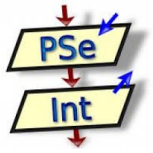


PSeInt – Introducción a la Programación

Apuntes Semana: 2
M. En C.M.V. Alejandro González Reyes

PSeInt

PSeInt es una herramienta para asistir a un estudiante en sus primeros pasos en programación. Mediante un simple e intuitivo pseudolenguaje en español (complementado con un editor de diagramas de flujo), le permite centrar su atención en los conceptos fundamentales de la algoritmia computacional, minimizando las dificultades propias de un lenguaje y proporcionando un entorno de trabajo con numerosas ayudas y recursos didácticos.

Elementos básicos de un programa

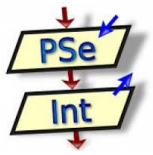
Antes de comenzar a desarrollar un programa de computadora. Lo que debe hacer un buen programador es reconocer a detalle cuáles son los **datos de entrada** que necesita, cuáles son los **procesos** que debe realizar con esos datos de entrada y cuáles son los **datos de salida** que se obtendrán como resultado de ese procesamiento.

Si comparamos un programa de computadora con una receta para preparar un pastel de chocolate, las entradas serían todos los ingredientes necesarios, la receta misma sería el algoritmo[1] y la salida sería nuestro flamante y delicioso pastel de chocolate listo para consumirse.

En una computadora, las entradas pueden venir de una gran variedad de dispositivos: el teclado, el mouse, un lector de código de barras, una memoria USB, etc., mientras que los datos de salida también se pueden mostrar en una amplia variedad de formas: en la pantalla, impresos en papel, en forma de sonidos en las bocinas, en forma de alguna acción de otro dispositivo (como encender o apagar la Wi-Fi del celular), etc.

Para lograr resolver un problema, necesitamos generar un algoritmo formado por distintas clases de instrucciones, las cuales pueden ser luego implementadas en

1 Conjunto ordenado de operaciones que permite hacer un cálculo y hallar la solución a un determinado problema.



cualquier lenguaje de programación (en nuestro caso PseInt, un lenguaje de programación falso). Entre los principales tipos se encuentran:

- Instrucciones de inicio-fin
- Instrucciones de asignación
- Instrucciones de lectura de datos (entrada)
- Instrucciones de escritura de resultados (salida)
- Instrucciones de control de flujo

Instrucciones de inicio-fin

Estas son las que usamos para indicarle al algoritmo cuando inicia y cuando termina. Generalmente se usan las palabras clave Proceso y FinProceso (en versiones recientes de PSeInt, se hace uso de Algoritmo y FinAlgoritmo).

Instrucciones de asignación

Una instrucción de asignación es la que se usa para darle un valor a una variable. ¿Y qué es una variable? pues es un valor que puede cambiar durante la ejecución de un programa. En un algoritmo para el cálculo del pago de nómina, las variables serían: **diasLaborados**, **salarioDiario**, **horasExtras**, etc. Para representar la asignación de un valor a una variable vamos a estar utilizando el símbolo de asignación o igualación (**=**).

Instrucciones de lectura de datos (entrada)

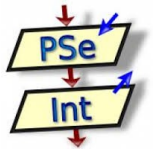
Son las que usamos para indicar la entrada de un dato para nuestro algoritmo por medio de algún dispositivo (como el teclado, el mouse, un escaner, etc) o incluso de algún otro programa. Generalmente usamos la palabra clave **Leer**.

Instrucciones de escritura de resultados (salida)

Son las que se usan para indicar la salida de resultados o la solicitud de información al usuario. Todo ello a través de ciertos dispositivos (como la pantalla, la impresora, las bocinas, etc). Generalmente usamos la palabra clave **Imprimir**.

Instrucciones de control de flujo[2]

2 *Estas situaciones son bastante comunes en nuestra vida diaria también. Pensemos en la decisión de la ropa que vamos a ponernos para salir a pasear en una tarde de sábado. Si el clima es caluroso, decidimos por un atuendo más ligero. Si no, nuestro atuendo es más cubierto. Por otro lado, cuando*



Estas nos permiten dividir la secuencia de ejecución o repetir ya sea algunas o todas las instrucciones de nuestro algoritmo con base a alguna condición. Si esa condición se cumple, se ejecuta (o repite) un conjunto de instrucciones y si no se cumple, se ejecuta (o repite) otro conjunto de instrucciones distintas.

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 podrán cambiar durante la ejecución del programa; a estos se les llama **variables**.

Identificadores

Como todo objeto que se usa dentro de un programa, tanto las **constantes** como las **variables** deben tener un nombre para que puedan ser reconocidas y utilizadas. A estos nombres los llamamos **identificadores.** PSeInt tiene ciertas reglas que se deben seguir cuando se trata de escribir identificadores. Entre ellas podemos mencionar:

- Usar caracteres como letras, dígitos y uno que otro símbolo (como el guión bajo)
- Comenzar con una letra
- No usar espacios en blanco
- Procurar no usar caracteres especiales como acentos, ñ o diéresis
- Ser significativo y tener relación con el objeto que representan (como **promedioGeneral**, **horas_trabajadas** y **tasa_impuesto**)

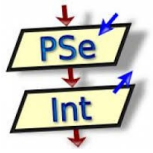
Tipos de datos

Un algoritmo requiere de datos para poder funcionar. Dependiendo del tipo de programa que necesitemos hacer, esos datos pueden ser de tipos muy diversos.

- **Numéricos:** Como su nombre lo indica, en estos tipos podemos almacenar números, los cuales a su vez pueden ser Enteros o Reales y también pueden

---

vamos a comer tacos a un lugar nuevo, decidimos repetir el pedido sólo si nos agradó la manera en que los preparan. Esto nos indica dos cosas: que programar es más común de lo que parece y que a mí me encantan los tacos...



ser positivos o negativos.

- **Alfanuméricos:** Estos están formados por todos los caracteres (letras, dígitos y símbolos) que reconoce una computadora. El tipo Caracter es el que está formado solamente por un caracter, ya sea letra, dígito o símbolo. Por otro lado, el tipo Cadena de Caracteres está formado por más de un caracter, como puede ser el nombre de una persona, un domicilio, una dirección de correo electrónico, entre otros muchos ejemplos. Los tipos alfanuméricos suelen delimitarse usando comillas simples o dobles (por ejemplo: "a", "Mambrú se fue a la guerra", "(555) 334-56-78", "03/Mar/2015").
- **Lógicos:** El tipo lógico, también llamado booleano es el que sólo puede tomar uno de dos valores: verdadero (true) o falso (false). Este tipo de datos se usa para representar alternativas a determinadas condiciones (como SI/NO, ENCENDIDO/APAGADO, LLENO/VACIO, etc). Por ejemplo, al decidir si un número es par, la respuesta se dará en términos de verdadero o falso.

Expresiones:

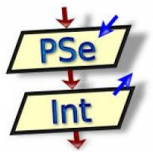
Las expresiones son combinaciones de constantes, variables, símbolos de operación, paréntesis y nombres de funciones especiales que utilizamos para darle instrucciones a una computadora. Una expresión puede ser algo tan simple como asignarle un valor a una variable o tan complejo como toda una fórmula matemática traducida a símbolos que la computadora entendería.

Existen distintos tipos de expresiones que podemos encontrarnos en un algoritmo común:

Expresiones Aritméticas

Estas son similares a las fórmulas matemáticas que aprendimos desde nuestros años de primaria y que nos sirven para representar algún modelo de la vida real. Desde el cálculo del área de alguna figura geométrica hasta la representación de un complejo modelo económico, las expresiones aritméticas serán bastante usuales en nuestra labor como programadores.

En una expresión aritmética usaremos únicamente variables o constantes de tipo numérico, además de algunos símbolos que se utilizan para representar las operaciones aritméticas más comunes. En esta tabla agrupamos los operadores



aritméticos que usaremos para construir nuestras expresiones así como algunos ejemplos de su uso:

Significado	Símbolo	Ejemplo
Suma	+	resultado= NumA + numB
Resta	-	resultado= NumA - numB
Multiplicación	*	resultado= NumA * numB
División	/	resultado= NumA / numB
Potencia	^	resultado= NumA ^ numB
Residuo o Sobrante (Módulo)	MOD	resultado= NumA MOD numB

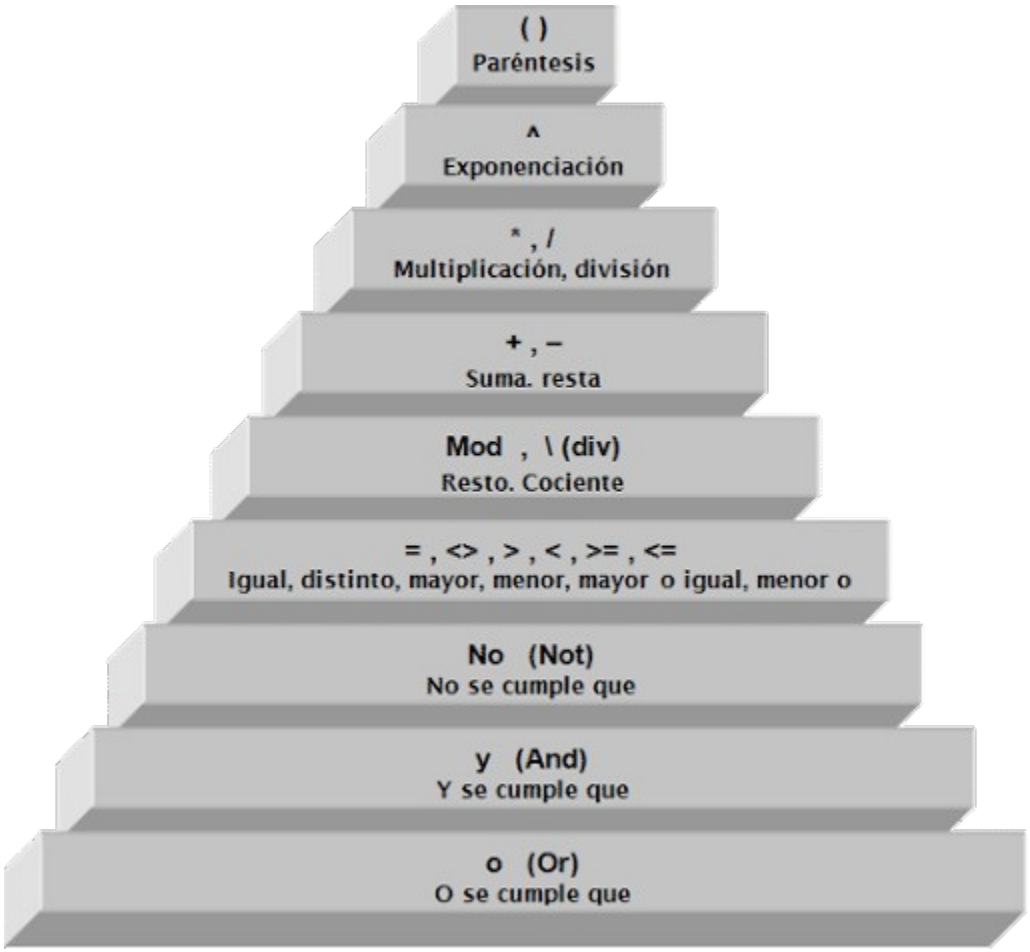
Algo muy importante que tenemos que destacar es que, cuando hay varias operaciones aritméticas en una expresión, la computadora las resuelve de un modo distinto a como nosotros lo haríamos. Supongamos, por ejemplo, que queremos calcular el valor para la variable C, que tiene asignada la siguiente expresión:

C = 5 * 4 + 8 / 2 – 1

Es importante enfatizar que la computadora ejecuta las operaciones según un determinado orden de jerarquía (prioridad de los operadores).

C = 23

Como vemos, la computadora llega a un resultado totalmente distinto al que obtendríamos nosotros. Es importante que sepamos cómo "piensa" una computadora para resolver estas operaciones dado que será nuestro trabajo escribir algunas fórmulas matemáticas en forma de expresiones que entienda y ejecute correctamente una computadora.



Con base en el esquema anterior, el orden de ejecución que sigue la computadora para determinar la solución a la formula anterior es como la que se describe a continuación:

C = 20 + 8 / 2 - 1 *Se multiplica 5 * 4*

C = 20 + 4 - 1 *Se divide 8 / 2*

C = 24 - 1 *Se suma 20 + 4*

C = 23 *Se resta 24 - 1*

En este sentido, en una expresión podemos usar también paréntesis para forzar la resolución de alguna operación independientemente de su nivel de jerarquía. Y es que, para una computadora, las operaciones que están agrupadas con paréntesis deben resolverse primero (igual que en la vida real).

M. En. C.M.V. Alejandro González Reyes

C = ((5 * 4) + 8) / (2) – 1

C = 13

Expresiones Lógicas

Un segundo tipo de expresiones es la **expresión lógica** o booleana, cuyo valor es siempre verdadero o falso. Recuerda que existen sólo dos posibles valores para una variable (o constante) de tipo booleano: *verdadero (true)* y *falso (false)*. En esencia, una expresión lógica es aquella cuyo resultado sólo puede tomar uno de esos dos valores.

Las expresiones lógicas se forman combinando constantes y variables lógicas con operadores lógicos como **NO**, **Y** y **O** y operadores relacionales (o comparación) como ==, <, >, <=, >=, !=.

Operadores de relación o comparación

Los operadores relacionales permiten realizar comparaciones entre valores o expresiones de cualquier tipo. Los operadores relacionales son:

Significado	Símbolo	Ejemplo
Menor que	<	Ventas < 12000
Mayor que	>	Promedio > 7
Menor o igual que	<=	Temperatura <= 5
Mayor o igual que	>=	Gastos >= 20000
Idéntico a	==	OpcionUsuario == “Aceptar”
Diferente de	!=	Impresora != “Epson”

Con base en lo explicado al inicio de este apartado, los operadores relacionales devuelven como resultado un valor de tipo lógico, por ejemplo:

7 > 45 FALSO

45 <= 165 VERDADERO

Su uso dentro del desarrollo de programas de computadora esta fuertemente relacionado con la toma de decisiones.

Funciones internas

Hemos hablado ya sobre las expresiones aritméticas y cómo podemos traducir fórmulas matemáticas usando diferentes clases de operadores aritméticos. Sin embargo, ¿qué pasa si en mi algoritmo necesito calcular una raíz cuadrada? ¿o si necesito calcular alguna función trigonométrica? ¿y qué tal si hay que redondear un valor numérico? Pues para esos casos existen las llamadas Funciones Internas, incorporadas o estándar, que forman parte de prácticamente todos los lenguajes de programación.

La idea de estas funciones es ahorrarnos trabajo puesto que no tenemos que hacer más que llamarlas por su nombre (o "invocarlas") y pasarles algunos parámetros para que nos regresen el resultado que buscamos. Por ejemplo, la función **RAIZ( )** nos permite calcular la raíz cuadrada de un número y lo único que tenemos que hacer es pasarle como parámetro el número cuya raíz cuadrada queremos calcular. Algo así:

resultado = **RAIZ(9)**

Después de esta instrucción, la variable resultado tendría el valor de 3.

Veamos ahora cómo usar estas funciones para traducir a expresiones algorítmicas una de las fórmulas matemáticas más conocidas: la fórmula general para resolver ecuaciones de segundo grado (sí recuerdan sus matemáticas de secundaria, ¿verdad?):

$$x1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

x1 = ((-b) + **RAIZ((b ^ 2) - (4 * a * c)))** / (2 * a)

$$x2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

x2 = ((-b) - **RAIZ((b ^ 2) - (4 * a * c)))** / (2 * a)

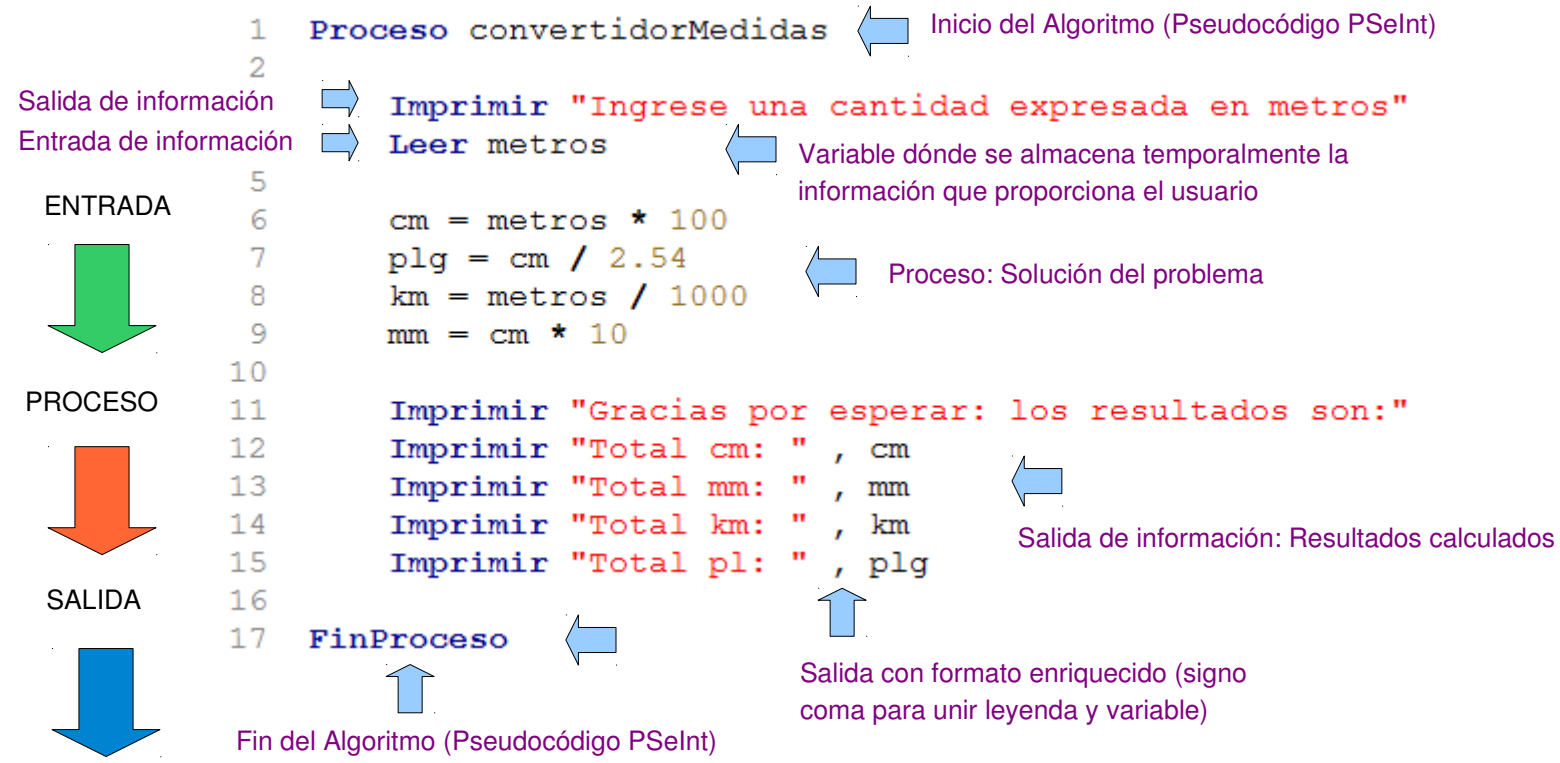
Para obtener más información acerca de las funciones internas, visite la ayuda

integrada en el software PseInt.

Ejemplo de Algoritmo diseñado mediante la técnica de Pseudocódigo

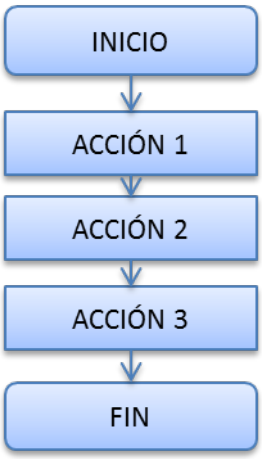
Desarrolle un programa de computadora mismo que, dada una cantidad expresada en metros por el usuario, determine su equivalente a centímetros, milímetros, kilómetros y pulgadas.

- Entrada: Cantidad expresada en metros
- Proceso: Determinar su equivalente a cm, mm, km, plg.
- Salida: El resultado de las conversiones, es decir, los cm, mm, km y plg.



Control de Flujo y estructuras de selección

Cuando hablamos del **control de flujo** nos referimos al control que como programadores podemos tener sobre el orden en el cual se van a ejecutar las diferentes instrucciones que forman a nuestro programa. Hasta ahora hemos estado desarrollando algoritmos cuyo **flujo** (o sea, el orden en que se van ejecutando cada una de las instrucciones del algoritmo) se dice que es **secuencial**. Esto quiere decir que las sentencias se ejecutan en secuencia, una después de otra, en el mismo orden en que se sitúan dentro del programa.



Pero este tipo de ejecución tiene grandes limitantes. Pensemos, por ejemplo, que el dueño de una tienda de abarrotes nos solicita desarrollar una pequeña aplicación para calcular el importe y cobrar las compras de sus clientes. El dueño de la tienda tiene un programa de recompensas en el cual aplica un determinado descuento si el total de la compra supera ciertos limites: si la compra es mayor a 1000 pesos, el descuento será del 5% y si supera 2500 pesos el descuento es del 10%.

En un caso como el descrito, el flujo secuencial no nos será de mucha utilidad por sí mismo porque ¿cómo define el programa cual es el descuento a aplicar? ¿Será el 5% o el 10%? Según la descripción, el descuento depende del monto de la compra pero ¿cómo decide eso el programa?

Ahí es donde entran en juego las **estructuras de selección**. Estas estructuras nos permiten habilitar a nuestro programa para tomar decisiones y ejecutar un determinado conjunto de instrucciones de acuerdo al cumplimiento de una o

varias condiciones.

En las estructuras de selección se evalúa una condición o conjunto de condiciones y de acuerdo al resultado se ejecuta una acción u otra. Para especificar las condiciones que se deben evaluar usamos expresiones lógicas. Existen tres estructuras de selección: *simples*, *dobles* y *múltiples*. Sin embargo, en este documento sólo vamos a conocer de un vistazo las estructuras de selección dobles.

- **Estructura de selección doble (si-entonces-si no)**

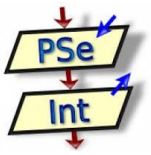
  Esta estructura evalúa una condición y, si es verdadera, ejecuta una acción determinada; pero si resulta falsa, ejecuta una acción distinta. A diferencia de la estructura simple, con la doble nuestro programa puede elegir entre dos cursos de acción diferentes.

La alternativa doble, como su nombre bien lo indica, nos permite dividir el flujo de nuestro programa en dos caminos distintos. El camino que se tome dependerá de

una condición o conjunto de condiciones. En cada uno de los caminos ejecutaremos una o más acciones. Al escribir un algoritmo, la representamos asi:

```
Si (condición) entonces
   acciones_caso_verdadero
Sino
   acciones_caso_falso
FinSi
```

En esta estructura, el conjunto de **acciones_caso_verdadero** se ejecutará si se cumple la condición, mientras que, si no se cumple, se ejecutará el conjunto de **acciones_caso_falso**. Para ejemplificar el uso de esta estructura, sírvase en atender la siguiente situación problema:



Desarrollar un programa de computadora mismo que, dadas dos calificaciones parciales por el usuario, determine el promedio general de las mismas. Además, si el promedio general es inferior a 6.0, se debe informar por pantalla que el usuario ha reprobado la materia, de lo contrario, se debe mostrar un mensaje de calificación aprobatoria.

- Entrada: Dos calificaciones parciales
- Proceso: Promedio de dichas calificaciones parciales
- Salida: El resultado del promedio general
- Condición: Si el promedio general es inferior a 6.0, mostrar un mensaje de situación reprobatoria, de lo contrario, mostrar un mensaje de situación aprobatoria.

ENTRADA

PROCESO

SALIDA

```
1  Proceso promedioCalifcaciones
2
3      Imprimir "PROGRAMA CALIFICACIONES PARCIALES - PREPA CDMX"
4      Imprimir "-----------------------------------------------"
5      Imprimir ""
6
7      Imprimir "Ingrese la primera calificación parcial"
8      Leer calificacionA
9      Imprimir "Ingrese la segunda calificación parcial"
10     Leer calificacionB
11
12     promedio = (calificacionA + calificacionB) / 2
13
14     Imprimir "El promedio general es: " , promedio
15
16     si promedio < 6 entonces
17         Imprimir "Alumno: valiste queso, ponte a estudiar"
18     Sino
19         Imprimir "Alumno: aprobaste la materia"
20     FinSi
21
22 FinProceso
```

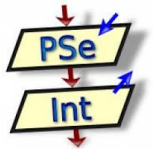
condición

Estructura condicional

Acciones caso verdadero

Acciones caso falso

M. En. C.M.V. Alejandro González Reyes



Observe que la alternativa doble es mucho más beneficiosa puesto que le da más versatilidad a nuestros programas, además de proporcionar una mejor experiencia de uso a nuestros usuarios.

Sabías que:

Para que un "algoritmo" pueda ser resuelto por una computadora, el mismo debe ser escrito en un "lenguaje de programación" de nuestra preferencia, siguiendo las reglas de sintaxis del mismo.

- Esta tarea se denomina programación y el algoritmo escrito se llama programa

Un Algoritmo es un conjunto ordenado y finito de operaciones que deben seguirse para la solución de un problema específico.

Características de los Algoritmos

PRECISO: Debe indicar el orden de realización de cada paso.
DEFINIDO: Si se ejecuta dos veces el algoritmo con los mismo datos, este debe arrojar el mismo resultado.
FINITO: Debe finalizar en algún momento.

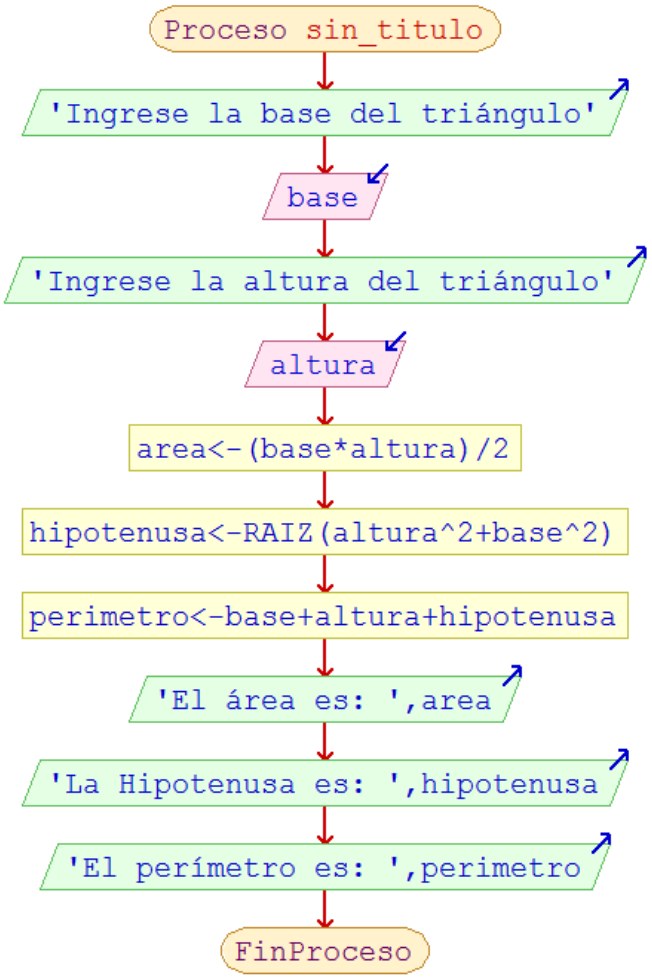
Elementos de un Algoritmo

Entrada – Proceso – Salida

Herramientas para el diseño de Algoritmos

Pseudocódigo: Consiste en un lenguaje de especificación de Algoritmos, que combina el lenguaje natural (humanos) y cualquier lenguaje de programación (ordenadores) específico.

Diagrama de Flujo: El diagrama de flujo o diagrama de actividades es la representación gráfica del algoritmo o proceso. Se utiliza en disciplinas como programación, economía, procesos industriales y psicología cognitiva.



Teorema fundamental de la programación:

Todo algoritmo (habido y por haber) se puede representar en pseudocódigo utilizando solamente las estructuras secuencial, condicional y repetitiva