

Módulo: Programación

Contenido

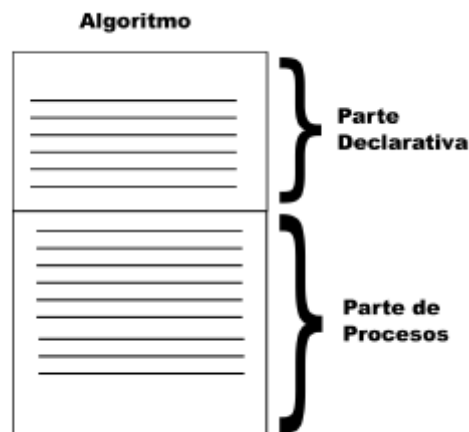
Estructura de un Algoritmo.....	3
Practicando.....	4
Técnicas de Programación.....	14
Introducción.....	14
Técnicas de programación algorítmica.....	14
Estructuras de Control.....	16
Detalle de los Sentencias de PSeInt Comandos.....	18
Inicio y Fin de un Proceso.....	18
Asignación.....	18
Definición de Variable.....	18
Expresiones y operadores.....	19
Expresiones Aritméticas.....	19
Expresiones lógicas.....	20
Funciones Internas o Bibliotecas:.....	22
Entrada y Salida de Datos.....	22
Practicando.....	23
Ejercicios Secuenciales.....	26
Anexo: Descripción de funciones y librerías.....	27

Diagramas de flujo y pseudocódigo

Estructura de un Algoritmo.

En esta materia vamos a usar la siguiente estructura de Algoritmo.

- Primero se escriben las sentencias referidas a todo lo declarativo, que es donde se describen los datos (nombres de variables y tipo de dato para esa variable) que se van a utilizar en el procedimiento de resolución.
- En la Parte de procesos, es donde se describen las acciones del algoritmo en sí (la lógica de resolución).



En algún momento mencionamos las palabras reservadas que no son otra cosa que las palabras propias por ejemplo del Pseudocódigo.

A continuación listamos las palabras reservadas de PSeInt.

- Proceso Subproceso Inproceso
- Definir Como Logico Entero Real Caracter
- Si Entonces Sino FinSi
- Para Paso Hacer Finpara
- Mientras FinMientras Repetir
- Escribir Leer Dimension Borra Pantalla

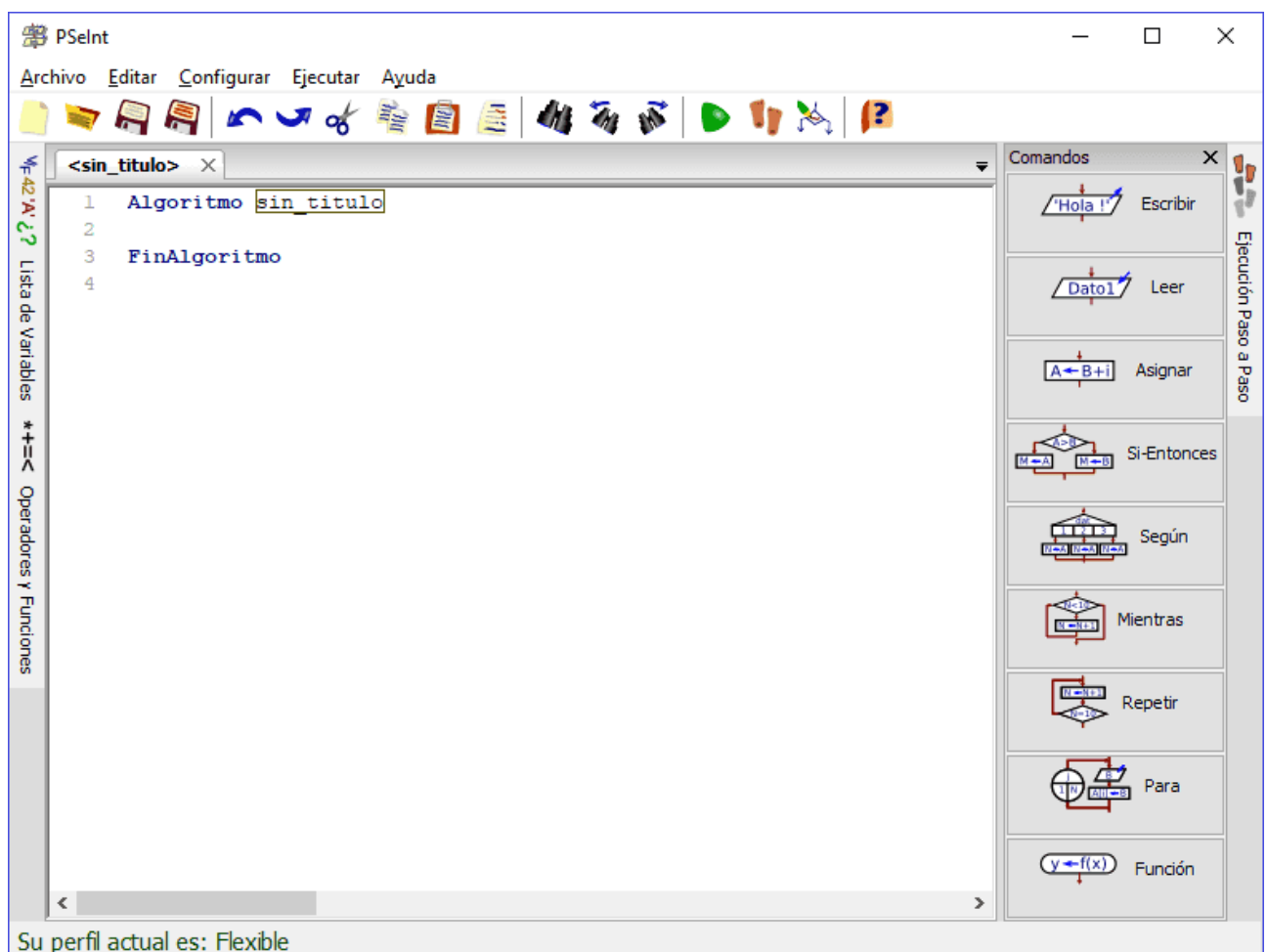
Practicando...

Vamos a empezar por lo más sencillo: escribir en pantalla.

Si queremos crear un programa que muestre algún texto en pantalla, en la mayoría de versiones de pseudocódigo usaremos la orden ESCRIBIR (en otras versiones puede ser IMPRIMIR o MOSTRAR). A continuación de esta palabra detallaremos, entre comillas, el texto que deseamos que aparezca en pantalla.

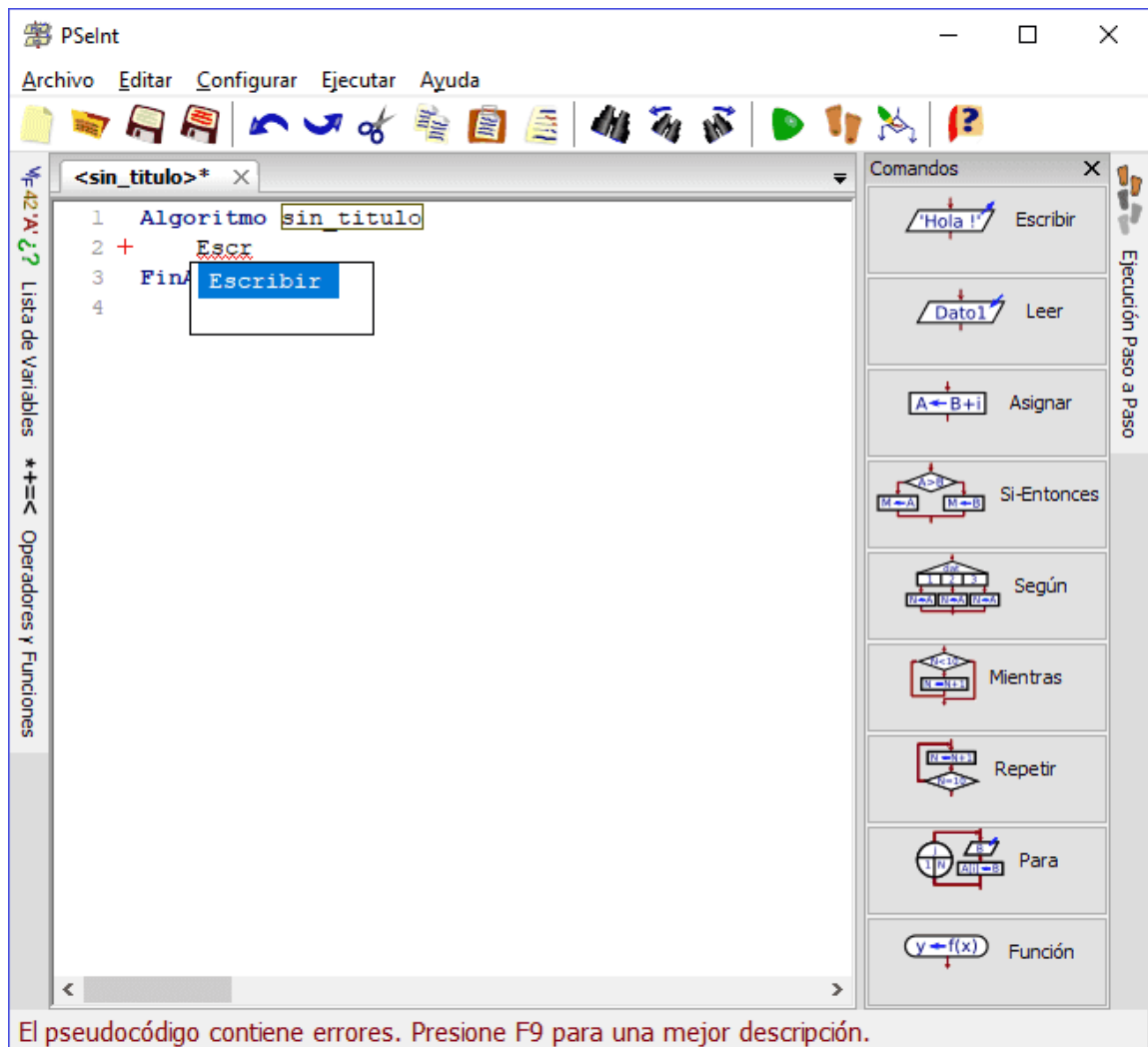
Escribir "Hola"

Hacerlo usando PSEINT también es casi igual de simple. En esta herramienta, cada programa debe encerrarse entre las palabras "Algoritmo" (para indicar dónde comienza) y "FinAlgoritmo" (para señalar dónde termina). Como curiosidad, si empleas una versión antigua de PseInt, quizá emplee las palabras "Proceso" y "FinProceso". En cualquier caso, eso no añade ninguna dificultad, porque cuando entramos a PSEINT, ese esqueleto de programa ya aparece escrito:

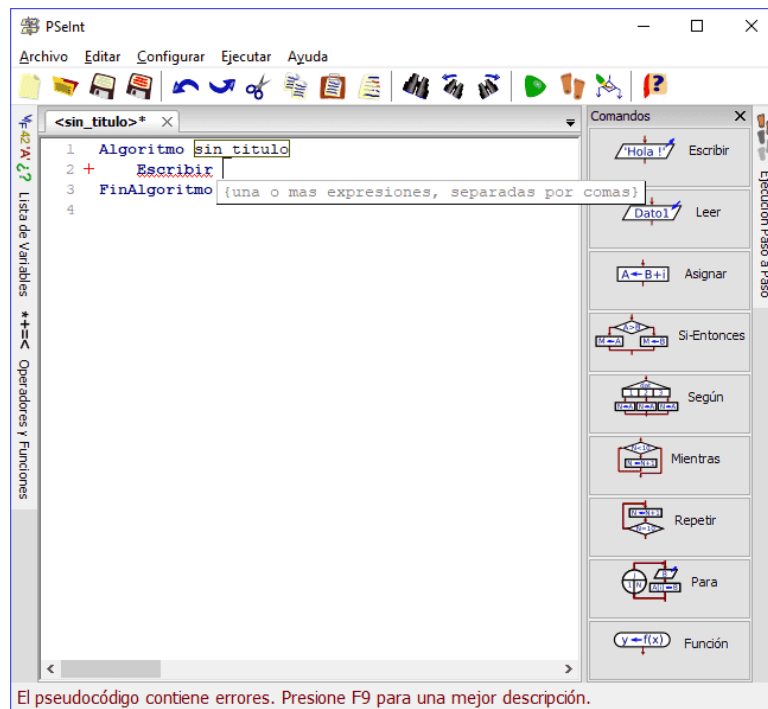


En la línea vacía que hay entre "Algoritmo" y "FinAlgoritmo" deberemos comenzar a teclear nuestro programa, en este caso empezando por la palabra "Escribir". Veremos que PSEINT se da

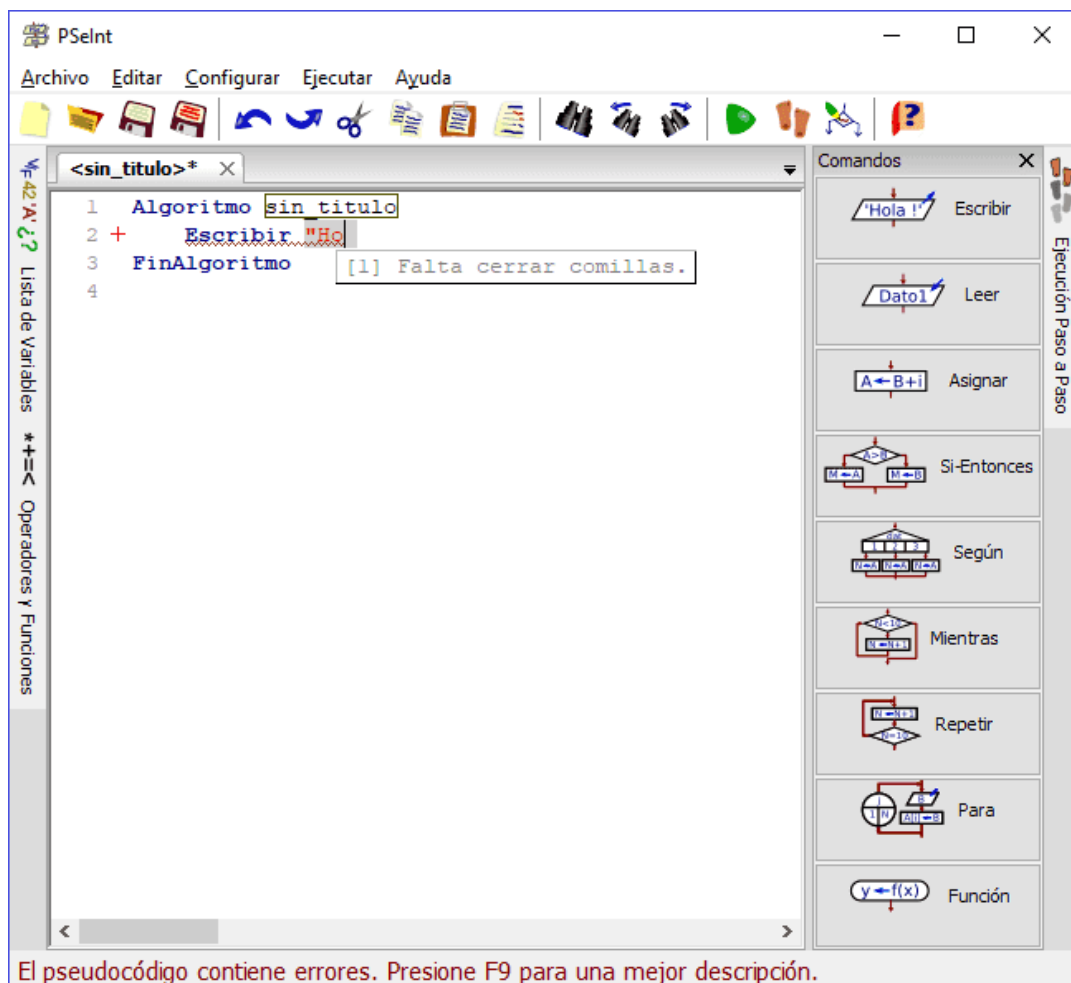
cuenta de que se trata de una orden que conoce, y nos recuerda cómo es la orden exacta, incluso cuando apenas llevamos unas pocas letras:



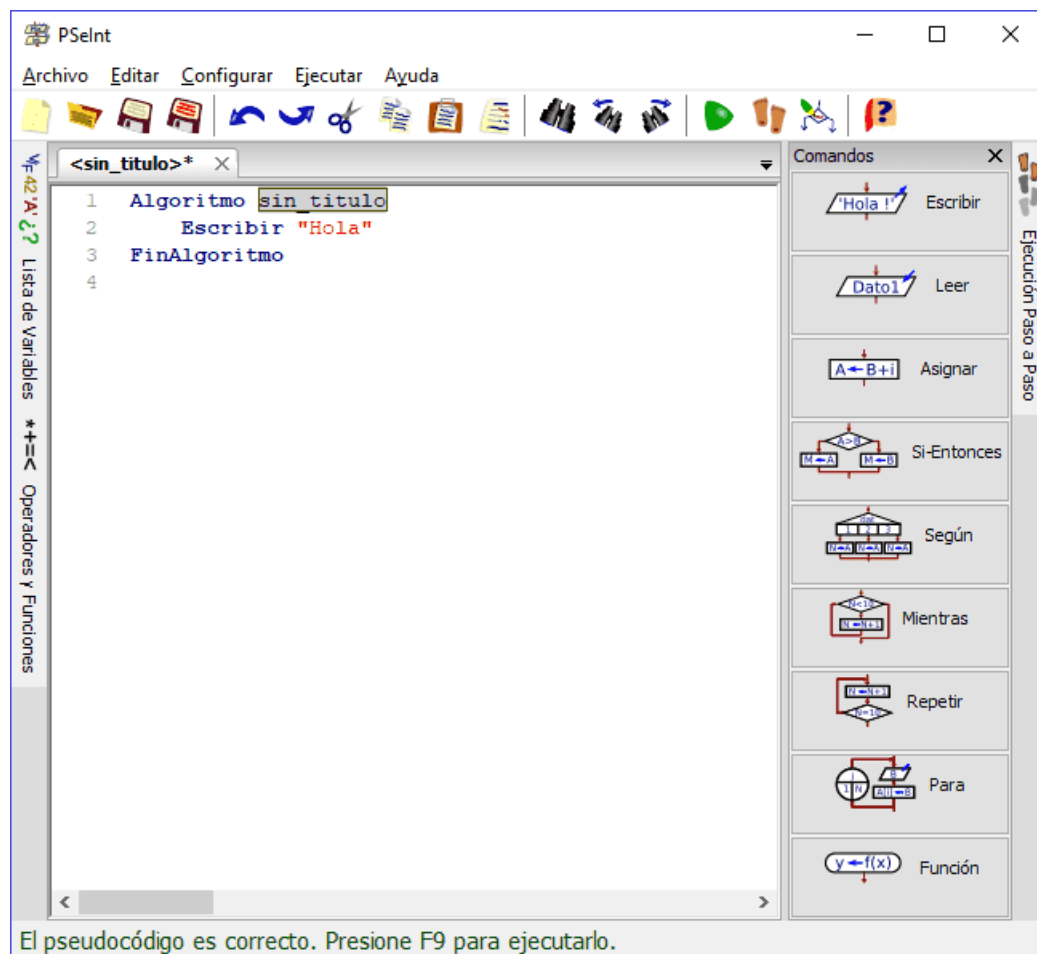
Podemos terminar de teclear la palabra, o bien pulsar Intro para que se complete automáticamente. En cualquier caso, tras terminar la palabra "Escribir", se nos propone que escribamos "una o más expresiones, separadas por comas"; en nuestro caso, será la palabra "Hola", entre comillas:



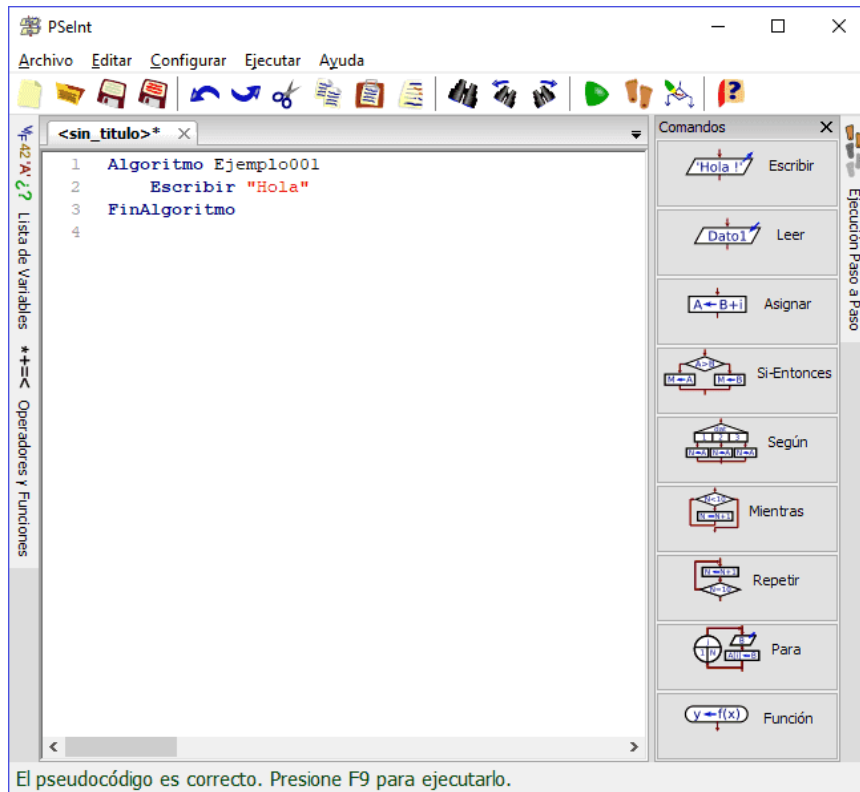
Mientras que estamos tecleando, PseInt mostrará un mensaje de aviso que nos recuerda que no será correcto hasta que terminemos de escribir el texto y cerremos las comillas:



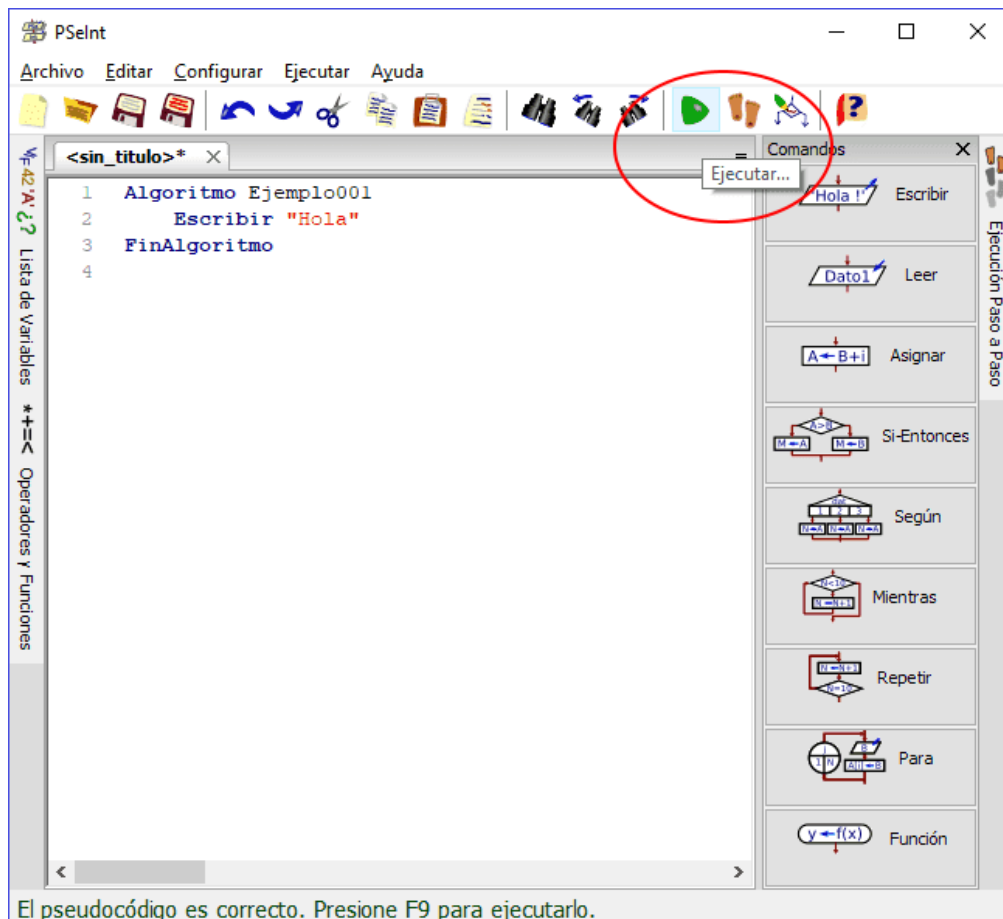
El programa casi completo, a falta de un pequeño detalle, debería quedar así:



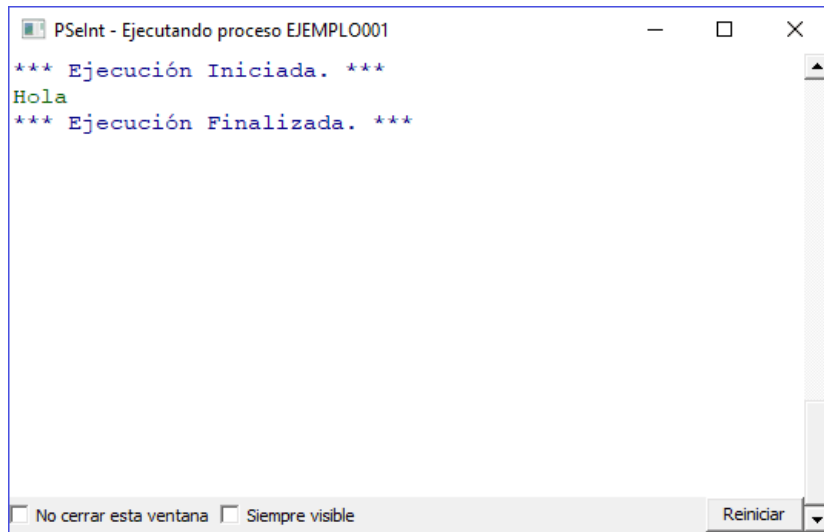
La primera línea todavía es "Proceso sin_titulo". Podemos emplear un nombre más significativo para nuestro programa. Como este es el primero de los ejemplos del curso, el nombre podría ser "Ejemplo001":



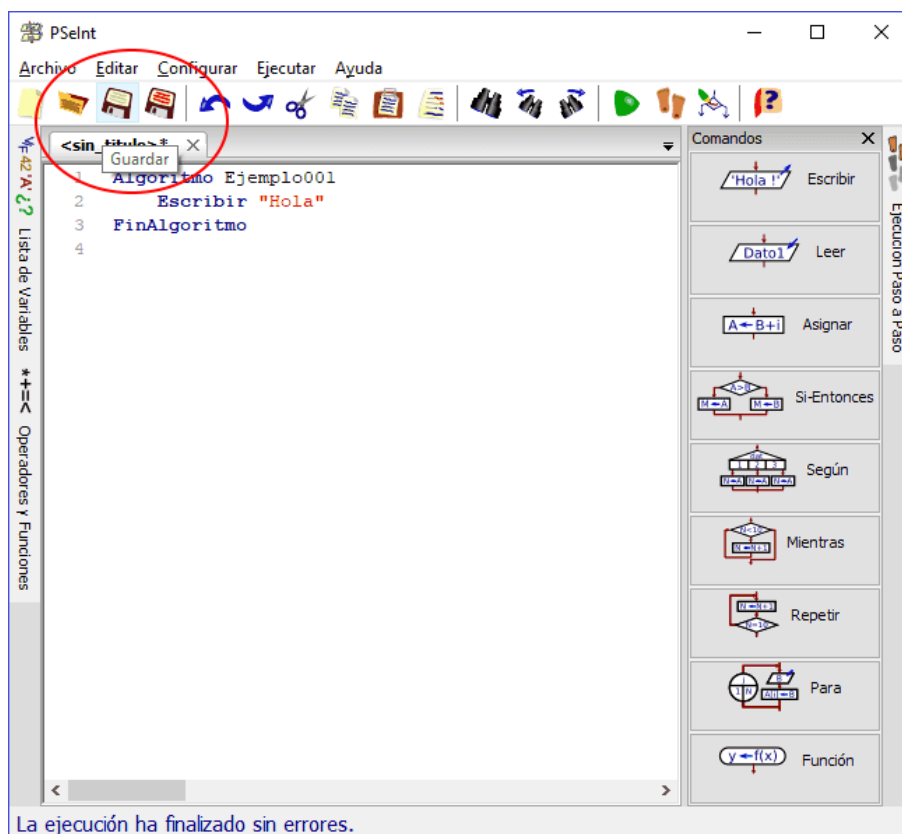
Nuestro programa está completo: podemos comprobar su funcionamiento pulsando el botón "Ejecutar", cuya imagen recuerda al "Play" de los equipos de música:



Y entonces aparecerá una nueva ventana que nos muestra el resultado de nuestro programa (por supuesto, se trata de la palabra Hola):

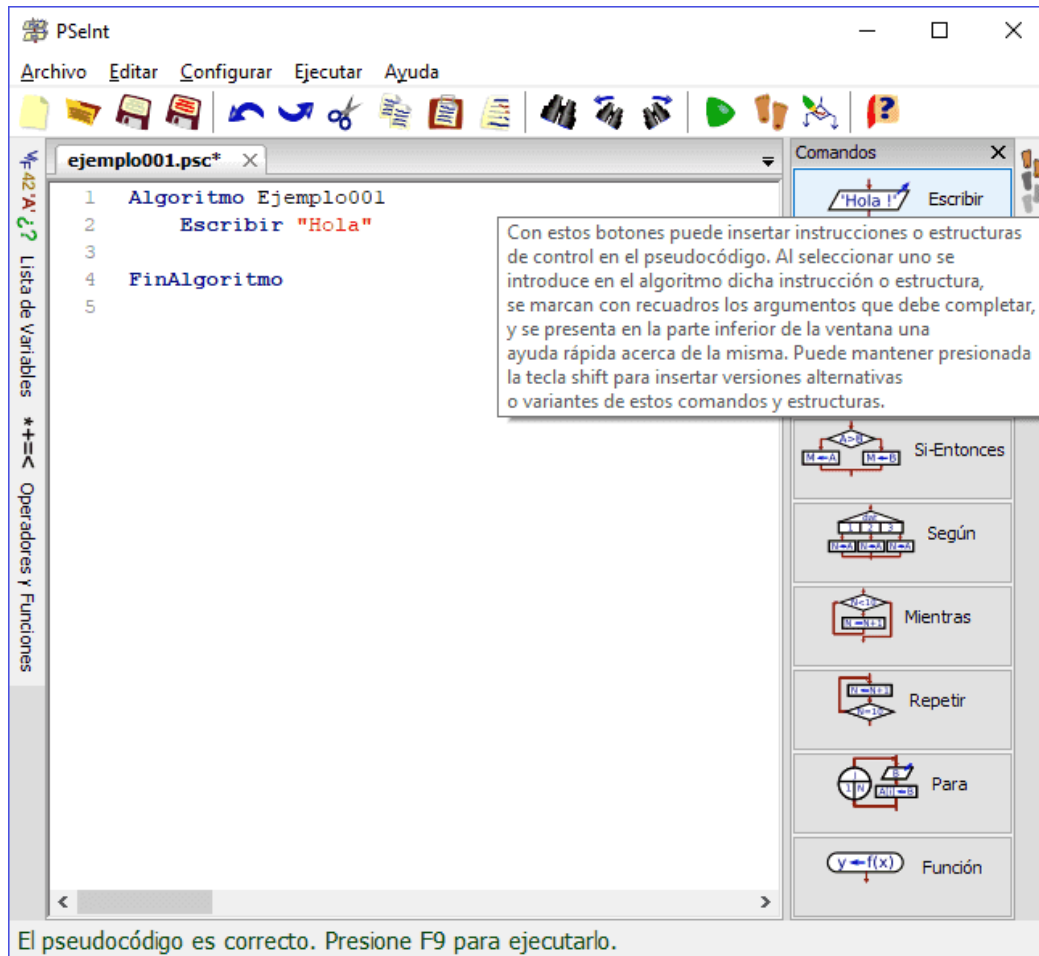


Si nos interesa guardar el programa para poder acceder a él más adelante, deberemos usar la opción "Guardar":

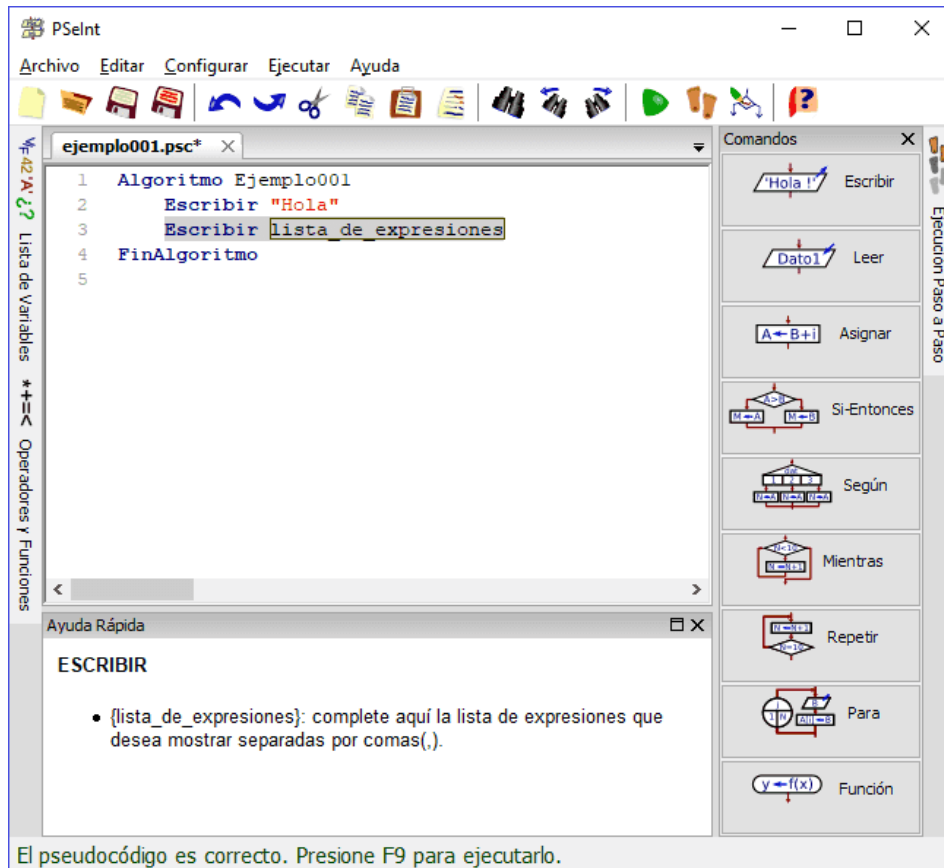


Un programa puede estar formado por más de una orden, claro. Bastará con pulsar Intro después del final de la orden "Escribir", para crear una nueva línea en blanco. De hecho, ni siquiera es necesario teclear la orden, porque PseInt permite utilizar las estructuras de

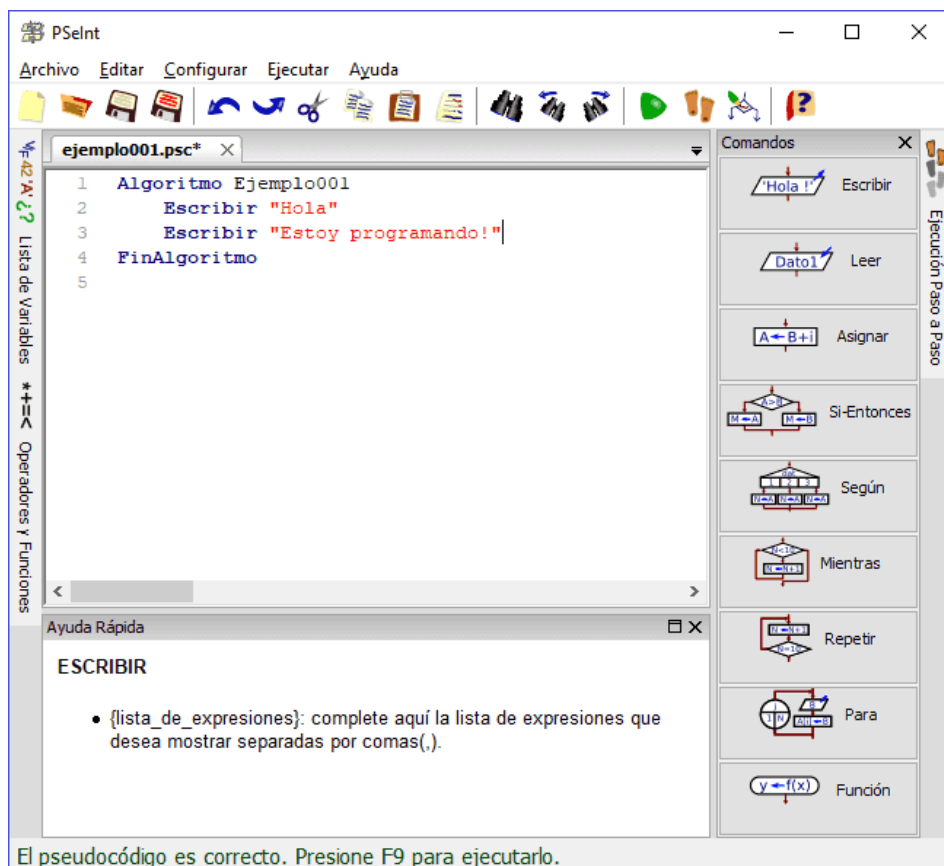
programación más habituales simplemente haciendo clic en el correspondiente botón del panel derecho:



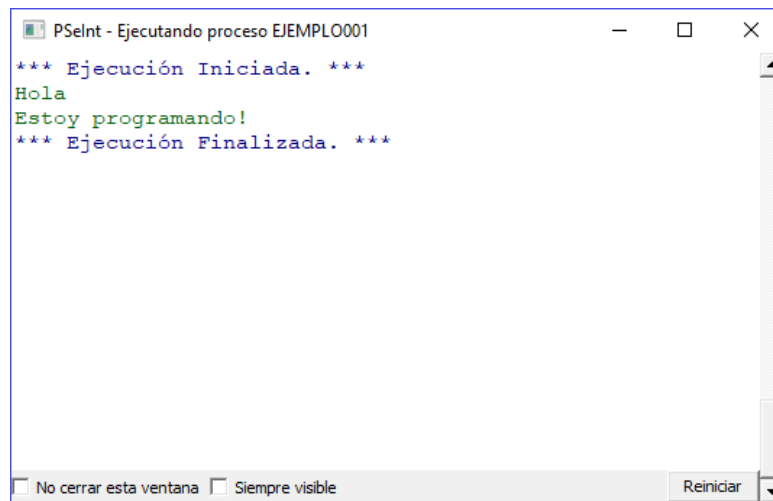
En ese caso, aparecerá un panel inferior, que nos recordará detalles adicionales sobre esa orden:



Añadimos otro texto que deseemos mostrar:



Y el nuevo resultado del programa sería:

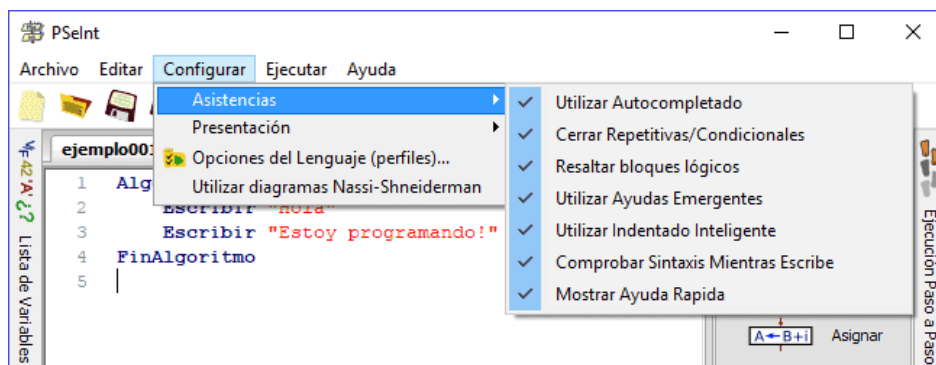


```
PSeInt - Ejecutando proceso EJEMPLO001

*** Ejecución Iniciada. ***
Hola
Estoy programando!
*** Ejecución Finalizada. ***

[ ] No cerrar esta ventana [ ] Siempre visible [Reiniciar]
```

Si te parece que PseInt te da demasiadas ayudas (aunque quizá nunca sean demasiadas cuando uno empieza), las podrías desactivar desde el menú Configurar / Asistencias:



Por supuesto, no sólo podemos escribir textos prefijados. Podemos usar nuestro ordenador como calculadora, que nos muestre el resultado de una operación aritmética:

```
Algoritmo Ejemplo002

    Escribir 20+30

FinAlgoritmo
```

Como es habitual en matemáticas, el símbolo + será el que utilizaremos para calcular una **suma**. La **resta** se indicará con -, la **multiplicación** con *, la **potencia** con ^ y la **división** con /. Una operación menos habitual, el **resto de la división**, también tiene un símbolo asociado: %. Lo usaremos con frecuencia para saber si un número es múltiplo de otro (por ejemplo, será múltiplo de 10 si su resto entre 10 es 0, o será impar si su resto entre 2 es 1).

```
Algoritmo Ejemplo002b
```

```
Escribir 15%2  
FinAlgoritmo
```

Se puede escribir varios textos en la misma línea si se emplea la orden ESCRIBIR SIN SALTAR (o Escribir Sin Bajar), como en este ejemplo:

```
Algoritmo Ejemplo002c  
    Escribir Sin Saltar "9876 * 54321 = "  
    Escribir 9876 * 54321  
FinAlgoritmo
```

También podemos incluir **comentarios**, aclaraciones, que nos ayuden a explicar la lógica del programa o cualquier detalle que no sea evidente. En el caso de PseInt, cualquier línea que comience con una doble barra (//) se considerará un comentario y no se tendrá en cuenta en el momento de analizar nuestro programa:

```
// Ejemplo de comentario en un fuente de PseInt
```

En el próximo apartado puntualizaremos un poco más lo que hemos hecho en este último programa y poco después veremos cómo comprobar condiciones.

Ejercicio de repaso propuesto 2.1: Crea un programa que escriba "Comienzo a aprender"

Ejercicio de repaso propuesto 2.2: Crea un programa que escriba el resultado de multiplicar 123 por 134

Ejercicio de repaso propuesto 2.3: Crea un programa que calcule el cuadrado de 25

Ejercicio de repaso propuesto 2.4: Crea un programa que escriba el resultado de dividir 37 entre 5

Ejercicio de repaso propuesto 2.5: Crea un programa que escriba el resto de la división de 37 entre 5

Técnicas de Programación.

Introducción

Muchas veces, encontrar la solución a un problema no resulta una tarea sencilla. Primeramente hay que comprender bien cuál es el problema que se nos plantea, al cual le debemos encontrar su solución. Luego tendremos que identificar bien a la información que nos explicita el enunciado del problema, es decir identificar los datos. Una vez _ados los datos debemos deducir del enunciado cual es el o los resultados que nos debe brindar la solución del problema. Cumplimentadas estas dos etapas: identificación de datos e identificación de resultado, recién estaremos en condiciones de comenzar a desarrollar una estrategia solución. Cuando la estrategia solución sea una secuencia de acciones, exactas, precisas y finitas que nuestro procesador (ente: persona o máquina) puede ejecutar con el solo hecho de enunciarlas, estaremos formulando el algoritmo solución de nuestro problema.

Técnicas de programación algorítmica

La elaboración de un algoritmo se puede realizar en forma totalmente libre, sin seguir los lineamientos de ningún modelo o en caso contrario aplicando las orientaciones de un prototipo. En el desarrollo de un algoritmo siguiendo un modelo establecido podemos citar, entre otros, a los siguientes modelos:

Modelo declarativo...

En este modelo se declara una serie de proposiciones, en general todo tipo de transformaciones que relatan el problema e individualizan su solución.

Algunos lenguajes de programación que responden a este paradigma son, por ejemplo: Prolog, Lisp, Maude, Sql.

Modelo imperativo (es el que emplearemos introductoriamente para aprender a programar).

En este modelo se detallan todos los pasos necesarios para encontrar la solución del problema.

Las acciones se ejecutan secuencialmente, siguiendo una estructuración. Esta estructuración se implementa a través de las estructuras de control.

Ejemplo:

Inicio

Acción 1

Acción 2

Acción 3

Hacer n veces

Si expresión es verdadera entonces

Acción 4

Acción 5

Acción 6

Sino

Acción 7

Acción 8

Fin selección

Fin repetición

Acción 9

Acción 10

Fin

Entre los lenguajes que responden a esta forma de desarrollar programas podemos mencionar, entre otros: Fortran, Pascal, C, etc.

Modelo orientado a objetos

Los algoritmos que siguen este modelo se caracterizan porque tienen en cuenta las relaciones que existen entre todos los objetos que intervienen. Cada objeto o entidad que interviene en la solución tiene una determinada conducta, estado e identificación. En este modelo no se debe preguntar: ¿qué hace el algoritmo?, sino preguntar: ¿quién o qué lo hace?. Algunos lenguajes de programación que siguen este modelo son: Smalltalk, C++, HTML, **Java**, etc.

Estructuras de Control

En la construcción de un algoritmo las acciones que lo integran deben agruparse de la forma que dicha resolución lo exija.

Existirán para un mismo algoritmo muchas formas de organizar sus acciones, primitivas o no, que conduzcan a la solución requerida. Pensaremos en esas organizaciones como no excluyentes y que se podrán incluir o asociar entre ellas.

Utilizaremos solamente tres únicas formas de disposición de las acciones.

- Hay veces que las acciones necesitan agruparse o estructurarse en forma de **serie o sucesión**.
- Otras veces las acciones deberán reunirse en un proceso **repetitivo**.
- En distintas ocasiones la agrupación de las acciones deberá responder a un proceso de **selección**, que divide el camino para agrupar las acciones siguientes entre dos o más opciones..

Llamaremos estructuras de control a las organizaciones que controlan la ejecución de las acciones en un algoritmo.

Las estructuras de control son las que establecen el orden de ejecución de las acciones.

Permiten especificar la coordinación y regulación del algoritmo, porque dirigen la dirección que debe seguir el flujo de información en el mismo.

La Técnica que va de lo “general” a lo “particular”, es decir, consiste en diseñar los algoritmos en etapas, partiendo de los conceptos generales hasta llegar a los detalles.

Se podría continuar con más niveles de división. La ejecución de cada uno de los algoritmos intervinientes constituirá la ejecución del algoritmo principal. Cada uno de estos algoritmos deberá reunir las siguientes características:

1. Tener un solo punto de inicio y un solo punto de finalización.
2. Toda acción de cada algoritmo es accesible, es decir, existe al menos un camino que va desde el inicio hasta el fin del mismo, que se puede seguir y pasa a través de dicha acción.
3. No existirán repeticiones que nunca concluyan, es decir, infinitas.

Podemos ver que esto coincide totalmente con la definición de ¿Qué es un Algoritmo?

"Un algoritmo podrá ser escrito utilizando únicamente tres tipos de estructuras u organización de las acciones: secuencial, selectiva y repetitiva".

Ejemplo: Desarrollar en pseudocódigo el algoritmo correspondiente al siguiente enunciado: Dados dos números reales calcular su suma y su producto y mostrar los resultados. El código secuencial sería:

Proceso suma

Definir n1, n2, sum, producto como real;

Leer n1;

Leer n2;

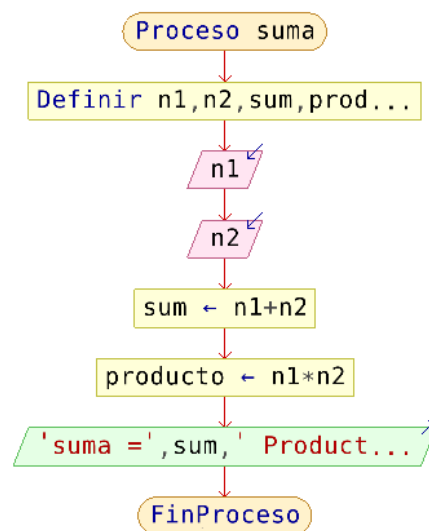
sum=n1+n2;

producto=n1*n2;

Escribir "suma =", sum, " Producto+", producto;

FinProceso

El Diagrama de Flujo para este Pseudocódigo sería:



Se puede decir que en este algoritmo las acciones Leer(), asignación de la variable suma, asignación de la variable producto y la acción Escribir() están en serie o secuencia. Es decir, cuando termina de ejecutarse una se continúa con la ejecución de la siguiente.

En esta sección comenzaremos a conocer la sintaxis y semántica del lenguaje que entiende nuestro procesador. Veremos algunas de las acciones primitivas del lenguaje de Psuedocódigo que utilizamos en PSeInt

Detalle de los Sentencias de PSeInt Comandos

Observar : Cada línea que termina una sentencia debe tener punto y coma (;).

Inicio y Fin de un Proceso

Para comenzar a escribir un programa es necesario iniciar con el comando Proceso seguido del nombre del **proceso** (Ej: Proceso Sumador), el nombre no debe llevar espacios y es recomendable que sea lo más representativo a lo que hace el código. Al finalizar se debe colocar el comando **FinProceso**, esto cierra el código del programa.

Ejemplo:

Proceso Sumador

..

..

FinProceso

Asignación

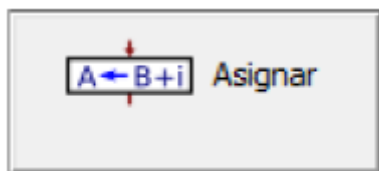
Esta instrucción permite almacenar el valor en una variable, ya sea resultado de una operación o bien el valor de otra variable

El comando es **<variable> = <expresión>;** primero evalúa la expresión de la derecha y luego asigna el resultado a la variable de la izquierda.

Es importante que el tipo de variable que involucra a la expresión coincida con el tipo de variable de asignación.

En caso de no definirse el tipo este se asignará automáticamente por la aplicación.

C = A + B; J = 8;



Definición de Variable

La instrucción Definir nos permite explicitar el tipo de una o más variables que se utilizarán en el programa. Antes de usar una variable en el proceso, es necesaria indicar que va a existir, esto es definirla y a la vez es necesario indicar de que tipo va a ser, para ello luego de los nombres

de las variables, se coloca el indicador **como**, y luego el tipo de variable. Los tipos de variables permitidos son :

- **REAL: números reales (decimales)**
- **ENTERO:**
- **LOGICO verdadero o falso**
- **CARACTER**
- Ejemplo:
- **Definir A como ENTERO;**
- **Definir b como REAL;**

Expresiones y operadores

Una **expresión** describe un cálculo a efectuar cuyo resultado es un único valor.

Una expresión está constituida por:

- **operandos.**
- **operadores.**

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

Las expresiones pueden ser de distintos tipos de acuerdo al tipo de dato de su resultado.

- numérica.
- relacional
- lógica
- de carácter/cadena de caracteres.

Expresiones Aritméticas.

Las expresiones aritméticas son análogas a las fórmulas matemáticas. Los elementos intervinientes son numéricos (reales, enteros) y las operaciones son aritméticas. Los operadores son:

Operador	Significado	Tipos de Operando	Tipo de Resultado
+	Suma	Enteros o Reales	Enteros o Reales (Ver Observación)
-	Resta	Enteros o Reales	Enteros o Reales (Ver Observación)
*	Multiplicación	Enteros o Reales	Enteros o Reales (Ver Observación)
/	Cociente	Enteros o Reales	Enteros o Reales (Ver Observación)

Observación a la hora de programar: Cuando en una operación los dos operandos son enteros el resultado es entero. Lo mismo sucede cuando son Reales. Pero si al menos uno de los operandos es real y el otro entero, se puede producir una conversión del tipo de dato, esto es conocido como **conversión implícita del Lenguaje, y el resultado dependerá del lenguaje de programación concreto.**

Expresiones lógicas

Para el caso de los operadores & y |, la evaluación se realiza en cortocircuito.

Esto significa

- que si existen dos expresiones unidas por el operador &. y la primera se evalúa como falso será falso y no se evalúa la segunda,
- para el caso de que estén unidas por el operador | y la primera se evalúa como Verdadero, la segunda no se evalúa ya que no altera el resultado.

Las siguientes tablas exhiben la totalidad de los operadores de este lenguaje reducido:

Operador	Significado	Ejemplo
Relacionales		
>	Mayor que	3>2
<	Menor que	'ABC'<'abc'
=	Igual que	4=3
<=	Menor o igual que	'a'<='b'
>=	Mayor o igual que	4>=5
<>	Distinto que	'a'<>'b'
Lógicos		
& ó Y	Conjunción (y).	(7>4) & (2=1) //falso
ó O	Disyunción (o).	(1=1 2=1) //verdadero
~ ó NO	Negación (no).	~(2<5) //falso
Algebraicos		
+	Suma	total <- cant1 + cant2
-	Resta	stock <- disp - venta
*	Multiplicación	area <- base * altura
/	División	porc <- 100 * parte / total
^	Potenciación	sup <- 3.41 * radio ^ 2
% ó MOD	Módulo (resto de la división entera)	resto <- num MOD div

La operación Y retornará VERDADERO si todos los operandos son verdaderos, en caso contrario retorna FALSO

La operación O retornará verdaderos cuando ALGUNO de los operandos sea verdadero, en caso contrario retornará FALSO

La operación NO retorna FALSO al aplicarse a VERDADERO y viceversa (Estos es niega el valor)

Orden de Precedencia de Evaluación.

1. Paréntesis (empezando por los más internos)
2. Potencias
3. Productos y Divisiones
4. Sumas y restas
5. Concatenación
6. Relacionales
7. Lógicos

Cuando hay dos operadores con la misma precedencia, se calcula primero la operación que está a la izquierda.

Practicando...

1) Si $A = 5$ y $B = 10$. ¿Qué valor asume C en cada caso?

- a) $C = A + B$
- b) $C = (B - A) / (A - 3)$
- c) $C = A * 5 + B$
- d) $C = 4 / 2 * 3 / 6 + B / 2 / 1 / 5 ^ 2 - A$
- e) $C = B + A + 25 + 4 * (A * B) / 2$
- f) $C = B ^ A + (A / B) - 20 / B$
- g) $C = A - B / 2 A * 5$
- h) $C = A * B * 5 (B - A) ^ 3$

2) ¿Cual es el resultado en cada caso?

- a) $10.5 / 3 =$
- b) $1 / 4 =$
- c) $1 / 4.0 =$
- d) $3 + 6 * 2 =$
- e) $-4 * 7 + 2 ^ 3 / 4 - 5 =$
- f) $12 + 3 * 7 + 5 * 4 =$

Funciones Internas o Bibliotecas:

Los lenguajes de programación traen incorporado funciones que conforman las funciones de biblioteca (library en inglés), estas no son ni mas ni menos que código que escribieron otras personas. En la mayoría de los lenguajes se pueden usar códigos escritos por otras personas, esto hace que nuestro trabajo sea mas sencillo. Estos códigos suelen estar muy bien probados.

En algunos casos esos códigos o funciones de librería no son libres y hay que pagar para obtenerlos.

Función	Significado
RC(X) o RAIZ(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
REDON(X)	Entero más cercano a X
AZAR(X)	Entero aleatorio en el rango [0;x-1]
ALEATORIO(A,B)	Entero aleatorio en el rango [A,B]
LONGITUD(S)	Cantidad de caracteres de la cadena S
MAYUSCULAS(S)	Retorna una copia de la cadena S con todos sus caracteres en mayúsculas
MINUSCULAS(S)	Retorna una copia de la cadena S con todos sus caracteres en minúsculas
SUBCADENA(S,X,Y)	Retorna una nueva cadena que consiste en la parte de la cadena S que va desde la posición X hasta la posición Y (incluyendo ambos extremos). Las posiciones utilizan la misma base que los arreglos, por lo que la primer letra será la 0 o la 1 de acuerdo al perfil del lenguaje utilizado.
CONCATENAR(S1,S2)	Retorna una nueva cadena resulta de unir las cadenas S1 y S2.
CONVERTIRANUMERO(X)	Recibe una cadena de caracteres que contiene un número y devuelve una variable numérica con el mismo.
CONVERTIRATEXTO(S)	Recibe un real y devuelve una variable numérica con la representación como cadena de caracteres de dicho real.

Practicando:

1) ¿Cual es el resultado en cada caso?

- a) $RC(25) =$
- b) $ABS(6) =$
- c) $ABS(-12.5) =$
- d) $TRUNC(6.64) =$
- e) $REDON(6.51) =$

Entrada y Salida de Datos

La instrucción **Escribir <variable>** nos permite mostrar en pantalla un valor, ya sea el contenido de una variable, el resultado directo de una expresión o bien un texto.

El comando **podría ser:**

Escribir <variable>;

Escribir <expresión>;

Escribir "TEXTO";

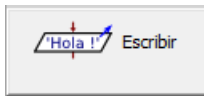
O una combinación de estas separadas por **coma (,)** donde se evalúa cada una de las expresiones y muestra en pantalla el contenido, en el caso de mostrar un texto siempre debe estar entre comillas dobles.

Ejemplo:

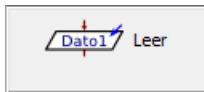
Definir A como ENTERO;

Leer A; Escribir "La Variable ", A, " es un entero";

En la representación de Escribir , se puede ver un Paralelogramo con una flechita hacia afuera.



En la representación de Leer , se puede ver un Paralelogramo con una flechita hacia afuera



Practicando....

Vamos a hacer algo un poco más complejo: vamos a sumar dos números que no estén prefijados dentro del programa, sino que deberá teclear el usuario.

Para eso, usaremos la orden "Leer", que nos permite obtener un dato que el usuario teclee y dejarlo guardado para utilizarlo después. Debemos dar un nombre temporal a estos datos que leemos del usuario. Parece razonable que el primer número que teclee el usuario se llame algo como "primerNumero", y el segundo sea algo como "segundoNumero". El resultado que queremos obtener será la suma de ese primer número y ese segundo número, así que nuestro programa podría quedar así:

Algoritmo Ejemplo003

Escribir "Dime un numero"

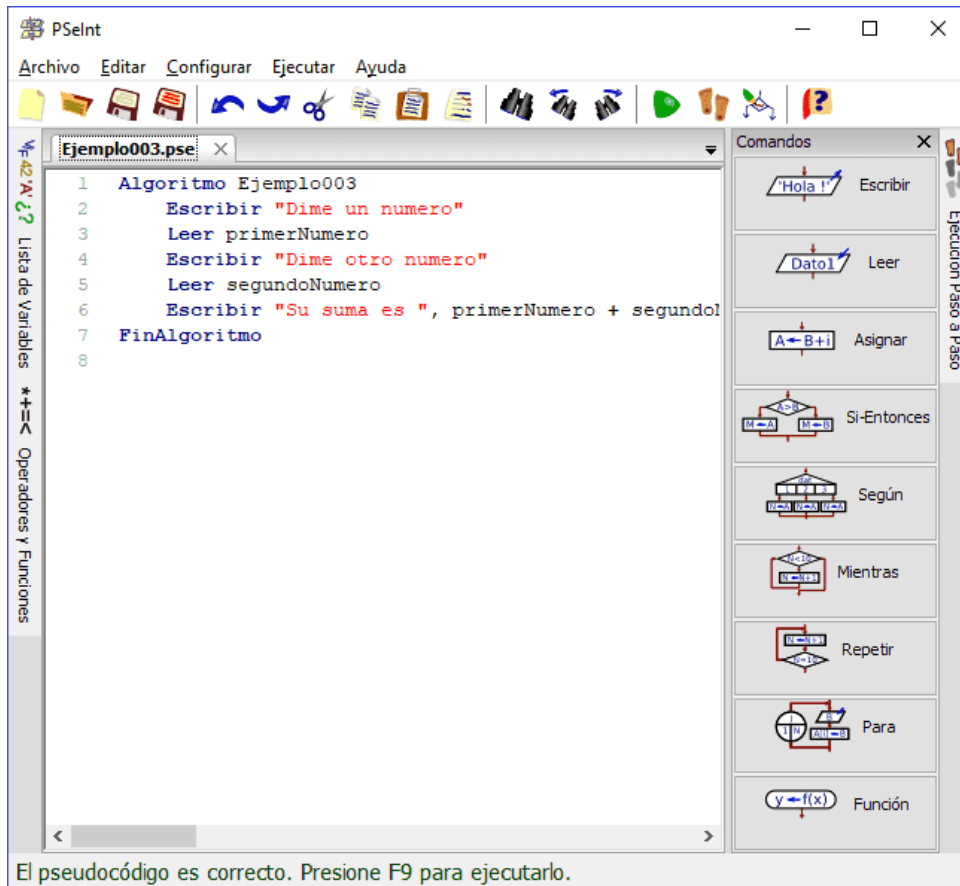
Leer primerNumero

Escribir "Dime otro numero"

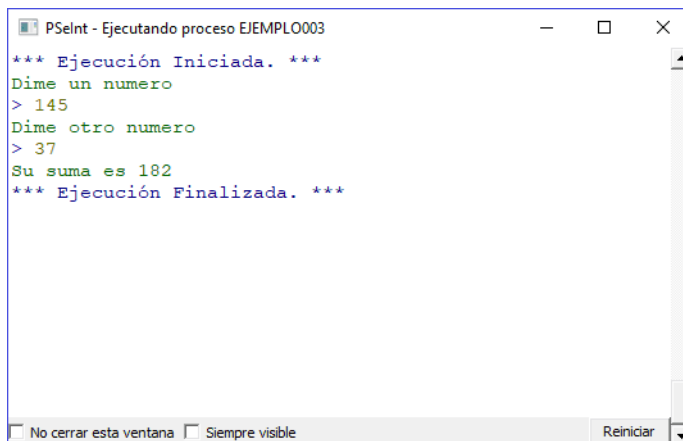
Leer segundoNumero

Escribir "Su suma es ", primerNumero + segundoNumero

FinAlgoritmo



El resultado de este programa debería ser algo como (dependiendo de los datos que se introduzcan):



Esas dos palabras, "primerNumero" y "segundoNumero" representan a números que no tienen un valor prefijado. Eso es lo que llamaremos "**variables**". En las variantes de pseudocódigo que emplean en muchas universidades y centros de estudio (ya habíamos avisado de que no existe ningún estándar totalmente asentado) debemos "declarar las variables" antes de usarlas: decir qué tipo de dato es el que queremos representar con ese nombre. En nuestro ejemplo anterior se trataba de dos números enteros, así que el programa sería:

Algoritmo Ejemplo003b

Definir primerNumero **como** Entero

Definir segundoNumero **como** Entero

Escribir "Dime un numero"

Leer primerNumero

Escribir "Dime otro numero"

Leer segundoNumero

Escribir "Su suma es ", primerNumero + segundoNumero

FinAlgoritmo

Esto se debe a que en la mayoría de lenguajes de programación "reales" es necesario detallar qué tipo de datos queremos guardar en cada variable, para que la herramienta de programación sepa exactamente qué cantidad de memoria será necesario reservar para guardar ese dato.

En el caso de PSEINT, se puede escoger entre distintas sintaxis. La sintaxis que viene "por defecto" (si no se cambia nada) es la llamada "flexible", que permite que no se declaren las variables antes de usarlas, pero existen otras variantes de sintaxis, empleadas por ciertas universidades, en las que sí puede ser obligatorio hacerlo.

Por otra parte, no siempre queremos que el valor de una variable lo introduzca el usuario. Habrá veces que seamos nosotros mismos los que demos el **valor inicial** a una variable desde nuestro programa, bien para usarlo en una serie de cálculos posteriores, o bien por legibilidad (es más fácil entender algo como "longitudCircunferencia = 2 * pi * radio" que algo como "longitudCircunferencia = 6.28 * radio").

La forma de dar un valor a una variable es con la secuencia de símbolos "<-":

```
radio <- 3  
longitudCircunferencia <- 2 * pi * radio
```

(esos símbolos representan una flecha, para indicar que el valor 3 se va a guardar dentro del espacio de memoria reservado para la variable llamada "radio").

¿Y qué ocurre si usamos una variable sin haberle dado valor? Esto sucede a veces por despiste, si tecleamos mal el nombre de una variable, como en este fragmento de programa:

```
primerNumero <- 2  
Escribir primerNmero
```

Si lees ese fragmento con cuidado, verás que el nombre de la variable que aparece en la segunda línea es incorrecto, falta la letra "u". ¿Qué sucede en ese caso? En algunos lenguajes (pocos, afortunadamente) se da por sentado que es una variable nueva, y se le da el valor 0; en el caso de PseInt, igual que en la mayoría de lenguajes actuales, obtendremos un mensaje de error que nos dirá que estamos usando una variable que no tenía valor.

Ejercicio de repaso propuesto 3.1: Crea un programa que escriba el resultado de multiplicar los dos números que introduzca el usuario

Ejercicio de repaso propuesto 3.2: Crea un programa que calcule la superficie de un rectángulo a partir de su base y su altura, y que después muestre el valor de dicha superficie.

Ejercicio de repaso propuesto 3.3: Crea un programa que calcule la superficie de un círculo a partir de su radio (la fórmula es " $\text{PI} * \text{radio}^2$ ") y que después muestre el valor de dicha superficie (pista: para calcular el cuadrado de un número puedes usar la operación "potencia": x^2 (mejor solución), o bien multiplicar el número por él mismo: $x^2 = x * x$).

Ejercicios Secuenciales

- Escribir un algoritmo que halle el promedio de tres valores A, B, C ingresados. El mismo debe mostrar los tres valores por separado y luego el valor promedio.
- Escribir un algoritmo que intercambie dos valores numéricos ingresados por teclado.

Anexo: Descripción de funciones y librerías

Función	Significado
RC(X) o RAIZ(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
REDON(X)	Entero más cercano a X
AZAR(X)	Entero aleatorio en el rango [0;x-1]
ALEATORIO(A,B)	Entero aleatorio en el rango [A;B]
LONGITUD(S)	Cantidad de caracteres de la cadena S
MAYUSCULAS(S)	Retorna una copia de la cadena S con todos sus caracteres en mayúsculas
MINUSCULAS(S)	Retorna una copia de la cadena S con todos sus caracteres en minúsculas
SUBCADENA(S,X,Y)	Retorna una nueva cadena que consiste en la parte de la cadena S que va desde la posición X hasta la posición Y (incluyendo ambos extremos). Las posiciones utilizan la misma base que los arreglos, por lo que la primer letra será la 0 o la 1 de acuerdo al perfil del lenguaje utilizado.
CONCATENAR(S1,S2)	Retorna una nueva cadena resulta de unir las cadenas S1 y S2.
CONVERTIRANUMERO(X)	Recibe una cadena de caracteres que contiene un número y devuelve una variable numérica con el mismo.
CONVERTIRATEXTO(S)	Recibe un real y devuelve una variable numérica con la representación como cadena de caracteres de dicho real.