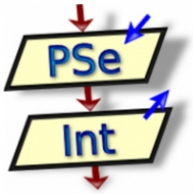
CURSO DE PROGRAMACIÓN FULL STACK

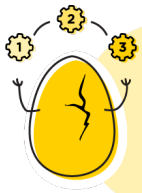
INTRODUCCIÓN A LA PROGRAMACIÓN CON PSEINT



EGG

¡¡NUESTRA PRIMERA GUÍA!!

Esta es tu primera guía, en la que vamos a empezar a ver algunos conceptos teóricos del mundo de la programación y el desarrollo web. Pero antes, queremos explicarte cómo debes leer las guías y qué encontrarás de ahora en adelante.



Cuando veas el indicador AMARILLO, estaremos explicando metodología de trabajo. Cómo deberás trabajar con tu equipo.

El indicador NARANJA te guiará respecto al contenido que estudiarás en esa guía. Te permitirá llevar un control de los conocimientos que debas adquirir y a hacer un seguimiento del mismo. No sólo en los aspectos técnicos, sino también en la implementación de la metodología. Te recomendamos que te tomes un tiempo para analizar si comprendiste correctamente la teoría, los ejemplos y pudiste aplicar lo aprendido en los ejercicios.

El indicador AZUL te mostrará ejemplos gráficos de la teoría y prácticos. Esto te permitirá esclarecer conceptos y llevar a la práctica los mismos.

El indicador MORADO te invitará a poner en práctica el concepto sobre el cual estabas leyendo. Suelen ser ejercicios muy breves en donde aplicarás sólo un concepto, o aquellos que ya hayas aprendido, más el que estás incorporando.

Encontrarás otro apartado llamado “MANOS A LA OBRA – CORRECCIÓN DE ERRORES” en dónde deberás copiar y pegar un ejercicio en tu IDE, chequear los errores en la consola y corregirlos de tal manera que, al ejecutar el programa, la consola se vea igual a la captura del ejercicio.

El indicador VERDE te proveerá tips, consejos y recomendaciones.

Objetivos de la Guía

En esta guía aprenderemos a:

- Definir todos los tipos de variables y nombrarlas correctamente.
- Asignarles valor a las variables.
- Utilizar métodos de escritura para mostrar mensajes por pantalla.
- Utilizar métodos de lectura para ingresar valores por teclado y alojarlo en las variables.
- Operar con las variables.
- Cooperar con el equipo de trabajo.
- Pedir ayuda al facilitador para realizar un ejercicio.
- Ayudar a un compañero a repasar la teoría o realizar un ejercicio en conjunto.

¿QUÉ ES LA PROGRAMACIÓN?

La programación es el acto de programar, es decir, **organizar una secuencia de pasos ordenados a seguir para hacer cierta cosa.** Este término puede utilizarse en muchos contextos. Por ejemplo, programar una salida con amigos, organizar unas vacaciones, etc.

En informática el término **programación** se refiere a la acción de crear programas y **programar** es la serie de instrucciones, que le vamos a dar a nuestra máquina, para lograr lo que nuestro programa necesite para funcionar.

Las partes que componen a nuestro programa son **el lenguaje de programación y los algoritmos.**

MANOS A LA OBRA!

EJERCICIO ALARMA

Ahora que sabemos que es programar, antes de ver en profundidad lenguajes de programación y algoritmos, vamos a programar algo muy sencillo.

Vamos a programar una alarma en el celular. Deberás entrar a su celular y poner una alarma para que suene a la hora en la que debes conectarte a tu clase, lo importante es que vean como al celular le damos una serie de instrucciones para realizar una tarea, que sería, hacer ruido a una hora que nosotros decidamos.

¿QUÉ ES UN LENGUAJE DE PROGRAMACIÓN?

Es un lenguaje formal que, mediante una serie de instrucciones, le permite a un programador escribir un conjunto de órdenes, acciones consecutivas, datos y algoritmos para, de esa forma, **resolver problemas.**

Las instrucciones que sigue la computadora para la creación de programas están escritas en un lenguaje de programación y luego son traducidas a un lenguaje de máquina que puede ser interpretado y ejecutado por el hardware del equipo.

Hay distintos tipos de lenguajes de programación:

- **Lenguaje máquina:** Es el más primitivo de los lenguajes y es una colección de dígitos binarios o bits (0 y 1) que la computadora lee e interpreta y son los únicos idiomas que las computadoras entienden. Ejemplo: **10110000 01100001**

- **Lenguajes de alto nivel:** Tienen como objetivo facilitar el trabajo del programador, ya que utilizan unas instrucciones más fáciles de entender.

Además, el lenguaje de alto nivel permite escribir códigos mediante idiomas que conocemos (español, inglés, etc.) y luego, para ser ejecutados, se traduce al lenguaje máquina mediante traductores o compiladores.

¿QUÉ ES UN ALGORITMO?

En el apartado anterior vimos que los lenguajes de programación son nuestro puente para poder comunicarnos con la máquina. Y de esa forma **darle instrucciones claras**, para poder **solucionar los problemas** que puede presentar la creación de un programa.

Estas instrucciones que le vamos a dar a nuestro programa, se conocen **como algoritmos**. Un algoritmo es un ***método*** para darle instrucciones a nuestro programa y resolver un problema.

Este consiste en la realización de un conjunto de pasos ***lógicamente ordenados*** tal que, partiendo de la información que le demos, permite obtener ciertos resultados que conforman la solución del problema.

Los algoritmos son **independientes** tanto del lenguaje de programación en que se expresan como de la computadora que los ejecuta. En cada problema el algoritmo se puede expresar en un lenguaje diferente de programación y ejecutarse en una computadora distinta; sin embargo, el algoritmo **será siempre el mismo**. Así, por ejemplo, en una analogía con la vida diaria, una receta de un plato de cocina se puede expresar en español, inglés o francés, pero cualquiera que sea el lenguaje, los pasos para la elaboración del plato se realizan sin importar el idioma del cocinero.

Los **algoritmos** son más importantes que los **lenguajes de programación** o las **computadoras**. Un lenguaje de programación es sólo un medio para expresar un algoritmo y una computadora es solo un procesador para ejecutarlo. Tanto el lenguaje de programación como la computadora son los medios para obtener un fin: conseguir que el algoritmo se ejecute y se efectúe el proceso correspondiente.

CARACTERÍSTICAS FUNDAMENTALES DE LOS ALGORITMOS

Las características fundamentales de los algoritmos son:

- Un algoritmo debe ser **preciso** e indicar el orden de realización de cada paso.

- Un algoritmo debe estar específicamente **definido**. Es decir, si se ejecuta un mismo algoritmo dos veces, con los mismos datos de entrada, se debe obtener el mismo resultado cada vez.

- Un algoritmo debe ser **finito**. Si se sigue un algoritmo, se debe terminar en algún momento; o sea, debe tener un número finito de pasos. Debe tener un inicio y un final.
- Un algoritmo debe ser **correcto**: el resultado del algoritmo debe ser el resultado esperado.
- Un algoritmo es **independiente** tanto **del lenguaje de programación** en el que se expresa **como de la computadora** que lo ejecuta.

El programador debe constantemente resolver problemas de manera algorítmica, lo que significa plantear el problema de forma tal que queden indicados los pasos necesarios para obtener los resultados pedidos, a partir de los datos conocidos. Lo anterior implica que un algoritmo básicamente consta de tres elementos: ***Datos de Entrada o Información de entrada, Procesos y la Información de Salida.***

Estructura de un Programa: Datos de entrada, proceso y Salida.

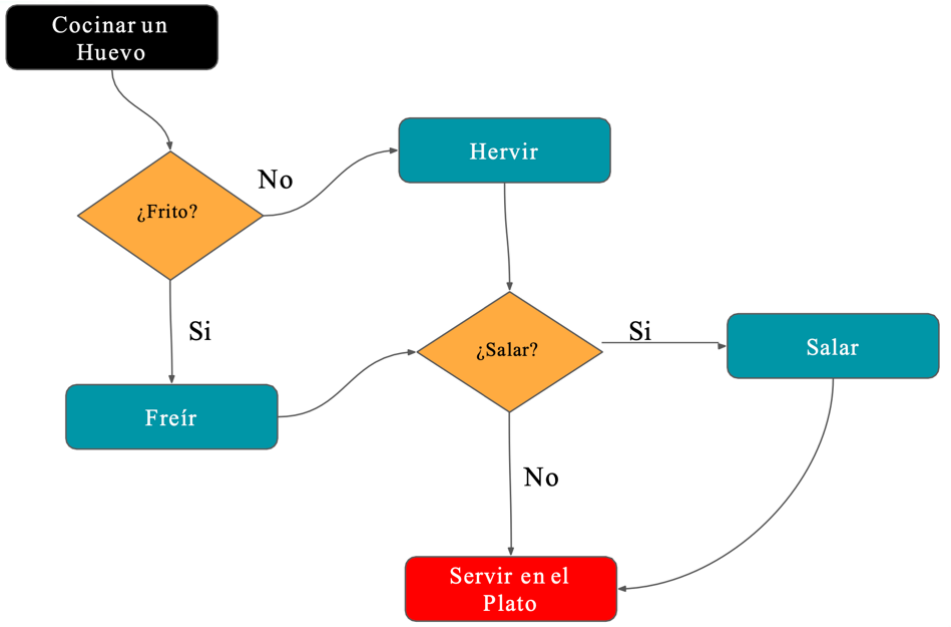
¿CÓMO REPRESENTAR UN ALGORITMO?

La mejor manera de representar un algoritmo, es a través de un diagrama de flujo.

Un Diagrama de Flujo representa la esquematización gráfica de un algoritmo, el cual muestra gráficamente los pasos o procesos a seguir para alcanzar la solución de un problema.

Se basan en la utilización de diversos símbolos para representar operaciones específicas, es decir, es la representación gráfica de las distintas operaciones que se tienen que realizar para resolver un problema, con indicación expresa el orden lógico en que deben realizarse.

Vamos a ver cómo sería un diagrama de flujo, de los pasos para cocinar un huevo:



Como podemos ver, nos muestra paso a paso cómo cocinar un huevo y cada opción que tenemos a la hora de cocinar un huevo.

¿QUÉ ES EL PSEUDOCÓDIGO Y POR QUÉ VAMOS A UTILIZARLO?

El lenguaje de programación que utilizaremos en **esta parte del curso**, para representar nuestros algoritmos es el pseudocódigo.

El ***pseudocódigo*** es una herramienta de programación en la que las instrucciones se escriben en palabras similares al inglés o español, que facilitan tanto la escritura como la lectura de programas. En esencia, el pseudocódigo se puede definir como un lenguaje de especificaciones de algoritmos. El uso de tal lenguaje hace el paso de codificación final (esto es, la traducción a un lenguaje de programación) relativamente fácil. El pseudocódigo se considera un primer borrador, dado que tiene que traducirse posteriormente a un lenguaje de programación.

PROGRAMA

En los apartados anteriores, explicamos que es la programación y que está compuesta de dos partes, los lenguajes de programación y los algoritmos, pero, ¿donde se van a ver reflejados estos dos conceptos? Estos se van a ver reflejados en nuestro **programa.**

Un programa no es más que una **serie de algoritmos escritos en algún lenguaje de programación de computadoras.** Un programa es, por lo tanto, un conjunto de instrucciones —órdenes dadas a la computadora— que producirán la ejecución de una determinada tarea. En esencia, un programa es un medio para conseguir un fin. El fin será probablemente definido como la información necesaria para solucionar un problema.

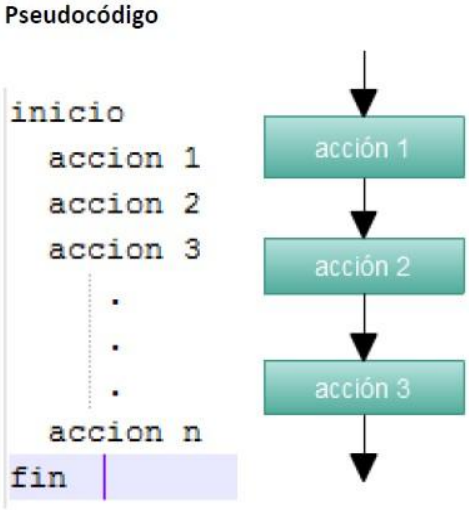
DISEÑO DEL PROGRAMA

Se puede utilizar algunas de las herramientas de representación de algoritmos, también conocidos como lenguajes de programación, para definir la secuencia de pasos que se deben llevar a cabo para conseguir el resultado que necesitamos.

ESPECIFICACIONES DE UN PROGRAMA

Tras la decisión de desarrollar un programa, el programador debe establecer el conjunto de especificaciones que debe contener el programa: entrada, salida y algoritmos de resolución, que incluirán las técnicas para obtener las salidas a partir de las entradas.

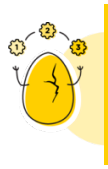
Un programa puede ser lineal (secuencial) o no lineal. Un programa es lineal si las instrucciones (acciones) se ejecutan secuencialmente como los ejercicios propuestos en esta guía, es decir, sin bifurcaciones, decisión ni comparaciones.



Estructura de un programa secuencial

CODIFICACIÓN

Una vez que tenemos las especificaciones de un programa pasaremos a la codificación del programa. La codificación es la operación de escribir la solución del problema (de acuerdo a la lógica del pseudocódigo), en una serie de instrucciones detalladas, en un código reconocible por la computadora. La serie de instrucciones detalladas se conoce como código fuente, el cual se escribe en un lenguaje de programación o lenguaje de alto nivel.



Leímos muchos conceptos nuevos hasta aquí, es un buen momento para preguntarle al resto del equipo si tienen alguna duda.

¿CÓMO DEBEN ESCRIBIRSE LOS ALGORITMOS/PROGRAMAS?

Ya sabemos que es un programa, el diseño de un programa, las especificaciones de un programa y su codificación. Ahora vamos a ver como es la escritura de estos algoritmos / programas. Un algoritmo consta de dos componentes: una cabecera de programa y un bloque algoritmo. La cabecera de programa es una acción simple que comienza con la palabra algoritmo. Esta palabra estará seguida por el nombre asignado al programa completo.

El bloque algoritmo es el resto del programa y consta de dos componentes o secciones: las ***acciones de declaración*** y las ***acciones ejecutables***.

Las declaraciones definen o declaran las variables que tengan nombres. Las acciones ejecutables son las acciones que posteriormente deberá realizar la computación cuando el algoritmo convertido en programa se ejecute.

```
algoritmo
cabecera del programa
sección de declaración
sección de acciones
```

CABECERA DEL PROGRAMA

Todos los algoritmos y programas deben comenzar con una cabecera en la que se exprese el identificador o nombre correspondiente con la palabra reservada que señale el lenguaje. En PSeInt, la palabra reservada es Algoritmo.

```
Algoritmo sin_titulo
      <Acciones>
FinAlgoritmo
```

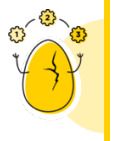
Donde la palabra sin título debe ser reemplazada por el nombre del algoritmo. Esto se vería así en Pseint.


```
1  Algoritmo sin_titulo
2
3
4  FinAlgoritmo
5
```

¿QUÉ ELEMENTOS POSEE UN PROGRAMA?

Los elementos de un programa, son básicamente, los componentes que conforman las instrucciones previamente mencionadas, para crear nuestro programa y resolver sus problemas. Estos elementos siempre estarán dentro de un algoritmo.

Los elementos de un programa son: **identificadores, variables, constantes, operadores, palabras reservadas.**



¿No recuerdas cómo identificar al facilitador de tu equipo? Es muy fácil. Ve a tu aula virtual. En el widget lateral podrás ver a tu equipo de trabajo del día. Quien tiene una insignia. **- ¿eres el facilitador? ¡Felicitaciones! ¿recuerdas cuál es tu rol? Puedes ir a refrescarlo en el Aula Virtual**

IDENTIFICADORES

Un identificador es un conjunto de caracteres alfanuméricos de cualquier longitud que sirve para identificar las entidades del programa (nombre del programa, nombres de variables, constantes, subprogramas, etc.). En PseInt los identificadores deben constar sólo de **letras, números y/o guión_bajo(_)**, comenzando siempre con una letra y **se suelen escribir siempre en minúsculas.** Estos tampoco pueden contar de tildes, ni de la letra Ñ, ya que generaría errores.

Otra cosa que es súper importante a la hora de pensar identificadores, es **poner nombres claros**, por ejemplo, si queremos tener una frase, que el identificador sea **frase** o si queremos una suma, le pondremos suma.

¿Y si necesitamos un identificador de más de una palabra?

Ahora, si queremos poner un identificador que sea de **más de una palabra**, usamos algo llamado ***camelCase***, consiste en poner la letra de la segunda palabra en mayúsculas, por ejemplo, queremos que el identificador refleje que suma números, le pondríamos **sumaNumero**s, la segunda palabra arranca con la N en mayúsculas. De está manera, mantenemos la regla de escribir siempre en minúsculas.

Esto no es solo para la segunda palabra, si queremos poner una tercera o una cuarta palabra, etc. haríamos lo mismo. Por ejemplo, **sumaNumerosEnteros**, para un identificador más claro.

VARIABLES Y CONSTANTES

Los programas de computadora necesitan **información** para la resolución de problemas. Está información puede ser un número, un nombre, etc. Para nosotros poder guardar esta información en algún lugar y que no esté “suelta”, para no perderla o poder acceder a ella cuando lo necesitemos es crucial. Para solucionar esto, vamos a guardar la información en algo llamado, **variables y constantes**. Las variables y constantes vendrían a ser como pequeñas cajas, que guardan algo en su interior, en este caso información. Estas, van a contar como previamente habíamos mencionado, con un identificador, un nombre que facilitará distinguir unas de otras y nos ayudará a saber que variable o constante es la contiene la información que necesitamos.

Dentro de toda la información que vamos a manejar, a veces, necesitaremos información que no cambie. Tales valores son constantes. De igual forma, existen otros valores que necesitaremos que cambien durante la ejecución del programa; esas van a ser nuestras **variables**.

Una variable es un objeto o tipo de datos cuyo valor puede cambiar durante el desarrollo del algoritmo o ejecución del programa. Dependiendo del lenguaje, hay diferentes tipos de variables, tales como enteras, reales, carácter, lógicas y de cadena. Una variable que es de un cierto tipo puede tomar únicamente valores de ese tipo. Una variable de carácter, por ejemplo, puede tomar como valor sólo caracteres, mientras que una variable entera puede tomar sólo valores enteros. **Ejemplo:** una variable que guardará el resultado de un cálculo, este valor puede cambiar, en alguna parte de nuestro programa.

Una constante es un dato, que al igual que la variable, puede ser de diferentes tipos como enteras, reales, carácter, lógicas y de cadena. Estas, también guardan solo valores de ese tipo, pero, permanecen sin cambios durante todo el desarrollo del algoritmo o durante la ejecución del programa. **Ejemplo:** el valor de Pi π

TIPOS DE DATOS EN PSEINT

Las variables y constantes como previamente habíamos mencionado, van a guardar información dependiendo del **tipo de dato** que le digamos que guarde esa variable. Por ejemplo, si digo que mi variable va a guardar números enteros, significa que el tipo de dato de esa variable es entero.

Los tipos de datos que podemos usar son:

- ✓ Entero: solo números enteros.
- ✓ Real: números con cifras decimales. Para separar decimales se utiliza el punto. Ejemplo: **3.14**
- ✓ Carácter: cuando queremos guardar un carácter. Los Caracteres se encierran entre comillas simples. un carácter (unidimensional): **‘a’, 'A'.**
- ✓ Lógico: cuando necesitamos guardar una expresión lógica (verdadero o falso)
- ✓ Cadena: cuando queremos guardar cadenas de caracteres. Las Cadenas se encierran entre comillas dobles. una cadena (multidimensional): **“esto es una cadena”, "hola mundo"**

Notas:

- ✓ Cadena y Carácter son términos equivalentes, no genera error que las escribamos indistintamente.
- ✓ El plural de Carácter es Caracteres o Cadena.

¿CÓMO CREO UNA VARIABLE?

A la hora de crear nuestra variable, vamos a tener que darle un **identificador, por lo que usaremos las reglas vistas en el apartado de identificadores**, y el tipo de dato que necesitamos que guarde. Para esto vamos a utilizar la palabra reservada **Definir**. La instrucción definir permite explicitar el tipo de una o más variables.

Después de la palabra **Definir**, va el nombre de la variable y por último el tipo de dato de la variable. Normalmente los identificadores de las variables y de las constantes con nombre deben ser declaradas en los programas antes de ser utilizadas. Entonces, la sintaxis de la declaración de una variable suele ser:

```
Definir <nombre_variable> como <tipo_de_dato>
```

Si queremos declarar una variable de tipo entero se escribe:

```
Definir varNumero Como Entero
```

varNumero se convierte en una variable de tipo entero.


```
Definir num Como Entero
```

EJERCICIO CREAR VARIABLE

Vamos a poner en práctica lo que acabamos de ver, vamos a crear en PseInt, una variable de cada tipo de dato posible.

Palabras Reservadas

Palabras que dentro del lenguaje significan la ejecución de una instrucción determinada, por lo que no pueden ser utilizadas con otro fin. En Pseudocodigo, las palabras reservas aparecen de color azul. Por ejemplo, la palabra **Algoritmo** y **FinAlgoritmo**.

DETECCIÓN DE ERRORES

Copia y pega el código que está a continuación, ahí vas a ver que el código tiene mal escritas las palabras reservadas, tu tarea es arreglar el código.

Para que pueda comprender mejor este tipo de ejercitación te dejamos el siguiente video:

```
Algoritmo Definicion_Variables
      Definir algoritmo Como Logico
      Definir numero Como Entero
      Definir cadena Como Caracter
FinAlgoritmo
```

Revisemos lo aprendido hasta aquí

- Definir variables de tipo lógico, entero, real y cadena.
- Nombrar correctamente las variables sin utilizar palabras reservadas.

TIPOS DE INSTRUCCIONES

Las instrucciones —acciones— básicas que se pueden implementar de modo general en un algoritmo y que esencialmente soportan todos los lenguajes son las siguientes:

INSTRUCCIONES DE INICIO/FIN

Son utilizadas para delimitar bloques de código. Por ejemplo, Algoritmo y FinAlgoritmo.

INSTRUCCIONES DE ESCRITURA O SALIDA

Ayer estuvimos trabajando un poco con esto, ahora si, te explicaremos cómo funcionan estas instrucciones.

Se utilizan para escribir o mostrar mensajes o contenidos de las variables en un dispositivo de salida. La salida puede aparecer en un dispositivo de salida (pantalla, impresora, etc.). La operación de salida se denomina escritura (escribir). En la escritura de algoritmos las acciones escritura se representa con el siguiente formato:

Escritura o salida

La instrucción Escribir permite mostrar información y valores de variables en la interfaz grafica o ambiente.

```
Escribir <exprl> , <expr2> , ... , <exprN>
```

Esta instrucción imprime en la interfaz grafica o ambiente (en este caso en la pantalla) los valores obtenidos de evaluar N expresiones, el valor de un variable o de un mensaje. Dado que puede incluir una o más expresiones, mostrará uno o más valores.

¿NECESITAS UN **EJEMPLO**?

```
1  Algoritmo ejemploEscribir
2
3    Definir num Como Entero
4
5    Escribir "Hola mundo"
6
7    Escribir 2 + 2
8
9    num = 10
10
11   Escribir num
12
13
14 FinAlgoritmo
```

```
PSeInt - Ejecutando proceso EJEMPLOESCRIBIR
*** Ejecución Iniciada. ***
Hola mundo
4
10
*** Ejecución Finalizada. ***
No cerrar esta ventana  Siempre visible  Reiniciar
```

En este ejemplo de escribir, vemos que nuestro primer escribir muestra un mensaje o cadena, que va entre comillas dobles, después nuestro segundo escribir muestra el resultado de una suma de dos números y nuestro último escribir, muestra el valor de una variable de tipo entero a la que se le asignó un valor previo.

Con el escribir también podemos mostrar variables o valores con un mensaje previo, para esto vamos a concatenar nuestra variable, usando una coma o un espacio en blanco, con un mensaje entre comillas.

¿NECESITAS UN **EJEMPLO**?

```
Escribir "Mensaje entre comillas" variable

Escribir "La suma de los números es:" suma
```

MANOS A LA OBRA!

EJERCICIO ESCRIBIR

Escribir un algoritmo en el cual se muestre nuestro nombre completo en la interfaz gráfica de PseInt.

Revisemos lo aprendido hasta aquí

- Utilizar instrucciones de escritura para mostrar mensajes por pantalla al ejecutar el programa.

INSTRUCCIONES DE LECTURA

Los cálculos que realizan las computadoras requieren, para ser útiles la entrada de los datos necesarios para ejecutar las operaciones que posteriormente se convertirán en resultados, es decir, salida.

Las operaciones de entrada permiten leer datos de un dispositivo de entrada y asignarlos a determinadas variables.

Esta entrada se conoce como operación de lectura (leer). Los datos de entrada se introducen al procesador mediante dispositivos de entrada (teclado, tarjetas perforadas, unidades de disco, etc.).

Lectura o entrada

La instrucción Leer permite ingresar información por teclado al usuario a través de la interfaz gráfica o ambiente de Pseint. Que se mostrará al correr nuestro algoritmo

```
Leer <variable1>, <variable2>, ..., <variableN>
```


```
1  Algoritmo ejemploLeer
2
3    Definir num como entero
4
5    Leer num
6
7
8  FinAlgoritmo
9
```

```
PSeInt - Ejecutando proceso EJEMPLOLEER
*** Ejecución Iniciada. ***
> 5
```

En este ejemplo definimos una variable de tipo entero llamada **num** y le asignamos un valor a través de la instrucción **Leer**.

EJERCICIO LECTURA Y ESCRITURA

Escribir un algoritmo en el cual el programa nos pida nuestro nombre completo, lo aloje en una variable de tipo Cadena llamada nombreCompleto y después mostrarlo usando la instrucción Escribir.

Revisemos lo aprendido hasta aquí

- Utilizar instrucciones de lectura para pedirle al usuario que guarde valores en las variables por pantalla al ejecutar el programa.

¿CÓMO ASIGNAMOS VALORES A LAS VARIABLES?

La instrucción de asignación permite almacenar un valor en una variable (previamente definida). Esta es nuestra manera de guardar información en una variable, para utilizar ese valor en otro momento. Se puede realizar de dos maneras, con el signo igual o una flecha:

```
<variable> <- <expresión>
```

```
<variable> = <expresión>
```

expresión es igual a una expresión matemática o lógica, a una variable o constante.

Al ejecutarse la asignación, primero se evalúa la expresión de la derecha y luego se asigna el resultado a la variable de la izquierda. El tipo de la variable y el de la expresión deben coincidir.


```
Algoritmo ejemploAsignacion

  Definir num Como Entero

  num = 4

FinAlgoritmo

```

En este ejemplo estamos definiendo una variable como entero y después asignándole un valor, en este caso el número 4.

EJERCICIO DEFINIR

Escribe un algoritmo definiendo la variable nombre como cadena, asigna allí tu información y luego muéstralo por pantalla escribiendo la variable.

OPERADORES

Este pseudolenguaje dispone de un conjunto básico de operadores que pueden ser utilizados para la construcción de expresiones más o menos complejas.

OPERADORES ALGEBRAICOS

Los operadores algebraicos o también conocidos como operadores aritméticos. Realizan operaciones aritméticas básicas: suma, resta, multiplicación, división, potenciación y modulo para datos de tipo numérico tanto enteros como reales. Estas operaciones son binarias porque admiten dos operandos.

Operador	Significado	Resultado
Algebraicos		
+	Suma	suma = 2 + 2
-	Resta	resta = 10 - 4
*	Multiplicación	multiplicación = 10 * 2
/	División	división = 9 / 3
^	Potenciación	potencia = 10 ^ 2
% o MOD	Módulo (resto de la división entera)	resto = 4 % 2

Reglas de prioridad:

Las expresiones que tienen dos o más operadores requieren unas reglas matemáticas que permitan determinar el orden de las operaciones, se denominan reglas de prioridad y son:

1. Las operaciones que están encerradas entre paréntesis se evalúan primero. Si existen diferentes paréntesis anidados (interiores unos a otros), las expresiones más internas se evalúan primero.
2. Las operaciones aritméticas dentro de una expresión suelen seguir el siguiente orden de prioridad:

   ✓ operador ( )

   ✓ operadores unitarios (potenciación),

   ✓ operadores *, /, % (producto, división, módulo) ✓ operadores +, – (suma y resta).

En caso de coincidir varios operadores de igual prioridad en una expresión o sub expresión encerrada entre paréntesis, el orden de prioridad en este caso es de ***izquierda a derecha,*** y a esta propiedad se denomina asociatividad.

MANOS A LA OBRA!

EJERCICIO VARIABLES

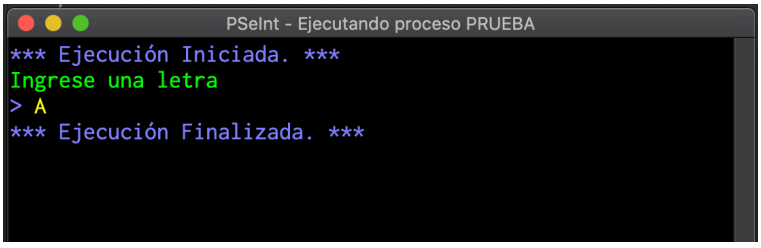
Define dos variables que guarden números enteros, defina una tercera variable donde aloje el resultado se sumarán ambas variables. Comente su código indicando qué finalidad tiene cada línea.

DETECCIÓN DE ERRORES

¿Puedes corregir esta porción de código para que cumpla el resultado esperado?

```
Algoritmo Prueba

Definir letra Como Entero

Escribir ingrese una letra

letra

FinAlgoritmo
```

¿CUÁL ES EL RESULTADO A LOGRAR?



Revisemos lo aprendido hasta aquí

- Asignar valor a las variables manualmente y operar con ellas.

PASOS PARA LA CONSTRUCCIÓN DE UN PROGRAMA

Hemos llegado al final de la guía, ahora pondremos en practica todo lo visto a través de una serie de ejercicios, pero, antes que arranquemos los ejercicios es importante entender los **pasos para construir un programa** y de esa forma trabajar de la mejor manera posible!!

El proceso de programación es un proceso de solución de problemas en el cual deben llevarse a cabo los pasos descritos a continuación.

¿ CÓMO VAMOS A ENCARAR ESTOS PROBLEMAS ?

Para poder resolver problemas vamos a tener que ejecutar una serie de pasos que nos van a ayudar a resolver el problema sin importar que tan grande o chico sea.

Los pasos serían:

1. **Lectura**

Leer el problema o la consigna dos veces, la primera para entender de manera general lo que debemos hacer y una segunda vez para entender el problema de manera más concreta y evitar saltarnos algún dato importante.

2. **Papel y lápiz**

Una vez que tenemos una idea clara de lo que debemos hacer, vamos a dejarlo por escrito, para esto lo mejor es usar papel y lápiz. Vamos a dejar por escrito que debemos realizar. Utilizamos la misma noción que cuando lo leímos, escribimos un esbozo general y luego un esbozo particular.

3. **Subproblemas**

Cuando estemos haciendo el esbozo particular, pensamos en el concepto divide y vencerás. Pensamos el problema que tenemos y lo dividimos en subproblemas. Un ejemplo sería hacer pan. Hacer pan es un problema que lo podemos dividir en subproblemas. Mezclar la harina con la levadura, amasar el pan, dejarlo levar, etc.

Básicamente, tomamos un problema grande y general para convertirlo en pequeños problemas más concretos y fáciles de afrontar.

4. **Herramientas del código**

En el siguiente paso, empezamos a pensar en código, antes estábamos tomando un problema de programación y resolviéndolo como si fuera un problema como hacer pan. Ahora vamos a tener que tomar esos subproblemas que habíamos creado y ver que herramientas de programación vamos a necesitar. Esto puede ser variables, constantes, bucles, condicionales, expresiones lógicas o matemáticas, etc. Cualquier herramienta que creamos necesaria para lograr nuestro cometido.

5. Pasaje a código

Y, por último, tomando todo lo que hicimos, lo pasaremos a código. Es importante que cuando empecemos a escribir el código y empecemos a trasladar los subproblemas a código, ir probando que cada subproblema funcione, o cumpla con el resultado esperado. Para facilitar este proceso Pseint, nos presenta la herramienta de ejecutar el código Paso a Paso y la prueba de escritorio.

¿ME BLOQUEE EN CUANTO A CÓDIGO?

- ¡¡Si hay errores rojos!! Leer la descripción del error y la línea donde está el error.
- Indentar el código, para tenerlo bien ordenado. Para indentar, tenemos que seleccionar todo el código, hacer click derecho y click izquierdo en indentar.
- ¿El código hace lo que quiero que haga? En caso que no, correr el código paso a paso y hacer prueba de escritorio.

HICE TODO LO ANTERIOR Y NO PUDE RESOLVERLO

- Primero consulto a mis compañeros como ellos encararon el ejercicio. Les pido que me expliquen en vez de mostrarme lo que hicieron.
- También puedes orientarte en cómo ir resolviendo los subproblemas en internet, es muy difícil encontrar la solución puntual, pero puedes orientarte en buscar parte de la solución
- Si no lo saben tus compañeros ni lo encontraste en internet, consúltale al o la profe de cómo se debería encarar la solución. El o la profe te dará las herramientas necesarias para resolverlo. En el caso de que después de la explicación del profe, seguís sin entender, plantea que parte no entendiste.

EJERCICIOS DE APRENDIZAJE

¡¡Vamos a poner en práctica todo lo que hemos visto en esta guía con los siguientes ejercicios y recordemos que lo más importante es disfrutar de nuestro aprendizaje!!

VIDEOS: Te sugerimos ver los videos relacionados con este tema, antes de empezar los ejercicios, los podrás encontrar en tu aula virtual o en nuestro canal de YouTube.

Los ejercicios van a tener el siguiente filtro de dificultad:

Dificultad Baja | Dificultad Media | Dificultad Alta

1. Conocido el número en matemática PI π, pedir al usuario que ingrese el valor del radio de una circunferencia y calcular y mostrar por pantalla el área y perímetro. Recuerde que para calcular el área y el perímetro se utilizan las siguientes fórmulas:

   area = PI * radio^2

   perimetro = 2 * PI * radio

2. Escribir un programa que calcule el precio promedio de un producto. El precio promedio se debe calcular a partir del precio del mismo producto en tres establecimientos distintos.

3. A partir de una conocida cantidad de metros que el usuario ingresa a través del teclado se debe obtener su equivalente en centímetros, en milímetros y en pulgadas.

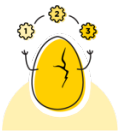
   Ayuda: 1 pulgada equivale a 2.54 centímetros.

4. Escribir un programa que calcule cuántos litros de combustible consumió un automóvil. El usuario ingresará una cantidad de litros de combustible cargados en la estación y una cantidad de kilómetros recorridos, después, el programa calculará el consumo (km/lt) y se lo mostrará al usuario.

5. Escriba un programa que permita al usuario ingresar el valor de dos variables numéricas de tipo entero. Posteriormente, el programa debe intercambiar los valores de ambas variables y mostrar el resultado final por pantalla.

   Por ejemplo, si el usuario ingresa los valores num1 = 9 y num2 = 3, la salida a del programa deberá mostrar: num1 = 3 y num2 = 9

   Ayuda: Para intercambiar los valores de dos variables se debe utilizar una variable auxiliar.



¿Lograste los objetivos de la guía?

¡Terminamos los ejercicios fundamentales de esta guía! **Te invitamos a ir a tu aula virtual y contestar una breve encuesta para revisar los objetivos de la esta guía**, reflexiona sobre cada uno de ellos haciéndote la siguiente pregunta ¿Pude aplicar este objetivo al realizar los ejercicios? Si es así, ¡Excelente!

¿Terminaste con los ejercicios fundamentales? hay más ejercicios para que puedas seguir practicando.

¿No pudiste cumplir los objetivos de la guía?

Si aún no has logrado tildar los objetivos propuestos, **no te preocupes**, a través de Slack contacta a un Coach para que te ayude a diseñar una estrategia personalizada para entender qué pasó y poder avanzar.

EJERCICIOS DE APRENDIZAJE EXTRA

Estos van a ser ejercicios para reforzar los conocimientos previamente vistos. Estos pueden realizarse cuando hayas terminado la guía y tengas una buena base sobre lo que venimos trabajando. Además, si ya terminaste la guía y te queda tiempo libre, podés continuar con estos ejercicios extra, recordando siempre que no es necesario que los termines para continuar con el tema siguiente. Por último, recordá que la prioridad es ayudar a los compañeros de tu equipo y que cuando tengas que ayudar, lo más valioso es que puedas explicar el ejercicio con la intención de que tu compañero lo comprenda, y no sólo mostrarlo. ¡Muchas gracias!

1. Un colegio desea saber qué porcentaje de niños y qué porcentaje de niñas hay en el curso actual. Diseñar un algoritmo para este propósito. Recuerda que para calcular el porcentaje puedes hacer una regla de 3 simple. El programa debe solicitar al usuario que ingrese la cantidad total de niños, y la cantidad total de niñas que hay en el curso.

2. Solicitar al usuario que ingrese la base y altura de un rectángulo, y calcular y mostrar por pantalla el área y perímetro del mismo

   area = base * altura

   perimetro = 2 * altura + 2 * base.

3. Escribir un programa que calcule el volumen de un cilindro. Para ello se deberá solicitar al usuario que ingrese el radio y la altura. Mostrar el resultado por pantalla.

   volumen = π * radio2 * altura

4. A partir de una conocida cantidad de días que el usuario ingresa a través del teclado, escriba un programa para convertir los días en horas, en minutos y en segundos. Por ejemplo

   1 día = 24 horas = 1440 minutos = 86400 segundos

5. Crear un programa que solicite al usuario que ingrese el precio de un producto al inicio del año, y el precio del mismo producto al finalizar el año. El programa debe calcular cuál fue el porcentaje de aumento que tuvo ese producto en el año y mostrarlo por pantalla.