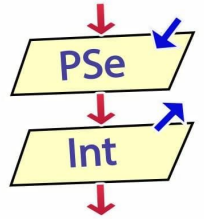
CODO A CODO INICIAL
Clase 11/12

Algoritmia

BA

Síntesis y Repaso



Les damos la bienvenida

Vamos a comenzar a grabar la clase

Clase 10

Algoritmia 10 - Estructuras Repetitivas II

- Contador.
- Acumulador.
- Bandera.
- Combinación de bucles y condicionales. Ejemplos.

Clase 11/12

Algoritmia 11/12 - Síntesis y repaso

- Integración de temas.
- Tipos de datos y variables.
- Entrada y salida de datos.
- Estructuras condicionales.
- Operadores lógicos
- Estructuras repetitivas.

Clase 13

Python 1 - Introducción al Lenguaje Python

- ¿Qué es Python?
- Historia y filosofía del lenguaje.
- Descarga e instalación del entorno de desarrollo.
- Uso de VSCode como editor de código..
- “Hola Mundo” en Python.

Hardware y Software

Hardware: Se refiere a los **componentes físicos** de un sistema informático. Esto incluye dispositivos como la unidad central de procesamiento (CPU), la memoria RAM, el disco duro, la tarjeta gráfica, la placa madre, periféricos como teclados y ratones, etcétera.

Software: Son los programas y aplicaciones que al ejecutarse trabajan con los datos. Es intangible y se compone de instrucciones que le indican a la computadora cómo realizar tareas específicas. El software incluye sistemas operativos, programas de aplicación, controladores de dispositivos y cualquier otro código que permita el funcionamiento de la computadora.

¿Qué es la programación?

La **programación** es la ciencia de crear conjuntos de instrucciones para que una computadora ejecute tareas específicas, implicando la traducción de ideas y lógica humana a un lenguaje comprensible por las máquinas. Los programadores utilizan lenguajes de programación para implementar algoritmos, resolver problemas y construir software funcional, lo cual requiere habilidades lógicas, claridad estructural y eficiencia en la resolución de problemas.

Su **objetivo** es codificar instrucciones para que las computadoras ejecuten sistemas, programas y aplicaciones efectivas, accesibles y amigables para el usuario.

Algoritmia

La **algoritmia** es la disciplina que se ocupa de la concepción, representación, diseño, análisis, implementación y aplicación de **algoritmos**.

Un algoritmo es un conjunto ***finito*** y ***ordenado*** de pasos o reglas ***bien definidas*** que describen la secuencia de operaciones necesarias para realizar una tarea o resolver un problema específico.

La algoritmia es fundamental en el campo de la informática y juega un papel crucial en el desarrollo de software y en la resolución eficiente de problemas computacionales.

Diagrama de flujo

Es una herramienta gráfica para representar los **algoritmos**. Se utiliza a la hora de diseñar un programa. Este método visual facilita la comprensión de los pasos y la lógica del proceso. Se pueden crear fácilmente sobre un simple papel, o utilizando herramientas especializadas de software.

Normalmente se usa el **lenguaje UML** (*Lenguaje Unificado de Modelado*), un estándar de la industria para visualizar, especificar y construir la estructura y el comportamiento de un sistema de software.

Análisis y resolución de Problemas

Utilizar una metodología apropiada para resolver problemas nos ayuda a conseguir un algoritmo eficiente y eficaz. Por lo general, los **pasos a seguir** son los siguientes:

1. Analizar el problema
2. Diseñar el algoritmo
3. Codificar
4. Depurar el programa

Siguiendo estos pasos correctamente se logra realizar cualquier algoritmo.

Programa informático

Un **programa informático** es una secuencia de instrucciones escritas en un **lenguaje de programación** que manipulan un conjunto de **datos** para resolver un problema o cumplir una función determinada.

La creación de un programa informático implica la **traducción de un algoritmo o diagrama de flujo**, concebido para resolver un problema, a un conjunto de instrucciones escritas en algún lenguaje de programación, para ser ejecutadas en un dispositivo programable (generalmente, una computadora).

¿Qué es un dato?

En sentido general, un dato es una pieza de información. **Es la unidad mínima de información.** En programación, un dato es esencialmente una pieza de información que almacenamos en algún sitio (por ejemplo, la memoria RAM o un pendrive). Los datos pueden ser de distintos tipos.

Independientemente de la naturaleza exacta de un dato, se almacena en la computadora como una serie de ceros y unos (utilizando el llamado “sistema binario”). Sin embargo, este proceso generalmente es “transparente” para el programador o el usuario.

Lenguaje de programación

Un **lenguaje de programación** es un conjunto de reglas y símbolos que permite a los programadores escribir instrucciones que una computadora puede entender y ejecutar. Es un medio de comunicación entre el ser humano y la máquina, facilitando la creación de software y el desarrollo de aplicaciones mediante la expresión de algoritmos y lógica de programación.

PSeInt

PSeInt (abreviatura de "*Pseudo Intérprete*") es un entorno de desarrollo y aprendizaje de programación diseñado para ayudar a estudiantes a comprender los conceptos básicos de la programación y la lógica algorítmica.

PSeInt no es un lenguaje de programación en sí mismo, sino un intérprete de pseudocódigo.

Proporciona una interfaz gráfica simple y herramientas que permiten practicar la construcción de algoritmos.

La experiencia adquirida con PSeInt sienta las bases para comprender la programación en otros entornos y lenguajes.

Salida de datos por pantalla

La **salida por pantalla** de un **programa** se refiere a la información que el programa muestra en la pantalla o dispositivo de salida.

Esta información incluye resultados de cálculos, mensajes informativos, interacciones con el usuario o cualquier otro dato relevante que el programa desee comunicar.

Es una parte esencial de la interacción entre el programa y el usuario, permitiendo la presentación clara y comprensible de los resultados generados durante la ejecución del programa.

Instrucción Escribir (output)

En PSeInt podemos indicar al programa que nos muestre un mensaje en la terminal. En otras palabras, podemos obtener la salida de valores por pantalla. Pueden ser datos, valores almacenados en variables, resultados de una función, etcétera. Se pueden mostrar uno o varios de estos objetos, siempre que los separemos con una “,” (coma).

Escribir «dato»

Escribir «unDato» , «otroDato» , «otroDatoMas»

Operador de asignación

En programación, el **operador de asignación** se encarga de asignar a la variable que se encuentra a su izquierda el valor resultante de evaluar la expresión que se encuentra a su derecha.

En **PSeInt** usamos el signo “**=**” (igual) como operador de asignación.

Operador de concatenación

Concatenar significa “unir o enlazar dos o más cosas” (RAE). En programación significa unir datos del tipo cadena de caracteres uno con otro.

En PSeInt, el operador de concatenación es el signo “**+**” (más).

Operadores aritméticos

Los **operadores aritméticos** permiten realizar operaciones con datos numéricos. Existen varios operadores aritméticos, uno para cada una de las **operaciones algebraicas** comunes (suma, resta, multiplicación, división, etc).

Operadores aritméticos

Operador	Símbolo	Acción
Suma	+	Suma dos operandos
Resta	-	Resta el segundo operando del primero
Multiplicación	*	Multiplica los dos operandos
División	/	Divide el primer operando por el segundo
Resto (módulo)	%	Devuelve el resto de la división entre dos operandos
Potencia	^	Eleva el primer operando a la potencia del segundo

Operadores relacionales

Los operadores relacionales en programación son herramientas clave para **comparar valores** y **evaluar condiciones**. Incluyen los símbolos “==”, “>”, “<”, “>=” , “<=” y “!=”, permitiendo establecer relaciones entre variables y valores. Estos operadores son fundamentales en **estructuras condicionales**, donde determinan la ejecución de bloques de código según si una condición dada es **verdadera** o **falsa**. Su uso es esencial para la toma de decisiones y el control del flujo en programas, brindando la capacidad de responder dinámicamente a diferentes situaciones.

¿Cuáles son los operadores relacionales?

Operador	Significado	Ejemplo
<	menor que	5 < 7 (Verdadero)
<=	menor o igual que	5 <= 4 (Falso)
>	mayor que	3 > 4 (Falso)
>=	mayor o igual que	13 >= 13 (Verdadero)
!=	distinto que	15 != 15 (Falso)
==	igual que	10 == 10 (Verdadero)

Estructuras condicionales

Las **estructuras condicionales** en programación son un tipo de estructuras de control que permiten **controlar el flujo de ejecución** de un programa de acuerdo con condiciones específicas.

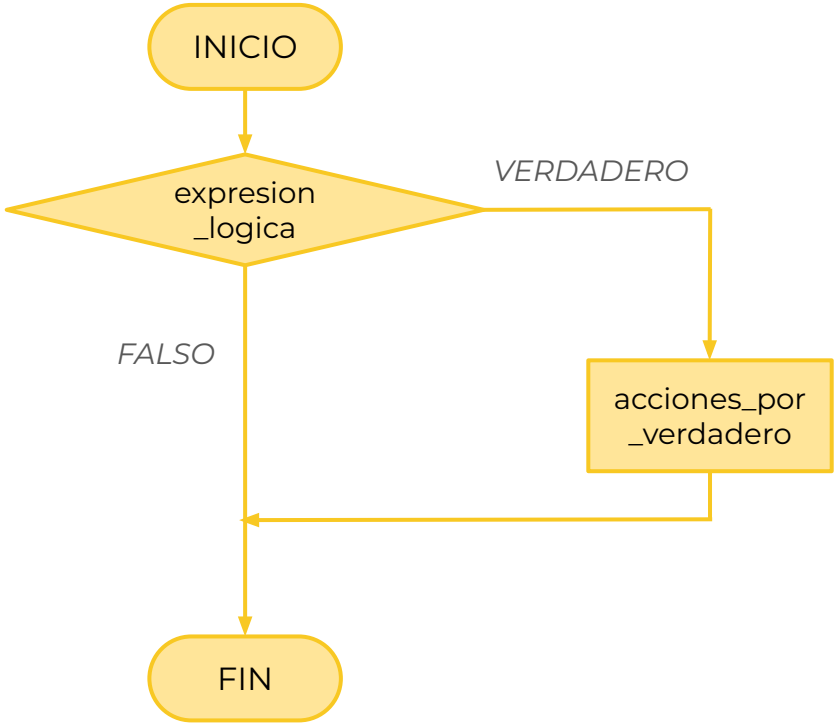
Utilizando declaraciones como "**Si... Entonces**", el flujo del programa bifurca, es decir, se dirige por diferentes caminos según si una condición es **verdadera** o **falsa**. Estas estructuras son fundamentales para la ejecución de código condicional, permitiendo que un programa tome decisiones dinámicamente en respuesta a diferentes situaciones, mejorando así la flexibilidad y adaptabilidad del software.

Condicional simple

Este estructura ejecuta una serie de instrucciones ***sólo si*** la evaluación de determinada expresión es **VERDADERA**. Su sintaxis es:

```
Si expresion_logica Entonces
    acciones_por_verdadero
Fin Si
```

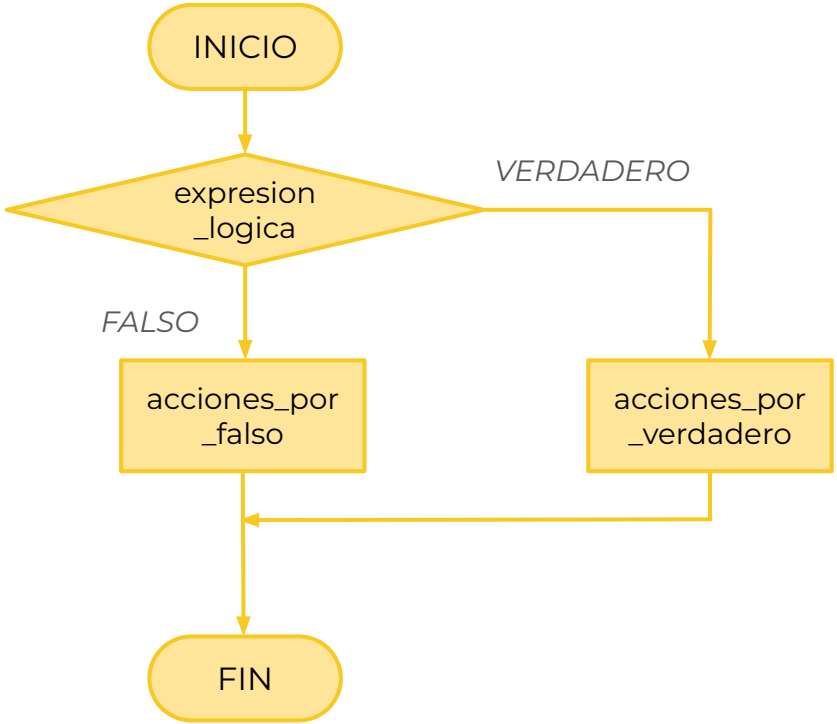
En *expresion_logica* se evalúa una condición. Si es verdadera se ejecutará el bloque *acciones_por_verdadero*.



Condicional doble

El **condicional doble** permite seleccionar la ejecución de uno de dos bloques de código. Uno se ejecuta ***sólo si*** la evaluación de la *expresion_logica* es **VERDADERA**. El otro se ejecuta ***sólo si*** la evaluación de la *expresion_logica* es **FALSA.**

```
Si expresion_logica Entonces
    acciones_por_verdadero
SiNo
    acciones_por_falso
Fin Si
```



Anidamiento de estructuras

El anidamiento de condicionales en programación se refiere a la práctica de incorporar estructuras condicionales dentro de otras, formando una jerarquía de evaluaciones. Esta técnica permite manejar situaciones complejas al tomar decisiones basadas en múltiples condiciones.
Es una herramienta poderosa, pero su uso debe equilibrarse. Se recomienda limitar el anidamiento para mantener un código claro y fácilmente comprensible. En situaciones simples, estructuras condicionales independientes pueden ser más apropiadas.

Anidamiento de condicionales | Ventajas

En anidamiento de condicionales, bien utilizado, presenta varias ventajas:

- **Granularidad:** Permite una clasificación detallada de casos al evaluar condiciones específicas en diferentes niveles.
- **Flexibilidad:** Facilita la adaptación del flujo del programa a situaciones variadas, mejorando la capacidad de respuesta.
- **Claridad en Casos Específicos:** Permite tratar casos específicos de manera detallada, lo que puede ser esencial en algoritmos más complejos.
- **Abstracción Jerárquica:** Crea una estructura jerárquica que refleja la lógica del problema, facilitando la comprensión del código.

Anidamiento de condicionales | Desventajas

Sin embargo, su uso descuidado puede tener algunos inconvenientes:

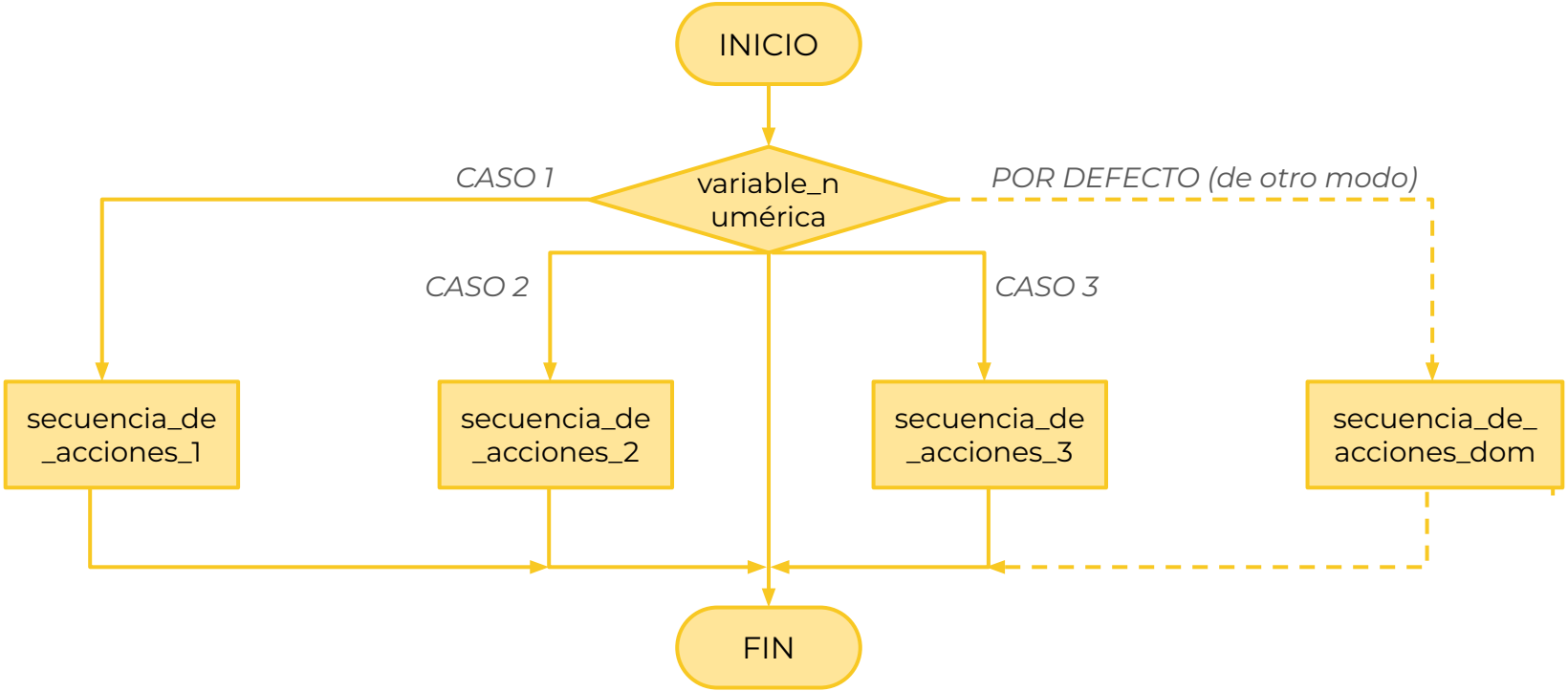
- **Complejidad:** A medida que se anidan más condicionales, la complejidad del código aumenta, lo que puede dificultar la comprensión y mantenimiento.
- **Posible Confusión:** El anidamiento excesivo puede llevar a la confusión, especialmente si no se documenta adecuadamente.
- **Dificultad en la Depuración:** La identificación y corrección de errores puede volverse más compleja al usar anidamientos profundos.
- **Posibilidad de Redundancia:** Se corre el riesgo de redundancias y repeticiones en la lógica condicional.

Condicional múltiple

La **estructura de control múltiple** en programación es una herramienta que permite ejecutar diferentes bloques de código según el valor de una variable o expresión, y es una alternativa organizada y legible a los múltiples condicionales.

En PSeInt, la estructura de control que permite ejecutar diferentes bloques de código según el valor de una variable se llama **"Según"**.

Condicional múltiple



Operadores lógicos

En programación, los **operadores lógicos** “AND”, “OR” y “NOT” son herramientas para evaluar y combinar condiciones. Permiten construir expresiones lógicas a partir de valores booleanos.

“**AND**” devuelve verdadero solo si ambas condiciones son verdaderas, “**OR**” es verdadero si al menos una de las condiciones es verdadera, y “**NOT**” invierte el valor de una condición.

Estos operadores proporcionan la capacidad de crear algoritmos flexibles y adaptativos en la programación.

Operador NOT (!) | Tabla de verdad

La tabla de verdad de este operador es es sencilla, ya que únicamente invierte el valor lógico.
En la columna de la izquierda, tenemos las entradas. En la de la derecha, las salidas, es decir, el resultado lógico de aplicar el operador a la entrada:

Entrada	!Entrada
Verdadero	Falso
Falso	Verdadero

Operador AND (&) | Tabla de verdad

El operador lógico **AND** es utilizado para combinar dos condiciones y produce un resultado verdadero (*true*) **solo si ambas condiciones son verdaderas**.

Entradas		Salida
A	**B**	**A & B**
Verdadero	**Verdadero**	**Verdadero**
Verdadero	**Falso**	**Falso**
Falso	**Verdadero**	**Falso**
Falso	**Falso**	**Falso**

Operador OR (|) | Tabla de verdad

El operador lógico OR se utiliza para combinar dos condiciones y produce un resultado verdadero (*true*) **si al menos una de las condiciones es verdadera**.

Entradas		Salida
A	**B**	**A | B**
Verdadero	**Verdadero**	**Verdadero**
Verdadero	**Falso**	**Verdadero**
Falso	**Verdadero**	**Verdadero**
Falso	**Falso**	**Verdadero**

Estructuras repetitivas

Las **estructuras repetitivas** son elementos clave que permiten la ejecución repetida de un bloque de código mientras se cumpla una **condición** específica.

Estas estructuras son esenciales para la implementación de bucles en algoritmos, ofreciendo una forma eficiente de realizar tareas repetitivas y controlar dinámicamente la ejecución del programa en **PSeInt**.

Existen distintas estructuras repetitivas en PSeint, aquí trabajaremos con **Mientras...Hacer** .

Estructuras repetitivas | Conceptos importantes

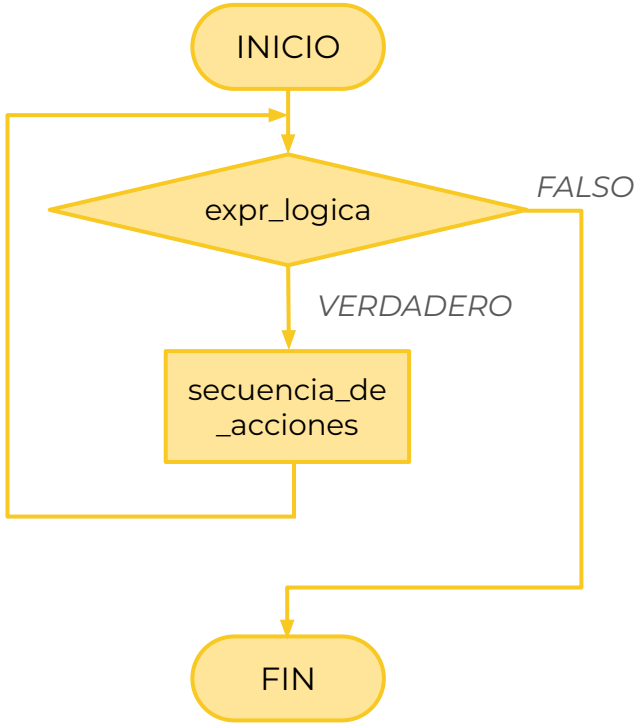
- **Bucle:** Es una estructura de control de flujo que repite un bloque de instrucciones un número determinado de veces. Es un sinónimo de *estructura repetitiva*.
- **Iteración:** Llamamos iteración a cada una de las veces que se produce la ejecución del bloque de instrucciones que contiene el bucle.
- **Condición:** Su valor de verdad determina si el bucle se repite una vez mas, o si ha finalizado. Todo bucle tiene que llevar asociada una condición, y debemos asegurarnos de que dicha condición en algún momento sea falsa, para finalizar la ejecución del bucle.

Estructura Mientras...Hacer

En PSeint, la estructura repetitiva **Mientras... Hacer** se utiliza para crear un bucle que ejecuta un conjunto de instrucciones mientras una condición específica sea verdadera. La sintaxis básica de esta estructura es la siguiente:

```
Mientras expresion_logica Hacer
    secuencia_de_acciones
Fin Mientras
```

expresion_logica es evaluada en cada iteración. En caso de ser verdadera se ejecutará la *secuencia_de_acciones*.



Contador

En programación, un **contador** dentro de una estructura repetitiva **es una variable que se incrementa o decrementa en cada iteración.**

Se utiliza para contar el número de repeticiones y controlar la ejecución del bucle. Se inicializa antes de la estructura repetitiva y se actualiza dentro del bucle. Su valor nuevo es el resultado del valor anterior más una constante (en general el valor 1).
Los contadores son valiosos para controlar el flujo de ejecución y realizar tareas repetitivas una cantidad específica de veces, evitando caer en un ciclo infinito.

Acumulador

Un **acumulador** dentro de una estructura repetitiva **es una variable que se utiliza para sumar o acumular valores en cada iteración del bucle**. Este tipo de variable se inicializa antes de entrar al bucle y se actualiza en cada repetición. Su valor nuevo es el resultado del valor anterior más una variable.

Los acumuladores son útiles para llevar un seguimiento de la suma total de valores y son comúnmente utilizados para calcular promedios, totales o realizar otras operaciones que involucren la acumulación de datos durante la ejecución del programa.

Bandera

Una variable **bandera** es una variable *booleana* (es decir, que puede tener dos valores: verdadero o falso) que se utiliza para indicar el estado de una condición o evento en un programa. Su nombre, "*bandera*" o "*flag*", sugiere que se utiliza como una señal para marcar o indicar algo. La variable bandera generalmente comienza con un valor inicial, y su valor se modifica si se cumple o deja de cumplirse cierta condición.

Actúa como una señal o indicador que puede cambiar su estado durante la ejecución del programa, y permite controlar ciclos donde no está definido el número de iteraciones.

Desafíos

Desafío 1: Piedra, papel o tijera (versión 2)

Con lo aprendido hasta ahora podemos implementar una versión mejorada del programa que escribimos para jugar a “piedra, papel o tijera”.

La nueva versión debe correr hasta que el usuario decida que no quiere continuar jugando (ingresando “s” o “n”). Además, debe contar cuantas partidas gano el usuario y cuantas la computadora.

Como desafío adicional, piensa en qué modificaciones debes hacer si deseas que se realice un campeonato que conste sólo de 5 partidas.

¿Podrías implementar cambios para que el número de partidas sea previamente ingresado por el usuario?

Desafío 2: Cálculo de Suma de Números Pares

Escribe un programa que solicite al usuario ingresar un número *n*.

Utiliza un bucle "*Mientras...Hacer*" para calcular la suma de los números pares desde 1 hasta *n*.

Muestra el resultado al finalizar el bucle.

Desafío 3: Tabla de Multiplicar Personalizada

Crea un programa que solicite al usuario ingresar un número *base* y otro número *límite*.

Utiliza un bucle "*Mientras...Hacer*" para mostrar la tabla de multiplicar de la *base* hasta el *límite* ingresado.

Desafío 4: Validación de Entrada

Diseña un programa que solicite al usuario ingresar un número.

Utiliza un bucle "*Mientras...Hacer*" para asegurarte de que el número ingresado sea positivo.

Si el usuario ingresa un número negativo, muestra un mensaje indicando que la entrada no es válida y solicita nuevamente el número.

No te olvides de dar el presente

Recordá:

- Revisar la Cartelera de Novedades.
- Hacer tus consultas en el Foro.
- Realizar el Ejercicio de Repaso.

Todo en el Aula Virtual.

Muchas gracias por tu atención.

Nos vemos pronto