

# Programación en:

# PSeInt



## ¿Qué es PSeInt?

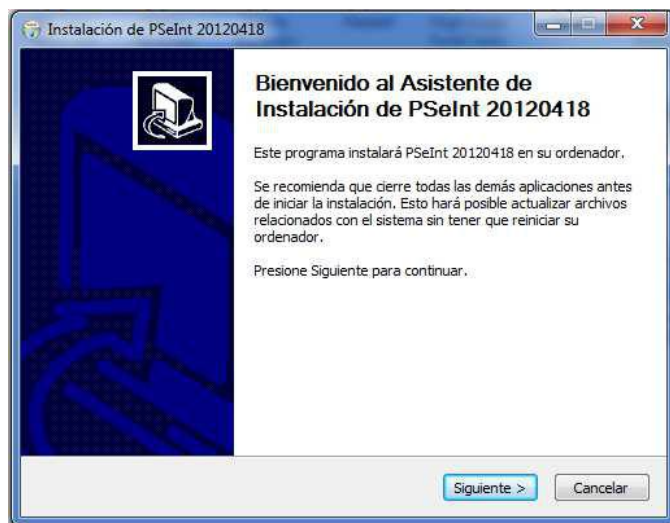
PSeInt, es la abreviatura de *Pseudocode Interpreter*, Intérprete de Pseudocódigo. Este programa fue creado como proyecto final para la materia *Programación 1* de la carrera *Ingeniería en Informática* de la Facultad de Ingeniería y Ciencias Hídricas de la Universidad Nacional del Litoral, del en aquel momento estudiante Pablo Novara.

El programa utiliza pseudocódigo, una descripción de un algoritmo computacional, cuya principal misión es que el programador pueda centrarse en los aspectos lógicos de la programación, dejando el apartado técnico para cuando se vea la sintaxis de un lenguaje de programación real.

## ¿Por qué usar PSeInt y no otro intérprete o compilador de pseudocódigo?

- 1) Porque es software libre y permite el acceso de cualquier usuario.
- 2) Está constantemente atendido por su creador y posee un foro para reportar errores y obtener ayuda, está también constantemente atendido por su creador, lo que ayuda a mejorar el programa.
- 3) Posee ayuda, que valga la redundancia ayuda a aprender a usarlo, y manejar r el lenguaje.
- 4) Está disponible su código fuente, y con instrucciones para ejecutarlo, de modo que si sabemos C++ podremos personalizarlo y corregirlo.
- 5) Posee previsualización y exportación a C, C++ y otros lenguajes para que podamos ver el mismo código implementado en C y otros lenguajes, lo que ayuda a aprender estos y otros lenguajes.
- 6) Se trata de un compilador que compila automática mente cuando el usuario pulsa ejecutar, el algoritmo se guarda automáticamente en un archivo del disco duro, dentro de la carpeta del ejecutable PSeInt, para su posterior ejecución, haciendo más cómodo su uso.

## Instalación



Abrir el archivo " pseint-win-32-xxxxxxx.exe " (xxxx es número de la versión actual), al hacer doble clic se ejecuta el instalador.

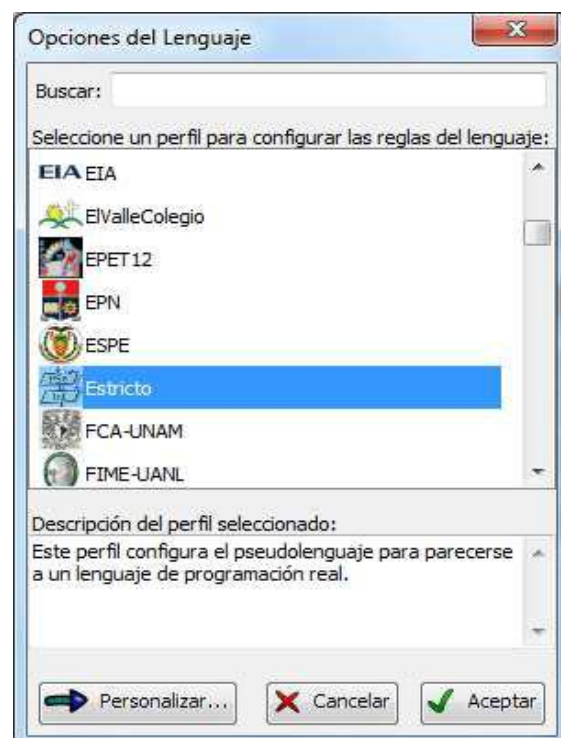
Luego presionamos siguiente -> siguiente y así sucesivamente hasta instalarlo

Cuando abrimos por primera vez PSeInt aparece un cartel preguntándonos que perfil deseamos utilizar, para evitar confusiones con el lenguaje.



Este manual se maneja con dos perfiles. El estricto, que es el más parecido a un lenguaje de programación real, se debe respetar al pie de la letra el formato del pseudocódigo propuesto por Novara. La sintaxis flexible la usamos para ejecutar ciertos códigos que requieren más flexibilidad a la hora de ejecutarse. A menos que se indique que se usa sintaxis flexible, se utilizará la sintaxis estricta.

***Nota:*** No confundir *Sintaxis flexible* con *Perfil flexible*



Vamos a Configurar → Opciones de Lenguaje → Elegimos Estricto y pulsamos aceptar.

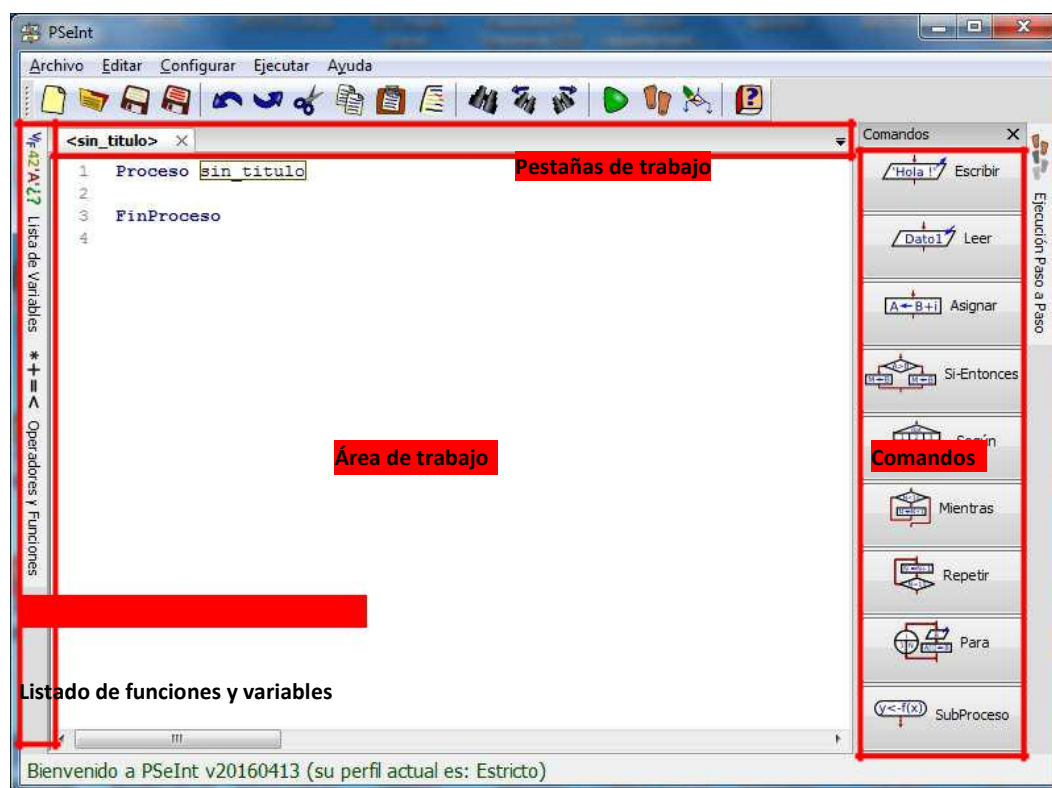
## Abrir PSeInt

Para abrir PSeInt damos doble clic en el acceso directo PSeInt del escritorio y nos abre el programa.

## Entorno de PseInt

Ahora que abrimos y configuramos por primera vez PSeInt, pasamos a detallar el entorno de programación de PSeInt.

Esta esta captura se detallan los nombres de las partes que componen el entorno o interfaz del programa



Como se explica en los textos de esta captura, podemos dividir al entorno en cuatro secciones: la de los botones de comando, arriba la de las pestañas de trabajo y el rectángulo blanco que contiene a las palabras Proceso y FinProceso y el listado de funciones y variables.

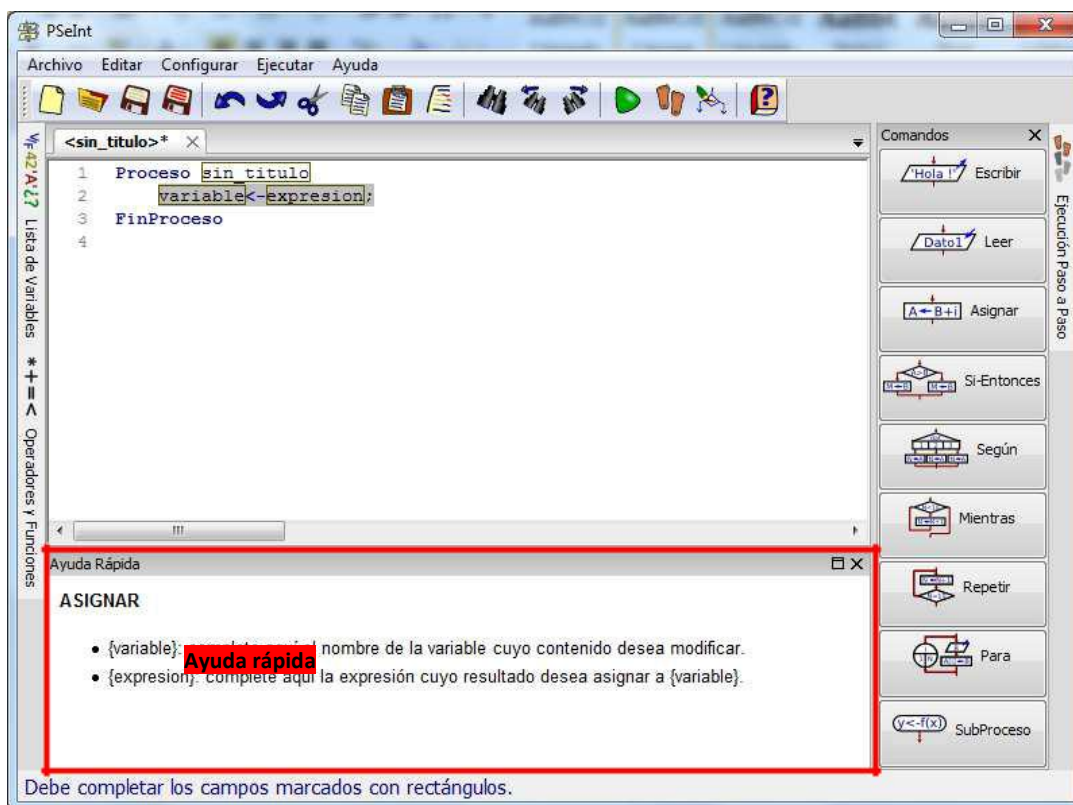
Pasamos a detallar cada una de sus partes de la siguiente forma:

## Botones de comando

PSeInt, al ser una herramienta didáctica y orientada a personas con poca o sin ninguna experiencia en programación, presupone que dicho usuario no conoce todavía la sintaxis válida en PSeInt. A tales efectos, como se ve en la captura, en este caso resaltados con rojo a la derecha del entorno, este dispone a su lado botones etiquetados con las estructuras usadas en este programa, que de ser presionadas escriben en el editor de texto la sintaxis válida de PSeInt, sirviendo de ayuda al programador o usuario:

Cuando el usuario pulsa cualquier botón de comando se escribe la sintaxis válida del pseudocódigo en PSeInt entre las líneas Proceso y FinProceso, excepto la estructura subproceso que se escribe fuera del proceso principal ya que proceso no puede contener ni uno ni más de un SubProceso.

Como parte también de la ayuda, cualquier estructura que se escriba en el área de trabajo abajo muestra su correspondiente descripción que especifica cómo se maneja la estructura citada, lo que en la captura de abajo se especifica como Ayuda rápida.



Pasamos a detallar los comandos:

## Acciones secuenciales:

### Botón Escribir

### Dibujo:



### Función del botón:

Escribir: Nos permite mostrar en pantalla algún tipo de dato, o varios separados por una coma (,) y esos datos deben estar entre comillas (")

**Nota:** La variable nunca va entre paréntesis

### Ejemplo de uso:

Ejemplos: `Escribir "hola mundo";` `Escribir "hola mundo, hola, 2, c";`

### Botón Leer

#### Dibujo:



### Función del botón:

Leer: nos permite recibir valores por teclado y guardarlos en variables.

### Ejemplo de uso:

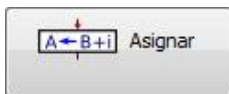
`Leer a;` //recibe el valor y lo almacena en a.

`Leer a, b, c;` //recibe 3 valores y los guarda en la variable que a, b y c respectivamente

**Nota:** La variable nunca va entre paréntesis

### Botón Asignar:

#### Dibujo:



### Función del botón:

Asignación: nos permite guardar un valor en una variable.

### Ejemplo de uso:

`c <- 2;` por lo tanto `c=2` (c tiene el valor dos), que es lo mismo decir c tiene el valor 2.

## Acciones selectivas o interrogativas:

### Botón Si

#### Dibujo:



#### Función del botón:

Nos permite evaluar la propiedad de una variable, y en función de esta, realizar una acción determinada

#### Ejemplo de uso:

```
Si cant_numeros != 0 entonces
    //sentencias
FinSi
```

### Botón Segun

#### Dibujo:



#### Función del botón:

Nos permite evaluar la propiedad de una variable, y después de comparar una a una las salidas con la evaluación, realizar la acción determinada

#### Ejemplo de uso:

```
Segun num_dia_sem Hacer
    1: Escribir "Lunes";
    2: Escribir "Martes";
    3: Escribir "Miércoles";
    4: Escribir "Jueves";
    5: Escribir "Viernes";
    6: Escribir "Sábado";
    7: Escribir "Domingo"; De
Otro Modo:
    Escribir "No es un día de la semana";
FinSegun
```

## Acciones repetitivas o interactivas:

### Botón Mientras

**Dibujo:****Función del botón:**

Permite realizar cierta acción determinada por la condición del Mientras

**Ejemplo de uso:**

```
Mientras num != 0 Hacer  
  
    Leer num;  
    cant_num<-cant_num+1;  
  
FinMientras
```

**Botón Repetir****Dibujo:****Función del botón:**

Repite una series de acciones hasta que se de cierta condición. Dicha acción la determina el operador =, que hace que salga del bucle

**Ejemplo de uso:**

```
Repetir  
  
    Leer num;  
  
    cant_num<-cant_num+1;  
  
Hasta Que num = 0;
```

**Botón Para****Dibujo:****Función del botón:**

Presenta un cierto rango de valores, y para ellos realiza una determinada acción

**Ejemplo de uso:**

```
Para i <- 1 Hasta 10 Con Paso 1 Hacer  
  
    Escribir i;  
  
FinPara
```



## Funciones y SubProcesos

Dibujo:



Función del botón:

SubProcesos: Permite añadir Funciones/SubProcesos al programa

Ejemplo de uso:

**SubProcesos que no devuelven valor, solo realizan una tarea específica**

```
SubProceso ImprimirResultado(x)

    Escribir "El resultado es: ",x;

FinSubProceso
```

**SuProcesos que devuelven valores**

```
SubProceso x <- LeerDato(cosa)

    Definir x Como Entero;

    Escribir "Ingrese ", cosa, ": ";

    Leer x;

FinSubProceso
```

**Nota:** En el caso que nos moleste o que ya no necesitemos este panel podemos cerrarlo con el botón cerrar ubicado a la derecha superior del mismo

**Nota 2:** Todas las estructuras tanto selectivas como repetitivas, así como también el uso de subprocessos se explican al detalle en cada sub apartado

### Área de trabajo

El lugar donde escribimos el código del pseudocódigo.

Los números a la izquierda indican el número de línea de código del programa.

### Pestañas de trabajo

Sobre la parte superior del área de trabajo vemos una pestaña que por defecto dice <sin\_titulo>

La pestaña activa se corresponde al área de trabajo actual.

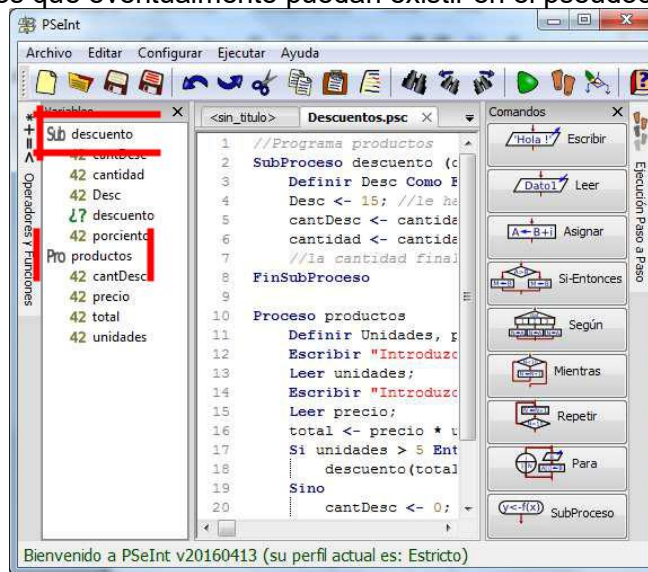
En caso de que guardemos el archivo en pseudocódigo, la pestaña tomará el nombre del nombre del archivo en pseudocódigo que hayamos guardado. PSeInt puede abrir varios archivos en pseudocódigo a la vez, mostrándose en la pestañas de trabajo.

Se pueden cerrar el proyecto con el botón cerrar de la pestaña.



## Listado de funciones y variables

A la izquierda vemos dos pestañas. La de más arriba, como su nombre lo indica, muestra la lista de variables. Los signos que aparecen antes del texto (V/F de verdadero/falso, números, letras y signos de interrogación) ofician de íconos del texto. Si hacemos clic izquierdo sobre el texto, se abre una solapa que detalla el Proceso y los SubProcesos que eventualmente puedan existir en el pseudocódigo.



En el panel, como se observa en la imagen, el Proceso se marca con el ícono Pro seguido por el nombre del Proceso. Por su parte, el SubProceso se marca con el ícono Sub seguido del nombre del SubProceso. Nótese que subordinado al ícono Sub y el nombre del SubProceso aparece el número 42 seguido de las variables. Estas variables son los argumentos de la función o la variable de retorno.

Este 42 significa las potenciales tipos de datos que determina el intérprete en caso de que el tipo de variable pueda deducirse antes de ejecutar el algoritmo

En caso de tratarse del proceso principal, estos textos que se muestran son las variables que se usan en ese proceso principal.

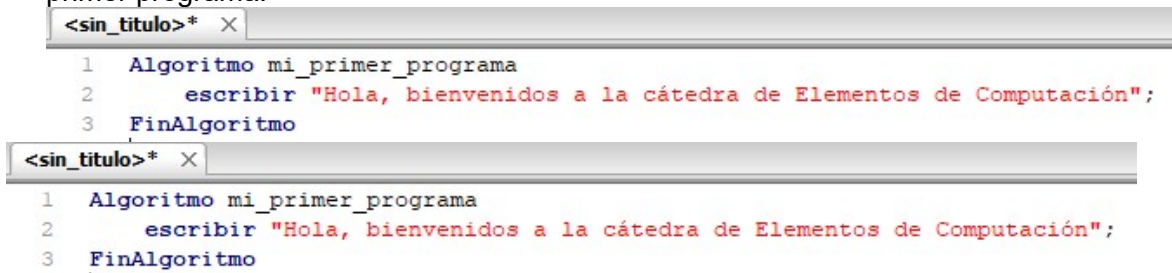
**Nota:** Como bien menciona Pablo Novara en este programa no puede hacer instrucciones fuera ni de los Procesos ni de los Subprocesos, por lo tanto no es posible declarar variables globales.

**Nota2:** Si ponemos el cursor sobre el 42 o la variable aparece un cartel con la ayuda de correspondiente

Si hacemos clic sobre el ícono Sub, PSeInt nos marca con azul en que parte del pseudocódigo se encuentra el SubProceso. Lo mismo sucede hacemos clic sobre las variables. Eso es útil cuando tenemos múltiplos SubProcesos, variables o parámetros y se nos hace difícil saber dónde está cada una de ellas/ellos.

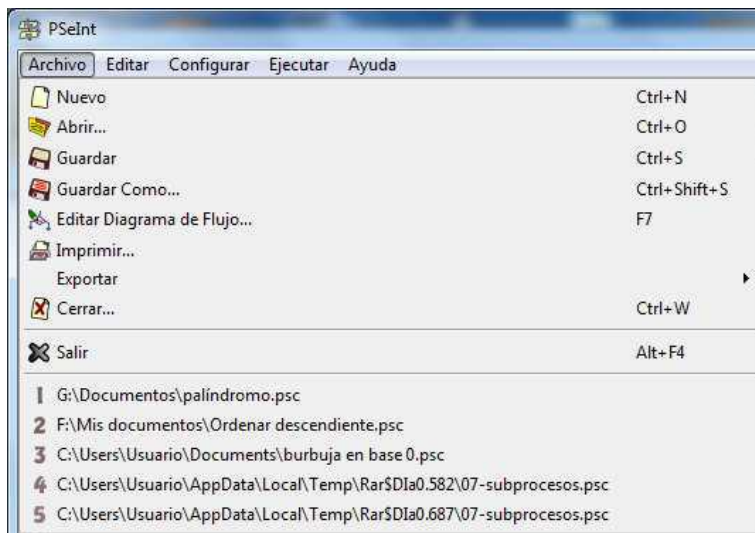
## Escribir mi primer programa

Ya analizado el entorno y conociendo sus partes, pasamos a escribir nuestro primer programa. Ya abierto PSeInt y habiendo configurado sintaxis estricta, este nos presenta el área de trabajo con dos palabras claves Proceso seguido del nombre sin\_titulo y FinProceso, entre estas dos líneas escribiremos nuestro primer programa:



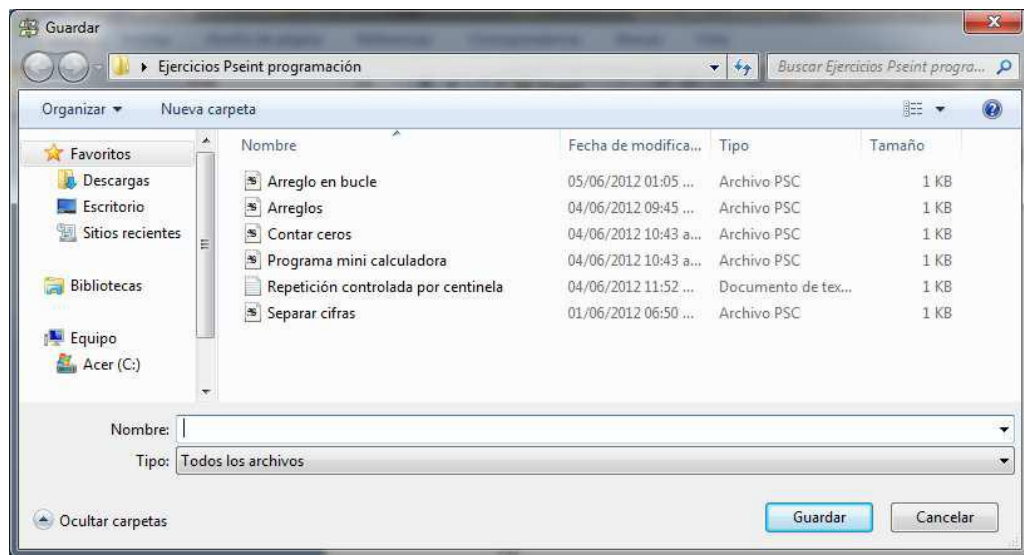
```
<sin_titulo>* X
1 Algoritmo mi_primer_programa
2 escribir "Hola, bienvenidos a la cátedra de Elementos de Computación";
3 FinAlgoritmo
```

Luego lo guardamos



Escribimos el nombre del programa en la ventana que nos aparece y luego

presionamos Guardar Como...



Ahora que los hemos guardado necesitamos que nuestro programa funcione y escriba en la pantalla Unitec, pero aunque PSeint subraya con rojo los errores de sintaxis, es bueno verificar sintaxis para ver los errores. Para ello vamos a ejecutar, → verificar sintaxis. De todos modos, si tuviéramos errores él nos subrayaría la frase donde se encuentre el error, luego lo corregimos y lo volvemos a ejecutar, hasta que no aparezca nada subrayado con rojo.

Luego que el programa no tiene errores de compilación, no aparecen líneas subrayadas con rojo, vamos al menú ejecutar, luego seleccionamos opción ejecutar, y en la pantalla aparecerá la palabra Uni tec que es la salida del programa, también para ejecutar el programa puede usar el ícono de ejecutar o pulsar F9:



Si la ejecución se realizó con éxito correcta al final aparecerá un mensaje diciendo que el programa se ejecutó correctamente.

Siempre que queremos escribir un programa en PSeint iniciamos debajo de la palabra

### Proceso sin\_titulo

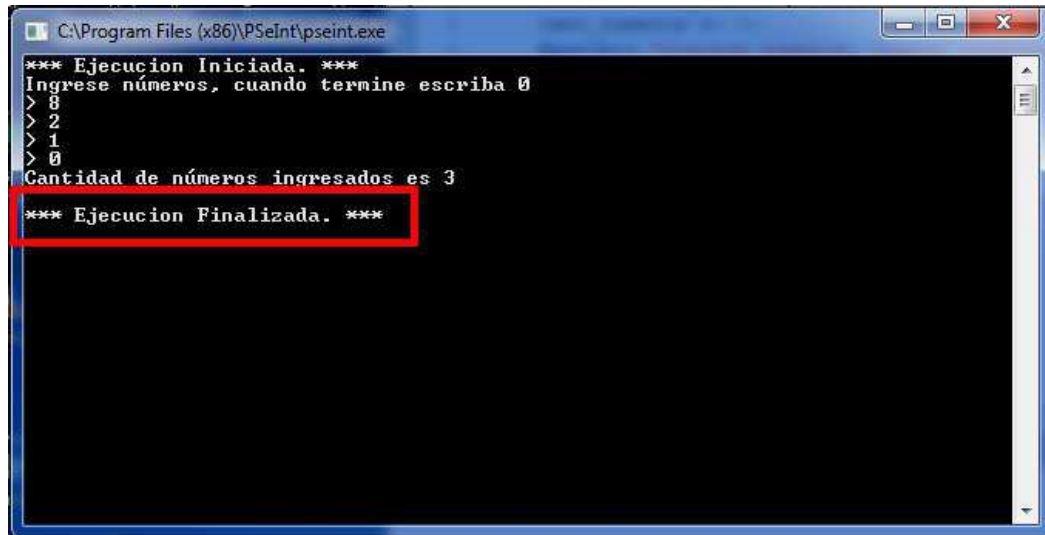
```
//escribimos el cuerpo del programa;
```

### FinProceso

Y el proceso principal se cierra con las palabras claves FinProceso que indica el final del programa principal.

Combine asignarle un nombre al programa, sustituyendo sin\_titulo por el nombre que queramos darle. Recordar que nombre del pseudocódigo en ninguna sintaxis puede tener espacios y en sintaxis estricta tampoco caracteres acentuados. No confundir el nombre del proceso con el del archivo en pseudocódigo.

La palabra reservada **Escribir** *escribe en la pantalla lo que está encerrado entre comillas*. En sintaxis flexible también podemos utilizar la palabra **Imprimir** o



```
C:\Program Files (x86)\PSeInt\pseint.exe
*** Ejecucion Iniciada. ***
Ingrese números, cuando termine escriba 0
> 8
> 2
> 1
> 0
Cantidad de números ingresados es 3
*** Ejecucion Finalizada. ***
```

**Mostrar.** *Reitero, a menos que se indique lo contrario, se utilizará siempre sintaxis Estricta.*

## Concatenar texto

```
Proceso primer_programa  
  
    Escribir "Mi primer programa";  
  
    Escribir " en PSeInt ";  
  
FinProceso
```

La salida del programa es

***Mi primer programa***

***En PSeInt***

Esto porque aunque se escriba otro texto, y se coloque la palabra clave Escribir, el texto sigue escribiéndose al final del otro, ahora si quisiéramos escribir:

***Mi primer programa En PSeInt***

El programa es de esta forma ejemplo

```
Proceso primer_programa  
  
    Escribir "Mi primer programa " Sin Saltar;  
  
    Escribir "En PSeInt ";  
  
FinProceso
```

Con esto deducimos que la instrucción Sin Saltar concatena, en este caso, el contenido de una cadena de texto, y el contenido del próximo Escribir se escribe a continuación del anterior y no en la línea de abajo.

Recordar que en sintaxis estricta la colocación del punto y coma al final de las sentencias es obligatoria, en flexible es opcional.

***Nota:*** Las comillas deben ser siempre simples y nunca tipográficas pues estas últimas son símbolos gráficos que ningún lenguaje de programación hasta el momento puede interpretar. Siempre por defecto en los editores de texto de los IDEs se escriben comillas simples, pero cuando se importa o se formatea pseudocódigo traído de afuera, hay que corregir el encomillado, de no hacerlo provocaría un error de compilación.

***Nota 2:*** PSeInt no es case sensitive, por lo tanto colocar Escribir con mayúsculas

y minúsculas es lo mismo y no genera errores de ningún tipo, pero por respeto a la sintaxis mostrada por los botones se debe escribir con mayúscula inicial, evitando así errores de formato.

***Nota 3:*** En sintaxis estricta, las sentencias siempre finalizan en punto y coma.

## Diagramas de flujo

PSeInt es capaz de interpretar los pseudocódigos y transformarlos a diagrama de flujo, para eso dispone de un visualizador y editor de diagramas de flujo. Esto es útil si queremos analizar el pseudocódigo desde un punto de vista gráfico.

Se accede pulsando el ícono  de la barra de tareas. PSeInt no sólo es capaz de visualizarlo, sino también editarlo.



Ejemplo:

Considera el siguiente programa

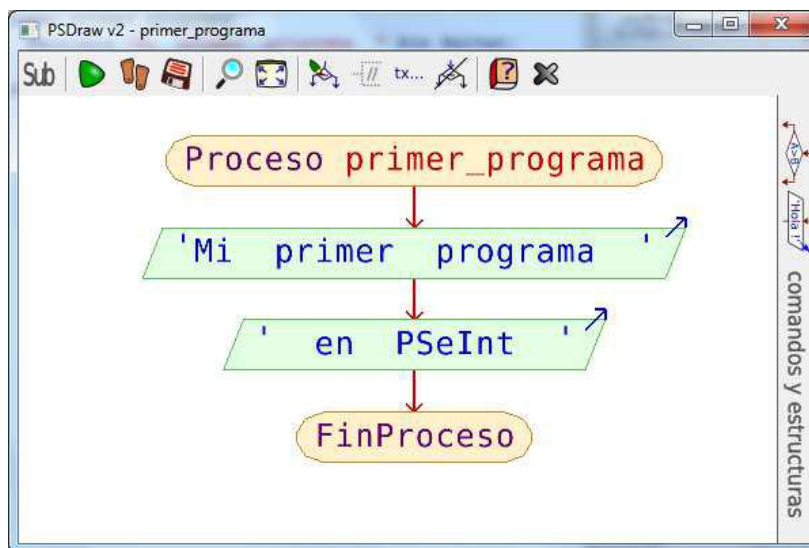
```
Proceso primer_programa
```

```
    Escribir "Mi primer programa " Sin Saltar;
```

```
    Escribir " en PSeInt ";
```

```
FinProceso
```

Su representación en diagrama de flujo es la siguiente:



Aquí vemos el inicio del proceso representado como una elipse, pues es donde comienza el programa, la sentencia escribir representada en un rectángulo, pues es una salida (obsérvese la flecha de salida) y abajo nuevamente una elipse que representa el fin del proceso.

## Declarar variables

En sintaxis estricta, siempre que necesitemos hacer un programa, tendremos que declarar variables para poder guardar la información que introduzcamos al programa.

Los tipos de datos básico soportados son los siguientes:

- Entero: solo números enteros.
- Real : números con cifras decimales.
- Caracter: cuando queremos guardar un carácter.
- Logico: cuando necesitamos guardar una expresión lógica (verdadero o falso)
- Cadena: cuando queremos guardar cadenas de caracteres.

**Nota 1:** *Cadena y Caracter son términos equivalentes, no genera error que las escribamos indistintamente.*

**Nota 2:** *El plural de Caracter es Caracteres o Cadena*

Ejemplos de declaración de variables:

*Si queremos declarar una variable de tipo entero sería así :*

Definir numero Como Entero;

Número se convierte en una variable de tipo entero

**Nota 2:** *En sintaxis estricta, ni los nombres de variables ni palabras claves pueden tener caracteres acentuados*

*Si queremos declarar una variable tipo Cadena para guardar el nombre sería así  
Dimension nombre [25];*

*Definir nombre Como Cadena;*

*Nombre sería una variable que guardaría solo 25 caracteres aunque podemos escribir más de 25 letras, él en la memoria solo guardara los primeros 25 caracteres.*

**Nota 3:** *Ver el apartado Dimensiones para más detalles.*



**Nota 4:** Aunque esto no genere errores en tiempo de ejecución, si se declaran varias variables a la vez, para evitar un error de formato – concordancia, de debe pluralizar el tipo de variable. Ej.: Definir a, b, c Como Enteros;

## Operadores

PSelnt proporciona los siguientes operadores y funciones

### Operador Función

Operador	Significado	Ejemplo
<b>Algebraicos</b>		
+	Suma	A+B
-	Resta	C-D
()	Agrupar expresiones	A+(C*D)
^	Potenciación	2^3
*	multiplicación	6*3
/	división	12/4
% o Mod	Módulo (resto de la división entera)	
<b>Lógicos</b>		
& ó y	conjunción (Y)	
ó O	Disyunción (O)	
~ ó NO	Negación (no)	
<b>Relacionales</b>		
>	Mayor que	
<	Menor que	
=	Igual que	
<=	Menor o igual que	
>=	Mayor o igual que	

**Nota:** Tanto en sintaxis flexible como estricta, podemos utilizar también los operadores & | y mod como y o y % respectivamente.

## Leer valores y almacenarlos en las variables

Cuando nosotros queremos leer un valor y almacenarlo en una variables usaremos la palabra **Leer < variable>**; y cuando queremos asignar un valor o una operación matemática, en sintaxis estricta, usaremos <- que es el símbolo de < mas - .

### Ejemplo sobre lectura de datos

```
Proceso lectura_datos
```

```
    Dimension nombre[25];
```

```
    Definir nombre Como Cadena;
```

```

Escribir "Ingrese su nombre ";

Leer nombre[24];

Escribir "Bienvenido ";

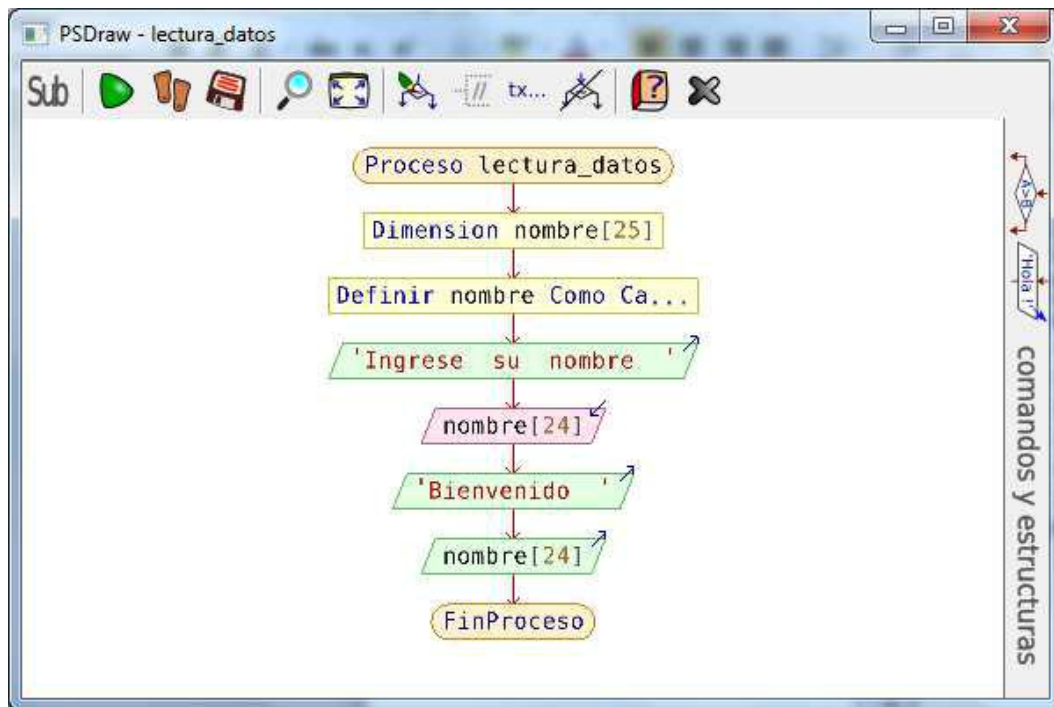
Escribir nombre[24];

FinProceso

```

El programa declara una variable para el nombre , que guarda 25 caracteres máximo, ingresa el nombre y luego escribe en la pantalla Bienvenido el nombre que se ingresó. Algo importante es que cuando se quiere presentar el valor de la variable esta no se escribe entre comillas.

Su diagrama de flujo:



En la tabla se nos muestra como se pudo sustituir un bloque del programa que nos daría el mismo resultado

Caso 1	Caso 2
<pre> Escribir "Bienvenido "; Escribir nombre; </pre>	<pre> Escribir "bienvenido " Sin Saltar , nombre; </pre>

**Nota:** No es necesario indicar de cuantos caracteres es la cadena que PSeInt debe leer, pero si se debe indicar si declaramos a la dimensión como un vector de caracteres.

## Asignaciones y Operaciones matemáticas en un programa.

En sintaxis estricta, el símbolo <- lo usaremos para asignar valores a las variables ejemplo **Sueldo<-500**; Con esta instrucción estamos asignando el valor de 500 a la variables sueldo que pudo declararse como entero o real

**Nombre<-"juan"**; con esta instrucción asignamos la cadena "Juan " a la variable nombre que es una variable de tipo cadena

***Nota:** En sintaxis estricta, también se puede utilizar := para asignar variables*

### **Ejemplo sobre asignaciones de valores a las variables**

```
Proceso programa_aumento

    Definir sueldo, aumento Como Enteros;

    Dimension nombre[25];

    Definir nombre Como Cadena;

    Escribir "Ingresar el nombre del empleado"; Leer
    nombre[24];

    Escribir "Ingresar el sueldo del empleado";

    Leer sueldo;

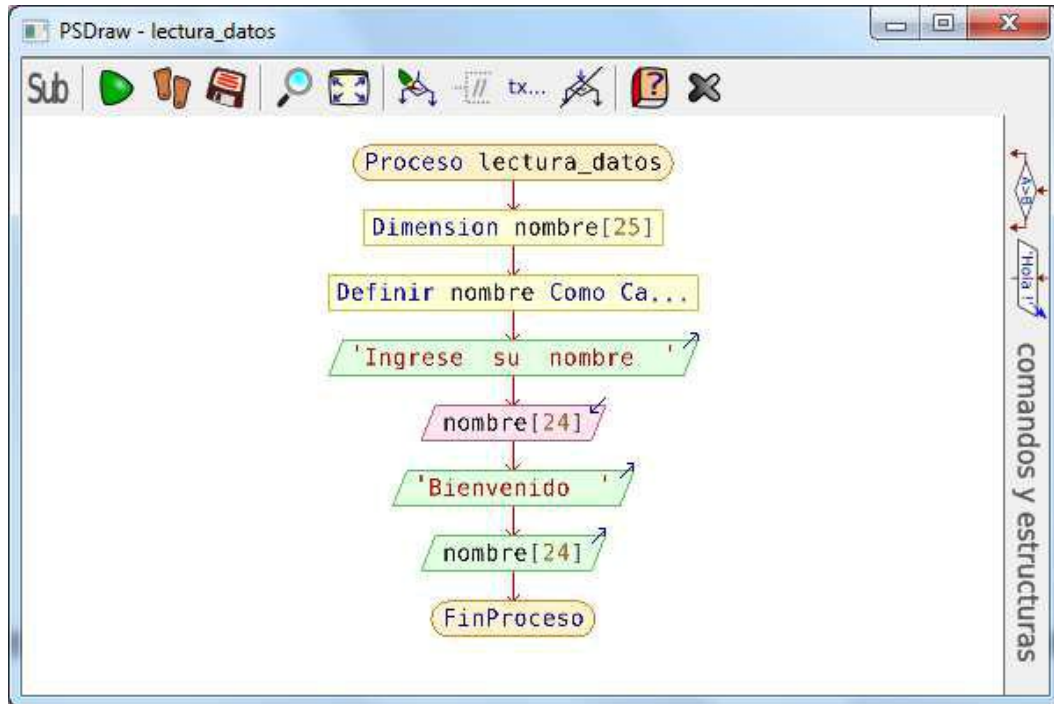
    Aumento<-trunc(sueldo*1.25);

    Escribir "Nuevo sueldo con el 25% de aumento";
    Escribir aumento;

FinProceso
```

El programa pide el nombre y el sueldo del empleado luego calcula el 25% de aumento de sueldo y lo guarda en la variable aumento y luego presenta el nuevo sueldo.

Diagrama de flujo:



**Nota:** Nótese que para calcular Aumento se lo tuvo que truncar utilizando la palabra clave trunc que es truncar la expresión. Es o es porque Aumento fue declarado como Entero, por lo tanto debe recibir un entero, pero la expresión (sueldo\*1.25) no da entero

### Ejemplo sobre suma de cadenas

**Nota:** En sintaxis flexible también se puede concatenar textos y arreglos con el operador +

Proceso suma\_de\_cadenas

```
Dimension nombre[25],apellido[25],completo[25];

Definir nombre,apellido,completo Como Cadenas;

Escribir "Su Nombre";

Leer nombre[24];

Escribir "Apellido";

Leer apellido[24];

Completo[24] <- concatenar(concatenar(nombre[24], "

"),apellido[24]);

Escribir "Nombre completo ", completo[24];

FinProceso
```

La variable completo toma el valor del nombre más un espacio en blanco mas el

apellido y lo guardamos en una variable donde ahora tenemos el nombre y el apellido.

**Nota:** *No es estrictamente necesario dimensionar cadenas de caracteres.*

## Instrucciones condicionales

Anteriormente hemos estado haciendo programas que solo hacen cálculos, pero la programación es más interesante cuando nuestros programas toman sus propias decisiones, en PSeInt existen instrucciones condicionales que se describen a continuación:

Instrucción Si:

sintaxis

Si condición Entonces

    instrucciones;

FinSi

ó

Si condición Entonces

    instrucciones;

Sino

    instrucciones;

FinSi

### **Ejemplo sobre decisiones**

Ingresar un numero y si el número es mayor a 100 , escribir en la pantalla el numero es mayor a 100.

Proceso decision

```
Definir num como Entero;  
  
Escribir "Ingresar un número";  
  
Leer num;  
  
Si num > 100 Entonces
```

En programa solo escribirá que el número fue mayor a 100 cuando cumpla con la condición **num > 100** sino cumple con la condición no hace nada .

### **Ejemplo sobre decisiones**

Ingresar el nombre del empleado, las horas trabajadas, luego Calcular pago bruto (50 lps la hora) IHSS y total a pagar, presentar los resultado del programa

**Nota:** el seguro social es 84 si el sueldo es mayor 2400 sino es el 3.5% del sueldo del empleado.

Proceso empleados

```
Definir horas como Enteros;  
  
Definir Pbruto,ihss,tp como Reales;  
  
Dimension nombre[25];  
  
Definir nombre Como Cadena;  
  
Escribir "Ingresar el nombre";  
  
Leer nombre[24];  
  
Escribir "Ingresar las horas trabajadas";  
  
Leer horas;  
  
Pbruto<-horas*50;  
  
Si pbruto > 2400 Entonces  
    Ihss<-84;  
  
Sino  
    Ihss<-0.035*pbruto;  
  
FinSi  
  
Tp<-pbruto-ihss;  
  
Escribir "Pago bruto " , pbruto;  
  
Escribir "Seguro Social " , ihss;  
  
Escribir "Total a pagar " , tp;
```

FinProceso

En este programa se usó en el cálculo del ihss una decisión que tiene dos salidas una cuando se cumple la condición que es el entonces y la otra cuando no se cumple la condición que es el sino, ahora esto nos ayuda a que nuestros programas puedan tomar una decisión cuando la condición se cumple y otra cuando no se cumple.

Ahora en el siguiente ejercicio que se presenta , ya no hay dos soluciones a la condición sino tres, cuando sucede esto se usan condiciones anidadas.

Sintaxis de una condición anidada :

```
Si condición 1 Entonces
    Instrucciones;
Sino Si condición 2 Entonces
    Instrucciones;
    Sino Si condición 2 Entonces
        Instrucciones;
        Sino
            Instrucciones;
    FinSi
FinSi
FinSi
```

### ***Ejemplo sobre decisiones anidadas***

Ingresar el nombre del empleado, la zona de trabajo, las ventas del empleado, luego calcular su comisión en base a un porcentaje basado en la zona de trabajo, luego determinar el IHSS y el total a pagar, presentar los datos.

Tabla para el caculo de la comisión

Zona	Porcentaje de Comisión
A	6%
B	8%
C	9%

Proceso Comision

```
Definir zona como Caracter;
Dimension nombre[25];
Definir nombre Como Cadena;
```

```

Definir  ventas, comis, ihss, tp Como Reales;

Escribir "Ingresar  el  nombre del empleado      ";

Leer  nombre[24];

Escribir "Ingresar las ventas del empleado "; Leer
ventas;

Escribir "Ingresar la zona de trabajo "; Leer
zona;

Si  zona  ='A'  Entonces

    comis<- 0.06  *  ventas;

    Sino  Si  zona='B'  Entonces

        comis<- 0.08  *  ventas;

        Sino  Si  zona='C'  Entonces

            comis<- 0.09  *  ventas;

            Sino

                comis<- 0;

        FinSi

    FinSi

FinSi

Si  comis  >  2400  Entonces

    ihss<-84;

    Sino

        ihss<-0.035*comis;

        tp<-comis-ihss;

    FinSi

Escribir "Comisión ganada ", comis;

Escribir "Seguro Social ", ihss;

Escribir "Total a pagar ", tp;

FinProceso

```

En este programa usamos decisiones anidadas para el cálculo de la comisión del empleado, esto porque se tenían varias opciones de la cuales elegir.



El ultimo sino donde la comisión es 0 se hace porque no estamos seguros de que la persona que opera el programa introduzca correctamente la zona, si se ingresó otra zona de las permitidas la comisión es cero.

## Estructura Según

Esta se usa como sustituto en algunos casos del si anidado, por ser más práctico al aplicarlo en la evaluación de algunas condiciones.

Sintaxis

Segun variable Hacer

valor1, valor2, valor3, ... :

instrucciones;

valor1, valor2, valor3, ... :

instrucciones;

▪

▪

[ De Otro Modo :

instrucciones;]

FinSegun

Los valores a evaluar, se separan por comas si hay varios, tal como aparece en la sintaxis valor1, valor2 etc., también se puede usar el De Otro Modo que nos indica, que en caso de no seleccionar ninguna de las instrucciones anteriores se ejecutan.

**Nota importante:** *En sintaxis estricta las opciones del Segun deben ser siempre del tipo numérico. Para poder evaluar opciones del tipo texto se debe personalizar el lenguaje utilizando sintaxis flexible, o yendo a Opciones del lenguaje (Perfiles) y destildar Limitar la estructura Según a variables de control numéricas en el editor o en su defecto utilizar el perfil taller de informática, o perfil flexible.*

### Ejemplo sobre la aplicación de la estructura Segun

En el ejercicio anterior usamos decisiones anidadas para determinar la comisión, ahora usaremos una estructura Según.

Para eso habilitamos sintaxis flexible yendo a personalizar lenguaje → Personalizar... → Utilizar sintaxis flexible

```

Proceso ejemplo_caso
    Definir zona Como Caracter;
    Dimension nombre[25];
    Definir nombre Como Cadena;
    Definir ventas,comis,ihss,tp Como Reales;
    Escribir "Ingresar el nombre del empleado ";
    Leer nombre[24];
    Escribir "Ingresar las ventas del empleado ";
    Leer ventas;
    Escribir "Ingresar la zona de trabajo ";
    Leer zona;
    Segun Zona Hacer
        'a','A' : comis<- 0.06 * ventas;
        'b','B' : comis<- 0.08 * ventas;
        'c','C' : comis<- 0.09 * ventas;
        De Otro Modo :
            comis<- 0;

    FinSegun

    Si comis > 2400 Entonces
        ihss<- 84;
    Sino
        ihss<-0.035*comis;

    FinSi

    tp<-comis - ihss;

    Escribir "Comisión ganada ", comis;

    Escribir "Seguro Social ", ihss;

    Escribir "Total a pagar ", tp;

FinProceso

```

Ahora nuestro programa reconoce las mayúsculas y minúsculas en la evaluación de la zona

## Uso del operador | ú o

El operador | (O) se utiliza cuando estamos evaluando dos o más condiciones y queremos que la condición se cumpla cuando una de las condiciones que estamos evaluando se hacen verdadera. Ejemplo

Cuando se introduce la zona en el ejercicio con la estructura Si solo evaluábamos una opción que la zona sea igual a la letra A y si el usuario escribía una a minúscula no se tomaba en cuenta pero esto se puede corregir de esta manera :

```

Si zona ='A'      |      zona ='a'      Entonces

    comis<- 0.06  *  ventas;

Sino Si zona='B' | zona='b' Entonces

    comis<- 0.08 * ventas;

Sino si zona='C' | zona='c' Entonces

    comis<- 0.09 * ventas;

Sino

    comis<- 0;

FinSi

FinSi

FinSi

```

Ahora la condición dice, **si zona es igual a la letra A o es igual a la letra a**, cualquiera que sea la zona a o A en ambos casos la condición es verdadera , ahora el usuario puede usar mayúsculas y minúsculas y el resultado será el mismo.

### Ejemplo sobre el operador |

Ingresar el nombre del cliente, luego la cantidad del producto, precio y tipo de cliente, calcular el subtotal, descuento, impuesto s/v, total a pagar, presentar los datos.

El descuento es del 10% si el cliente es de tipo A o la cantidad de cualquier producto es mayor a 100 sino es de 5%.

Proceso descuento

```

Definir precio,st,des,tp,isv Como Reales;
Dimension nombre[25]; Definir nombre Como
Cadena;

Definir tipoM Como Caracter;

Definir cant Como Entero;

Escribir "Nombre del cliente";

Leer nombre[24];

Escribir "Ingresar el Tipo de cliente";
Leer tipoM;

Escribir "Ingresar el precio del producto";
Leer precio;

Escribir "Ingresar la cantidad ";

Leer cant;

```

```

St<- precio*cant;

Si tipoM ='a' | tipoM='A' | cant>100
Entonces Des<- st*0.10
Sino
    Des<- st*0.05
FinSi

Isv<-(st-des)*0.12;

Tp<-(st-des)+isv;

Escribir "Subtotal ", st;

Escribir "Descuento ", des;

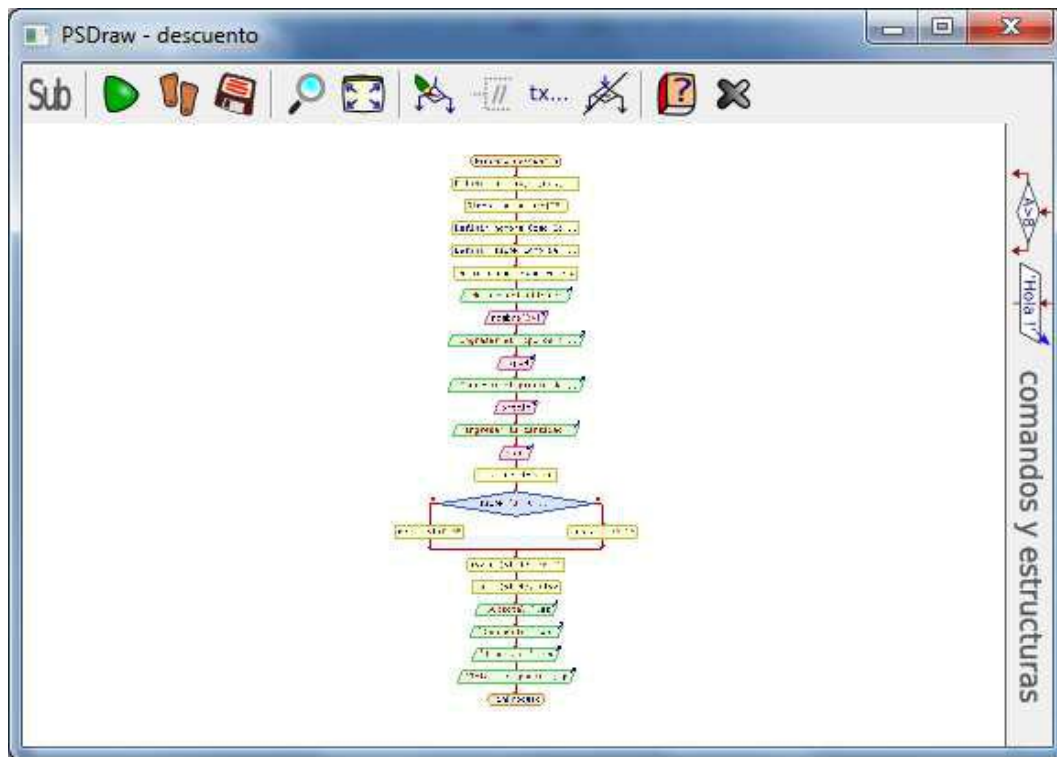
Escribir "Impuesto ", isv;

Escribir "Total a pagar ",tp;

FinProceso

```

Su representación en diagrama de flujo:



Como vemos, el proceso es tan largo, que aparece con la letra muy chica, para que se vea más grande movemos el scroll para que se agrande.

## Uso del operador & ó y

El operador Y (&) se utiliza cuando estamos evaluando dos o más condiciones y queremos que la condición se cumpla cuando las dos condiciones que estamos evaluando se hacen verdadera. Ejemplo

### Ejemplo sobre el operador &

Se ingresa un número y se desea saber si dicho número está entre 50 y 100.

```
Proceso ejemplo_operador_y

    Definir num Como Entero;

    Escribir "Número a evaluar";

    Leer num;

    Si num >=50 & num<=100 Entonces
        Escribir "El número está entre 50 y 100";
    Sino
        Escribir "Fuera del rango 50 y 100";
    FinSi
FinProceso
```

## Exportación a C

PSelnt puede exportar el programa el algoritmo a C, C++, C# y otros lenguajes. Genera un archivo con la extensión .c. No es necesario guardar previamente el archivo en pseudocódigo para que se exporte a C.

Simplemente vamos a Archivo → Exportar y seleccionamos Convertir el código a C (c).

También podemos ver la vista previa yendo a archivo → Exportar → Vista previa

**Nota:** Puede que el código generado por el interpretador no sea del todo correcto, esto se va a ir solucionando en las próximas versiones de PSelnt

## Instrucciones de ciclo

Hemos hecho programas que solo se repiten una vez, pero en la programación necesitamos que los programas corran varias veces y que nos presenten información al final de correr varias veces, en estos casos usaremos ciclos, que son estructuras de repetición, que se repiten hasta cumplir con una condición o simplemente indicamos cuantas veces se van a repetir.

**Nota:** Para evitar ambigüedades, todos los ciclos deben cerrarse siempre, no es posible que hayan "Ciclos abiertos".

## Ciclo Mientras:

### Sintaxis

Mientras condición Hacer

instrucciones;

FinMientras

El ciclo mientras se utiliza cuando se quiere ejecutar repetidamente un bloque instrucciones basado en una condición, el ciclo se repite mientras la condición se cumple.

### ***Ejemplo sobre el ciclo Mientras usando un contador***

Ingresar 10 nombres

Proceso contar\_nombres

Definir contador Como Entero;

Dimension nombre[25];

Definir nombre Como Cadena;

Contador<-0;

Mientras contador<10 Hacer

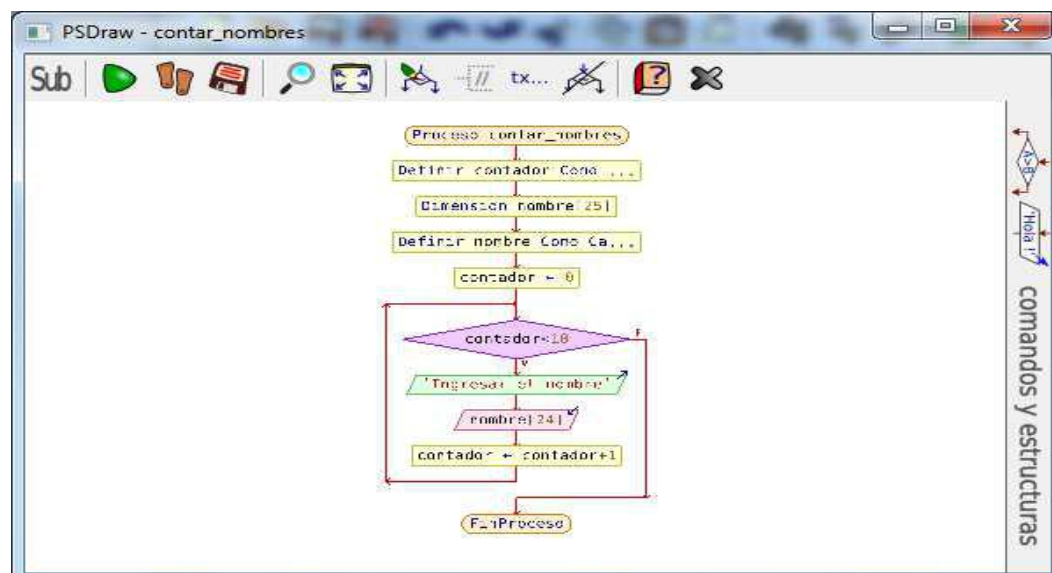
Escribir "Ingresar el nombre";

Leer nombre[24];

contador<-contador+1;

FinMientras

FinProceso



En este programa introducimos el concepto de contador, que es una variable que se incrementa su valor en 1 y de esta manera contamos cuantos nombres se van ingresando para parar cuando ingresemos 10, esto nos dice que la condición ya no se cumple porque cuando el contador vale 10 la condición de  $\text{contador} < 10$  ya no se cumple porque es igual y el ciclo termina.

### ***Ejemplo sobre el ciclo Mientras usando acumuladores***

Ingresar 10 números y al final presentar la suma de los números.

Proceso acumuladores

```
Definir Contador, Suma, Num Como Enteros;  
  
Contador<-0;  
  
Suma<-0;  
  
Mientras contador<10 Hacer  
  
    Escribir "Ingresar un número";  
  
    Leer Num;  
  
    Contador<-Contador+1;  
  
    Suma<-Num+Suma;  
  
FinMientras  
  
Escribir "Suma de los 10 números ", Suma;
```

FinProceso

***Nota:*** Para evitar ambigüedades, los números se deben ingresar de a uno pulsando enter sucesivamente. Ingresarlos en una fila separados por espacios provocaría un error de no coincidencia de tipos ya que se toma el espacio como un tipo de dato de ingreso más y un espacio no es un dato de tipo numérico.

El ciclo recorre 10 veces y pide los 10 números, pero la línea `suma<- suma + num`, hace que la variable suma, incremente su valor con el número que se introduce en ese momento, a diferencia del contador, un acumulador se incrementa con una variable, acumulando su valor hasta que el ciclo termine, al final se presenta la suma, solo en ese momento se debe de presentar un acumulador, porque antes no reflejaría la suma de todos los números.

Siempre que usemos un contador o acumulador debemos darle un valor inicial de generalmente será 0.

Ejemplo sobre el ciclo mientras usando una respuesta para controlar la salida del ciclo.

Ingresar el nombre del cliente, el precio del producto, cantidad y luego calcular el subtotal, iva y total a pagar, presentar los datos luego preguntar si desea continuar, al final presentar el monto global de la factura.

Proceso producto

```
    Definir Resp Como Caracter;

    Dimension nombre[25];

    Definir nombre Como Cadena;

    Definir Precio, cantidad, totalglobal, st, isv, tp Como
Reales;

    Totalglobal<-0;

    Resp<-'S';

    Mientras resp <>'N' Hacer

        Escribir "Nombre del cliente";

        Leer nombre[24];

        Escribir "Ingresar la cantidad del producto ";

        Leer cantidad;

        Escribir "Ingresar el precio de producto ";

        Leer precio;

        St<- precio*cantidad;

        Isv<-st*0.012;

        Tp<-st-isv;

        Totalglobal<-totalglobal+st;

        Escribir "Subtotal ", st;

        Escribir "Impuesto sobre venta ", isv;

        Escribir "Total a pagar ", tp;

        Escribir "Desea continuar S/N";

        Leer Resp;

    FinMientras

    Escribir "Total de la venta ", totalglobal;

FinProceso
```

En este ejercicio, observamos que el ciclo lo controla una respuesta que se pide al final S para seguir o N para terminar , pero daría el mismo resultado si escribe cualquier letra distinta a S, aunque no sea N siempre seguiría funcionando el programa, la validación de los datos de entrada lo estudiaremos más adelante.



### ***Ejemplo sobre estructuras de condición dentro del ciclo Mientras.***

Ingresar el nombre del alumno, la nota examen y nota acumulada, luego calcular la nota final, y presentar la nota final y la observación del alumno.

Preguntar si desea continuar, al final presentar el número de aprobados y reprobados.

```
Proceso aprobado
  Definir Resp Como Caracter;
  Dimension nombre[25];
  Definir nombre Como Cadena;
  Definir na,ne,nf Como Reales;
  Definir cr,ca Como Enteros;
  cr<-0;
  ca<-0;
  Resp<-'S';
  Mientras resp<>'N' Hacer
    Escribir "Nombre del alumno ";
    Leer nombre [24];
    Escribir "Nota acumulada ";
    Leer na;
    Escribir "nota examen ";
    Leer ne;
    nf<-na+ne;
    Si nf >= 60 Entonces
      Escribir "Tu estás Aprobado";
      ca<-ca+1;
    Sino
      Escribir "Tu estás Reprobado";
      cr<-cr+1;
    FinSi
    Escribir "Nota final ", nf;
    Escribir "Desea continuar S/N";
    Leer Resp;
  FinMientras
  Escribir "Total de reprobados ", cr;
  Escribir "Total de aprobados ", ca;
FinProceso
```

***Nota:*** Las variables no pueden declararse inicializadas, se declaran primero y se inicializan después.

Como podemos observar en las líneas del programa, usamos dentro del ciclo mientras, decisiones para poder contar los reprobados y aprobados que resulten del ingreso de los alumnos, si la nota es mayor a 60 escribe aprobado e incrementa el contador y sino hace lo contrario, escribir reprobado e incrementar el contador.

## **Ciclo Para**

### **Sintaxis**

Para variable <- valor\_inicial Hasta valor\_final Con Paso Paso Hacer

instrucciones

FinPara

## Descripción

El ciclo Para se utiliza generalmente para ejecutar un conjunto de instrucciones que se repiten un número de veces, establecido antes de ejecutar el ciclo.

**Variable:** es de tipo entero

**Valor\_inicial:** este puede ser un número entero o una variable entera.

**Valor\_final:** este puede ser un número entero o una variable entera.

**Paso :** este puede ser un número entero o una variable entera.

***Nota: el paso 1 puede omitirse, tanto en sintaxis estricta como flexible***

***Ejemplo : presentar los números del 1 al 10 en la pantalla.***

```
Proceso ciclo_Para
    Definir I Como Entero;
    Para I<-1 Hasta 10 Con Paso 1 Hacer
        Escribir I;
    FinPara
FinProceso
```

El programa el ciclo para establece el número de veces que se repetirá el ciclo

indicando **1 hasta 10** luego la variable I toma el valor 1 a 10 según el ciclo se va

ejecutando, es por eso que al escribir

la primera vez escribe 1

la segunda vez 2

y así hasta llegar al

final que es 10.

***Ejemplo : sobre el uso de variables en el rango del ciclo Para.***

```
Proceso ciclo_Para_2
    Definir I, final Como Enteros;
    Escribir "Ingresar el número de veces a repetir el ciclo ";
    Leer final;
    Para I<-1 Hasta final Con Paso 1 Hacer
        Escribir I;
    FinPara
FinProceso
```

Ahora el programa se vuelve más dinámico, nosotros podemos indicar el número de veces que se repetirá el ciclo, usando una variable entera para indicar el final del ciclo.

### **Ejemplo uso del ciclo Para, en el cálculo del factorial de un número.**

Proceso ciclo\_Para\_con\_factorial

```
Definir I, numero, factorial Como Enteros;  
factorial<-1;
```

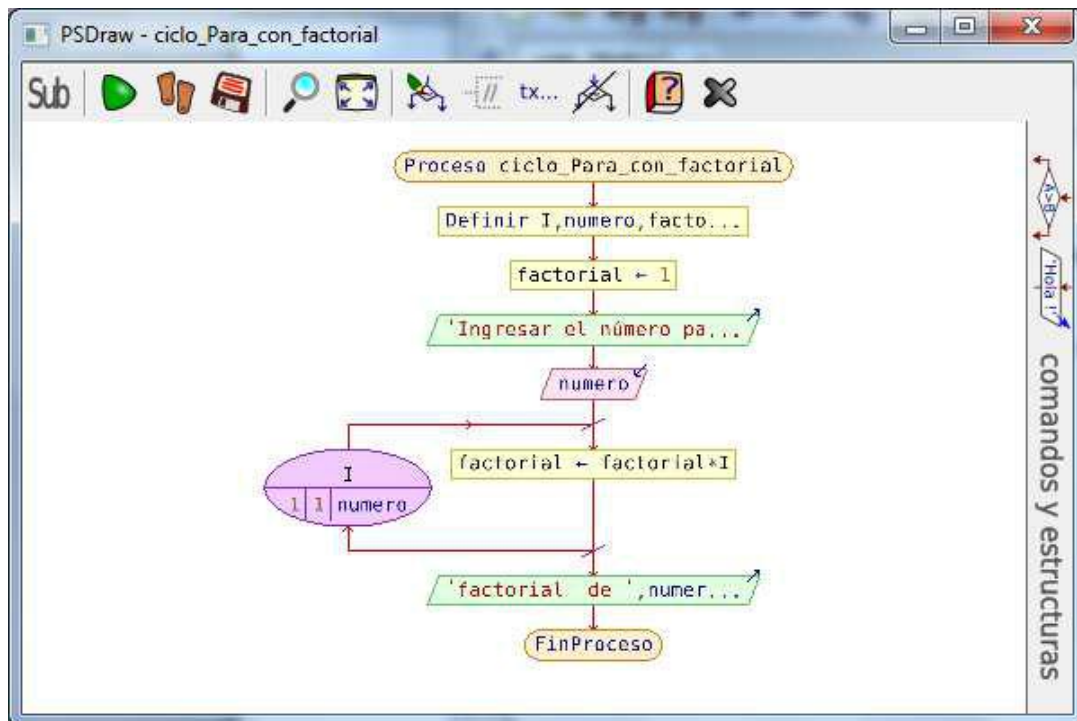
```
Escribir "Ingresar el número para determinar su factorial ";  
Leer numero;
```

```
Para I<-1 hasta numero Con Paso 1 Hacer  
    factorial<- factorial*I;
```

```
FinPara
```

```
Escribir "factorial de ", numero ," es ", factorial;  
FinProceso
```

En este ejercicio se inicia el factorial en 1 porque acumulara una multiplicación y si lo iniciamos en cero nos daría el resultado cero, si nosotros ingresar 3, el ciclo se ejecutara 3 veces , el factorial tomaría el valor de 1x2x3. Diagrama de flujo:



### **Ciclos con paso negativo**

PSelnt también puede realizar ciclos inversos para mostrar, por ejemplo secuencias de mayor a menor, solamente invirtiendo el orden de los números del ejercicio anterior y colocando como Paso -1

Proceso ciclo\_Para\_negativo

```
Definir I Como Entero;
```

```
Para I<-10 Hasta 1 Con Paso -1 Hacer  
    Escribir I;
```

```
FinPara
```

```
FinProceso
```

**Nota:** Puede omitirse el paso negativo en el ciclo para yendo a Configurar Opciones de lenguaje (perfiles)... → Personalizar → Permitir omitir el paso -1 en los ciclos Para.

## Ciclos anidados

Cuando un ciclo se encuentra dentro de otro ciclo se le llama ciclo anidado.

### **Ejemplo de un ciclo anidado**

Producir la siguiente salida en la pantalla

11111

22222

33333

44444

```
Proceso ciclo_Para_anidado
  Definir I,k Como Enteros;
  Para I <- 1 Hasta 4 Hacer
    Para K <-1 Hasta 5 Hacer
      Escribir I Sin Bajar;
    FinPara
    Escribir "";
  FinPara
FinProceso
```

Cuando usamos ciclos anidados, las variables para manejar los ciclos para deben de ser diferentes pues cada una de ellas toma un valor diferente, en este ejercicio necesitamos que se haga 5 veces el ciclo que está dentro, que es el que presenta 4 veces el valor de la I , luego salta una línea , para que aparezcan los grupos de números en cada línea.

### **Ejemplo de un ciclo anidado**

Ingresar 5 números y calcular el factorial para c/u de los números.

En este ejercicio necesitamos ingresar 5 números pero cada vez que ingresemos un numero debemos de calcular el factorial , entonces necesitaremos una variable para el calculo del factorial, que forzosamente tiene que inicializarse en 1 cada vez que el ciclo que calcula el factorial inicie, des esta manera estaremos seguro que la variable no ha acumulado el valor del factorial anterior.

Ahora con lo anterior deducimos que necesitamos un ciclo para pedir los datos y otro para calcular el factorial.

Proceso factorial

```
Definir I,k,fac,num Como Enteros;  
Para I <- 1 Hasta 5 Hacer  
    Escribir " ingresar un número ";  
    Leer Num;  
    fac<-1;  
    Para k <-1 Hasta num Hacer  
        fac<-fac*K;  
    FinPara  
    Escribir "factorial de ", num , " es ",fac;  
FinPara  
FinProceso
```

## Ciclo Repetir

### Sintaxis:

Repetir

    Instrucciones;

Hasta Que condición

### Descripción

El ciclo Repetir es lo contrario al ciclo Mientras, en éste la ejecución se lleva a cabo hasta que se cumple la condición impuesta.

La diferencia con el ciclo Mientras radica en que este evalúa la condición desde el principio, y si está no se cumple, el código que está encerrado dentro del cuerpo del mientras no se ejecuta.

En cambio, el Repetir - Mientras Que evalúa la condición para seguir ejecutándose luego de haber ejecutado el código dentro de su cuerpo, es decir siempre se ejecuta por lo menos una vez el código.

**Nota:** En perfil flexible, habilitando sintaxis flexible o en personalizar también es posible usar la estructura

    Hacer

        //Instrucciones;

    Mientras Que

o

    Repetir

        //Instrucciones;

    Mientras Que

*como alternativa a Repetir – Mientras Que correspondiente a la sintaxis estricta. Recordar que en este caso la condición sale por el distinto, a diferencia del Repetir que sale por el igual.*

### **Ejemplo del Repetir**

Ingresar el nombre del alumno, la nota , luego preguntar si desea continuar , al final presentar el número de aprobados y reprobados .

```
Proceso ejemplo_Repetir
    Definir resp Como Caracter;
    Definir nota Como Real;
    Definir ca,cr Como Enteros;
    Dimension nombre[25];
    Definir nombre como Cadena;
    ca<-0;
    cr<-0;
    Repetir
        Escribir "ingresar el nombre del alumno ";
        Leer nombre[24];
        Escribir "ingresar la nota del alumno ";
        Leer nota;
        Si nota >= 60 Entonces
            ca<-ca+1;
        Sino
            cr<-cr+1;
        FinSi
        Escribir "Desea continuar S/N";
        Leer resp;
        Hasta Que resp='n' | resp='N';
        Escribir "Aprobados ",ca;
        Escribir "Reprobados ",cr;
FinProceso
```

si comparamos este programa con los hechos con el ciclo mientras notaremos que la variable Resp le damos un valor inicial de 'S' , para que sea distinta de N , ya que la condición se verifica al inicio del ciclo , pero ahora con el ciclo repita ya no es necesario pues el primer valor de resp lo toma dentro del ciclo , que es la pregunta que hacemos si desea continuar, y luego verificamos la condición.

Algo importante del ciclo Repetir es, como ya se dijo, que se ejecuta por lo menos una vez, antes de validar la condición de salida del ciclo, es por esto , que siempre que escribamos un programa que verifique la condición antes de entrar ciclo se debe de usar el ciclo Mientras.

El programa anterior no es la versión final, puesto que debemos hacer que el usuario solo ingrese S o N cuando responda si desea continuar , esto nos lleva a escribir un ciclo repetir dentro del ciclo repetir, para pedir la respuesta y hacer que se salga del ciclo solo cuando responda S o N , de esta manera estaremos Seguros de que la repuesta es correcta.

```

Proceso ejemplo_Repetir
  Definir resp Como Caracter;
  Definir nota Como Real;
  Definir ca,cr Como Enteros;
  Dimension nombre[25];
  Definir nombre como Cadena;
  ca<-0;
  cr<-0;
  Repetir
    Escribir "Ingresar el nombre del alumno ";
    Leer nombre[24];
    Escribir "Ingresar la nota del alumno ";
    Leer nota;
    Si nota >= 60 Entonces
      ca<-ca+1;
    Sino
      cr<-cr+1;
  FinSi
  Repetir
    Escribir "Desea continuar S/N";
    Leer resp;
    Hasta Que resp='N' | resp='S'
  Hasta Que resp='N';
  Escribir "Aprobados ",ca;
  Escribir "Reprobados ",cr;
FinProceso

SubProceso calculo(alum Por Referencia)
  Definir i Como Entero;
  Para i <- 0 Hasta 4 Con Paso 1 Hacer
    alum[i].nf<-alum[i].na + alum[i].ne alum[i].obs<-
      observacion(alum[i].nf);
  FinPara
FinSubProceso

SubProceso presentar (alum)
  Definir i Como Entero;
  Para i <- 0 Hasta 4 Con Paso 1 Hacer
    Escribir "Nombre del alumno ",alum[i].nombre;
    Escribir "Nota Final ",alum[i].nf; Escribir
      "Observación ",alum[i].obs;
  FinPara
FinSubProceso

Proceso Principal
  // delcaración del arreglo de tipo registro
  Definir alum Como reg_alumno; Ingreso(alum);
  Calculo(alum);
  Presentar(alum);

FinProceso

```

### Ejemplo arreglos con estructura.

Se declara una estructura con las variables de nombre ventas, comisión ihss y total a pagar, se laboran una función para el seguro social, luego se elabora un procedimiento de ingreso de datos donde se el nombre y las ventas, después el procedimiento de cálculo, donde se determina la comisión que es el 5% de las ventas, el seguro usando la función del Seguro y el total a pagar, luego se presentan los datos usando un procedimiento.

```

Registro emple <- Empleado
    Dimension nombre[30];
    Definir nombre Como Cadena;
    Definir ventas,comis,ihss,tp Como Reales;
FinRegistro

SubProceso retorno <- seguro
    Dimension[5] empleado;
    Definir empleado Como emple;
    Definir sueldo, retorno Como Real;
    Si sueldo > 2400 Entonces
        retorno <- 84;
    Sino
        retorno <- 0.035*sueldo;
    FinSi
FinSubProceso

SubProceso Ingreso (emple)
    Dimension[5] empleado;
    Definir empleado Como emple;
    Definir i Como Entero;
    Para i <- 0 Hasta 1 Con Paso 1 Hacer
        Escribir "ingresar Nombre del Empleado ";
        Leer emple[i].nombre;
        Escribir "Ingresar las ventas ";
        Leer emple[i].ventas;
    FinPara
FinSubProceso

SubProceso Calculo(emple)
    Dimension[5] empleado;
    Definir empleado Como emple;
    Definir I Como Entero;
    Para i <- 0 Hasta 1 Con Paso 1 Hacer
        emple[i].comis<-emple[i].ventas*0.05;
        emple[i].ihss<-seguro(emple[i].comis);
        emple[i].tp<-emple[i].comis-emple[i].ihss;
    FinPara
FinSubProceso

SubProceso Presentar (emple)
    Dimension[5] de empleado;
    Definir empleado Como emple;
    Definir i Como Entero;
    Para i <- 0 Hasta 1 Hacer
        Escribir "Empleado ",emple[i].nombre;
        Escribir "";
        Escribir "Comisión ..:", emple[i].comis;
        Escribir "";
        Escribir "Seguro Social..:", emple[i].ihss;
        Escribir "";
        Escribir "Total a Pagar ..:", emple[i].tp;
        Escribir ""
        Escribir "";
    FinPara
FinSubProceso

Proceso principal
    Escribir Ingreso(emple);
    Calculo(emple);
    Presentar(emple);
FinProceso

```

**Bibliografía :** <http://studylib.es/doc/380831/funciones>

- <http://www.slideshare.net/juanrobyn/manual-de-pseint>
- <http://slideplayer.es/slide/9449505/>