexión del Día

Hoy has aprendido a instalar y familiarizarte con PSeInt, además de crear y ejecutar tu primer programa. Este es un paso importante, ya que te has introducido al entorno de desarrollo y has escrito tus primeras líneas de pseudocódigo. Mañana, aprenderemos sobre variables y tipos de datos.

## Día 2: Variables y Tipos de Datos

### Objetivo

Comprender el concepto de variables y los diferentes tipos de datos en PSeInt. Crear un programa que solicite al usuario su nombre y edad, y luego imprima un mensaje personalizado.

### Conceptos Básicos

Variables:

* Una variable es un espacio de almacenamiento en memoria con un nombre asignado que puede contener datos que pueden cambiar durante la ejecución del programa.
* En PSeInt, se definen usando la palabra clave Definir.

Tipos de Datos:

* **Entero:** Números sin decimales (e.g., 10, -5).
* **Real:** Números con decimales (e.g., 3.14, -2.71).
* **Cadena:** Secuencias de caracteres (e.g., "Hola", "Mundo").
* **Lógico:** Valores booleanos (Verdadero o Falso).

### Crear un Programa con Variables y Tipos de Datos

Nuevo Algoritmo:

* Abre PSeInt.
* En el menú "Archivo", selecciona "Nuevo" para crear un nuevo archivo.

Definir Variables:

* Escribe el siguiente pseudocódigo en el área de edición:

Algoritmo VariablesYTiposDeDatos

// Definir variables

Definir nombre Como Cadena

Definir edad Como Entero

// Solicitar al usuario que ingrese su nombre y edad

Escribir "Ingrese su nombre:"

Leer nombre

Escribir "Ingrese su edad:"

Leer edad

// Imprimir un mensaje personalizado

Escribir "Hola ", nombre, ", tienes ", edad, " años."

FinAlgoritmo

Guardar el Algoritmo:

* En el menú "Archivo", selecciona "Guardar" o "Guardar Como".
* Asigna un nombre al archivo, por ejemplo, VariablesYTiposDeDatos.pse.

Ejecutar el Algoritmo:

* En el menú "Ejecutar", selecciona "Ejecutar".
* Ingresa los datos solicitados en la consola de resultados y verifica el mensaje personalizado.

### Actividades Adicionales

* Modifica el programa para solicitar también la ciudad de origen del usuario y su altura en metros. Imprime un mensaje que incluya todos estos datos.
* Modifica el programa para convertir la edad del usuario en días (aproximadamente) y la altura en centímetros. Imprime los resultados.

### Reflexión del Día

Hoy has aprendido sobre variables y tipos de datos, que son fundamentales para almacenar y manipular información en tus programas. Has escrito un programa que solicita datos del usuario y los utiliza para generar mensajes personalizados. Mañana, exploraremos operadores y expresiones.

### Día 3: Operadores y Expresiones

### Objetivo

Comprender el uso de operadores aritméticos, de comparación y lógicos en PSeInt. Crear un programa que realice operaciones aritméticas básicas con dos números ingresados por el usuario.

### Conceptos Básicos

****Operadores Aritméticos:****

* **Suma (+):** Añade dos números.
* **Resta (-):** Resta el segundo número del primero.
* **Multiplicación (\*):** Multiplica dos números.
* **División (/):** Divide el primer número por el segundo.
* **Módulo (%):** Devuelve el resto de la división del primer número por el segundo.

****Operadores de Comparación:****

* **Igual (==):** Comprueba si dos valores son iguales.
* **Distinto (!=):** Comprueba si dos valores son diferentes.
* **Mayor que (>):** Comprueba si el primer valor es mayor que el segundo.
* **Menor que (<):** Comprueba si el primer valor es menor que el segundo.
* **Mayor o igual (>=):** Comprueba si el primer valor es mayor o igual que el segundo.
* **Menor o igual (<=):** Comprueba si el primer valor es menor o igual que el segundo.

****Operadores Lógicos:****

* **Y (&&):** Devuelve Verdadero si ambos operandos son verdaderos.
* **(||):** Devuelve Verdadero si al menos uno de los operandos es verdadero.
* **No (!):** Invierte el valor lógico del operando.

### Crear un Programa con Operadores Aritméticos

****Nuevo Algoritmo:****

* Abre PSeInt.
* En el menú "Archivo", selecciona "Nuevo" para crear un nuevo archivo.

**Escribir el Algoritmo:**

* Escribe el siguiente pseudocódigo en el área de edición:

Algoritmo OperadoresAritmeticos

// Definir variables

Definir num1, num2 Como Real

// Solicitar al usuario que ingrese dos números

Escribir "Ingrese el primer número:"

Leer num1

Escribir "Ingrese el segundo número:"

Leer num2

// Realizar y mostrar las operaciones aritméticas

Escribir "Suma: ", num1 + num2

Escribir "Resta: ", num1 - num2

Escribir "Multiplicación: ", num1 \* num2

Escribir "División: ", num1 / num2

Escribir "Módulo: ", num1 % num2

FinAlgoritmo

Guardar el Algoritmo:

* En el menú "Archivo", selecciona "Guardar" o "Guardar Como".
* Asigna un nombre al archivo, por ejemplo, OperadoresAritmeticos.pse.

Ejecutar el Algoritmo:

* En el menú "Ejecutar", selecciona "Ejecutar".
* Ingresa los datos solicitados en la consola de resultados y verifica las operaciones aritméticas.

### Crear un Programa con Operadores de Comparación y Lógicos

**Nuevo Algoritmo:**

* En el menú "Archivo", selecciona "Nuevo" para crear un nuevo archivo.

**Escribir el Algoritmo:**

* Escribe el siguiente pseudocódigo en el área de edición:

Algoritmo OperadoresComparacionYLogicos

// Definir variables

Definir num1, num2 Como Real

// Solicitar al usuario que ingrese dos números

Escribir "Ingrese el primer número:"

Leer num1

Escribir "Ingrese el segundo número:"

Leer num2

// Realizar y mostrar comparaciones

Escribir "¿Son iguales? ", num1 = num2

Escribir "¿Son diferentes? ", num1 != num2

Escribir "¿El primer número es mayor? ", num1 > num2

Escribir "¿El primer número es menor? ", num1 < num2

Escribir "¿El primer número es mayor o igual? ", num1 >= num2

Escribir "¿El primer número es menor o igual? ", num1 <= num2

// Realizar y mostrar operaciones lógicas

Escribir "¿Ambos números son mayores que cero? ", (num1 > 0) Y (num2 > 0)

Escribir "¿Al menos uno de los números es mayor que cero? ", (num1 > 0) O (num2 > 0)

Escribir "¿No es el primer número mayor que cero? ", No (num1 > 0)

FinAlgoritmo

Guardar el Algoritmo:

* En el menú "Archivo", selecciona "Guardar" o "Guardar Como".
* Asigna un nombre al archivo, por ejemplo, OperadoresComparacionYLogicos.pse.

Ejecutar el Algoritmo:

* En el menú "Ejecutar", selecciona "Ejecutar".
* Ingresa los datos solicitados en la consola de resultados y verifica las comparaciones y operaciones lógicas.

### Actividades Adicionales

Modifica el programa para realizar otras operaciones aritméticas, como la potencia o la raíz cuadrada.

Crea un programa que combine operadores aritméticos, de comparación y lógicos para resolver un problema más complejo. Por ejemplo, calcular si un número es positivo, par y mayor que otro número ingresado por el usuario.

### Reflexión del Día

Hoy has aprendido sobre operadores aritméticos, de comparación y lógicos, que son esenciales para realizar cálculos y tomar decisiones en tus programas. Has escrito programas que utilizan estos operadores para realizar operaciones y comparaciones con datos ingresados por el usuario. Mañana, exploraremos las estructuras de control.

### Día 4: Estructuras de Control

### Objetivo

Comprender y utilizar estructuras de control de flujo en PSeInt, como condicionales y bucles, para dirigir el flujo de un programa según condiciones específicas.

### Conceptos Básicos

**Condicionales:** Las condicionales permiten que el programa tome decisiones basadas en condiciones específicas.

* **Si...Entonces...Sino:** Ejecuta un bloque de código si una condición es verdadera, y otro bloque si la condición es falsa.

Si (condicion) Entonces

// Bloque de código si la condición es verdadera

Sino

// Bloque de código si la condición es falsa

FinSi

* **Si...Entonces:** Ejecuta un bloque de código solo si una condición es verdadera.

Si (condicion) Entonces

// Bloque de código si la condición es verdadera

FinSi

**Bucles:** Los bucles permiten repetir un bloque de código varias veces.

* **Mientras:** Repite un bloque de código mientras una condición sea verdadera.

Mientras (condicion) Hacer

// Bloque de código a repetir

FinMientras

* **Repetir...Hasta Que:** Repite un bloque de código hasta que una condición sea verdadera.

Repetir

// Bloque de código a repetir

Hasta Que (condicion)

* **Para:** Repite un bloque de código un número específico de veces.

Para i <- valor\_inicial Hasta valor\_final Con Paso paso Hacer

// Bloque de código a repetir

FinPara

### Crear un Programa con Condicionales

**Nuevo Algoritmo:**

* Abre PSeInt.
* En el menú "Archivo", selecciona "Nuevo" para crear un nuevo archivo.

**Escribir el Algoritmo:**

* Escribe el siguiente pseudocódigo en el área de edición:

Algoritmo Condicionales

// Definir variables

Definir numero Como Entero

// Solicitar al usuario que ingrese un número

Escribir "Ingrese un número:"

Leer numero

// Verificar si el número es positivo, negativo o cero

Si (numero > 0) Entonces

Escribir "El número es positivo."

Sino

Si (numero < 0) Entonces

Escribir "El número es negativo."

Sino

Escribir "El número es cero."

FinSi

FinSi

FinAlgoritmo

**Guardar el Algoritmo:**

* En el menú "Archivo", selecciona "Guardar" o "Guardar Como".
* Asigna un nombre al archivo, por ejemplo, Condicionales.pse.

**Ejecutar el Algoritmo:**

* En el menú "Ejecutar", selecciona "Ejecutar".
* Ingresa los datos solicitados en la consola de resultados y verifica las respuestas condicionales.

### Crear un Programa con Bucles

**Nuevo Algoritmo:**

* En el menú "Archivo", selecciona "Nuevo" para crear un nuevo archivo.

**Escribir el Algoritmo:**

* Escribe el siguiente pseudocódigo en el área de edición:

Algoritmo Bucles

// Definir variables

Definir i Como Entero

// Mostrar los números del 1 al 10 usando Mientras

i <- 1

Mientras (i <= 10) Hacer

Escribir i

i <- i + 1

FinMientras

// Mostrar los números del 1 al 10 usando Repetir

i <- 1

Repetir

Escribir i

i <- i + 1

Hasta Que (i > 10)

// Mostrar los números del 1 al 10 usando Para

Para i <- 1 Hasta 10 Con Paso 1 Hacer

Escribir i

FinPara

FinAlgoritmo

**Guardar el Algoritmo:**

* En el menú "Archivo", selecciona "Guardar" o "Guardar Como".
* Asigna un nombre al archivo, por ejemplo, Bucles.pse.

**Ejecutar el Algoritmo:**

* En el menú "Ejecutar", selecciona "Ejecutar".
* Verifica que los números del 1 al 10 se impriman correctamente en la consola de resultados.

### Actividades Adicionales

Crea un programa que solicite al usuario su edad y verifique si es un niño (menor de 12 años), un adolescente (entre 12 y 17 años), un adulto (entre 18 y 64 años) o un adulto mayor (65 años o más).

Crea un programa que imprima una tabla de multiplicar (por ejemplo, del 1 al 10) utilizando bucles anidados.

### Reflexión del Día

Hoy has aprendido sobre las estructuras de control de flujo, que son fundamentales para dirigir la ejecución de un programa basado en condiciones y repeticiones. Has escrito programas que utilizan condicionales y bucles para tomar decisiones y repetir tareas. Mañana, exploraremos las funciones y procedimientos.

### Día 5: Funciones

### Objetivo

Comprender y utilizar funciones en PSeInt para modularizar y organizar mejor el código, facilitando su mantenimiento y reutilización.

### Conceptos Básicos

**Funciones:** Las funciones son bloques de código reutilizables que realizan una tarea específica. Una función puede recibir parámetros de entrada y devolver un valor.

**Definición de Función:** Una función se define con la palabra clave Funcion seguida del nombre de la función, los parámetros de entrada y el tipo de retorno.

Funcion nombreFuncion (parametro1, parametro2) Regresa tipo

// Cuerpo de la función

Retornar valor

FinFuncion

**Llamada a la Función:** Para utilizar una función, se llama a su nombre seguido de los argumentos entre paréntesis.

resultado <- nombreFuncion(arg1, arg2)

### Crear una Función Simple

**Nuevo Algoritmo:**

* Abre PSeInt.
* En el menú "Archivo", selecciona "Nuevo" para crear un nuevo archivo.

**Escribir el Algoritmo con Función Simple:**

* Escribe el siguiente pseudocódigo en el área de edición:

Algoritmo FuncionSimple

// Definir función que suma dos números

Funcion Sumar (a, b) Regresa Real

Retornar a + b

FinFuncion

// Variables principales

Definir x, y, resultado Como Real

// Solicitar al usuario que ingrese dos números

Escribir "Ingrese el primer número:"

Leer x

Escribir "Ingrese el segundo número:"

Leer y

// Llamar a la función Sumar y mostrar el resultado

resultado <- Sumar(x, y)

Escribir "La suma es: ", resultado

FinAlgoritmo

**Guardar el Algoritmo:**

* En el menú "Archivo", selecciona "Guardar" o "Guardar Como".
* Asigna un nombre al archivo, por ejemplo, FuncionSimple.pse.

**Ejecutar el Algoritmo:**

* En el menú "Ejecutar", selecciona "Ejecutar".
* Ingresa los datos solicitados en la consola de resultados y verifica la suma de los números.

### Crear una Función con Condicionales

**Nuevo Algoritmo:**

* En el menú "Archivo", selecciona "Nuevo" para crear un nuevo archivo.

**Escribir el Algoritmo con Función que Verifica Paridad:**

* Escribe el siguiente pseudocódigo en el área de edición:

Algoritmo VerificarParidad

// Definir función que verifica si un número es par

Funcion EsPar (numero) Regresa Logico

Si (numero % 2 = 0) Entonces

Retornar Verdadero

Sino

Retornar Falso

FinSi

FinFuncion

// Variables principales

Definir num Como Entero

Definir resultado Como Logico

// Solicitar al usuario que ingrese un número

Escribir "Ingrese un número:"

Leer num

// Llamar a la función EsPar y mostrar el resultado

resultado <- EsPar(num)

Si (resultado) Entonces

Escribir "El número es par."

Sino

Escribir "El número es impar."

FinSi

FinAlgoritmo

**Guardar el Algoritmo:**

* En el menú "Archivo", selecciona "Guardar" o "Guardar Como".
* Asigna un nombre al archivo, por ejemplo, VerificarParidad.pse.

**Ejecutar el Algoritmo:**

* En el menú "Ejecutar", selecciona "Ejecutar".
* Ingresa los datos solicitados en la consola de resultados y verifica la paridad del número.

### Actividades Adicionales con Funciones

Crea un programa que incluya una función para calcular la factorial de un número.

Crea un programa que incluya una función para calcular el MCD de dos números usando el algoritmo de Euclides.

### Reflexión del Día

Hoy has aprendido sobre las funciones, que son bloques de código reutilizables y modulares. Has escrito programas que utilizan funciones para realizar cálculos específicos y tomar decisiones. Mañana, exploraremos los procedimientos y cómo se diferencian de las funciones.

### Día 6: Estructuras de Datos

### Objetivo

Comprender y utilizar las estructuras de datos en PSeInt, como vectores y matrices, para organizar y manipular conjuntos de datos de manera eficiente.

### Conceptos Básicos

**Vectores (Arreglos Unidimensionales):** Un vector es una estructura de datos que almacena una secuencia de elementos del mismo tipo.

* **Declaración de un Vector:**

Definir numeros Como Entero[5]

**Matrices (Arreglos Bidimensionales):** Una matriz es una estructura de datos que almacena elementos en una cuadrícula de filas y columnas.

* **Declaración de una Matriz:**

Definir matrizNumeros Como Entero[3, 3]

### Crear un Programa con Vectores

### Ejercicio 1: Cálculo de Promedio de Calificaciones

**Nuevo Algoritmo:**

* Abre PSeInt.
* En el menú "Archivo", selecciona "Nuevo" para crear un nuevo archivo.

**Escribir el Algoritmo para Calcular Promedio de Calificaciones:**

* Escribe el siguiente pseudocódigo en el área de edición:

Algoritmo CalcularPromedio

// Definir un vector para almacenar las calificaciones de 5 estudiantes

Definir calificaciones Como Real[5]

Definir i Como Entero

Definir suma, promedio Como Real

// Solicitar al usuario que ingrese las calificaciones

Para i <- 1 Hasta 5 Con Paso 1 Hacer

Escribir "Ingrese la calificación del estudiante ", i, ":"

Leer calificaciones[i]

FinPara

// Calcular la suma de las calificaciones

suma <- 0

Para i <- 1 Hasta 5 Con Paso 1 Hacer

suma <- suma + calificaciones[i]

FinPara

// Calcular el promedio

promedio <- suma / 5

// Mostrar el promedio

Escribir "El promedio de las calificaciones es: ", promedio

FinAlgoritmo

**Guardar el Algoritmo:**

* En el menú "Archivo", selecciona "Guardar" o "Guardar Como".
* Asigna un nombre al archivo, por ejemplo, CalcularPromedio.pse.

**Ejecutar el Algoritmo:**

* En el menú "Ejecutar", selecciona "Ejecutar".
* Ingresa las calificaciones solicitadas y verifica el cálculo del promedio.

### Crear un Programa con Matrices

### Ejercicio 2: Inventario de Productos

**Nuevo Algoritmo:**

* En el menú "Archivo", selecciona "Nuevo" para crear un nuevo archivo.

**Escribir el Algoritmo para Gestionar un Inventario de Productos:**

* Escribe el siguiente pseudocódigo en el área de edición:

Algoritmo InventarioProductos

// Definir una matriz para almacenar el inventario de productos

Definir inventario Como Real[3, 2]

Definir i, j Como Entero

Definir totalStock Como Real

// Solicitar al usuario que ingrese el nombre, cantidad y precio de los productos

Para i <- 1 Hasta 3 Con Paso 1 Hacer

Escribir "Ingrese el nombre del producto ", i, ":"

Leer inventario[i, 1]

Escribir "Ingrese la cantidad del producto ", i, ":"

Leer inventario[i, 2]

FinPara

// Calcular el valor total del inventario

totalStock <- 0

Para i <- 1 Hasta 3 Con Paso 1 Hacer

totalStock <- totalStock + inventario[i, 2]

FinPara

// Mostrar el valor total del inventario

Escribir "El valor total del inventario es: ", totalStock

FinAlgoritmo

**Guardar el Algoritmo:**

* En el menú "Archivo", selecciona "Guardar" o "Guardar Como".
* Asigna un nombre al archivo, por ejemplo, InventarioProductos.pse.

**Ejecutar el Algoritmo:**

* En el menú "Ejecutar", selecciona "Ejecutar".
* Ingresa los datos solicitados y verifica el cálculo del valor total del inventario.

### Actividades Adicionales con Estructuras de Datos

**Ejercicio 1: Registro de Temperaturas Diarias:** Escribir el Algoritmo para Registrar Temperaturas Diarias

**Ejercicio 2: Calificación de Estudiantes en Varias Materias:** Escribir el Algoritmo para Calificar Estudiantes en Varias Materias:

### Reflexión del Día

Hoy has aprendido sobre las estructuras de datos, que son fundamentales para organizar y manipular conjuntos de datos en un programa. Has escrito programas que utilizan vectores y matrices para almacenar y procesar información de manera práctica y realista. Mañana, exploraremos la gestión de archivos.

### Día 7: Proyecto final

### Objetivo

Aplicar todos los conceptos aprendidos durante la semana en un proyecto final que integre variables, operadores, estructuras de control, funciones y estructuras de datos en PSeInt, creando un juego interactivo.

### Descripción del Proyecto

El proyecto final consistirá en crear un juego del ahorcado en el que el usuario debe adivinar una palabra secreta letra por letra. El programa debe proporcionar retroalimentación al usuario, mostrando las letras adivinadas correctamente y las letras incorrectas, y controlar la cantidad de intentos permitidos.

### Requisitos del Proyecto

**Definir una palabra secreta:** La palabra secreta será predefinida o seleccionada al azar de una lista de palabras.

**Solicitar suposiciones al usuario:** Permitir al usuario ingresar letras y verificar si están en la palabra secreta.

**Mostrar el progreso:** Mostrar las letras adivinadas correctamente y los espacios restantes.

**Controlar los intentos:** Llevar un conteo de los intentos incorrectos y finalizar el juego si se alcanzan los intentos permitidos.

**Finalizar el juego:** Informar al usuario si ha ganado o perdido el juego y mostrar la palabra secreta.

### Planificación del Proyecto

#### 1.1 Variables Principales

* palabraSecreta: Almacena la palabra que el usuario debe adivinar.
* palabraMostrada: Vector que muestra el progreso de la palabra adivinada.
* letrasIncorrectas: Vector que almacena las letras incorrectas.
* intentos: Cuenta el número de intentos incorrectos realizados.
* maxIntentos: Número máximo de intentos permitidos.

#### 1.2 Funciones Principales

* InicializarJuego: Función para inicializar las variables del juego.
* SolicitarLetra: Función para solicitar una letra al usuario.
* ActualizarPalabraMostrada: Función para actualizar la palabra mostrada basada en la letra adivinada.
* MostrarEstado: Función para mostrar el estado actual del juego.

Algoritmo Ahorcado

// Función para inicializar el juego

Funcion InicializarJuego()

palabraSecreta <- "PROGRAMACION"

maxIntentos <- 6

intentos <- 0

Para i <- 1 Hasta Longitud(palabraSecreta) Con Paso 1 Hacer

palabraMostrada[i] <- "\_"

FinPara

FinFuncion

// Función para solicitar una letra al usuario

Funcion SolicitarLetra() Regresa Caracter

Definir letra Como Caracter

Escribir "Ingrese una letra:"

Leer letra

Retornar letra

FinFuncion

// Función para actualizar la palabra mostrada basada en la letra adivinada

Funcion ActualizarPalabraMostrada(letra) Regresa Logico

Definir acierto Como Logico

acierto <- Falso

Para i <- 1 Hasta Longitud(palabraSecreta) Con Paso 1 Hacer

Si Subcadena(palabraSecreta, i, 1) = letra Entonces

palabraMostrada[i] <- letra

acierto <- Verdadero

FinSi

FinPara

Retornar acierto

FinFuncion

// Función para mostrar el estado actual del juego

Funcion MostrarEstado()

Definir i Como Entero

Escribir "Palabra: "

Para i <- 1 Hasta Longitud(palabraMostrada) Con Paso 1 Hacer

Escribir palabraMostrada[i], " "

FinPara

Escribir ""

Escribir "Letras incorrectas: ", letrasIncorrectas

Escribir "Intentos restantes: ", maxIntentos - intentos

FinFuncion

// Variables principales

Definir palabraSecreta Como Caracter

Definir palabraMostrada Como Caracter[100]

Definir letrasIncorrectas Como Caracter[100]

Definir letra Como Caracter

Definir intentos, maxIntentos, aciertos Como Entero

// Inicializar el juego

InicializarJuego()

// Bucle principal del juego

Mientras intentos < maxIntentos Y aciertos < Longitud(palabraSecreta) Hacer

MostrarEstado()

letra <- SolicitarLetra()

Si No ActualizarPalabraMostrada(letra) Entonces

letrasIncorrectas[intentos] <- letra

intentos <- intentos + 1

Sino

aciertos <- aciertos + 1

FinSi

// Comprobar si se ha adivinado toda la palabra

aciertos <- 0

Para i <- 1 Hasta Longitud(palabraSecreta) Con Paso 1 Hacer

Si palabraMostrada[i] <> "\_" Entonces

aciertos <- aciertos + 1

FinSi

FinPara

FinMientras

// Mostrar el resultado del juego

Si aciertos = Longitud(palabraSecreta) Entonces

Escribir "¡Felicidades! Adivinaste la palabra: ", palabraSecreta

Sino

Escribir "Has perdido. La palabra era: ", palabraSecreta

FinSi

FinAlgoritmo

### Descripción del Proyecto

El proyecto final consistirá en crear un juego de "Tic-Tac-Toe" en el que dos jugadores se turnan para marcar los espacios en una cuadrícula de 3x3. El primer jugador que consiga alinear tres de sus marcas (horizontal, vertical o diagonalmente) gana el juego.

### Requisitos del Proyecto

**Mostrar el tablero:** Mostrar el tablero de juego actualizado después de cada movimiento.

**Solicitar movimientos a los jugadores:** Permitir a los jugadores ingresar sus movimientos y verificar si son válidos.

**Controlar el flujo del juego:** Alternar entre los jugadores y verificar el estado del juego después de cada movimiento.

**Determinar el ganador:** Verificar si un jugador ha ganado el juego o si hay un empate.

### Planificación del Proyecto

#### 1.1 Variables Principales

* tablero: Matriz que representa el tablero de juego.
* jugadorActual: Variable que indica el jugador actual ('X' o 'O').
* movimientos: Contador de movimientos realizados.

#### 1.2 Funciones Principales

* InicializarTablero: Función para inicializar el tablero de juego.
* MostrarTablero: Función para mostrar el tablero actual.
* SolicitarMovimiento: Función para solicitar un movimiento al jugador.
* VerificarGanador: Función para verificar si hay un ganador o un empate.

Algoritmo TicTacToe

// Función para inicializar el tablero de juego

Funcion InicializarTablero()

Para i <- 1 Hasta 3 Con Paso 1 Hacer

Para j <- 1 Hasta 3 Con Paso 1 Hacer

tablero[i, j] <- " "

FinPara

FinPara

FinFuncion

// Función para mostrar el tablero actual

Funcion MostrarTablero()

Escribir "Tablero:"

Para i <- 1 Hasta 3 Con Paso 1 Hacer

Para j <- 1 Hasta 3 Con Paso 1 Hacer

Si j < 3 Entonces

Escribir tablero[i, j], " | "

Sino

Escribir tablero[i, j]

FinSi

FinPara

Si i < 3 Entonces

Escribir "--|---|--"

FinSi

FinPara

FinFuncion

// Función para solicitar un movimiento al jugador

Funcion SolicitarMovimiento() Regresa Logico

Definir fila, columna Como Entero

Escribir "Jugador ", jugadorActual, ", ingrese su movimiento (fila y columna):"

Leer fila, columna

Si fila >= 1 Y fila <= 3 Y columna >= 1 Y columna <= 3 Entonces

Si tablero[fila, columna] = " " Entonces

tablero[fila, columna] <- jugadorActual

Retornar Verdadero

Sino

Escribir "Espacio ya ocupado. Intente de nuevo."

Retornar Falso

FinSi

Sino

Escribir "Movimiento inválido. Intente de nuevo."

Retornar Falso

FinSi

FinFuncion

// Función para verificar si hay un ganador o un empate

Funcion VerificarGanador() Regresa Caracter

// Verificar filas y columnas

Para i <- 1 Hasta 3 Con Paso 1 Hacer

Si tablero[i, 1] = tablero[i, 2] Y tablero[i, 2] = tablero[i, 3] Y tablero[i, 1] <> " " Entonces

Retornar tablero[i, 1]

FinSi

Si tablero[1, i] = tablero[2, i] Y tablero[2, i] = tablero[3, i] Y tablero[1, i] <> " " Entonces

Retornar tablero[1, i]

FinSi

FinPara

// Verificar diagonales

Si tablero[1, 1] = tablero[2, 2] Y tablero[2, 2] = tablero[3, 3] Y tablero[1, 1] <> " " Entonces

Retornar tablero[1, 1]

FinSi

Si tablero[1, 3] = tablero[2, 2] Y tablero[2, 2] = tablero[3, 1] Y tablero[1, 3] <> " " Entonces

Retornar tablero[1, 3]

FinSi

// Verificar empate

Si movimientos = 9 Entonces

Retornar "E"

FinSi

Retornar "N"

FinFuncion

// Variables principales

Definir tablero Como Caracter[3, 3]

Definir jugadorActual Como Caracter

Definir movimientos Como Entero

Definir ganador Como Caracter

// Inicializar el juego

InicializarTablero()

jugadorActual <- "X"

movimientos <- 0

ganador <- "N"

// Bucle principal del juego

Mientras ganador = "N" Hacer

MostrarTablero()

Si SolicitarMovimiento() Entonces

movimientos <- movimientos + 1

ganador <- VerificarGanador()

Si ganador = "N" Entonces

Si jugadorActual = "X" Entonces

jugadorActual <- "O"

Sino

jugadorActual <- "X"

FinSi

FinSi

FinSi

FinMientras

// Mostrar el resultado del juego

MostrarTablero()

Si ganador = "E" Entonces

Escribir "¡Es un empate!"

Sino

Escribir "¡Felicidades! Jugador ", ganador, " ha ganado."

FinSi

FinAlgoritmo