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 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 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 no2 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 |

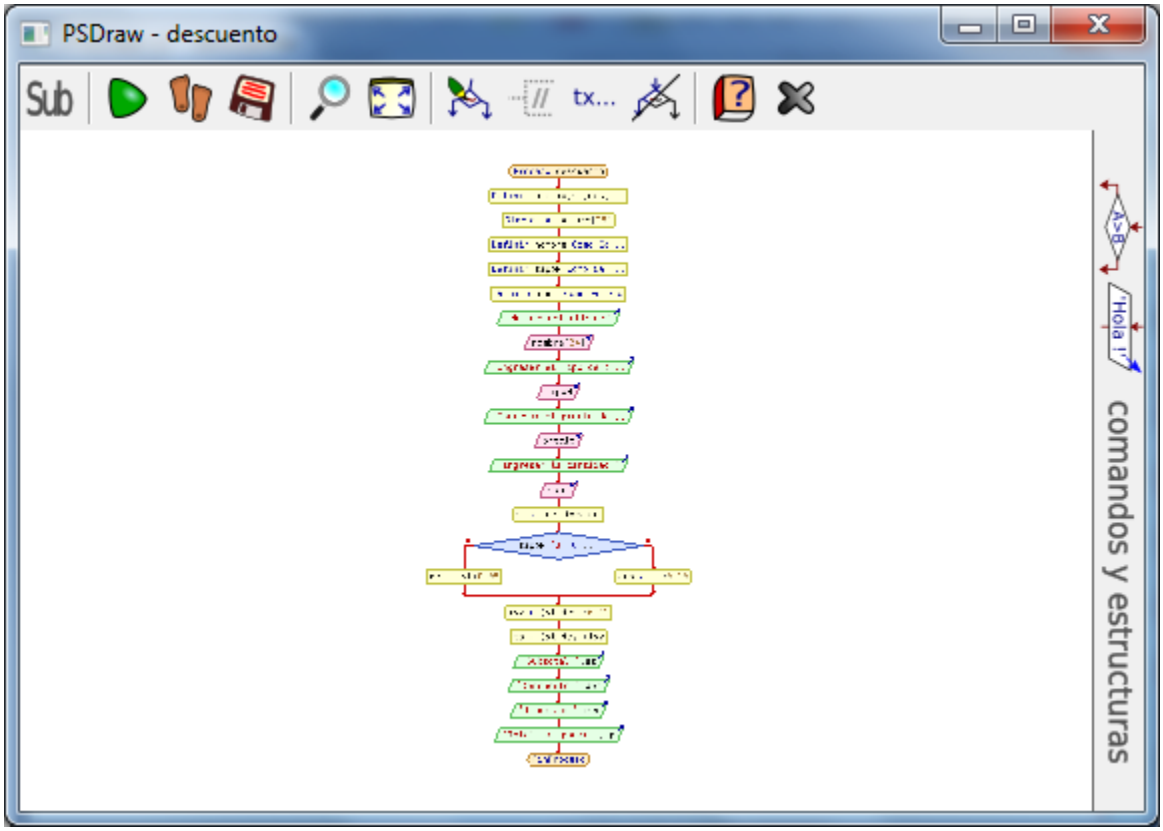
Ingresar el nombre del cliente, luego la cantidad del producto, precio y tipo de cliente, calcular el subtotal, descuento, impuesto s/v, total a pagar, presentar los datos.

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

PSeInt 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 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
```

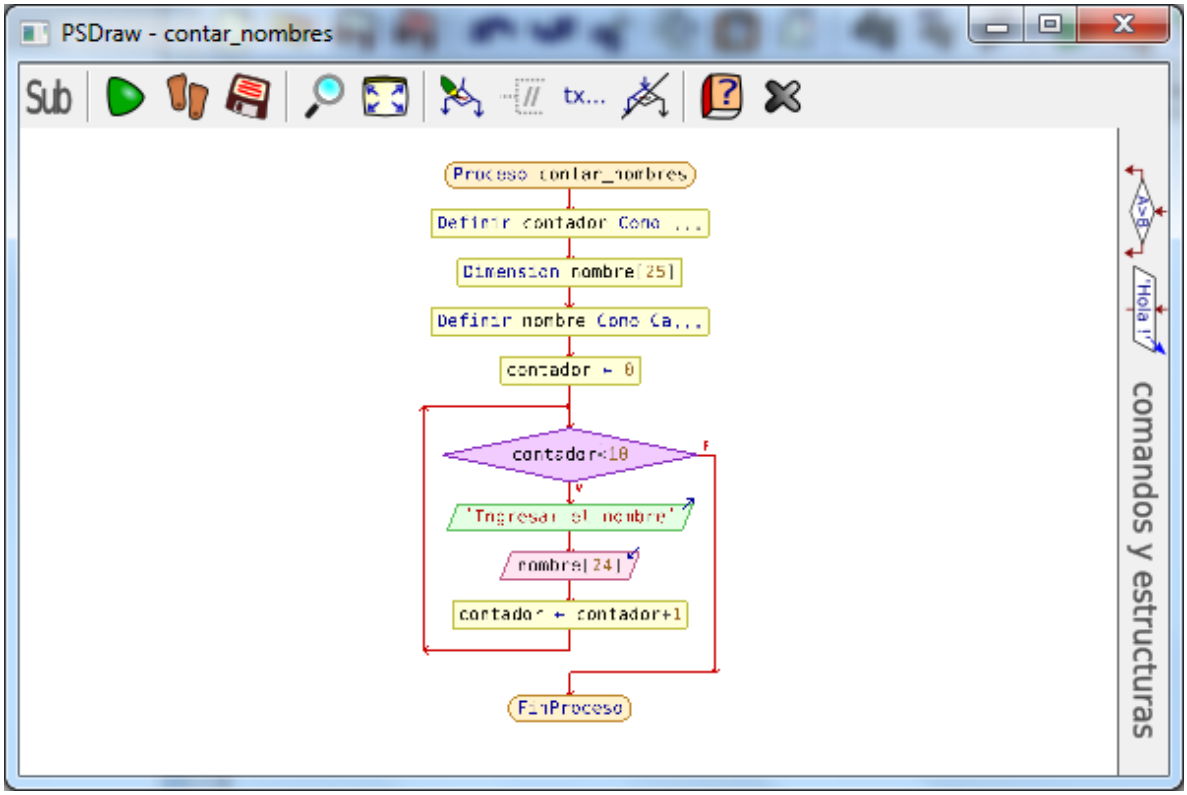
El ciclo mientras se utiliza cuando se quiere ejecutar repetidamente un bloque instrucciones basado en una condición, el ciclo se repite mientras la condición se

cumple.

Ejemplo sobre el ciclo Mientras usando un contador

Ingresar 10 nombres

```
Proceso 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 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, 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;
      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 "Tú estás Aprobado";
                ca<-ca+1;
            Sino
                Escribir "Tú 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**: la expresión "Con 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_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
```

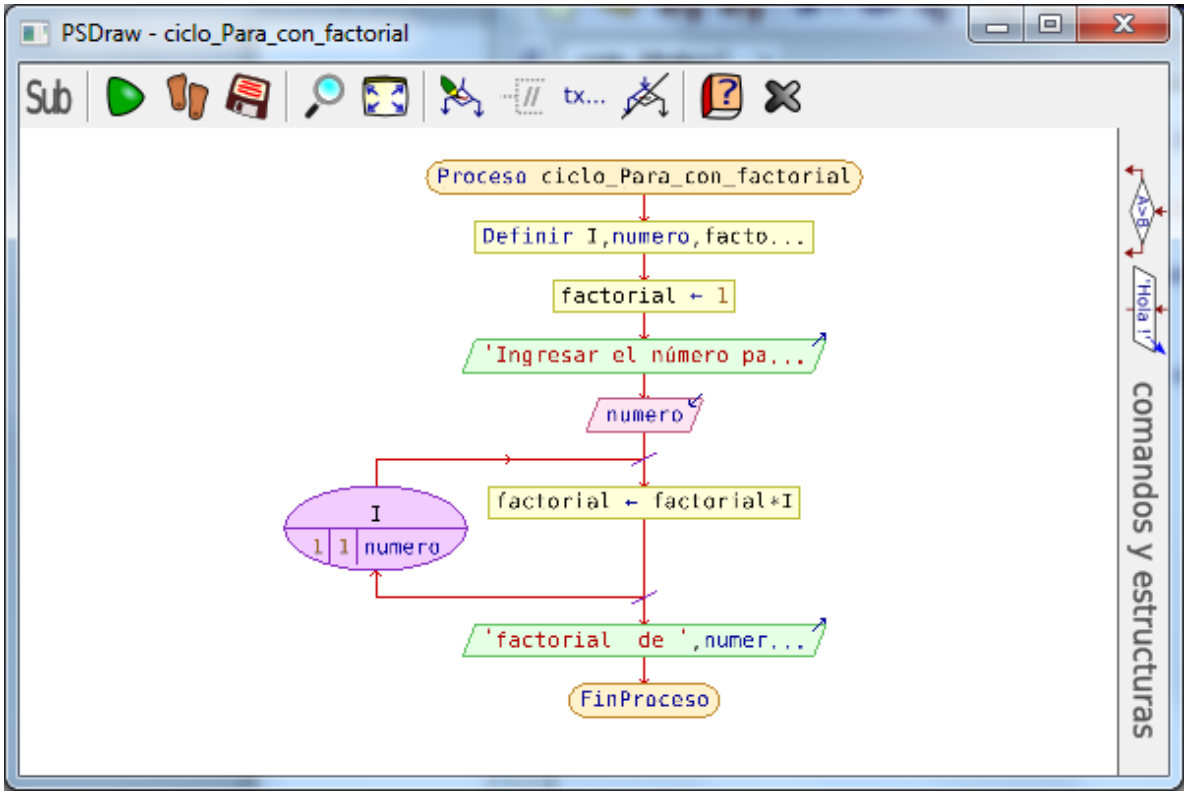
Ahora el programa se vuelve más dinámico, nosotros podemos indicar el número de veces que se repetirá el ciclo, usando una variable entera para indicar el final del ciclo.

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

PSeInt 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 la expresión "Con Paso -1" 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 caculo 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 respuesta 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
```

SubProcesos

Un subproceso es un subprograma, procedimiento o función que realiza una tarea específica y que puede ser definido mediante 0, 1 o más parámetros. Tanto en entrada de información al subproceso como la devolución de resultados desde la subproceso se realiza mediante parámetros, el cual nos sirve para introducir o modificar información del programa principal.

Sintaxis

```
SubProceso  NombreSubProceso

      // ...hacer algo con los argumentos

FinSubProceso
```

Los subprocesos pueden o no tener retorno. En este caso, este subproceso no devuelve nada, los subprocesos que retornan argumentos los veremos más adelante.

Siempre que usemos parámetros estos deben de ser del mismo tipo datos, esto nos dice que la variable del programa, debe de del mismo tipo del parámetro del procedimiento y pasados en el mismo orden en que están colocados en el subproceso.

Nota: *Las variables han de definirse en todos los subprocesos, a no ser que pasen y/o entren por referencia o valor desde otro subproceso*

Ejemplo: elaborar un subproceso que presente 5 asteriscos en una línea horizontal.

```
SubProceso asteriscos
      Definir I Como Entero;
      Para i <- 1 Hasta 5 Hacer
            Escribir "*" Sin Bajar;
```

```
      FinPara
FinSubProceso

Proceso Principal
      Dimension nombre[25];
      Definir nombre como Cadena;
      Escribir "Ingresar el nombre ..:";
      Leer nombre[24];
      asteriscos;
      Escribir "";
      Escribir nombre[24];
      Escribir "";
      asteriscos;
      Escribir "";
FinProceso
```

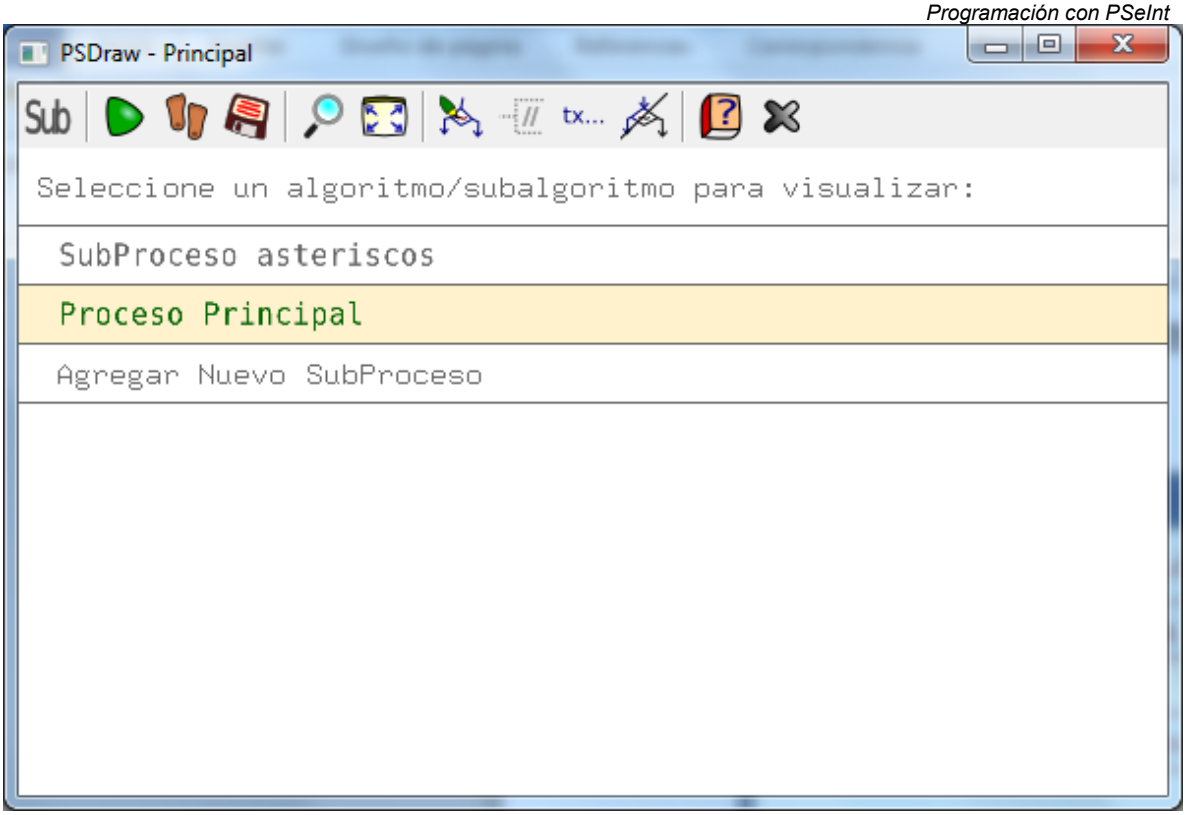
En este programa usamos un subproceso (función -palabra equivalente, PSeInt también la toma-, o procedimiento) para escribir 5 asteriscos, si no lo hubiéramos hecho de esta manera donde se encuentra la instrucción asteriscos; tendríamos que escribir el ciclo, y lo haríamos dos veces, de la forma en que lo escribimos es más estructurado, pues se divide ese proceso en un subprograma, que cuando necesitamos una línea de 5 asteriscos solo llamamos el procedimiento .

***Nota**: Los subprocesos sin parámetros se llaman desde el proceso principal simplemente por su nombre sin más argumentos, se pueden abrir y cerrar paréntesis, pero esto es opcional.*
En cambio, si el subproceso contiene parámetros, estos si deben especificarse cuando se lo llama debe estar precedido por la palabra Escribir, de lo contrario marca error.

Ahora en el programa anterior usa un procedimiento estático, siempre escribirá 5 asteriscos, ahora lo podemos hacer dinámico usando parámetros para indicar cuantos asteriscos queremos presentar en la línea.

Visualizador de diagramas de flujo

Los subprocesos en el diagrama de flujo se muestran de la siguiente manera:



Una lista con los SubProcesos marcados con rojo:

Se elige a cual subproceso entrar pulsando sobre el subproceso. Como dice la captura, también es posible agregar nuevos SubProcesos.

Parámetros de valor

Este tipo de parámetro se le conoce con el nombre de ***parámetro de valor***, que se debe especificar si es por valor o por referencia, por defecto es por valor, este último tipo de parámetro aunque durante el procedimiento su valor cambie el valor no será asignado a la variable del programa principal , por ejemplo si la variable num del programa que presentamos abajo se le asigna otro valor diferente al 10, este cambio se reflejaría en la variable num , y por esto en el programa principal, es este tipo de parámetros que se le conoce como parámetros de valor.

Ejemplo Subproceso con valor

```
SubProceso asteriscos
	Definir num, I Como Enteros;
	num <- 10;
```

```
      Para i <- 0 Hasta num-1 Con Paso 1 Hacer
            Escribir "*" Sin Bajar;
      FinPara
      Escribir "";
FinSubProceso

Proceso principal
      Dimension nombre[25];
      Definir nombre Como Cadena;
      Definir num Como Entero;
      num<-10;
      Escribir "Ingresar el nombre ..:";
      Leer nombre[24];
      asteriscos;
      Escribir "";
      Escribir nombre[24];
      Escribir "";
      asteriscos;
FinProceso
```

En la línea ***num <-10*** estamos asignando al parámetro num de asteriscos el valor de 10, esto hace que el ciclo recorra 10 veces, luego más abajo del programa en la instrucción ***asteriscos;*** se pasó una variable como parámetro asignando el valor de num-1 a número, el cual número en el programa principal tiene un valor de 10 el cual se le asigna a numero en el paso del valor de parámetro.

Parámetros de variable

El siguiente programa, nos enseña el uso de los parámetros de variable o referencia, los cuales se les antepone la palabra reservada VAR para indicar que esa variable será un parámetro de referencia o variable, esto nos indica que cualquier cambio que sufra la variable del procedimiento , la variable del programa principal también lo sufrirá, de esta manera podemos enviar información modificarla y envirar resultados al programa principal.
La sintaxis es la siguiente:

Ejemplo parámetros de variable o referencia.

Elaborar un programa donde se ingrese el nombre y el apellido usando un procedimiento y luego presentar los datos.

```
SubProceso Pedir_datos (nombre Por Referencia, apellido Por Valor)
      Escribir "Ingresar el nombre ";
      Leer nombre;
      Escribir "Ingresar el apellido ";
      Leer apellido;
FinSubProceso

Proceso Principal
      Definir nombre, apellido Como Cadenas;
      nombre<-"No hay cambios en nombre";
      apellido<-"No hay cambios en apellido";
      Pedir_datos(nombre,apellido);
      Escribir "Nombre completo ", nombre," ", apellido;
FinProceso
```

Nota: *En caso de que la variable se deba pasar por referencia siempre se debe indicar. En cambio, si se pasa por valor, la indicación de pase puede omitirse. Siempre por defecto se pasa por valor.*

En el programa anterior, se inician las variables de apellido y nombre, luego se pasan como parámetros al subproceso, el nombre por referencia y el apellido por valor, luego escribimos los valores y solo el nombre presentara el cambio que sufrió en el subproceso y el apellido seguirá escribiendo el mismo valor que tenía al empezar el programa esto porque no se pasó como parámetro de variable (por referencia) sino como de valor y no se le permitió sufrir alguna modificación.

Para mejorar el programa anterior el procedimiento tendría que escribirse así, usando un parámetro de salida, que veremos más adelante:

```
SubProceso nombre, apellido <- pedir_datos (nombre por Referencia)
      Dimension apellido[30];
      Definir apellido Como Cadena;
      Escribir "Ingresar  el  nombre  ";
      Leer nombre;
```

```
      Escribir "Ingresar el apellido";
      Leer apellido;
FinSubProceso
```

Ejemplo

Ingresar la base y el exponente y luego calcular la potencia.

En este programa usaremos un subproceso para el ingreso de los datos y otro para calcular la potencia.

```
SubProceso Ingreso (base Por Referencia, expo Por Referencia)
      Escribir "Ingresar la base ";
      Leer base;
      Escribir "Ingresar el exponente ";
      Leer expo;
FinSubProceso

SubProceso pot <- Potencia (base, expo, pot Por Referencia)
      pot<-1;
      Para I <- 1 Hasta expo Con Paso 1 Hacer
            pot<-pot*base;
      FinPara
FinSubProceso

Proceso principal
      Definir pot como Entero;
      Ingreso(base,expo);
      Escribir "Potencia es ",Potencia(base,expo,pot);
FinProceso
```

En el subproceso de ingreso los dos datos , exponente y base son de tipo entero y parámetros de variable, esto porque necesitamos que el procedimiento nos devuelva los valores para luego introducirlos en el procedimiento potencia pero aquí, base, expo son de tipo valor, esto porque no necesitamos modificar o leer su valor como anteriormente los hicimos en el procedimiento de ingreso , luego la variable pot si se pasa como parámetro de variable debido a que necesitamos modificar su valor y presentarlo en el programa principal.

Nota: *Los subprocesos no se pueden llamar igual que las variables que se declaran*

en el programa.

Nota 2: *Las funciones o subprocesos que retornan valores deben utilizarse como parte de expresiones. Generalmente, el programa pide que se le anteponga la palabra escribir antes del nombre de la función*

SubProcesos que devuelven valor o con retorno

Las SubProcesos también pueden devolver un valor, pero solo uno.

Sintaxis

```
Sintaxis

SubProceso valor_de_retorno <- nombre_SubProceso [( parámetros ) ]
      //[variables locales];

      //instrucciones;

FinSubProceso
```

Si notamos en la sintaxis de la función observamos que hay dos variables entre una flecha que apunta a la izquierda, esta está apuntado a la variable "retorno" la cual devuelve un valor.

Nota: *También se puede usar indistintamente la palabra funcion en lugar de subproceso. En PSeInt, son términos equivalentes.*

Ejemplo: cálculo de la potencia

Usaremos el mismo ejercicio que usamos para los subprocesos, para hacer una demostración de cómo cambiaría el programa usando un subproceso sin retorno para el cálculo de la potencia.

```
SubProceso resp <- potencia (base, expo Por Referencia)
      Definir i, resp Como Enteros;
      resp<-1;
      Para I <- 1 Hasta expo Con Paso 1 Hacer
            resp<-resp*base;
      FinPara
FinSubProceso
```

```
SubProceso Ingreso (base Por Referencia, expo Por Referencia)
      Escribir "Ingresar la base ";
      Leer base;
      Escribir "Ingresar el exponente ";
      Leer expo;
FinSubProceso

Proceso principal
      Definir base, expo, pot Como Enteros;
      Ingreso(base,expo);
      pot<-Potencia(base,expo);
      Escribir "Potencia es ", pot;
FinProceso
```

Nota: *Como se ve en el ejemplo anterior, cuando se llama a funciones, además de deberla llamar con la palabra escribir, cuando se coloca el nombre de la función a llamar los argumentos deben estar pegados al nombre de la función. De los contario aparecerá un cartel que dice Los argumentos para invocar a un subproceso deben ir entre paréntesis*

Ahora veremos cómo dibuja el diagrama de flujo el intérprete de diagramas de flujo:

Diagrama de flujo del procedimiento ingreso:

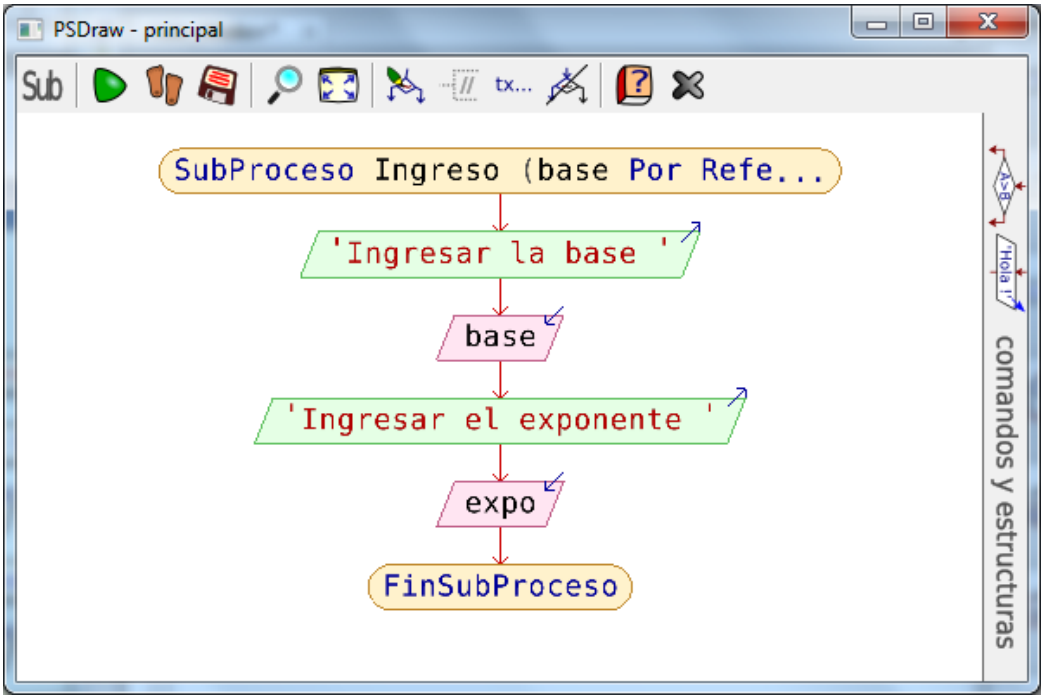
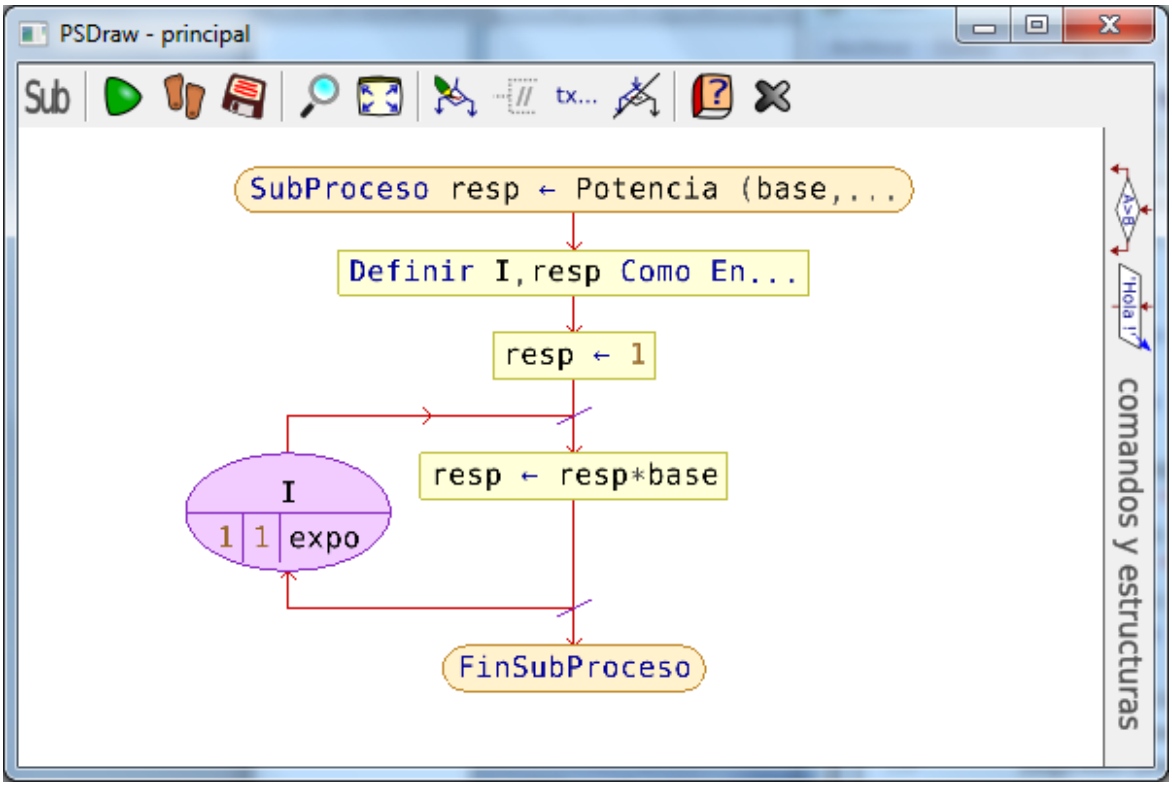


Diagrama de flujo del procedimiento potencia:



Si miramos este diagrama de flujo o el pseudocódigo, observamos que en la

función Potencia se declaran una variable ***I*** que es para el ciclo y la otra ***resp*** que es para el cálculo de la potencia, la cual usaremos como acumulador de la multiplicación de la potencia, y después de la variable ***resp***, a su vez después de la palabra clave subproceso, que es lo que nos devuelve el valor , y lo asigna en a la variable pot en el programa principal, cuando usamos la instrucción **pot<- potencia(base Por Referencia, expo Por Referencia);.**

En conclusión las funciones siempre nos retornaran un valor que es producto de uno o más cálculos, y se devuelve el valor de la variable que pusimos después de la palabra clave SubProceso.

Ejemplo de planilla (SubProcesos con y sin retorno)

Se ingresan el nombre, las ventas y la zona del empleado usando un procedimiento, luego se calcula la comisión en base a la zona de trabajo , ihss y total a pagar, luego se presentan los datos.

Nota:

- *se deberá de usar un subproceso con retorno para los cálculos y la presentación de los datos.*
- *Usar un subproceso con retorno para el cálculo del ihss.*
- *Usar un subproceso con retorno para el cálculo de la comisión.*

Subproceso de ingreso

En este subproceso sin retorno se ingresan los datos, validando que la zona solo sea A,B,C

Subproceso de cálculo

Se calcula la comisión e ihss usando los subprocesos sin retorno declarados anteriormente, luego el total a pagar, algo que debemos de notar es que las ventas y la zona se pasan como parámetros de valor y las demás ihss, comis y tp como parámetros de variable porque necesitamos modificar su valor

SubProceso presentar

Presentamos los cálculos y pasamos las variables como parámetros de valor, porque solo las necesitamos presentar

```
SubProceso vihss <- seguro(comis)
      Definir Vihss Como Real;
      Si comis >2400 Entonces
            vihss<-84;
      Sino
            vihss<-0.035*comis;
      FinSi
FinSubProceso

SubProceso vcomis <- comision(zona,ventas)
      Definir vcomis como Real;
      Segun zona Hacer
            'A' : vcomis<-0.05*ventas;
            'B' : vcomis<-0.06*ventas;
            'C' : vcomis<-0.09*ventas;
      FinSegun
FinSubProceso

SubProceso ingreso (nombre Por Referencia, zona Por Referencia, ventas
Por Referencia)
      Escribir "Ingresar  el  nombre  ";
      Leer  nombre;
      Escribir "Ventas  mensuales      ";
      Leer  ventas;
      Repetir
            Escribir  "Zona  A,B,C  ";
            Leer  zona;
      Hasta Que zona  ='B'  |  zona  ='C' |  zona ='A'
FinSubProceso

SubProceso calculos (zona, ventas, comis Por Referencia, ihss Por
Referencia, tp Por Referencia)
      comis<-comision(zona,ventas);
```

```
      ihss<-seguro(comis);
      tp<-comis-ihss;
FinSubProceso

Subproceso presentar (comis,ihss,tp)
      Escribir "Comisión ",comis;
      Escribir "Seguro Social ", ihss;
      Escribir "Total a pagar ", tp;
FinSubProceso

Proceso principal
      Definir ventas,comis,ihss,tp Como Reales;
      Definir nombre Como Cadena;
      Definir zona Como Caracter;
      Ingreso(nombre,zona,ventas);
      Calculos(zona,ventas,comis,ihss,tp);
      Presentar(comis,ihss,tp);
FinProceso
```

En este caso los subprocesos con retorno los declaremos antes de los subprocesos sin retorno solo porque estas se usaran en el subproceso sin retorno cálculos, y es más legible al momento de leer un programa, pero, a los efectos de la ejecución, PSeInt, no tiene en cuenta el orden del proceso y de los subprocesos.

Nota: *En sintaxis estricta, la variable de retorno debe ser declarada*

Dimensiones

Es una colección de datos del mismo tipo, que se almacenan en posiciones consecutivas de memoria y reciben un nombre común.
Y para referirse a un determinado elemento tendremos de acceder usando un índice para especificar la posición que queremos extraer o modificar su valor. Las dimensiones pueden ser:

1-Unidimensionales: solo tiene una solo dimensión una fila y una columna

2-Bidimensionales: tablas o matrices.

3-Multidimensionales: de 3 o más dimensiones.

Dimension de l Capacidad

Declaración:

Dimension <*Nombre de la dimension*> [<capacidad>*];
Definir <Nombre de la variable de la dimension> Como <tipo de la variable>;

Capacidad: es el tamaño de la dimension, es un número entero con el cual indicamos el número de elementos que queremos guardar con el mismo tipo.

Nombre de la variable: es el nombre con el cual vamos a ser referencia en el programa principal

Tipo de datos: es el tipo de datos que queremos que sea la colección, puede ser entero, real, cadena, carácter o una estructura.

Nota: *En sintaxis estricta, se debe definir siempre la variable antes o después de dimensionarla. A diferencia de otros lenguajes de programación, dimensionar una variable no implica declararla.*

Ejemplo:

Dimension numero [9];

Con esta declaración estamos creando una colección de 10 números enteros

3	5	7	8	3	6	9	2	45	67
0	1	2	3	4	5	6	7	8	9

Nota: *Al igual que en los lenguajes de programación reales, en sintaxis estricta, la base de la dimensión es 0, pero en sintaxis flexible o perfil flexible es base 1. Para utilizar dimensiones variables debemos habilitar la opción, permitir utilizar variables para dimensionar arreglos en las opciones del perfil.*

Siempre que nosotros queremos hacer referencia a uno de los elementos de la dimension, tendremos que indicar la posición, con un número entero que este dentro del rango.

*Seg*uidamente definimos el tipo de dimensión:

Definir numero Como Entero:

Si que queremos escribir el valor de posición 7 tendremos que escribir:

```
Escribir  numero[7]; // de esta indicamos escribir la posición 7
```

o

```
I<- 7 //asignamos un valor a una variable de tipo entero
Escribir  numero[ 0 ];  // luego usamos la variable I para indicar la posición
```

que queremos presentar.

// Si deseamos asignar valores a un elemento de la dimension lo podremos
// hacer:

```
Leer numero[2];
```
// indicamos directamente la posición que queremos leer

```
I<-6
```
// Asignamos un valor a una variable entero y luego la usamos

```
Leer numero[ i ];
```
// para indicar la lectura de elemento que queremos leer

Ejemplo

Ingresar 10 números a una dimension de 10 elementos y luego presentar los números.

En este programa tendremos que usar un ciclo que la variable I tome un valor de 0..9, para leer los valores o asignar valores a la dimension, luego usaremos otro ciclo para presentar los datos.
Cuando guardamos los datos en una dimension, sus valores son almacenados en la memoria y no se borrar después al leer el siguiente número, como en los

programas anteriores, cuando usábamos una variable para ingresar 10 números, pero la variable al final del ingreso solo guardaba el último número que se introdujo, ahora con los arreglos se guardan los 10 números en la memoria.

***Nota:** Si PSeInt está configurado para trabajar en base 0, se define una dimensión, y por ejemplo, se recorre una dimensión con un Para que comience en 1 y finalice con el número de elementos que se declaró a la dimensión, por ejemplo para llenar el vector de números, el último elemento ingresado no va a tener posición de memoria en la dimension ingresada. Esto lo podemos cambiar definiendo el Para desde base 0 y el número de la dimensión-1, personalizando el perfil o utilizando sintaxis flexible.*

```
// programa de ingreso de 10 números a una dimension
Proceso dimension_10
      Dimension numero[10];
      Definir numero Como Entero;
      Definir  I Como Entero;


      Para  I <- 0 Hasta 9 Con Paso 1 Hacer
          Escribir "Ingrese el número de la pos# ", I ,"....:";
          Leer numero[I];
      FinPara

      Para I <- 0 Hasta 9 Con Paso 1 Hacer
            Escribir numero[I];
      FinPara
FinProceso
```

Ejemplo

Ingresar el nombre del empleado en una dimension y el sueldo en otra dimension, luego de ingresar los datos determinar el ihss, el total a pagar para cada uno de los empleados.

En este programa se almacena el nombre del empleado y el sueldo en dos arreglos

diferentes el nombre en un arreglos de cadena y el sueldo en una dimension de tipo real, primero se ingresa los datos en la dimension luego se calculan los datos en otro ciclo con el fin de enfatizar que los arreglos guardan los datos en la memoria durante el programa funciona y los podemos usar después de ingresados los datos, lo que antes no podíamos hacer pues al ingresar el elemento 10 en la variable solo ese podíamos guardar, es por ello que los cálculos se hacían en el mismo ciclo.

```
Proceso dimension_empleado
  Dimension nombre[5];
  Definir nombre Como Cadena;
  Dimension sueldo[5];
  Definir sueldo como Entero;
  Definir ihss,tp Como Reales;
  Definir I Como Entero;

      Para  I <- 0  Hasta 4 Hacer
                  Escribir "Nombre del empleado [",i+1,"]..:";
                  Leer nombre[i];
                  Escribir "Sueldo del empleado ...:";
                  Leer sueldo[i];
      FinPara
      Para  I  <-  0  Hasta  4  Con Paso 1 Hacer
                  Si sueldo[i]>2400 Entonces
                        ihss<-84;
                  Sino
                        ihss<-0.035*sueldo[i];
                  FinSi
                  tp<-sueldo[i]-ihss;
                  Escribir "Nombre ...:", nombre[i];
                  Escribir "Sueldo  ...:",sueldo[i];
            Escribir "Ihss  ...:",ihss;
            Escribir "Total  pagar..:",tp;
      FinPara
FinProceso
```

Nota: *Para poder utilizar dimensiones de dimension variable es necesario habilitar perfil flexible en el editor.*

Uso de arreglos o dimensiones como parámetros en los subprocesos y funciones

En el ejemplo que, se presenta se demuestra el uso de los arreglos o dimensiones en los subprocesos y parámetros, el ejemplo muestra un subproceso sin retorno para el ingreso de datos a una dimension de 5 números enteros, luego se usa una función de mayor que nos devuelve el número de la dimension.

```
SubProceso nummayor <- mayor (num)
      Definir nummayor, i Como Enteros;
      nummayor<-0;
      Para i <-0 Hasta 4 Con Paso 1 Hacer
            Si num[i]>nummayor Entonces
                  nummayor<-num[i];
            FinSi
      FinPara
FinSubProceso

SubProceso ingreso (num)
      Definir i como Entero;
      Para  i <- 0 Hasta 4 Con Paso 1 Hacer
            Escribir "Ingresar un número ";
            Leer num[i];
      FinPara
FinSubProceso

Proceso Principal
      Dimension num[5];
      Definir num, max Como Enteros;
      Ingreso(num);
      Max<-mayor(num);
      Escribir "Mayor  ",  max;
FinProceso
```

Nota: *Por defecto, los arreglos siempre se pasan por referencia. No intentes pasarlo por valor o provocarías un error.*

Función mayor

En esta función se determina el número mayor comparando los números que se

ingresan, cuando se inicia la función nummayor vale cero pero supongamos que ingresamos en el arreglos 3-5-4-2-8

Cuando el elemento uno de la dimension se compara con 3, hay una nueva asignación para nummayor que es 3, cuando se compara con 5 el 3 es menor al 5 hay una nueva asignación a nummayor es 5, cuando se compara con 4 el 5 no es menor al cuatro, así que nummayor no se asigna ningún valor y se queda con el 5 ahora cuando se compara con 8 nummayor se le asigna el 8 porque el 5 es menor a 8.

Num	Nummayor
cuando num[0] es 3	Entonces vale 3
cuando num[0] es 5	Entonces vale 5
cuando num[0] es 4	No hay cambio y sigue valiendo 5
cuando num[0] es 2	No hay cambio y sigue valiendo 5
cuando num[0] es 8	Entonces vale 8

Dimension de II capacidad (de doble capacidad)

Declaración:

Dimension *<Nombre de la variable>* [*<Líneas>, <Columnas>*];

También se les denomina matrices o tablas. Una dimension bidimensional es una tabla que ahora tiene líneas y columnas, donde las líneas indican la primera dimensión y las columnas la segunda dimensión.

	0	1	2	3
0				
1				
2				
3				
4				

La tabla que se muestra nos representa un dimension de 2 dimensiones con 5 líneas (4 posiciones) y 4 columnas (3 posiciones), el código para declarar este dimension sería:

```
Dimension numero[4,3];
```

La referencia a un determinado elemento de la matriz, requiere el empleo de un primero subíndice que indica la fila y el segundo que indica la columna. Ambos subíndices deberán de ser de tipo entero.

Por ejemplo si quisiéramos guardar el valor de 30 en la línea 4 columna 3 el código en PSeInt sería:

```
Numero[3,2]<-30;
```

El siguiente ejemplo nos muestra como ingresar datos a una dimension de 5 líneas y 4 columnas para luego presentar los datos en la pantalla:

```
Proceso dimension_5_lineas
      Dimension numero[5,4];
      Definir numero Como Entero;
      Definir L, C Como Enteros;

      Para L <- 0 Hasta 4 Con Paso 1 Hacer
           Para  C <- 0  Hasta 3 Con Paso 1 Hacer
               Escribir "Número[", L , ",", C, "]";
               Leer numero[L,C];
           FinPara
      FinPara

      Limpiar pantalla;

      Para L <- 0 Hasta 4 con Paso 1 Hacer
           Para C <- 0 Hasta 3 Con Paso 1 Hacer
                  Escribir numero[L,C]," " Sin Bajar;
           FinPara
      Escribir "";
      FinPara
FinProceso
```

En este programa usamos dos variables enteras L que se usa para las líneas y C que se usa para las columnas, usamos ciclos anidados porque necesitas recorrer por cada línea, todas las columnas, esto sucede así:

Cuando la L tiene el valor de 0 la C toma el valor de 0 a 3 esto hace que se puede leer el elemento Numero [0,1], Numero [0,2], Numero [0,3], luego cuando la L tiene el valor de 2 entonces la l vuelve a iniciar de 0 a 3 haciendo lo mismo 4 veces que es el número de las líneas.

Suma de líneas y columnas de un dimension bidimensional

El programa que se presenta, ingresa los datos y los presenta usando un subproceso sin retorno.

```
SubProceso sum <- SumaLinea (numero, linea)
      Definir sum, C Como Enteros;
      sum<-0;
      Si linea>=0 | linea<=4 Entonces
            Para C<-0 Hasta 3 Con Paso 1 Hacer
                  sum<-sum + numero [linea,C];
            FinPara
      FinSi
FinSubProceso

SubProceso sum <- SumaColumna (numero, col)
      Definir sum, L Como Entero;
      sum<-0;
      Si col>=0 | col<=3 Entonces
            Para L <- 0 Hasta 4 Con Paso 1 Hacer
                  sum<-sum + numero [L,col];
            FinPara
      FinSi
FinSubProceso

SubProceso ingreso(numero)
      Definir L,C Como Enteros;
      Para  L <- 0 Hasta 4 Con Paso 1 Hacer
            Para  C <- 0 Hasta 3 Con Paso 1 Hacer
                  Escribir "Ingresar un número ..:";
                  Leer numero[L,C];
            FinPara
          Escribir "";
      FinPara
FinSubProceso

SubProceso presentar (numero)
      Definir L, C Como Enteros;
      Limpiar pantalla;
      Para L <- 0 Hasta 4 Con Paso 1 Hacer
          Para C <- 0 Hasta 3 Con Paso 1 Hacer
```

```
                Escribir numero[L,C]," " Sin Bajar;
            FinPara
            Escribir "";
        FinPara
FinSubProceso

Proceso principal
        Dimension numero[5,4];
        Definir numero Como Entero;
        Definir linea,col,sumaC,sumaL Como Enteros;
        Ingreso(numero);
        Presentar(numero);
        Escribir "Línea a sumar";
        Leer linea;
        Escribir  "Columna  a  sumar";
        Leer  col;
        sumaL<-sumaLinea(numero,linea);
        sumaC<-sumaColumna(numero,col);
        Escribir "Suma  de  la  columna  ", col,  "  es  ",  sumaC;
        Escribir "Suma  de  la  línea ", linea,  "  es  ",  sumaL;
FinProceso
```

Nota1: *Los arreglos del parámetro no se declaran en el subproceso, porque siempre son por referencia. De hacerse, aparecerá un cartel que dice No se debe redefinir el tipo de argumento.*

Nota2: *Los arreglos del parámetro, tanto cuando se pasan los arreglos a otro proceso/subproceso, como cuando se hace referencia al subproceso desde el proceso principal no se especifica su capacidad.*

Información teórica

Estructuras o registros

Nota: *La información de estructuras se toma como teórica. De momento, PSeInt no soporta estructuras o registros.*

Una estructura o registro es un dato estructurado, formado por elementos lógicamente relacionados, que pueden ser del mismo o de distintos tipos, a los que se les denomina campos. Los campos de una estructura podrían ser de los tipos previamente definidos por PSeInt (entero, real etc.) o bien por una estructura definido anteriormente

Ejemplo: demostración de estructuras

En este programa usaremos una estructura para guardar la información del alumno usando una estructura que se llama Alum.

Siempre que queremos acceder a una estructura se hace

Estructura.Variable;

Entonces si queremos acceder a nombre escribiríamos

Alum.nombre;

Alum porque así se llama la variable que es de tipo estructura re_alumno.

```
Registro regAlum
      Dimension nombre[30];
      Definir nombre Como Cadena;
      Dimension carrera[30];
      Definir carrera Como Cadena;
      Definir cuenta Como Entero;
FinRegistro

Proceso principal
      Definir Alum Como reg_alum;
      Escribir "El nombre del Alumno ";
      Leer Alum.nombre;
      Escribir "Cuenta  del  Alumno ";
      Leer  Alum.cuenta;
      Escribir "Carrera que estudia ";
      Leer Alum.carrera;
      Escribir "El  alumno ",  Alum.nombre;
      Escribir "Con  cuenta ", Alum.cuenta,  "  Estudia  ",
      Alum.carrera;
FinProceso
```

Ahora lo más importante es que podamos usar estructuras o registros como parámetros en los subprocesos con retorno y sin retorno para hacer más fácil el pasar información como parámetro.

Ejemplo estructuras o registros con subprocesos

Se desea elaborar un programa donde se ingrese el nombre del alumno , la nota acumulada, examen, nota final y observación, luego que se determine la nota final y observación.

Usaremos una estructura para guardar la información, un subproceso sin retorno para el ingreso de datos, otro para calcular la nota final y la observación (se usará una función para el cálculo de la observación).

Siempre debemos de tomar en cuanta cuales son los parámetros de variable y de valor, en este programa usa en los subprocesos ingreso y cálculo de variable y en presentar de valor porque no se modifica ninguna variable.

```
//  declaración de la estructura

Registro reg_alum
      Dimension nombre[30];
      Definir nombre Como Cadena;
      Definir na,ne,nf Como Reales;
      Dimension obs[10];
      Definir obs Como Cadena;
FinRegistro
SubProceso vobs <- observacion (nota)
      Definir vobs Como Cadena;
      Si nota>= 60 Entonces
            vobs<-"aprobado";
      Sino
            vobs<-"reprobado";
      FinSi
FinProceso
```

```
SubProceso ingreso(alum Por Referencia)
      Definir Alum Como reg_alum;
      Escribir  "Ingresar  el  nombre       ";
      Leer  Alum.nombre;
      Escribir  "Ingresar  la  nota  examen ";
      Leer  Alum.ne;
      Escribir  "Ingresar  la  nota  acumulada      ";
      Leer  Alum.na;
FinSubProceso

SubProceso calculo(alum Por Referencia)
      Alum.nf<-Alum.na+Alum.ne;
      Alum.obs<-observacion(Alum.nf);
FinSubProceso

SubProceso presentar(alum)
      Escribir "Nota Final ",Alum.nf;
      Escribir "Observación ",Alum.obs;
FinSubProceso

Proceso principal
      Definir I Como Entero;
      Para I<- 1 Hasta 5 Con Paso 1 Hacer
            ingreso(Alum);
            calculo(Alum);
            presentar(Alum);
      FinPara
FinProceso
```

Dimensiones con estructuras

Nota: *Información teórica*

Hasta ahora nuestros arreglos solo han guardado un solo datos ya sea real, entero cadena o caracter, cuando se quiere guardar más de un dato en una casilla de la dimension se declara una estructura y la dimension se declara que es del tipo estructura que declaramos.

Ejemplo:

```
Registro emple
       Definir codigo Como Entero;
       Dimension nombre[30];
       Definir nombre Como Cadena;
FinRegistro
Dimension empleado[5];
Definir empleado Como emple;
```

Código	Código	Código	Código	Código
Nombre	Nombre	Nombre	Nombre	Nombre
0	1	2	3	4

Declaramos el estructura empleado y luego declaramos la dimension que será de tipo empleado ahora para acceder a la dimension:

Lectura de datos

```
Escribir "Ingresar Nombre del Empleado ";
Leer emple[3].nombre;
Escribir "Ingresar el codigo de estructura ";
Leer emple[3].codigo;
```

Al momento de leer, se tiene que especificar la posición de la dimension que deseo leer emple(3).nombre nos indica que se leerá de posición 3 el nombre.

Escribir datos

```
Escribir "Nombre del Empleado ", emple[3].nombre;
Escribir "Código de estructura ", emple[3].codigo;
```

Igual que al leer los datos para escribir especificamos el elemento de la dimension, del cual queremos presentar los datos de la estructura

Ejemplo dimensiones con registro.

En este ejemplo declaramos la estructura, luego se declara la dimension de tipo estructura, se elabora un subproceso sin retorno para el ingreso de los datos de la dimension y otro para presentar los estructuras de la dimension.

Cuando declaramos `Dimension emple[5];` y después `Definir emple Como Empleado;` en el subproceso de ingreso nos referimos a que tenemos una dimension de 5 elementos que es de tipo empleado (la estructura) y que la variable se llama emple.

En ambos subprocesos se recorre la dimension y luego por cada una de las posiciones de la dimension se lee el nombre y el código.

```
Registro emple <- Empleado
      Definir codigo Como Entero;
      Dimension nombre[30];
      Definir nombre Como Cadena;
FinRegistro

SubProceso Ingreso (emple)
      Definir i Como Entero;
      Para  i <- 0 Hasta 4 Con Paso 1 Hacer
           Escribir "Ingresar Nombre del Empleado ";
           Leer emple.nombre;
           Escribir "Ingresar el código de estructura ";
           Leer emple.codigo;
      FinPara
FinSubProceso

SubProceso Presentar (emple)
      Definir i Como Entero;
      Limpiar Pantalla;
      Para i <- 0 Hasta 4 Con Paso 1 Hacer
           Escribir "Nombre del Empleado ",emple.nombre;
           Escribir "Código de estructura ", emple.codigo;
      FinPara
```

```
FinSubProceso

Proceso principal
      Ingreso(emple);
      Presentar(emple);
FinProceso
```

Ejemplo de dimensiones con registro.

En este ejemplo declaramos el estructura luego, se declara la dimension de tipo de tipo estructura alumno, luego usamos una función para determinar la observación, no se introduce todo el estructura porque solo se ocupa un dato, para determinar la observación, luego en el procedimiento de cálculo al momento de enviar la nota para usar la observación indicamos el elemento de la dimension y la parte de la estructura que queremos enviar que es la nota:

```
            alum[i].obs<-observacion(alum[i].nf);



// declaración del registro
Registro alum <- reg_alumno
      Dimension nombre[30];
      Definir nombre Como Caracter;
      Definir na,ne,nf Como Reales;
      Dimension obs[10];
      Definir obs Como Cadena;
FinRegistro

SubProceso vobs <- observación (nota)
      Dimension vobs[10];
      Definir vobs Como Cadena;
      Si nota>= 60 Entonces
            vobs<-"aprobado";
      Sino
            vobs<-"reprobado;
      FinSi
FinSubroceso
```

```
SubProceso  ingreso(alum Por Referencia)
      Definir  i Como Entero;
      Para  i <- 0 Hasta 4 Con Paso 1 Hacer
            Escribir "Ingresar  el  nombre        ";
            Leer  alum[i].nombre;
            Escribir "Ingresar  la  nota  examen  ";
            Leer  alum[i].ne;
            Escribir "Ingresar  la  nota  acumulada      ";
            Leer alum[i].na;
      FinPara
FinSubProceso

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

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

Manejo De Archivos De Texto

Nota: *Información teórica*

Hasta esta parte, todos los resultados de los programas se borran de la memoria al terminar el programa, en este capítulo aprenderemos de forma teórica como guardaríamos la información en un archivo de texto para su posterior utilización.

Sintaxis

Declarar un tipo archivo

Declarar un tipo archivo secuencial es necesario para , declarar variable de este tipo ejemplo :

```
Tipo  Arch  Es  Archivo  Secuencial;
```

Abrir un archivo

Sintaxis

Abrir nombre_archivo como variable [para lectura, escritura]

ejemplo :

```
Abrir  "empleados.txt"  Como  Archemple  Para  Lectura;
```

Descripción

Esta instrucción sirve para abrir el archivo. Las operaciones permitidas para el archivo son lectura, escritura o ambas. En la sintaxis variable se refiere a variable de tipo archivo que se usará para referenciar el archivo.

Cerrar un archivo

Sintaxis

Cerrar variable de tipo archivo

Ejemplo :

```
Cerrar  archemple;
```

Descripción

Esta instrucción sirve para cerrar un archivo. Variable

Leer de un archivo

Sintaxis

Leer variable_archivo, variable_datos

ejemplo :

```
Leer  archemple,emple.nombre;
```

Descripción

Esta instrucción lea una variable desde un archivo. La primera variable de la instrucción debe ser de tipo archivo, la segunda puede ser de cualquier tipo, eso dependerá del tipo de archivo.

Escribir en un archivo

Sintaxis

Escribir variable_archivo, variable_datos;

ejemplo:

```
Escribir  archemple,  emple.nombre;
```

Descripción

Esta instrucción escribe una variable en un archivo. La primera variable de la instrucción debe ser de tipo archivo, la segunda puede ser de cualquier tipo, eso dependerá del tipo de archivo.

Ejemplo Ingreso de datos a un archivo secuencial (texto).

Lo primero que tenemos que hacer es crear con Windows un archivo de texto, con el notepad, y lo salvamos con el nombre de empleados, en el mismo directorio donde salvaremos el programa de ingreso de datos.

Declaramos el tipo de archivo secuencial

```
Tipo  Arch  es  archivo  secuencial;
```

luego el estructura que usaremos para ingresar los datos

```
Estructura  Empleado
     Dimension nombre[50];
     Definir nombre Como Cadena;
     Definir sueldo Como Real;
     Definir sexo como Caracter;
FinEstructura
```

luego declaramos la variable para manejar el archivo de texto, que de tipo arch y la variable de tipo estructura

```
Definir Empleado Como emple;
Definir ArchEmple Como Arch;
Definir resp como Caracter;
```

Luego en el programa lo primero que se hace es abrir el archivo para escritura, luego se piden los datos y se salvar en el archivo , al final se cierra el archivo de texto, ahora si nosotros queremos saber si guardo los datos , podremos abrir empleados con el notepad y veremos los datos que se salvaron en el archivo.

```
Tipo  Arch  Es  Archivo  Secuencial;
Estructura emple <- Empleado
     Dimension nombre[50];
     Definir nombre Como Cadena;
     Definir sueldo Como Real;
     Definir sexo como Caracter;
FinEstructura
Proceso principal
     Definir emple Como Empleado;
     Definir ArchEmple Como Arch;
     Definir resp como Caracater;
     Abrir "empleados.txt" Como archemple Para Escritura;
     Repetir
               Escribir "Nombre del emnpleado..:";
```

```
                Leer emple.nombre;
                Escribir "Sueldo del empleado...:";
                Leer emple.sueldo;
                Escribir "Sexo ..:";
                Leer emple.sexo;
                Escribir archemple, emple.nombre;
                Escribir archemple, emple.sueldo;
                Escribir archemple, emple.sexo;
                Escribir "Desea Continuar ..:";
                Leer resp;
            Hasta Que resp="S" | resp="N";
      Hasta Que resp='N';
      Cerrar archemple;
FinProceso
```

Ejemplo Listar el contenido de un archivo secuencial (texto).

Se declara el tipo del archivo, el estructura y las variables para usar el estructura y el archivo de texto, luego se abre el archivo para lectura y se hace un ciclo mientras no sea fin de archivo, esto se logra con la función FDA que nos devuelve verdadero cuando se encuentra al final del archivo y falso cuando no lo está.
Se usa la instrucción Leer, para recuperar los valores que se guardaron en el archivo de texto, luego usando un procedimiento se escriben los valores de la estructura en la pantalla

```
Definir Tipo  Arch  Como Archivo  Secuencial;

Estructura emple <- Empleado
      Dimension nombre[50];
      Definir sueldo Como Real;
      Definir nombre Como Cadena;
      Definir sexo Como Caracter;
      Definir emple Como Empleado;
FinEstructura

SubProceso presentar(emple)
      Definir Detener Como Caracter;
      Escribir "Nombre del empleado ...:",emple.nombre;
```

```
      Escribir "";
      Escribir "Sueldo....:",emple.sueldo,
      Escribir "";
      Escribir "Sexo......:",emple.sexo;
      Escribir "";
      Leer  detener;
FinSubProceso

Proceso principal
      Definir ArchEmple Como Arch;
      Abrir  "empleados.txt"  Como  archemple  Para  Lectura;
      Mientras ~fda(archemple) Hacer
            Leer archemple,emple.nombre;
            Leer archemple,emple.sueldo;
            Leer archemple,emple.sexo;
            presentar(emple);
      FinMientras;
      Cerrar archemple;
FinProceso
```

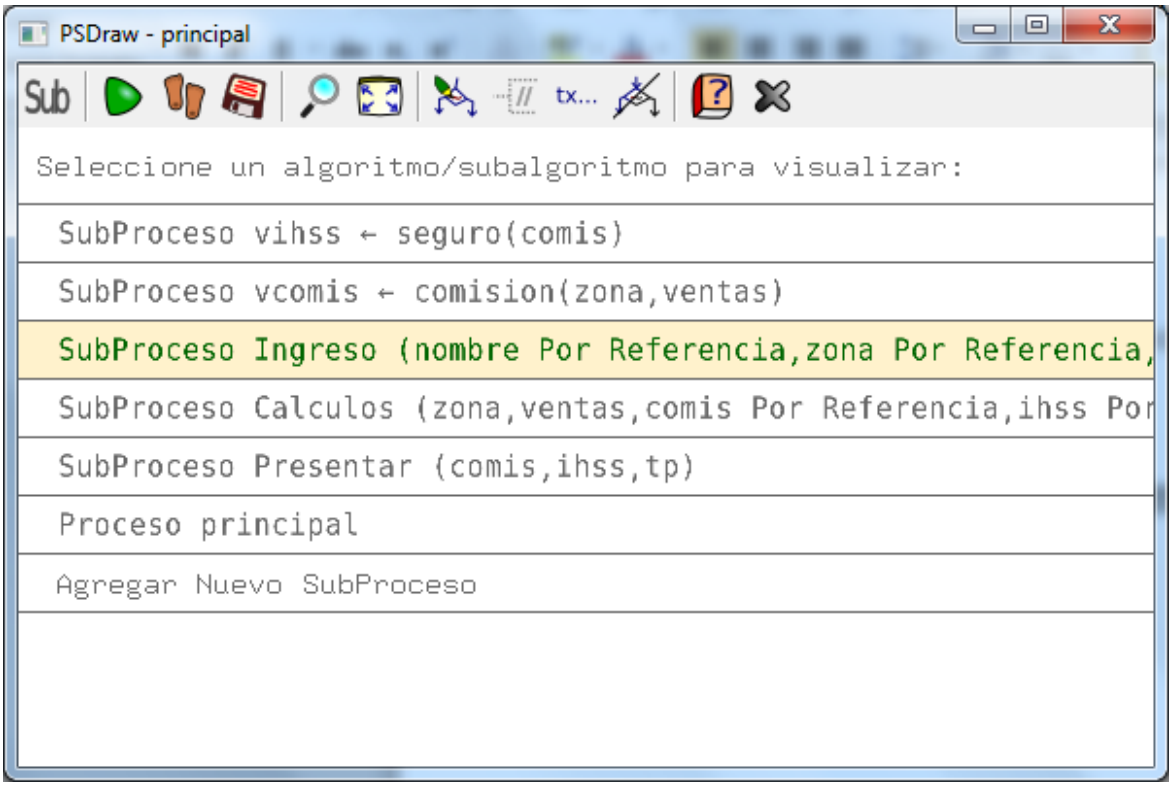
Anexo:

Editar diagramas de flujo

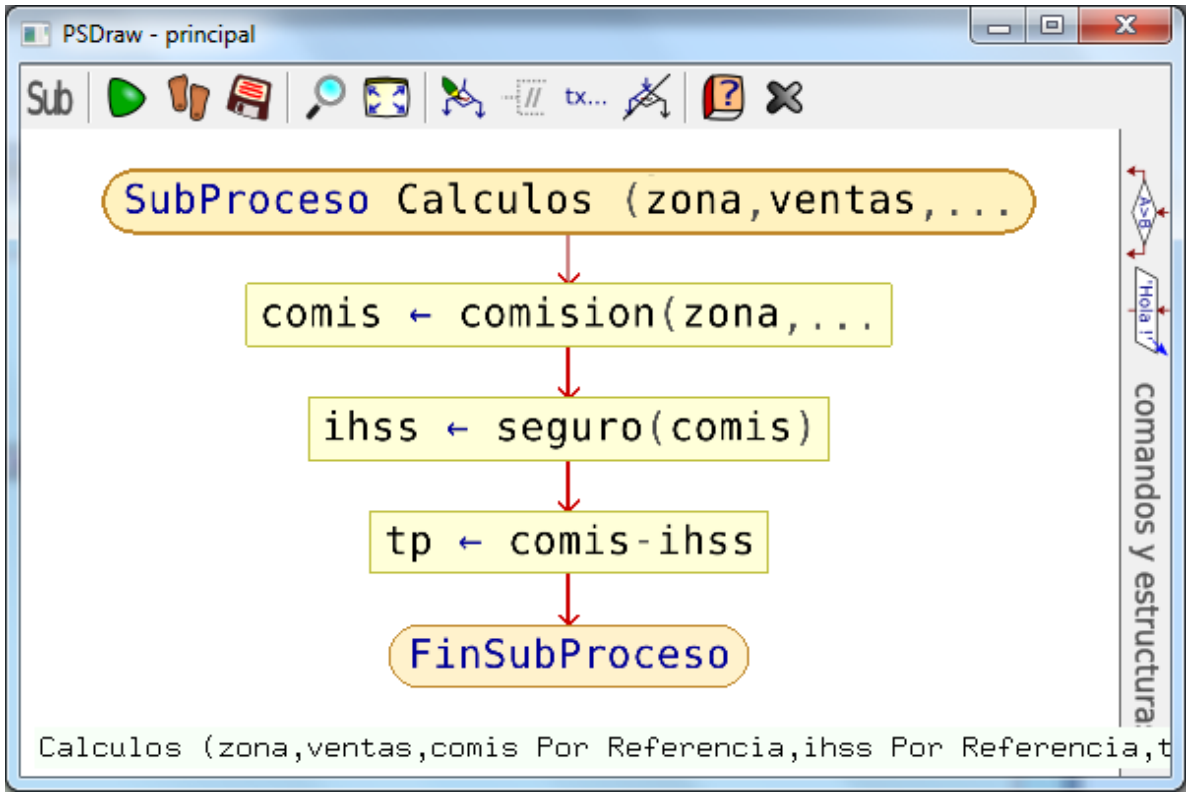
PSeInt permite editar el diagrama de flujo, luego editar los cambios, para que pueda ser ejecutado desde pseudocódigo.

Accedemos al editor de diagramas de flujo yendo a Archivo → Editar diagramas de flujo:

Elegimos un subproceso y hacemos clic en uno de ellos



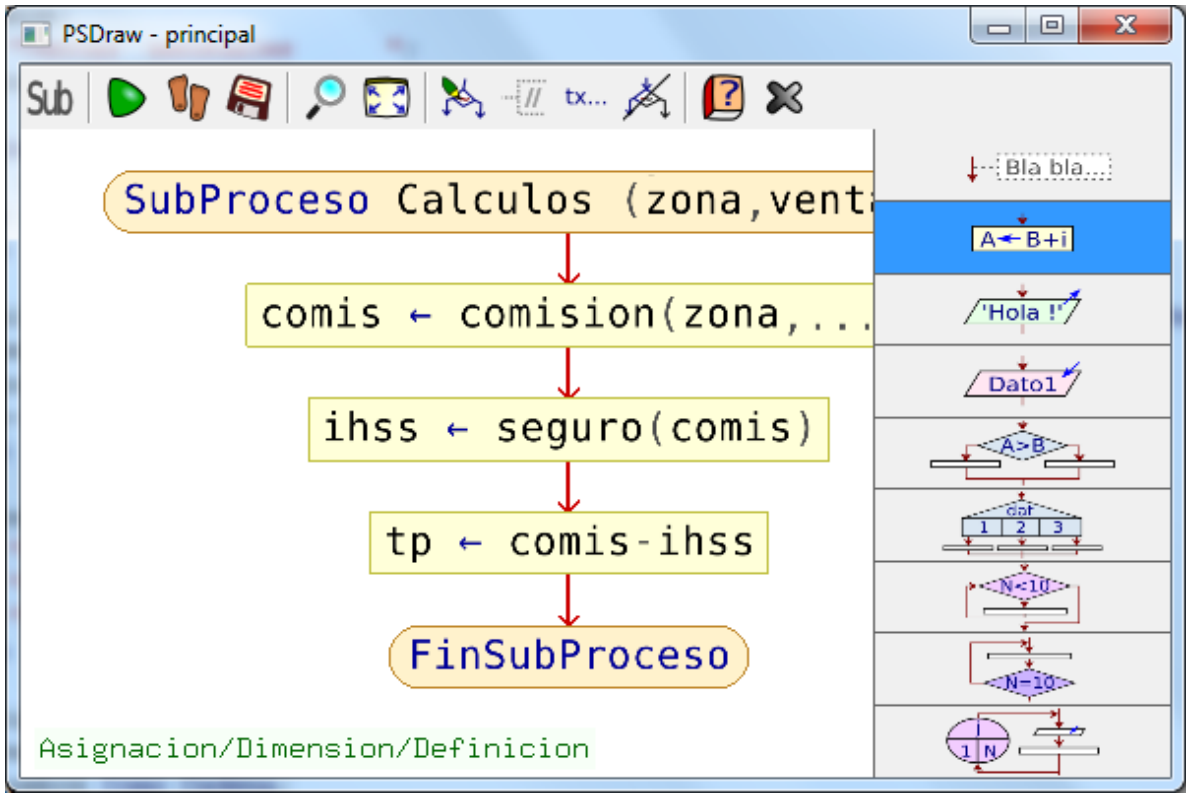
Hacemos clic en el proceso principal o cualquiera de los subprocesos, en este caso el subproceso Ingreso.



Se nos presenta una pantalla mostrando el diagrama de flujo correspondiente al subproceso que estamos ejecutando.

A la derecha encontramos una pestaña que aparecen dos iconos y al costado el título comandos y estructuras

Pasamos el mouse sobre la misma.



Se nos presenta un dibujo con las estructuras usadas, y al costado izquierdo inferior aparece su nombre.

Si queremos añadir un nuevo bloque al diagrama de flujo, lo que hacemos es cliquear en un bloque y sin soltar el botón izquierdo del mouse arrastrarlo hasta el diagrama de flujo. Para fijar el bloque, presionamos la tecla escape.

En las sentencias escribir, el texto se debe poner entre comillas.

Guardar cambios

Para guardar los cambios, vamos al botón que se encuentra al costado izquierdo superior y hacemos clic en guardar cambios.

No se ejecutan diagramas de flujo que no sean guardados.

Nota: *También se pueden crear diagramas sin necesidad de escribir su pseudocódigo correspondiente.*

Nota: *Por la forma de trabajar del intérprete de diagrama de flujo, si se guardan los*

cambios desde el editor de diagrama de flujo, hay modificaciones en el pseudocódigo, por ejemplo, pasado de comillas a apóstrofes, etc. Estos errores se pueden ir resolviendo a medida que salgan nuevas versiones de PSeInt.

Des instalar PSeInt

PSeInt dispone de un des instalador, que se accede desde agregar o quitar programas. Se desinstala como cualquier otro programa.

Abrir el código fuente

En estos blog se explica cómo lo que debemos hacer para abrir el código fuente del programa: