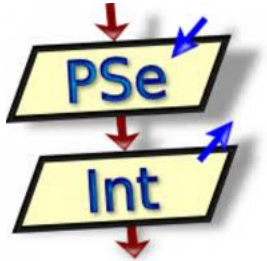
1.2 Pseudocódigo

Encarnación Sánchez Gallego

- Imprimir por pantalla
- Comentarios
- Variables
- Condicionales. Estructura SI
- Operadores
- Condicionales. Estructura SEGUN
- Bucles. Estructura mientras
- Bucles. Estructura Repetir Hasta Que
- Bucles. Estructura Para

  Extra: Funciones



Imprimir por pantalla

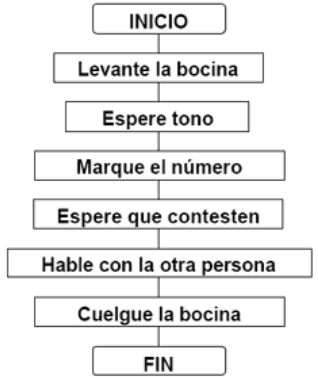
Si queremos crear un programa que muestre algún texto en pantalla, en la mayoría de versiones de pseudocódigo usaremos la orden ESCRIBIR (en otras versiones puede ser IMPRIMIR o MOSTRAR). A continuación de esta palabra detallaremos, entre comillas, el texto que deseamos que aparezca en pantalla.

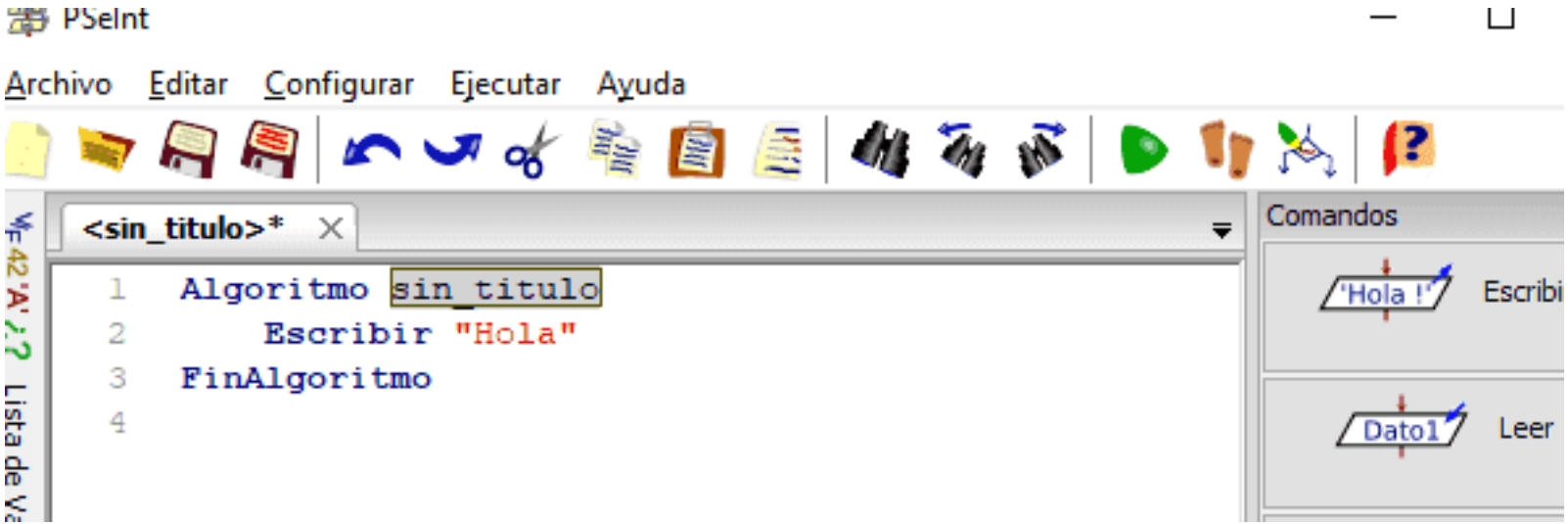
Escribir "Hola"

Pseudocódigo:

```
INICIO
  Levante la bocina
  Espere tono
  Marque el número
  Espere que contesten
  Hable con la otra persona
  Cuelgue la bocina
FIN
```

Diagrama de flujos:




PSeInt
Archivo Editar Configurar Ejecutar Ayuda
<sin_titulo>*
1 Algoritmo sin_titulo
2 Escribir "Hola"
3 FinAlgoritmo
4
Comandos
'Hola !'
Escribi
Dato1
Leer

COMENTARIO

Un comentario, es una opción que permite un lenguaje de programación para establecer algunas oraciones y párrafos y así comprender mejor el código que estamos programando. Tipos de comentarios en PSeint:

- Por línea // Esto es un comentario
- Por párrafo

/* esto es

un comentario*/

Variables

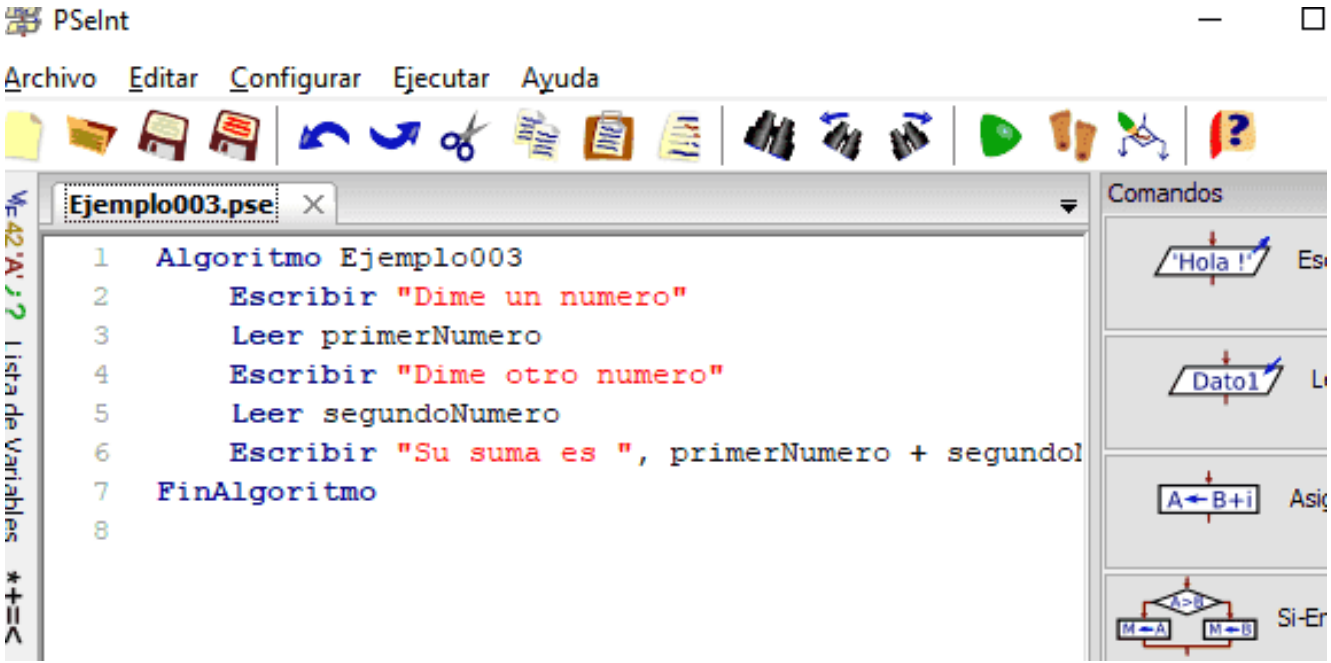
Es un espacio de memoria reservado para almacenar un valor, el cual se le reconoce como una etiqueta po nombre.

Con esa etiqueta o nombre se tiene acceso a la información que contiene.

Tipos de variables en PSeint:

- Numérico: Contiene número decimales o enteros
- Carácter: contiene cadena de caracteres

Variables


PSeInt
Archivo Editar Configurar Ejecutar Ayuda
Ejemplo003.pse
1 Algoritmo Ejemplo003
2 Escribir "Dime un numero"
3 Leer primerNumero
4 Escribir "Dime otro numero"
5 Leer segundoNumero
6 Escribir "Su suma es ", primerNumero + segundol
7 FinAlgoritmo
8
Lista de Variables
Comandos
'Hola !'
Dato1
A←B+i
A>B
M←A
M←B

Variables

Para eso, usaremos la orden "Leer", que nos permite obtener un dato que el usuario teclee y dejarlo guardado para utilizarlo después.

Deberemos dar un nombre temporal a estos datos que leemos del usuario.

Ejemplo: Sumar dos números

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:

Condicionales. Estructura SI

En casi cualquier secuencia de instrucciones para un ordenador, será vital poder comprobar si se cumple alguna condición.

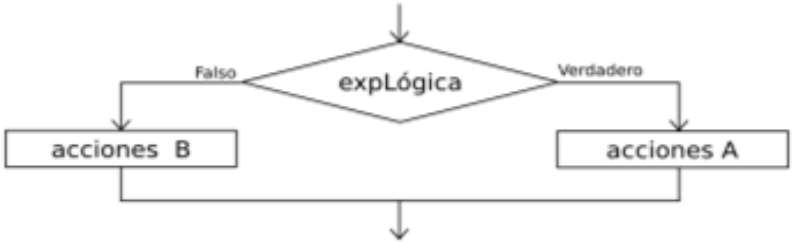
Una primera forma básica de comprobar condiciones es con la orden "SI". Su uso básico sería

Si condición

Entonces pasos_a_dar_si_es_verdadero

SiNo pasos_a_dar_si_es_falso

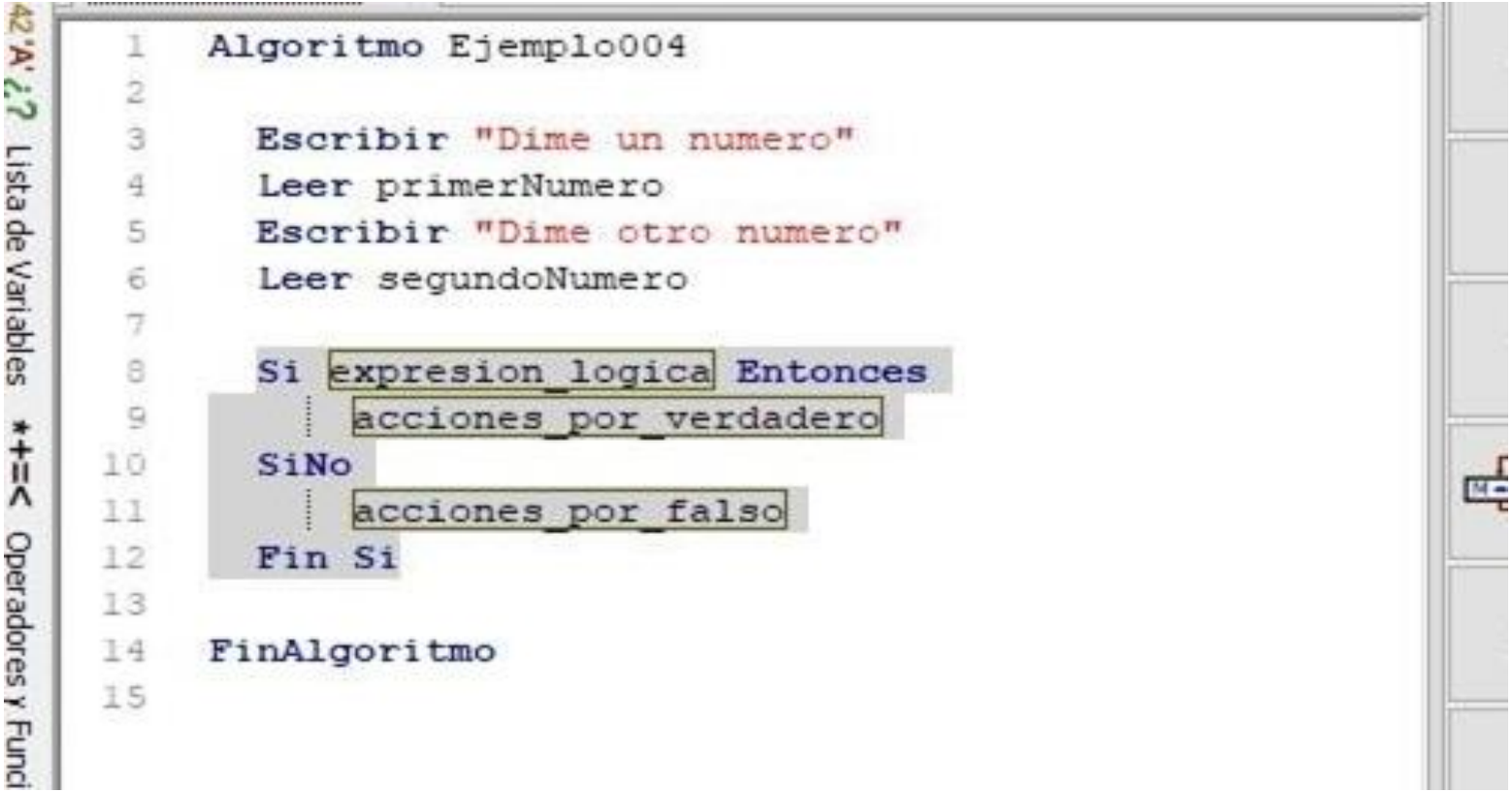
FinSi



Condicionales. Estructura SI

El bloque "SiNo" es opcional: podemos optar por no indicar lo que queremos que se haga cuando no se cumpla la condición.

Para ayudarnos a planificar el comportamiento de una secuencia de instrucciones, se suele usar como ayuda los llamados "diagramas de flujo". En estos diagramas, una condición se representa como un rombo, del que salen dos flechas: una para la secuencia de acciones a realizar si se cumple la condición y otra para cuando no se cumple:


Algoritmo Ejemplo004
Escribir "Dime un numero"
Leer primerNumero
Escribir "Dime otro numero"
Leer segundoNumero
Si expresion_logica Entonces
acciones_por_verdadero
SiNo
acciones_por_falso
Fin Si
FinAlgoritmo
Lista de Variables
Operadores y Funci

Operadores

Operador relacional	Significado	Ejemplo
>	Mayor que	3>2
<	Menor que	2<3
=	Igual que	3=3
<=	Menor o igual que	5<=5
>=	Mayor o igual que	6>=5

Operadores

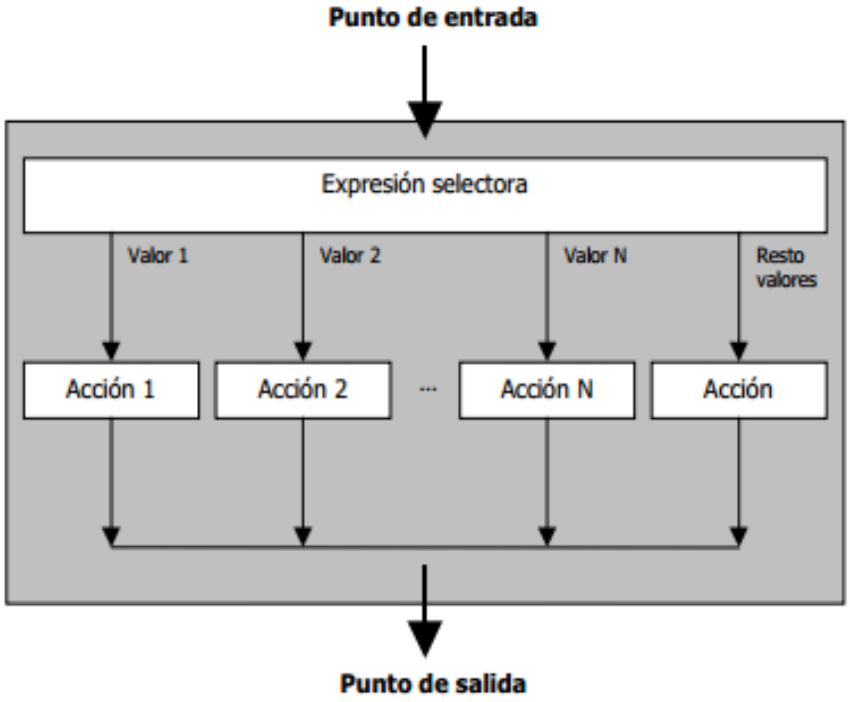
Operador lógico	*Significado*	*Ejemplo*
& ó Y	Conjunción (y).	(7>4) & (2=1) //falso
| ó O	Disyunción (o).	(1=1 | 2=1) //verdadero
~ ó NO	Negación (no).	~(2<5) //falso

```
Algoritmo Ejemplo004c

    Escribir "Dime un numero"
    Leer primerNumero
    Escribir "Dime otro numero"
    Leer segundoNumero

    Si primerNumero > segundoNumero Entonces
        Escribir "El primero es mayor"
    Sino
        Si primerNumero < segundoNumero Entonces
            Escribir "El segundo es mayor"
        Sino
            Escribir "Son iguales"
        FinSi
    FinSi

FinAlgoritmo
```

Condicionales. Estructura SEGUN

- Es frecuente tener que comprobar más de una condición a la vez, o bien varias condiciones consecutivas.
- Por ejemplo, en el sistema de notas del colegio, ciertas notas numéricas tienen "nombres" asociados: un 5 es un aprobado, un 9 y un 10 son sobresaliente, etc.
- Si queremos hacer un programa que convierta de la nota numérica a su equivalente escrito, podríamos emplear varias órdenes IF, una tras la otra. Pero en muchos lenguajes de programación (y, por tanto, también en muchas variantes de pseudocódigo) existe una alternativa más compacta y más legible: la orden "SEGUN".
- Esta orden permite hacer unas cosas u otras según el valor que tome una variable. Su uso sería así:

```
Segun variable Hacer
  valor1: pasos_a_dar_si_es_el_valor1
  valor2: pasos_a_dar_si_es_el_valor2
  valor3: pasos_a_dar_si_es_el_valor3
          …
  valorn: pasos_a_dar_si_es_el_valorn
De Otro Modo:
  pasos_a_dar_si_es_otro_valor
FinSegun
```

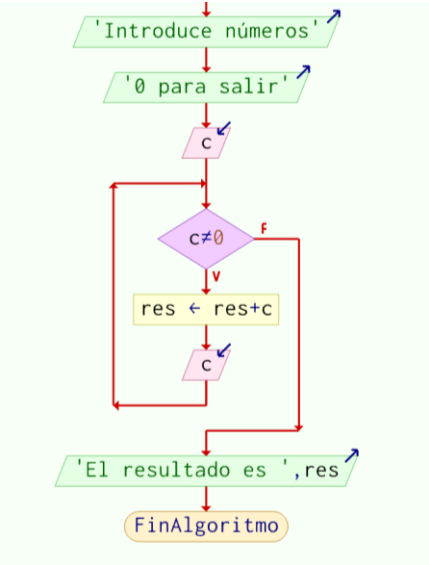


```
1  Algoritmo EjemploNota
2      Escribir "Introduzca la nota";
3      Leer nota;
4      Segun nota Hacer
5          10:
6              Escribir "Ha obtenido un sobresaliente alto";
7          9:
8              Escribir "Ha obtenido un sobresaliente bajo";
9          8:
10             Escribir "Ha obtenido un notable alto";
11         7:
12             Escribir "Ha obtenido un notable bajo";
13         6:
14             Escribir "Ha obtenido un aprobado alto";
15         5:
16             Escribir "Ha obtenido un aprobado";
17         De Otro Modo:
18             Escribir "Ha suspendido";
19     FinSegun
20 FinAlgoritmo
21
```

Bucles.

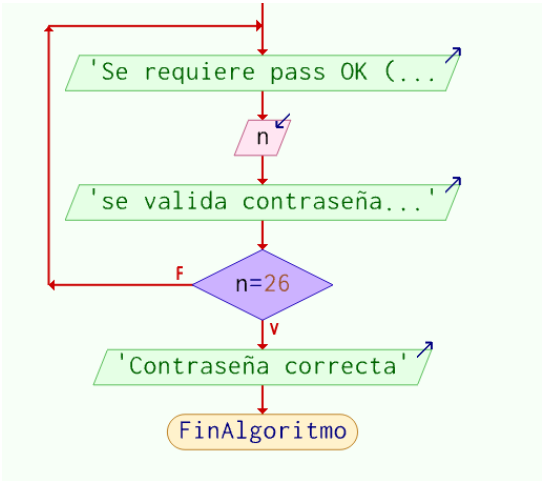
Mientras (while)

Se recomienda cuando:
- Queremos repetir una ejecución un número indeterminado de veces.
- En base a una condición.
- No se requiere realizar ninguna ejecución de base.



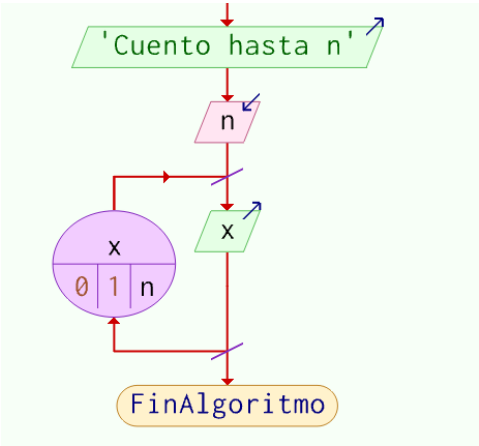
Repetir … hasta (do … while)

Se recomienda cuando:
- Queremos repetir una ejecución un número indeterminado de veces.
- En base a una condición.
- Sí se requiere realizar alguna ejecución de base.



Para (for)

Se recomienda cuando:
- Queremos repetir una ejecución un número concreto de veces.
- En base a un aumento o disminución concreta en los valores tratados.
- Se realizan todas las ejecuciones que se definan de base.

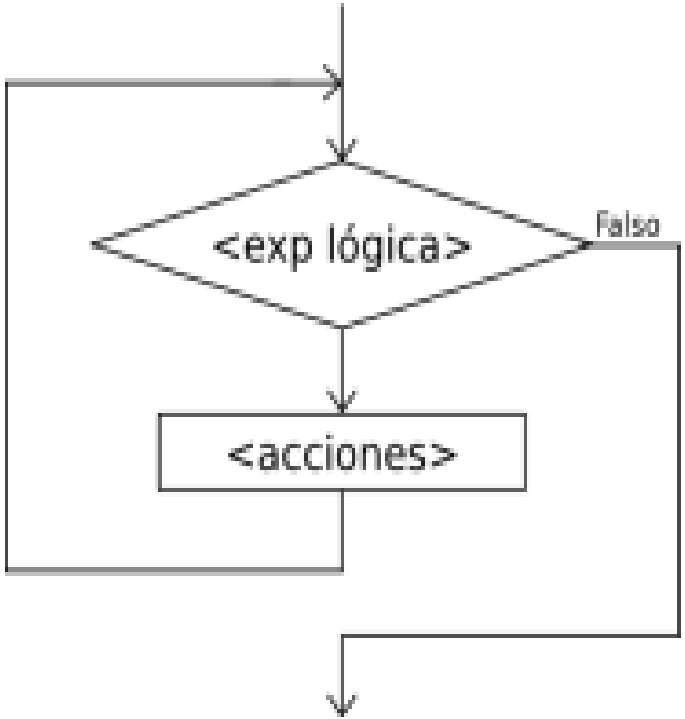


Bucles. Estructura mientras

Habitualmente, una condición se deberá comprobar más de una vez. Por ejemplo, una condición de error puede repetirse: el usuario que introduce mal una contraseña por primera vez puede equivocarse también en una segunda ocasión.

Por eso, igual que cualquier lenguaje de programación tiene una orden "si", la gran mayoría de ellos tendrá una orden "mientras", que permite que un fragmento de un programa se repita mientras una cierta condición se siga cumpliendo (por ejemplo, mientras la contraseña que teclee el usuario sea incorrecta, el usuario deberá volver a introducirla).

También existe un símbolo habitual en los diagramas de flujo para representar este tipo de condiciones repetitivas, en las que si se cumple la condición, se realiza una serie de acciones y se vuelve a comprobar la condición, y así sucesivamente hasta que la condición no se cumpla:


<exp lógica>
Falso
<acciones>

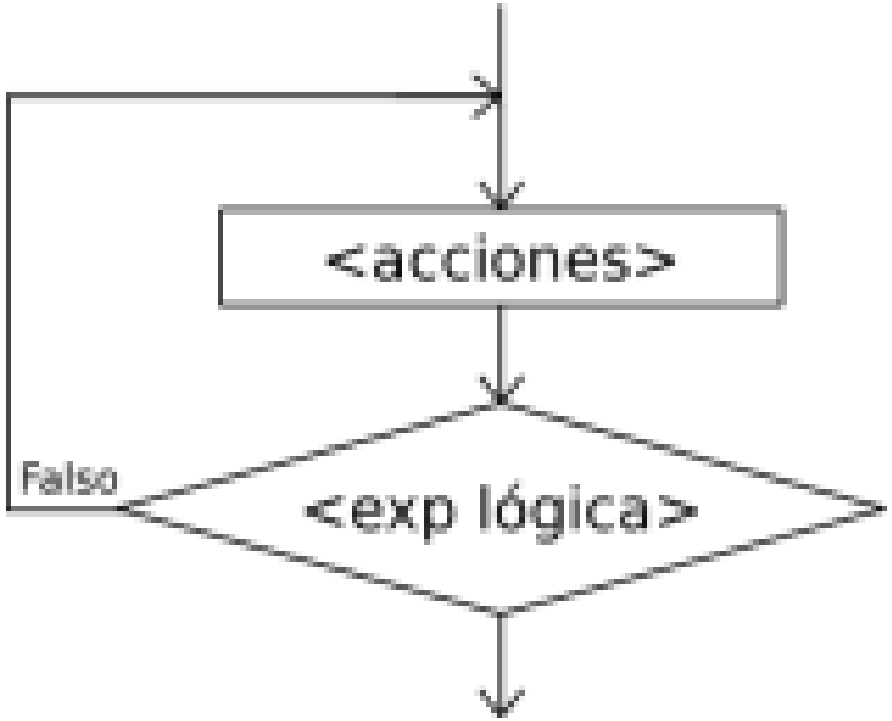
```
1   Algoritmo Mientras01
2       Escribir "Dime un numero";
3       Leer x;
4       suma = 0;
5       Mientras x <> 0 Hacer
6           suma = suma + x;
7           Escribir "Hasta ahora, la suma es ", suma;
8           Escribir "Dime otro numero";
9           Leer x;
10      FinMientras
11      Escribir "Terminado";
12  FinAlgoritmo
13
```

Bucles. Estructura Repetir Hasta Que

Es también muy frecuente que un bloque de programa que quizá se repita, deba ejecutarse al menos una vez.

Por ejemplo, si queremos pedir un dato al usuario, quizá exista algún error y haya que insistir, pero al menos deberemos pedírselo una primera vez.

En estos casos, la estructura "mientras" no es la más adecuada: no podemos comprobar la condición al principio, sino después de haber pedir el valor. En estos casos (que son muy frecuentes), sería más razonable usar otra estructura de programación en la que la condición se compruebe después de dar ciertos pasos. Esa estructura es "repetir... hasta":


<acciones>
Falso
<exp lógica>

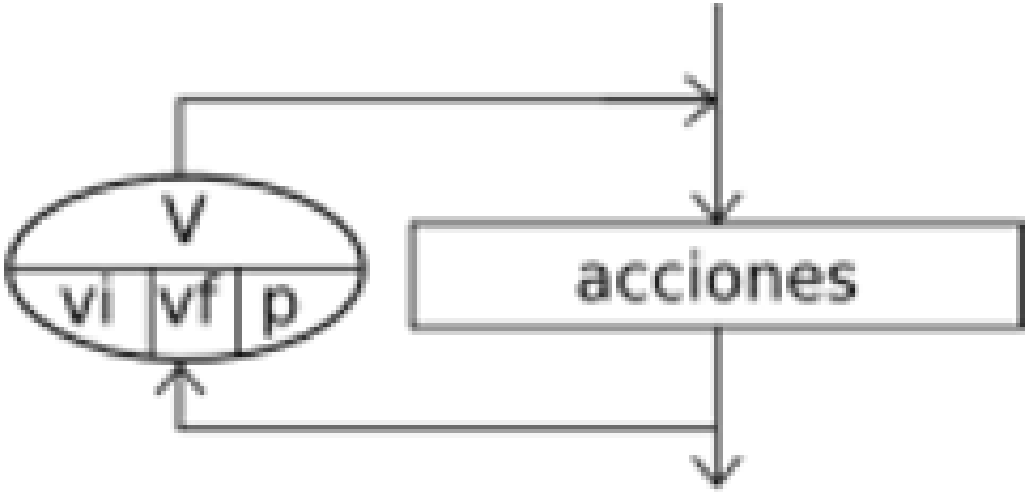
```
1   Algoritmo Repetir01
2       Repetir
3           Escribir "Dime tu clave de acceso";
4           Leer clave;
5           Si clave <> 1234 Entonces
6               Escribir "Clave incorrecta";
7           FinSi
8       Hasta Que clave=1234
9       Escribir "Bienvenido!";
.0  FinAlgoritmo
.1
.2
```

Bucles. Estructura Para

En muchas ocasiones, no queremos que una serie de instrucciones se repitan mientras se cumpla una condición, sino un cierto número de veces.

Por ejemplo, para escribir "Hola" 3 veces en pantalla existe una orden más cómoda que la orden "mientras" o la orden "repetir... hasta".

Es la orden "para", que hace que una variable tome una serie de valores que se van incrementando. Por ejemplo, una estructura como "para x con valores desde 2 hasta 4" haría que un bloque de programa se repitiera 3 veces. En la primera repetición, la variable "x" tendría el valor 2, en la segunda tendría el valor 3 y en la tercera tendría el valor 4.


V
vi
vf
p
acciones

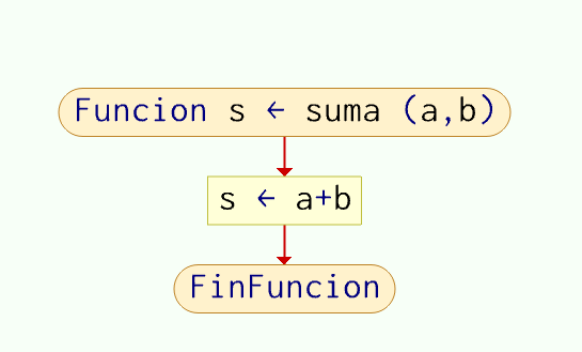
```
1  Algoritmo Para01
2      Para x = 1 Hasta 10 Hacer
3          Escribir x;
4      FinPara
5  FinAlgoritmo
6
```

Función

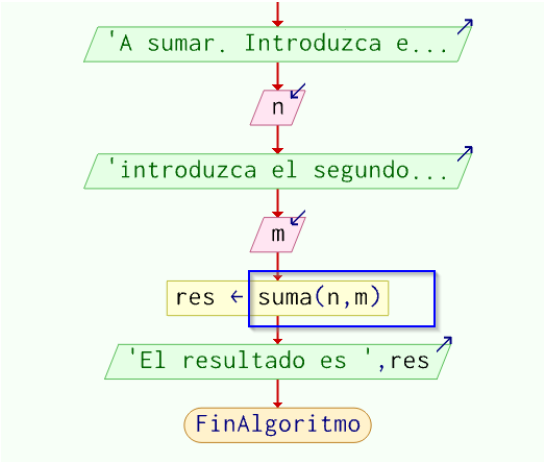
Una función es una sección de un programa que calcula un valor de manera independiente al resto del programa.

Se puede definir, y luego ser llamada en mitad del código tantas veces como se quiera. El programa durante la llamada irá a la definición de la función, la recorrerá, y dará el resultado adecuado.

Definición:



Llamada:



Bibliografía

- http://pseint.sourceforge.net/
- https://pseint.site/