

Programación en:

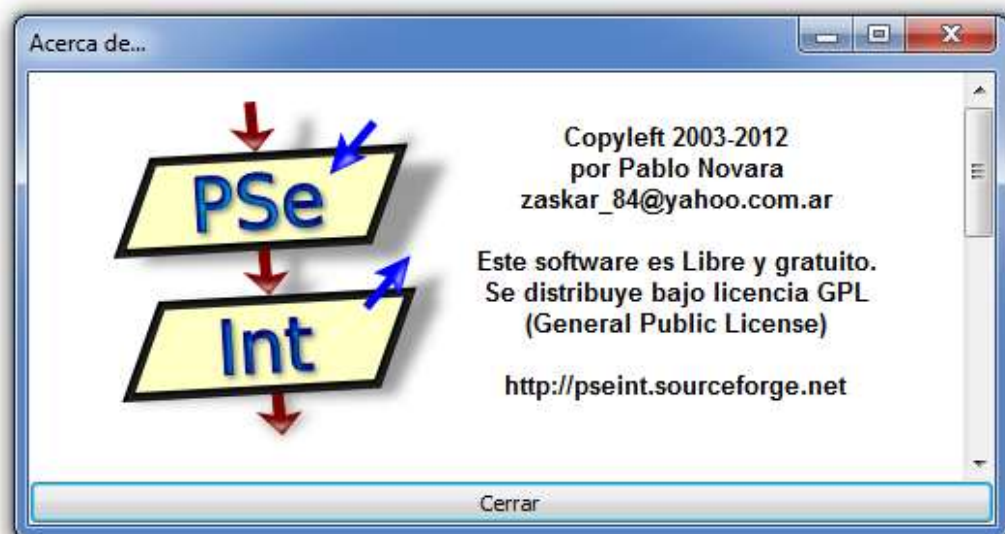
PSeInt

Por Lic. Rommel Castillo Suazo

Original para LPP

Implementado en PSeInt por

Alejandro Caro



Índice

Tema	Pág.
¿Qué es PSeInt?	3
Instalación	4
Mi primer programa	5
Diagramas de flujo	10
• Editor de diagramas de flujo	
Declarar variable	12
Operadores	13
Asignaciones y Operaciones matemáticas en un programa.	14
Instrucciones Condicionales	
• Si	16
• Si anidado	18
• Segun	20
• Operador	22
• Operador &	24
Instrucciones de ciclo	
• Ciclo Mientras	26
• Ciclo Para	32
○ Ciclos Anidados	32
○ Ciclo Para negativo	34
• Ciclo Repetir	37

¿Qué es PSeInt?

PSeInt, es la abreviatura de *Pseudocode Interpreter*, Intérprete de Pseudocódigo. Este programa fue creado como proyecto final para la materia *Programación I* 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, un lenguaje de programación ficticio 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 verdadero.

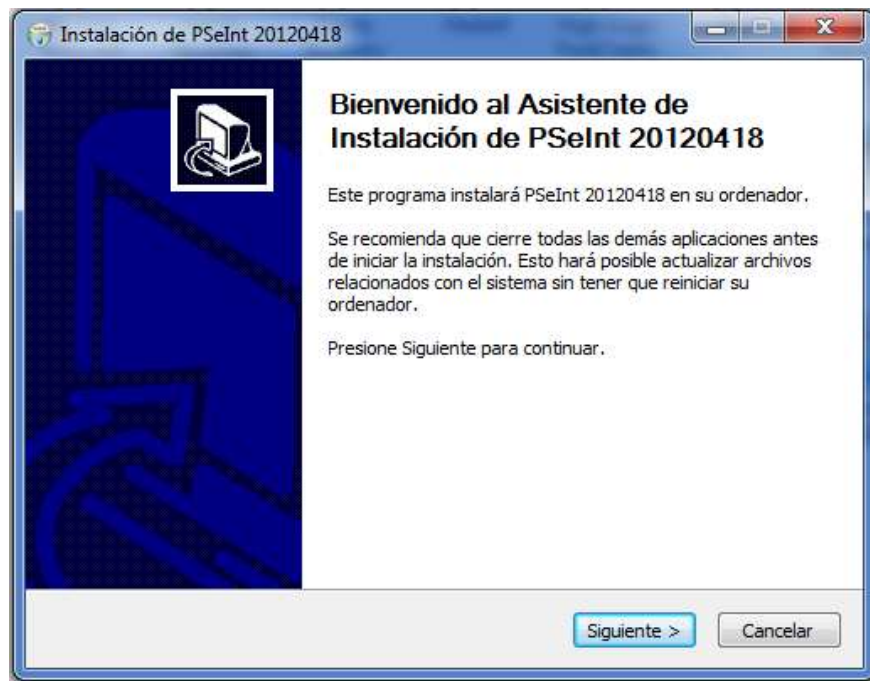
PSeInt incluye en su editor diversas herramientas para que podamos crear y almacenar programas en este peculiar lenguaje, ejecutarlos directamente desde su interfaz, o incluso corregir posibles defectos que encontremos en su desarrollo.

Se puede encontrar un power en odelys2003.files.wordpress.com/2011/10/pseint.pptx

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

- 1) Porque es software libre, sin necesidad de andar gastando dinero, haciendo giros, etc., violando los derechos de autor ni andar creando o consiguiendo cracks, que a veces sus link están inactivos y/o los programas no dejan craquearse.
 - 2) Está constantemente atendido por su creador, a diferencia de los otros compiladores e intérpretes de pseudocódigo que están descontinuados.
 - 3) Posee un foro para reportar errores y obtener ayuda, está también está constantemente atendido por su creador, lo que ayuda a mejorar el programa.
 - 4) Posee una extensa ayuda, que valga la redundancia ayuda a aprender a usarlo, y a aprender el lenguaje.
 - 5) Está disponible su código fuente, y con instrucciones para ejecutarlo, de modo que si sabemos C++ podremos personalizarlo y corregirlo.
- Posee exportación a C++ para que podamos ver el mismo código en C++, lo que ayuda a aprender C++;
- 5) Se trata de un intérprete y no un compilador, el archivo no tienen por qué ser guardado en disco para ser ejecutado, haciendo más cómodo su uso.

Instalación



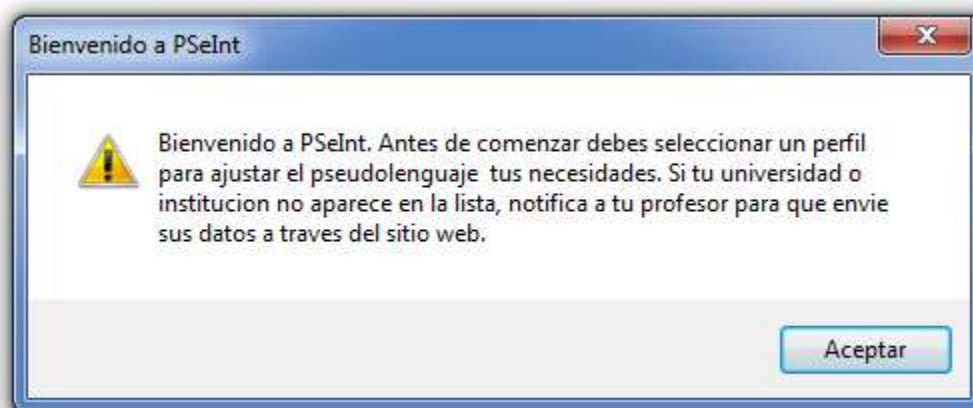
Abrir el archivo " pseint-win-32-xxxxxxx.exe " (xxxx es número de la versión actual), el cual será proporcionado por la página del proyecto, al hacer doble clic se ejecuta el instalador.

Luego presionamos siguiente -> siguiente y así sucesivamente hasta instalarlo.

Apuntes preliminares

Antes de empezar a programar, es conveniente tener una idea del funcionamiento general de PSeInt.

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



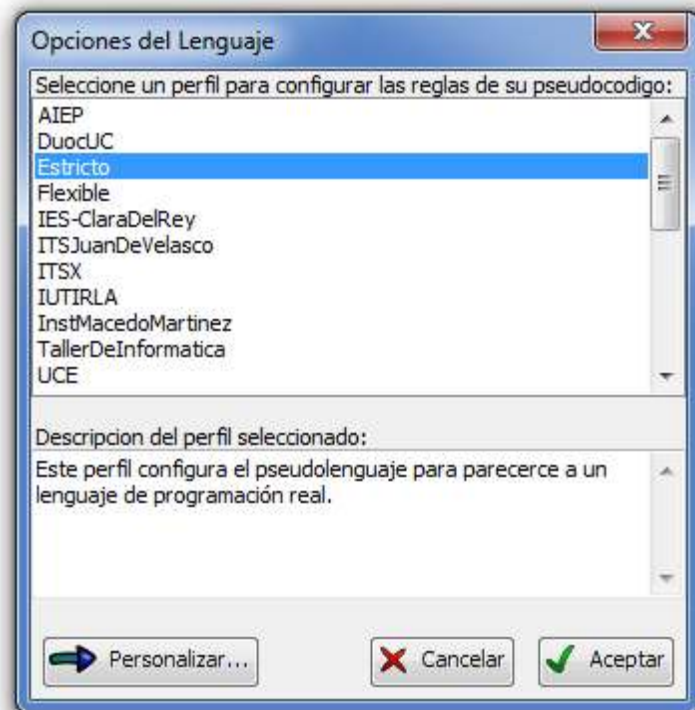
Este manual se maneja con dos perfiles. El estricto, que es el más parecido a un lenguaje de programación real, se debe respetar al pie de la letra el formato del pseudocódigo propuesto por Novara. La sintaxis flexible la usamos para ejecutar ciertos códigos que requieren más flexibilidad a la hora de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

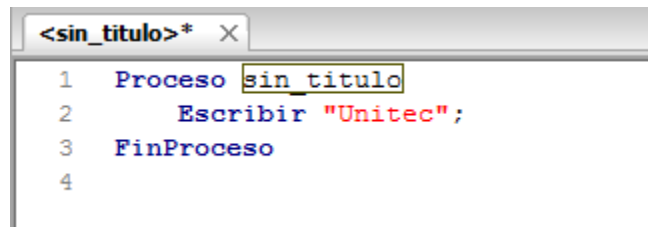
Para abrir PSeInt damos doble clic en el acceso directo PSeInt del escritorio y nos abre el programa.



Escribir mi primer programa

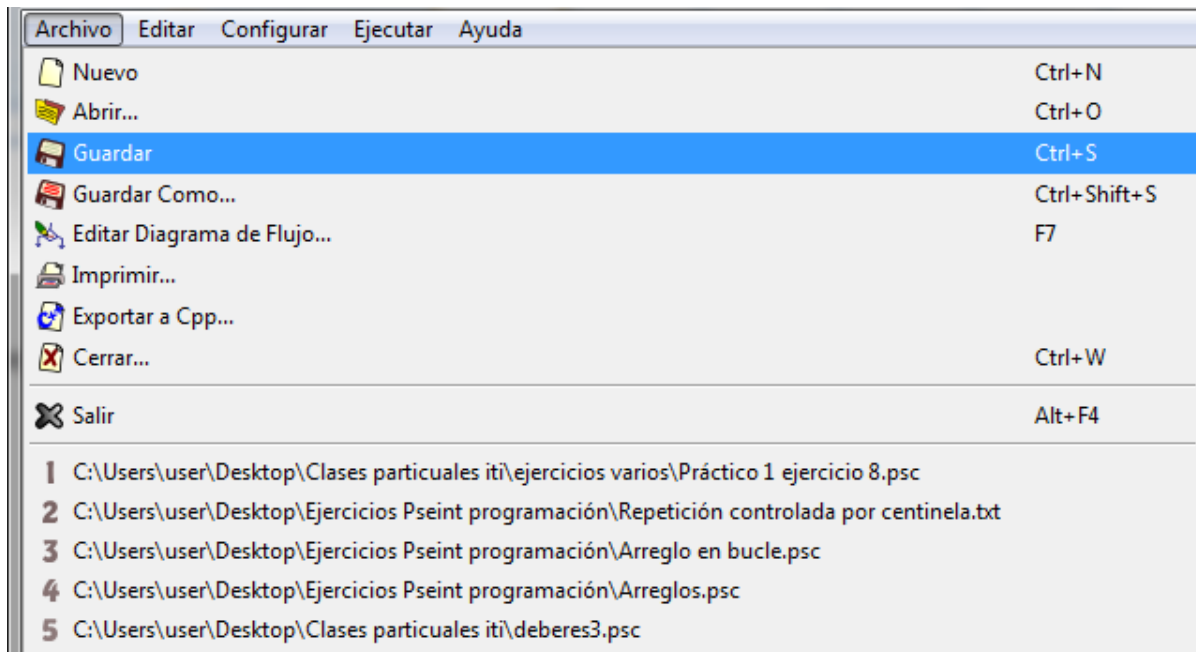
Una vez que hemos abierto PSeInt y habiendo configurado sintaxis estricta, este nos presenta una página que dice Inicio sin_titulo y FinProceso, entre

estas dos líneas escribiremos nuestro primer programa:

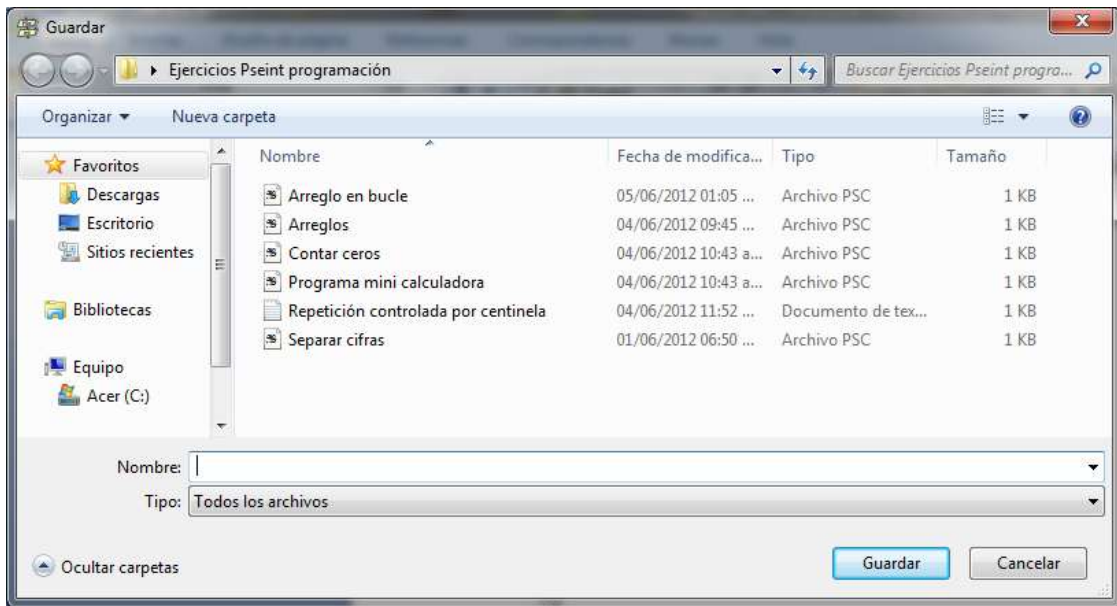


```
<sin_titulo>* X
1 Proceso sin_titulo
2   Escribir "Unitec";
3 FinProceso
4
```

Luego lo guardamos



Escribimos el nombre del programa en la ventana que nos aparece y luego presionamos Guardar Como... .

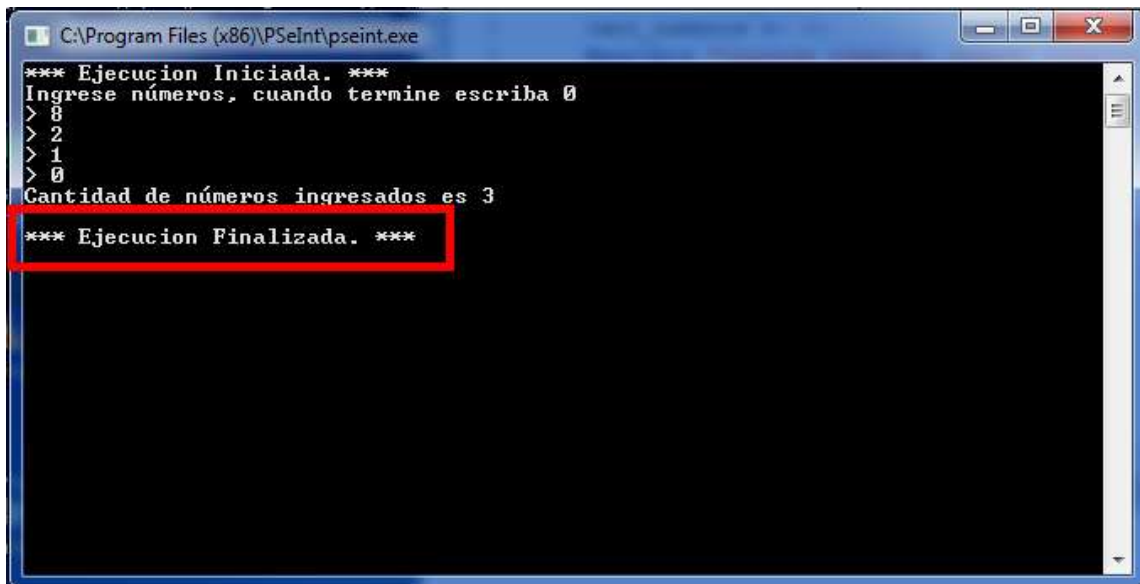


Ahora que los hemos guardado necesitamos, que nuestro programa funcione y escriba en la pantalla Unitec, aunque PSeInt subraya con rojo los errores de sintaxis, también es bueno verificar sintaxis para ver los errores. Para ello vamos a ejecutar, → verificar sintaxis. De todos modos, si tuviéramos errores 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, seleccionamos ejecutar, luego ejecutar, y en la pantalla aparecerá la palabra Unitec 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 esta encerrado entre comillas*. En sintaxis flexible también podemos utilizar la palabra **Imprimir** o **Mostrar**. *Reitero, a menos que se indique que se utiliza sintaxis flexible, nosotros vamos a utilizar siempre sintaxis Estricta.*

Concatenar texto

```
Proceso Ver
    Escribir "Mi primer programa ";
    Escribir " en PSeInt ";
```


FinProceso

La salida del programa es

Mi primer programa en PSeInt

Esto porque el final de la línea hace un retorno y baja a la siguiente, ahora si quisiéramos escribir:

Mi primer programa

En PSeint

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.

El programa sería 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 e contenido de una cadena de texto y el contenido del próximo escriba se escribe en la primera línea.


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

Nota: 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 2: 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.



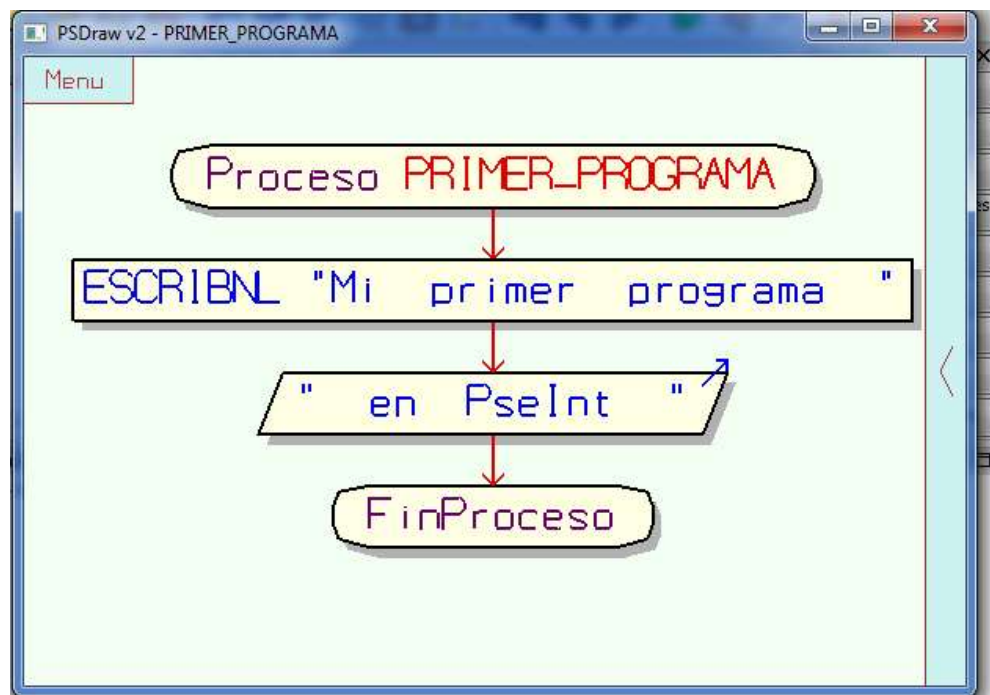
Ejemplo:

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, la sentencia escribir representada en un rectángulo, pues es un cartel



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:

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

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

Ejemplos

Si queremos declarar una variable de tipo entero sería así :

Definir numero Como Entero;

Numero se convierte en una variable de tipo entero

Nota: En sintaxis estricta, las variables no pueden tener caracteres acentuados
Si queremos declarar una variable tipo Cadena para guardar el nombre sería así

Dimension nombre [25];

Definir nombre Como Cadena;

Nota: en sintaxis estricta las variables no pueden tener caracteres acentuados

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

Nota: Ver el apartado Dimensiones para más detalles.

Nota: 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;

Nota2: El plural de Caracter es Cadena

Operadores

PSeInt proporciona los siguientes operadores:

Operador Función

()	Agrupar expresiones
^	Operador para exponenciación
*	Operador de multiplicación
/	Operador de división
% ó Mod	Operador de cálculo de residuo
trunc(valor1 / valor2);	Sintaxis de división entera
& ó y	Operador lógica y
+	Operador de suma
-	Operador de Resta
ó o	Operador lógico o

Nota: En sintaxis flexible, 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 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];
```

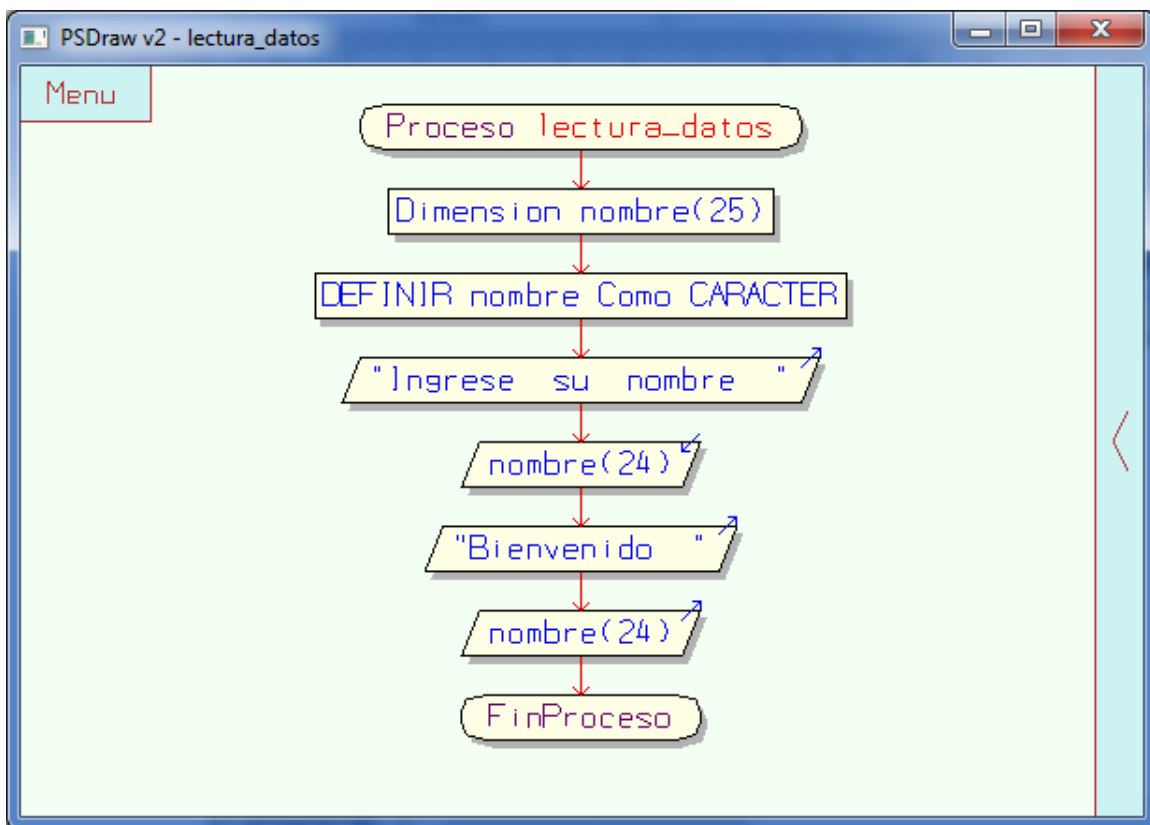
```

    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 ingreso. 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.

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

Ejemplo sobre asignaciones de valores a las variables

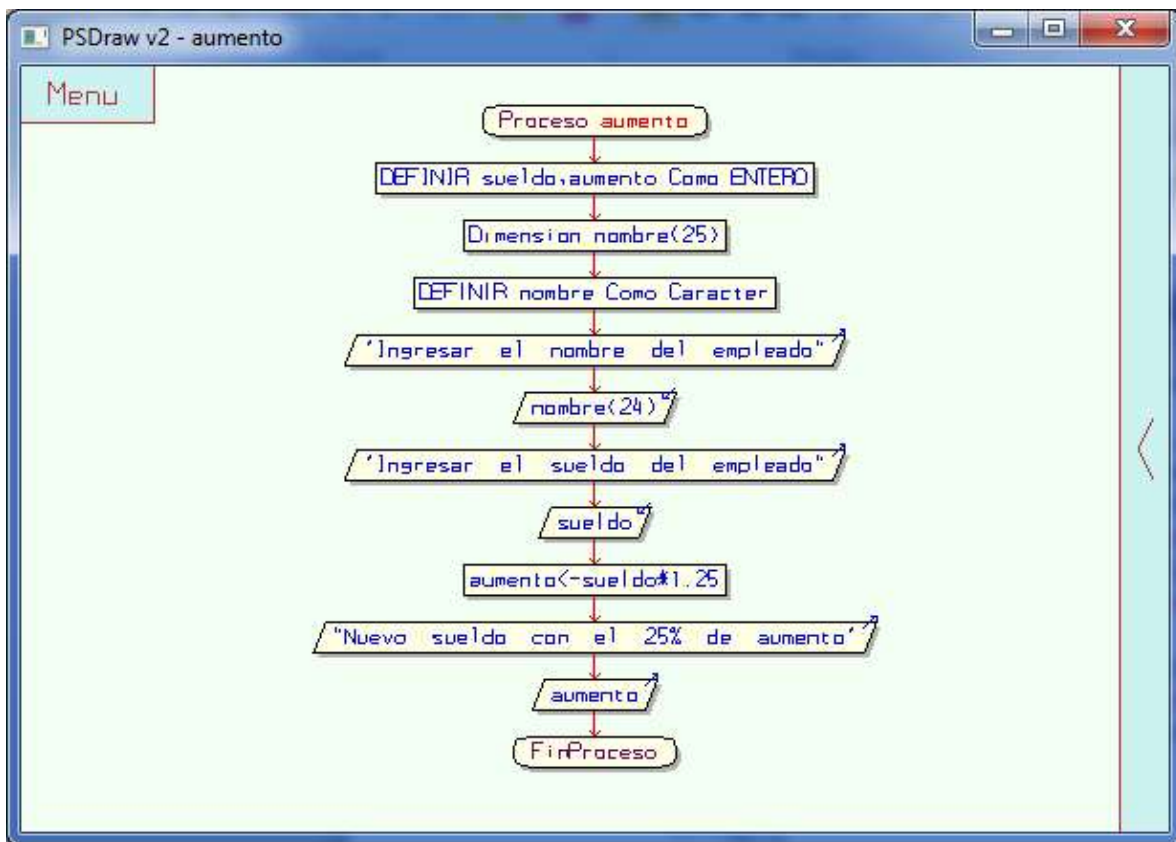
Proceso aumento

```
Definir sueldo, aum 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;
aum <- sueldo * 1.25;
Escribir "Nuevo sueldo con el 25% de aumento";
Escribir aum;
```

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:



Ejemplo sobre suma de cadenas

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[25] <- nombre[24] + " " + apellido[24];
Escribir "Nombre completo " , completo[25];

```

FinProceso

La variable completo toma el valor del nombre mas 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. Véase la página que trata el tema de dimensiones.

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 uso en el calculo 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 calculo 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 ingreso otra zona de las permitidas la comisión es cero.

Estructura Segun

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 sino 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 en el editor.

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 " Comsión 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 |

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

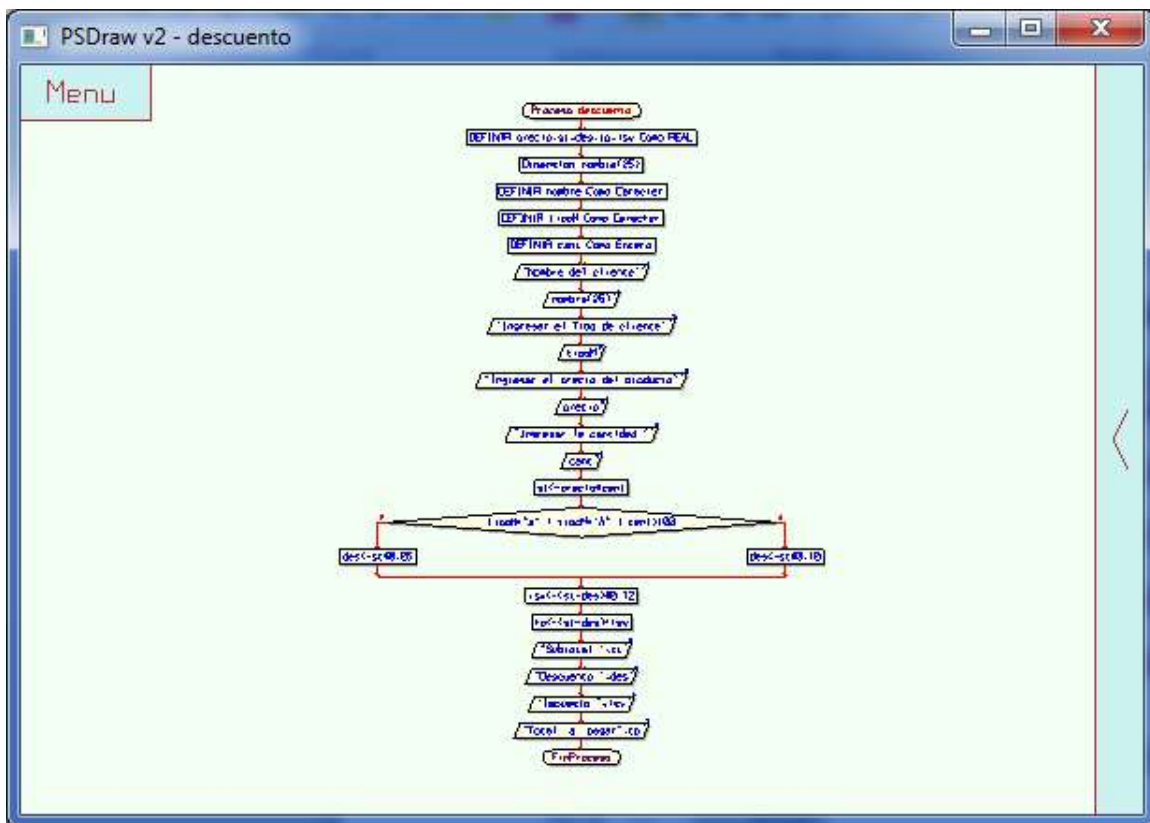
```

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[25];
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 hacia nosotros para que se agrande.

Uso del operador 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++

PSeInt puede exportar el programa el algoritmo a C++. Genera solo un archivo con la extensión .cpp (abreviatura de *C plus plus*, c++) .No es necesario guardar previamente el archivo en pseudocódigo para que se exporte a C++.

Simplemente vamos a Archivo y seleccionamos Exportación a C++

Nota: Al estar el modo experimental, puede que el código generado no sea del todo correcto, esto se va a ir solucionando en las próximas versiones de PSeInt

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.

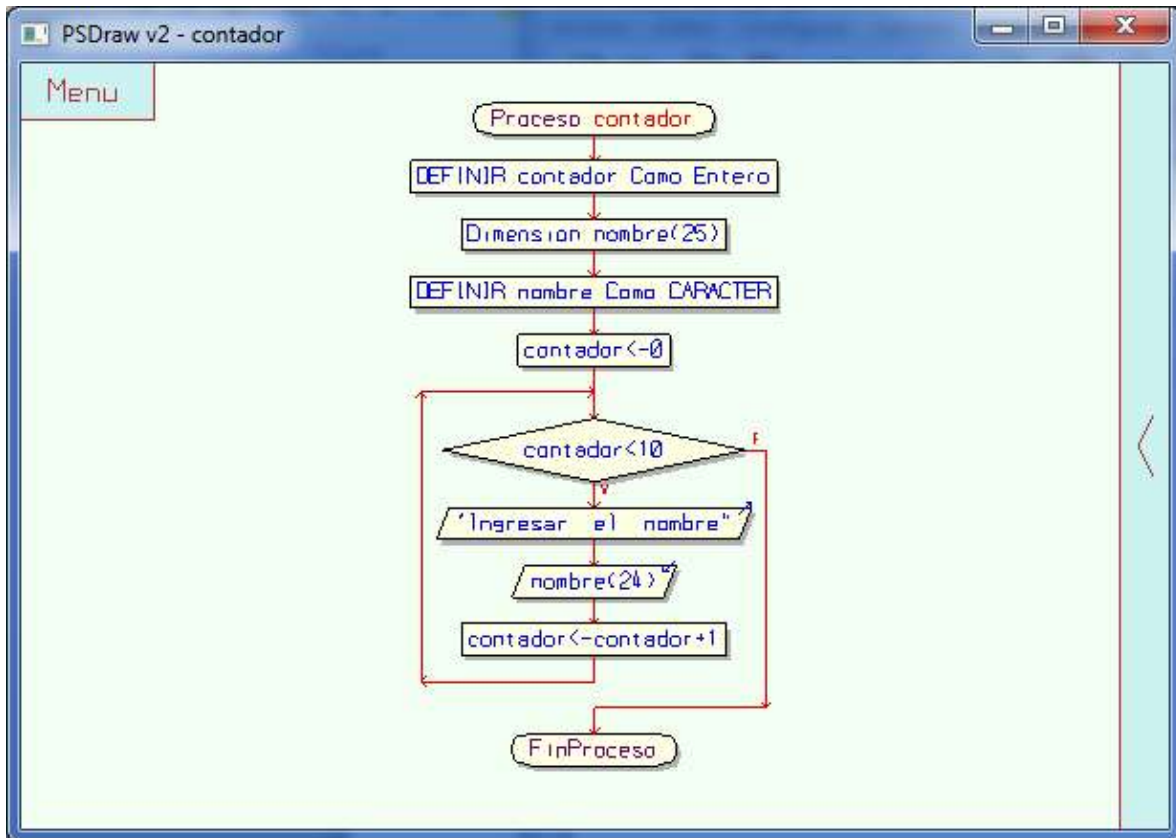
Ejemplo sobre el ciclo Mientras usando un contador

Ingresar 10 nombres

```

Proceso contador
  Definir cont Como Entero;
  Dimension nombre[25];
  Definir nombre Como Cadena;
  Cont<-0;
  Mientras cont<10 Hacer
    Escribir "Ingresar el nombre";
    Leer nombre[24];
    cont<- cont + 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 ingresan 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 numero 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, isv 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;
    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;
        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 mas 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 numero de aprobados y reprobados.

Proceso aprobado

```

Definir Resp Como Caracter;
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;
    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 I 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
    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 numero de veces que se repetirá el ciclo, usando una variable entera para indicar el final del ciclo.

Ejemplo uso del ciclo Para , en el calculo del factorial de un número.

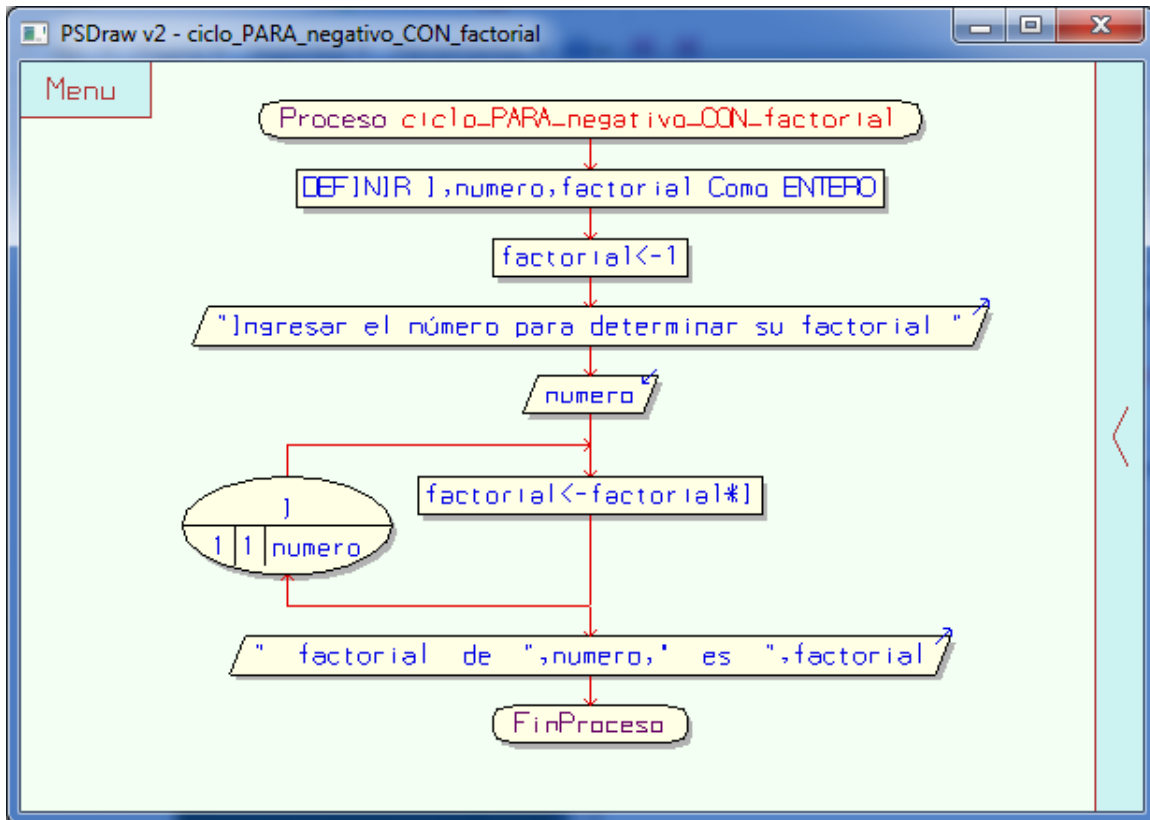
```

Proceso ciclo_Para_negativo_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 negativos

PSeInt también puede realizar ciclos negativos 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: En ciclos negativos el paso no puede omitirse.

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_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 esta 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 número debemos de calcular el factorial, entonces necesitaremos una variable para el cálculo del factorial, que forzosamente tiene que inicializarse en 1 cada vez que el ciclo que calcula el factorial inicie, de esta manera estaremos seguros 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 numero de aprobados y reprobados.

```

Proceso ejemplo_repetir
    Definir resp Como Caracter;
    Definir nota Como Real;
    Definir ca,cr Como Enteros;
    Definir nombre como Cadena;
    ca<-0;
    cr<-0;
    Repetir
        Escribir "ingresar el nombre del alumno ";
        Leer nombre;
        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;
    Definir nombre como Cadena;
    ca<-0;
    cr<-0;
    Repetir
        Escribir "ingresar el nombre del alumno ";
        Leer nombre;
        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

```