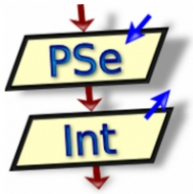
CURSO DE PROGRAMACIÓN FULL STACK

Introducción a la Programación con PSeInt



G&L GROUP | EGG

Guía de Programación con PSeInt

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 la informática el termino programación se refiere a la acción de crear programas y programar es la serie de instrucciones, que le vamos a dar a nuestra maquina, para lograr lo que nuestro programa necesite para funcionar.
Las partes que componen a nuestro programa son el lenguaje de programación y los algoritmos.

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 para crear programas que controlen el comportamiento físico y lógico de una máquina.
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 maquina: 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.

Algoritmo

Entonces, los lenguajes de programación son nuestro puente para poder comunicarnos con la maquina. Y de esa forma darle instrucciones claras, para poder solucionar los problemas que puede presentar la creacion de un programa.

Estas instrucciones que le vamos a dar un 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 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 realizarán sin importar el idioma del cocinero.

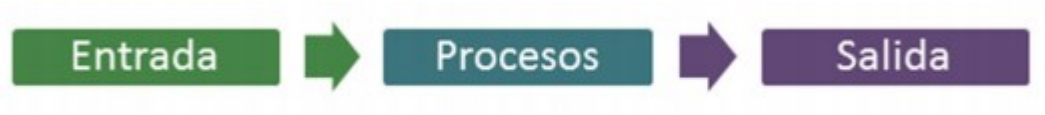
Los algoritmos son más importantes que los lenguajes de programación o las computadoras. Un lenguaje de programación es tan sólo un medio para expresar un algoritmo y una computadora es sólo un procesador para ejecutarlo. Tanto el lenguaje de programación como la computadora son los medios para obtener 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 o Información de entrada, Procesos y la Información de Salida.*



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

Programa

Entonces para resumir, un programa no es mas 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.

IDE

Nuestro programa va a ser escrito en un IDE (.Entorno de Desarrolllo Integrado). Un ide es una aplicación informatica, como es el word, en donde vamos a poder escribir algoritmos y crear programas con un lenguaje de programacion. Este ide, va a proveernos de muchas faciliades para lograr nuestro cometido, desde ayudas, hasta correcciones de codigo para evitar errores.

Entonces, basicamente, el ide es la herramienta que vamos a utilizar para escribir nuestro programa, con el que nos vamos a comunicar con la computadora, además, el ide va a cambiar dependiendo del lenguaje con el que querramos trabajar.

DISEÑO DEL PROGRAMA

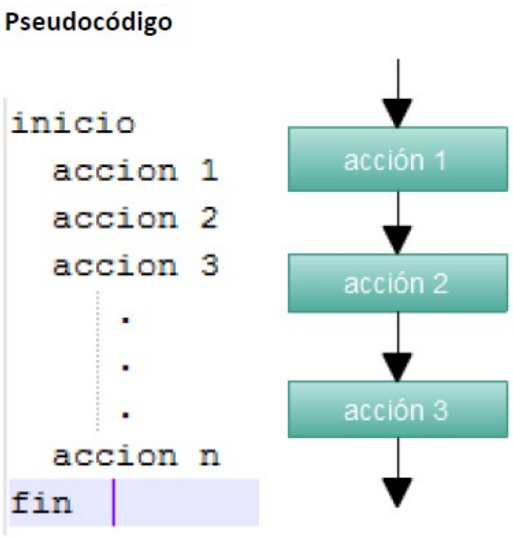
Se puede utilizar algunas de las herramientas de representación de algoritmos, también conocidos como lenguajes de programación, para definir la secuencia de pasos que se deben llevar a cabo para conseguir el resultado que necesitamos. El lenguaje de programación que utilizaremos en esta parte de curso,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

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.

PRUEBA Y DEPURACIÓN

Hay veces que necesitamos comprobar si nuestro algoritmo realiza lo necesario, para esto usaremos la prueba de escritorio. 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 datos de entrada y durante la codificación, 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.

Escritura de Algoritmos/Programas

Ya sabemos que es un programa, el diseño de un programa, las especificaciones de un programa, su codificación y su depuración. Ahora vamos a ver como es la escritura de estos 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.

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.

Elementos de 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.

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.

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.

Variables y Constantes

Los programas de computadora necesitan información para la resolución de problemas. Está información puede ser un numero, un nombre, etc. Para nosotros poder guardar esta información en algún lugar y que no este “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 facilitara distinguir unas de otras y nos ayudara 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 las **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 calculo, ese 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 ese 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. 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. 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

CREACION DE VARIABLES

A la hora de crear nuestra variable, vamos a tener que darle un identificador 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 el ultimo 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>
```

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.
Recordemos que los tipos de instrucciones eran: Instrucciones de inicio/fin - Instrucciones de asignación - Instrucciones de lectura - Instrucciones de escritura

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). Esta es nuestra manera de guardar información en una variable, para utilizar ese valor en otro momento. 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:

```
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 numero 4.

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)
```

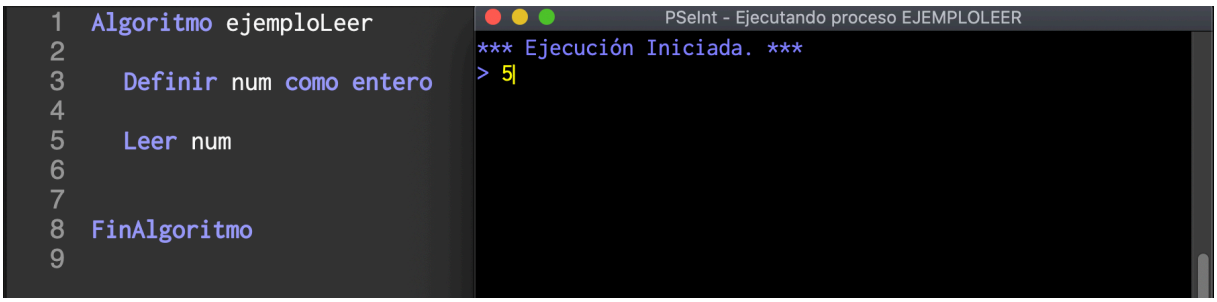
Lectura o entrada

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

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

Esta instrucción lee los N valores que escribamos por teclado, desde la interfaz grafica o ambiente, 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 en Pseint. A la izquierda está el algoritmo y a la derecha la interfaz grafica:



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

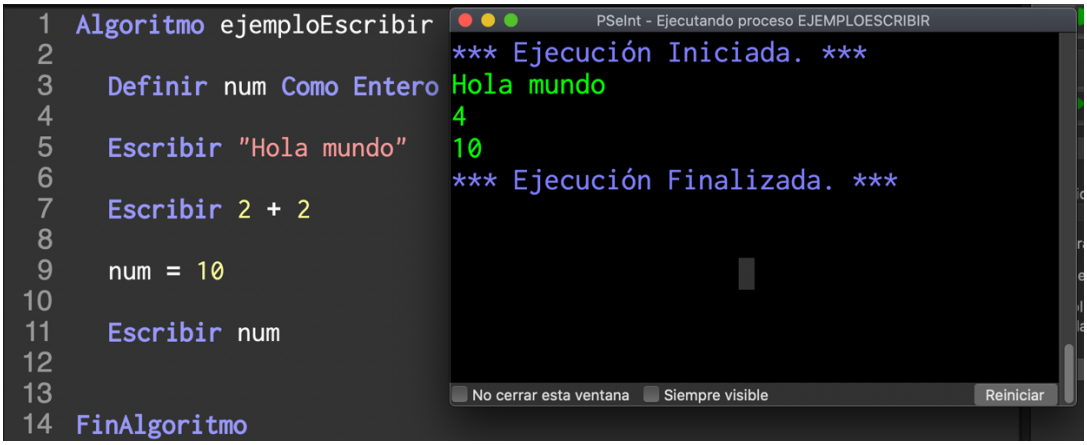
Escritura o salida

La instrucción Escribir permite mostrar 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.

Ejemplo de Escribir. A la izquierda está el algoritmo y a la derecha la interfaz grafica:



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.

Ejemplo:

```
Escribir “Mensaje entre comillas” variable
Escribir “La suma de los números es:” suma
```

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	1 < 5 // 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

Estos se utilizan cuando necesitamos las expresiones lógicas con múltiples variantes y nos proporcionan un resultado a partir de que se cumpla o no una cierta condición, estos producen un resultado lógico, y sus operadores 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 ambas expresiones son verdaderas. En caso contrario el resultado es falso.

Operador O

Este operador devuelve verdadero si alguno de las expresiones es verdadera. En caso contrario devuelve falso.

Operador NO

Este operador cambia la devolución de un expresión, 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 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).

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.

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) ¿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

4) 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

5) 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

6) 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 variable de tipo entero
c) A debe ser una variable de tipo cadena
d) A puede ser tanto una variable de tipo real como de tipo cadena

7) 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

8) 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

9) 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

10) 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

11) 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

12) 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

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

a) (4 >= 40 y 8 <= 10) o (2 < 20 o 10 > 100)
b) (8 >= 10 o 4 <= 8) y (3 <> 10 y 10 >= 4)
c) (8 >= 4 y 8 >= 10) o (5 = 5 y 4 < 8)
d) (4 > 4 o 10 >= 8) y (2 > 5 o 8 < 4)

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

a) (50 > 49 y 7 = 5) o (15 <= 14 o 10 > 100)
b) (6 <> 6 o 4 = 5) y (10 > 9 y 20 <= 20)
c) no(no(10 >= 8) y 1 > 3) o (2 <> 3 y 2 < 8)
d) (4 > 2 o 7 > 6) y no(3 < 6 o 2 > 0)

15) Si a = verdadero y b = falso, la expresión no (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 descargalo 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" que se encuentra en el Drive. 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.

Pero antes de empezar, recomendamos leer este breve inciso sobre como encararemos la resolución de los siguientes ejercicios / programas.

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 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 lapiz

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 codigo

En el siguiente paso, empezamos a pensar en código, antes estábamos tomando un problema de programación y resolviendolo 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 codigo

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.

EJEMPLO DE RESOLUCIÓN DE PROBLEMAS

Teniendo en cuenta que ya leimos el problema y lo pasamos a papel y lapiz. Lo que hariamos con el siguiente problema seria esto:

Leer el radio de un circunferencia y calcular e imprimir su superficie y su circunferencia.

Se puede dividir en tres subproblemas más sencillos:

- Leer Radio
- Calcular Superficie : PI * radio ^ 2
- Calcular Longitud : 2 * PI * radio
- Escribir resultados

Herramientas necesarias para hacer estos subproblemas:

- Variables: radio, superficie, longitud.
- Constantes: PI
- Funciones: Leer – Escribir - Definir
- Expresiones: Multiplicar - Potencia

Traslado de los subproblemas a codigo:

- Definir radio, superficie, longitud como entero
- Leer radio
- superficie = PI * radio ^ 2
- longitud = 2 * PI * radio
- Escribir radio, longitud, superficie

¿ ME BLOQUEE EN CUANTO A CODIGO ?

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

NO PUEDO EMPEZAR EL EJERCICIO E HICE TODO LO ANTERIOR

- 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, consultale al o la profe de como 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, seguis sin entender, plantea que parte no entendiste.

Reglas de oro:

- Preguntar TODO. No tengamos miedo a preguntar, pensar que los demas tambien se benefician de nuestra duda y que el otro va a estar dispuesto a ayudarnos.
- Leer TODA la teoría.
- Ver TODOS los videos.
- Participar en TODOS los debates.
- Tener paciencia y entender que con el tiempo y con practica, iremos aprendiendo que necesitamos para resolver estos problemas de manera más sencilla.
- Divertirse, ver los problemas como un desafio y no como algo que si no lo podemos lograr, nos haces más o menos inteligentes. Recordemos que esto es un proceso de aprendizaje y que no todos aprendemos al mismo tiempo.

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. Conocido el número en matemática PI ℗, 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

3. 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.

4. 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.

5. Escribir un programa que calcule cuántos litros de combustible consumió un automóvil. El usuario ingresase 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.

6. 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.*

7. 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.

Material 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 en las mesas, podes continuar con estos ejercicios extra, recordando siempre que no es necesario que los termines para continuar con el tema siguiente. Por ultimo, recordá que la prioridad es ayudar a los compañeros de la mesa 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 = Ⓟ * radio^2 * 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.