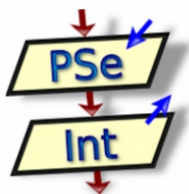


CURSO DE PROGRAMACIÓN FULL STACK

Introducción a la Programación con PSeInt



Guía de Programación con PSeInt

RESOLUCIÓN DE PROBLEMAS

El programador de computadora es antes que nada una persona que resuelve problemas, por lo que para llegar a ser un programador eficaz se necesita aprender a resolver problemas de un modo riguroso y sistemático. Para poder resolver un problema hay que comprender su contexto.

CONTEXTO DE UN PROBLEMA

El *contexto de un problema* implica la realidad cercana a un determinado problema, se refiere a todo aquello que rodea, ya sea física o simbólicamente, a un acontecimiento. A partir del contexto, por lo tanto, se puede interpretar o entender un hecho para luego proponer una solución, por ejemplo, a través de un algoritmo.

Algoritmo

Un algoritmo es un *método* para resolver un problema, que consiste en la realización de un conjunto de pasos *lógicamente ordenados* tal que, partiendo de ciertos datos de entrada, 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 realizarán sin importar el idioma del cocinero.

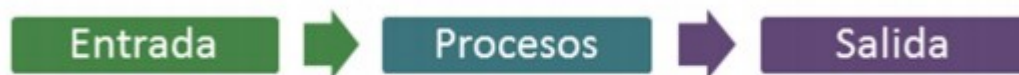
Los algoritmos son más importantes que los lenguajes de programación o las computadoras. Un lenguaje de programación es tan sólo un medio para expresar un algoritmo y una computadora es sólo un procesador para ejecutarlo. Tanto el lenguaje de programación como la computadora son los medios para obtener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

CARACTERÍSTICAS DE LOS ALGORITMOS

Las características fundamentales que debe cumplir todo algoritmo 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, Procesos y la Información de Salida.**



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

Programa

Un programa no es mas que **un algoritmo escrito 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.

PASOS PARA LA CONSTRUCCIÓN DE UN PROGRAMA

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

DEFINICIÓN DEL PROBLEMA

En este paso se determina la información inicial para la elaboración del programa. Es donde se determina que es lo que debe resolverse con el computador, el cual requiere una definición clara y precisa.

Es importante que se conozca lo que se desea que realice la computadora; mientras la definición del problema no se conozca del todo, no tiene mucho caso continuar con la siguiente etapa.

ANÁLISIS DEL PROBLEMA

Una vez que se ha comprendido lo que se desea de la computadora, es necesario definir:

- ✓ Los datos de entrada.
- ✓ Los datos de salida.
- ✓ Los métodos y fórmulas que se necesitan para procesar los datos.

Una recomendación muy practica es la de colocarse en el lugar de la computadora y analizar qué es lo que se necesita que se ordene y en que secuencia para producir los resultados esperados.

DISEÑO DEL ALGORITMO

Se puede utilizar algunas de las herramientas de representación de algoritmos para definir la secuencia de pasos que se deben llevar a cabo para conseguir la salida identificada en el paso anterior. La herramienta de representación de algoritmos que utilizaremos 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.

CODIFICACIÓN

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.

PRUEBA Y DEPURACIÓN

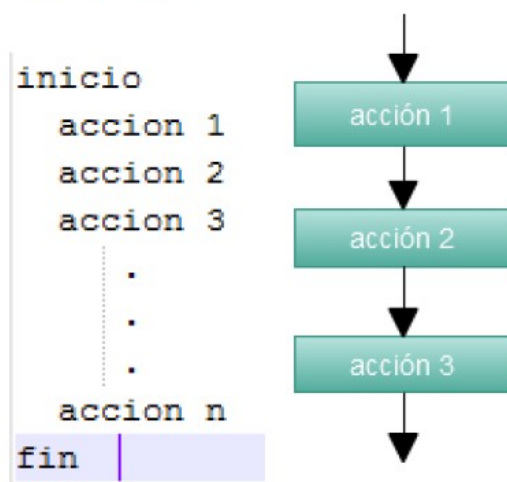
Se denomina *prueba de escritorio* a la *comprobación* que se hace *de un algoritmo para saber si está bien realizado*. Esta prueba consiste en tomar datos específicos como entrada y seguir la secuencia indicada en el algoritmo hasta obtener un resultado, el análisis de estos resultados indicara si el algoritmo esta correcto o si por el contrario hay necesidad de corregirlo o hacerle ajustes.

PARTES CONSTITUTIVAS 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.

Pseudocódigo



Estructura de un programa secuencial

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.
- ✓ **Instrucciones de asignación**, se utilizan para asignar el resultado de la evaluación de una expresión a una variable. El valor (dato) que se obtiene al evaluar la expresión es almacenado en la variable que se indique.
`<nombre de la variable> ← <expresión>`
expresión es igual a una expresión matemática o lógica, a una variable o constante.
- ✓ **Instrucciones de lectura**, se utilizan para leer datos de un dispositivo de entrada y se asignan a las variables.
- ✓ **Instrucciones de escritura**, se utilizan para escribir o mostrar mensajes o contenidos de las variables en un dispositivo de salida.
- ✓ **Instrucciones de bifurcación**, mediante estas instrucciones el desarrollo lineal de un programa se interrumpe. Las bifurcaciones o al flujo de un programa puede ser según el punto del programa en el que se ejecuta la instrucción hacia adelante o hacia atrás.

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.

Variables y Constantes

Los programas de computadora contienen ciertos valores que no deben cambiar durante la ejecución del programa. Tales valores se llaman constantes. De igual forma, existen otros valores que cambiarán durante la ejecución del programa; a estos valores se les llama variables.

Una constante es un dato que permanece sin cambios durante todo el desarrollo del algoritmo o durante la ejecución del programa.

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.

DECLARACIÓN DE VARIABLES

Normalmente los identificadores de las variables y de las constantes con nombre deben ser declaradas en los programas antes de ser utilizadas. La sintaxis de la declaración de una variable suele ser:

```
<tipo_de_dato> <nombre_variable> [=<expresión>]
```

Entrada y Salida de Información

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 determinados valores 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.). 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 de lectura y escritura se representan por los formatos siguientes:

```
leer (lista de variables de entrada)  
escribir (lista de variables de salida)
```

Escritura de Algoritmos/Programas

Un algoritmo constará 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_titulo debe ser reemplazada por el nombre del algoritmo.

SECCIÓN DE DECLARACIÓN

En esta sección se declaran o describen todas las variables utilizadas en el algoritmo, listándose sus nombres y especificando sus tipos (ver apéndice al final de la guía). La declaración de variables en PseInt comienza con la palabra reservada **Definir** seguida del *nombre de la variable* (identificador) la palabra reservada **como** y luego el tipo de dato.

```
Definir <identificador> como <Tipo de dato>
```


TIPOS DE DATOS EN PSEINT

Los tipos de datos básico soportados son los siguientes:

- ✓ **Entero**: solo números enteros.
- ✓ **Real**: números con cifras decimales. Para separar decimales se utiliza el punto. Ejemplo:
`3.14`
- ✓ **Caracter**: cuando queremos guardar un carácter.
- ✓ **Logico**: cuando necesitamos guardar una expresión lógica (verdadero o falso)
- ✓ **Cadena**: cuando queremos guardar cadenas de caracteres.

Notas:

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

Ejemplo de declaración de variables:

Si queremos declarar una variable de tipo entero se escribe:

`Definir varNumero Como Entero`

`varNumero` se convierte en una variable de tipo entero .

SECCIÓN DE ACCIONES

En esta sección se describe paso a paso cada una de las instrucciones que llevará a cabo el algoritmo/programa para obtener una solución a un determinado problema.

PRIMITIVAS SECUENCIALES (COMANDOS DE ENTRADA, PROCESO Y SALIDA)

- ✓ **Asignación (Proceso).**
- ✓ **Lectura (Entrada).**
- ✓ **Escritura (Salida).**

Asignación o proceso

La instrucción de **asignación** permite almacenar un valor en una variable (previamente definida), se puede realizar de dos maneras:

```
<variable> <- <expresión>
<variable> = <expresión>
```

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.

Ejemplo de asignación:

```
1 Algoritmo ejemploAsignacion
2
3   Definir num Como Entero
4
5   num = 4
6
7 FinAlgoritmo
8
```

En este ejemplo estamos definiendo una variable como entero y después asignándole un valor(4).

Lectura o entrada

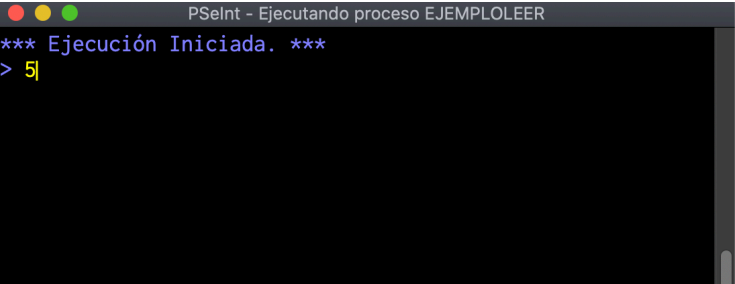
La instrucción **Leer** permite ingresar información desde la interfaz grafica o ambiente. Que es mostrará al corre nuestro algoritmo

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

Esta instrucción lee N valores desde la interfaz grafica o ambiente, lo que escribamos por teclado y los asigna a las N variables mencionadas. Pueden incluirse una o más variables, por lo tanto el comando leerá uno o más valores.

Ejemplo de Leer:

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



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

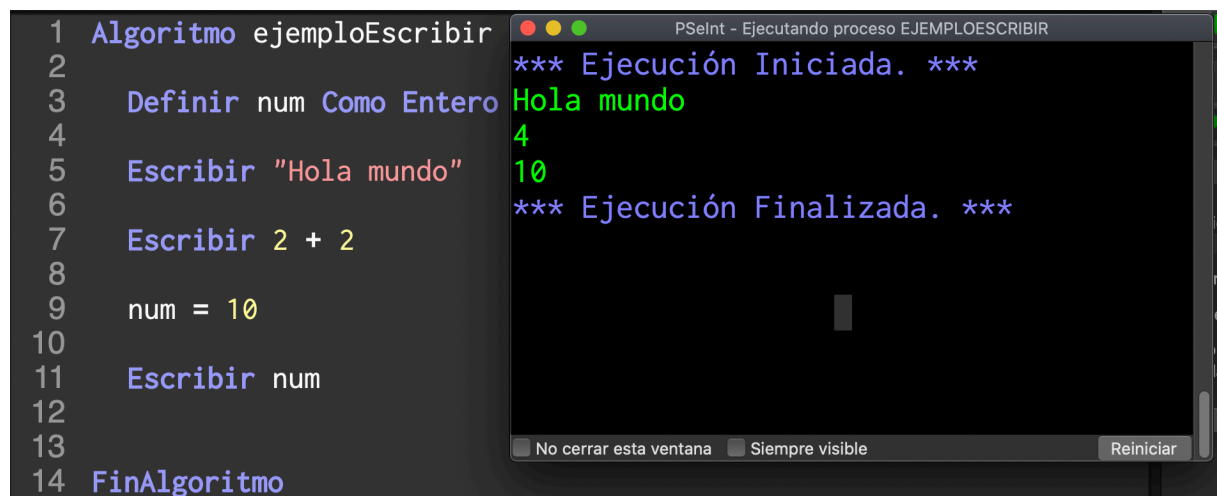
Escritura o salida

La instrucción **Escribir** permite mostrar valores en la interfaz grafica o ambiente.

Escribir <expr1> , <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.

Ejemplo de Escribir:



```
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
```

*** Ejecución Iniciada. ***
Hola mundo
4
10
*** Ejecución Finalizada. ***

☐ No cerrar esta ventana ☐ Siempre visible

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 asigno un valor previo.

OPERADORES

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

OPERADORES RELACIONALES

Los operadores relacionales son símbolos que se usan para comparar dos valores. Si el resultado de la comparación es correcto la expresión considerada es verdadera, en caso contrario es falsa.

Operador	Significado	Ejemplo
Relacionales		
>	Mayor que	3 > 2 // verdadero
<	Menor que	"ABC" < "abc" // verdadero
=	Igual que	4 = 4 // verdadero
>=	Mayor o igual que	4 >= 5 // falso
<=	Menor o igual que	'a' <= 'b' // verdadero
<>	Distinto que	10 <> 8 // verdadero

Operadores Lógicos

Nos proporcionan un resultado a partir de que se cumpla o no una cierta condición, producen un resultado lógico, y sus operandos son también valores lógicos o asimilables a ellos.

Operador	Significado	Ejemplo
Lógicos		
Y	Conjunción	(2 < 4 Y 3 > 5) // falso
O	Disyunción	(7 <= 8 O 10 >= 9) // verdadero
NO / no	Negación	no(1 = 1) // falso

Operador Y

Devuelve un valor lógico verdadero si ambos operandos son ciertos. En caso contrario el resultado es falso.

Operador O

Este operador devuelve verdadero si alguno de los operandos es cierto. En caso contrario devuelve falso.

Operador NO

Este operador cambia la devolución de un operando, al caso contrario. Si es verdadero lo hace falso y si es falso lo hace verdadero.

A la hora de trabajar con operadores lógicos, para saber si una expresión lógica nos devuelve como resultado Verdadero o Falso, debemos observar la siguiente tabla de la verdad:

Conjunción

A	Operador	B	Resultado
V	Y	V	V
V	Y	F	F
F	Y	V	F
F	Y	F	F

Disyunción

A	Operador	B	Resultado
V	O	V	V
V	O	F	V
F	O	V	V
F	O	F	F

Negación

A	Resultado	B	Resultado
no(V)	F	no(F)	V

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 operandos requieren unas reglas matemáticas que permitan determinar el orden de las operaciones, se denominan reglas de prioridad o precedencia 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).
3. Las operaciones lógicas dentro de una expresión suelen seguir el siguiente orden de prioridad:
 - ✓ operador ()
 - ✓ operador negación NO
 - ✓ operador conjunción Y
 - ✓ operador disyunción O

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.

FUNCIONES MATEMÁTICAS

Las funciones en el pseudocódigo se utilizan de forma similar a otros lenguajes. Se coloca su nombre seguido de los argumentos para la misma encerrados entre paréntesis (por ejemplo, trunc(x)). Se pueden utilizar dentro de cualquier expresión, y cuando se evalúe la misma, se reemplazará por el resultado correspondiente.

Actualmente, todas las funciones disponibles son matemáticas (es decir que devolverán un resultado de tipo numérico) y reciben un sólo parámetro de tipo numérico. A continuación, se listan las funciones integradas disponibles:

Función	Significado
RC(X)	Raíz cuadrada de x
ABS(X)	Valor absoluto de x
LN(X)	Logaritmo natural de x
EXP(X)	Función exponencial de x
SEN(X)	Seno de x
COS(X)	Coseno de x
TAN(X)	Tangente de x
ASEN(X)	Arcoseno de x
ACOS(X)	Arcocoseno de x
ATAN(X)	Arcotangente de x
TRUNC(X)	Parte entera de x
REDOND(X)	Entero más cercano de x
AZAR(X)	Entero aleatorio entre 0 y x -1
ALEATORIO(X,X)	Entero aleatorio entre valor mínimo y máximo

Notas:

- ✓ La función raíz cuadrada no debe recibir un argumento negativo.
- ✓ La función exponencial no debe recibir un argumento menor o igual a cero.

Preguntas de Aprendizaje

1) Los dispositivos de entrada permiten:

- a) Guardar datos en la computadora
- b) Desplegar información almacenada en el equipo
- ☒ c) Ingresar datos a la computadora
- d) Ninguna de las anteriores

2) Los dispositivos de salida permiten:

- a) Guardar datos en la computadora
- ☒ b) Desplegar información almacenada en el equipo
- c) Ingresar datos a la computadora
- d) Ninguna de las anteriores

3) Las unidades de almacenamiento permiten:

- a) Ingresar datos a la computadora
- ☒ b) Guardar datos en la computadora
- c) Desplegar información almacenada en el equipo
- d) Ninguna de las anteriores

4) ¿Qué es un algoritmo?

- ☒ a) Un conjunto de instrucciones o reglas bien definidas, ordenadas y finitas que permiten realizar una actividad mediante pasos sucesivos que no generen dudas a quien deba realizar dicha actividad
- b) Es una igualdad entre dos expresiones algebraicas, denominadas miembros, en las que aparecen valores conocidos o datos, y desconocidos o incógnitas, relacionados mediante operaciones
- c) Es una relación de variables que pueden ser cuantificadas para calcular el valor de otras de muy difícil o imposible cálculo y que suministra una solución para un problema
- d) Ninguna de las anteriores

5) La prueba de escritorio se usa para:

- a) Programar órdenes
- ☒ b) Verificar si el algoritmo es correcto
- c) Eliminar virus informáticos
- d) Todas las anteriores

6) Una variable es

- ☒ a) Un lugar de almacenamiento, cuyo contenido podrá variar durante el proceso y finalmente se obtendrán los resultados con los datos contenidos en ellas
- b) Un lugar de almacenamiento, cuyo contenido no varía durante el proceso y finalmente se obtendrán los resultados con los datos contenidos en ellas
- c) Una palabra reservada del lenguaje de programación
- d) Ninguna de las anteriores

7) La ejecución de la siguiente sentencia de asignación: A = "4.5"

- a) A debe ser una variable de tipo real
- b) A debe ser una constante de tipo real
- ☒ c) A debe ser una variable de tipo cadena
- d) A puede ser tanto una variable de tipo real como de tipo cadena

8) entero, carácter, lógico y real son:

- a) Funciones de acceso a datos
- b) Instrucciones de acceso a datos
- c) Sentencias de control
- ☒ d) Tipos de datos

9) Un operador es

- a) un lugar de almacenamiento de datos
- ☒ b) un símbolo especial que indica al compilador que se debe realizar una operación matemática o lógica
- c) una variable
- d) Ninguna de las anteriores

10) Los operadores relacionales se usan en:

- ☒ a) operaciones de comparación
- b) operaciones de suma y resta
- c) operaciones de multiplicación y división
- d) Ninguna de las anteriores

11) Una estructura secuencial es aquella que ejecuta:

- ☒ a) Una evaluación de una expresión y, dependiendo del resultado, se decide la siguiente sentencia a ejecutar
- b) Una sentencia detrás de otra
- c) Una repetición de un bloque de sentencias mientras sea verdadera una determinada condición
- d) Ninguna de las anteriores

12) La instrucción leer base, altura permite:

- ☒ a) Almacenar los datos ingresados por el usuario en algún lugar de la computadora
- ☒ b) Almacenar los datos ingresados por el teclado en las variables base y altura
- c) Almacenar tres datos ingresados por teclado en las variables leer, base y altura
- d) Ninguna de las anteriores

13) La instrucción escribir "Ingrese 25 números enteros" permite:

- ☒ a) Visualizar en pantalla el mensaje entre comillas
- b) Guardar en la variable pantalla los datos ingresados por teclado
- c) Verificar si el algoritmo está bien hecho
- d) Ninguna de las anteriores

14) La ejecución de la siguiente sentencia de asignación: A = B

- a) A debe ser variable y B constante
- b) A debe ser constante y B variable
- ☒ c) Tanto A como B deben ser variables
- d) Debe haber compatibilidad de tipos entre A y B

15) Seleccione la expresión que da como resultado el valor lógico falso.

- a) $(4 \geq 40 \vee 8 \leq 10)$ o $(2 < 20 \vee 10 > 100)$
- b) $(8 \geq 10 \vee 4 \leq 8)$ y $(3 < 10 \vee 10 \geq 4)$
- c) $(8 \geq 4 \vee 8 \geq 10)$ o $(5 = 5 \vee 4 < 8)$
- d) $(4 > 4 \vee 10 \geq 8)$ y $(2 < 5 \vee 8 < 4)$

16) Seleccione la expresión que da como resultado el valor lógico verdadero.

- a) $(50 > 49 \vee 7 = 5)$ o $(15 \leq 14 \vee 10 > 100)$
- b) $(6 < 6 \vee 4 = 5)$ y $(10 > 9 \vee 20 \leq 20)$
- c) $\text{no}(\text{no}(10 \geq 8) \vee 1 > 3)$ o $(2 < 3 \vee 2 < 8)$
- d) $(4 > 2 \vee 7 > 6)$ y $\text{no}(3 < 6 \vee 2 > 0)$

17) Si a = verdadero y b = falso, la expresión no (a o b) y no (a) toma el mismo resultado que:

- a) a y b
- b) no (a o no b)
- c) b o (a y b)
- ☒ d) no (no a o b) o no b

Ejercicios de aprendizaje

Primero tenemos que tener descargado Pseint, para descargarlo haga click aquí:

<http://pseint.sourceforge.net/?page=descargas.php>.

PSeInt permite personalizar algunos aspectos del lenguaje a través de la configuración de diferentes perfiles. En este curso utilizaremos el perfil "Perfil_EggTech_PseInt". Para agregar el perfil se debe ingresar al menú "Configurar->Opciones del Lenguaje (Perfiles)" y seleccionar la opción "Personalizar". Esta opción abrirá una nueva ventana a través de la cual en la parte inferior (ícono de una carpeta) podrá seleccionar el archivo y cargarlo en PSeInt. Una vez cargado el perfil se selecciona la opción "Aceptar" y posteriormente ya se podrá comenzar a programar.

Para cada uno de los siguientes ejercicios realizar el análisis del problema e indicar cuáles son los datos de entrada y cuáles son los datos de salida. Escribir luego el algoritmo en PSeInt. Al final de la guía encontrará un apéndice con los tipos de datos y operadores disponibles en PSeInt

VER VIDEO: Mi Primer Programa

1. Escribir un algoritmo en el cual se consulte al usuario que ingrese ¿cómo está el día de hoy? (soleado, nublado, lloviendo). A continuación, mostrar por pantalla un mensaje que indique "El día de hoy está ...", completando el mensaje con el dato que ingresó el usuario.

VER VIDEO: Trabajando con Datos

2. 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.
3. Conocido el número en matemática 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:

$$\text{Área} = \pi * \text{radio}^2$$

$$\text{Perímetro} = 2 * \pi * \text{radio}$$

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

Área = base * altura

Perímetro = 2 * altura + 2 * base.

5. 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

6. 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.
7. 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

8. 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.
9. 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.
10. 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.

11. 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 $\text{num1} = 9$ y $\text{num2} = 3$, la salida a del programa deberá mostrar: $\text{num1} = 3$ y $\text{num2} = 9$

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

12. Escriba un programa que lea dos números enteros y realice el cálculo de la suma, resta, multiplicación y división entre ambos valores. Los resultados deben mostrarse por pantalla.