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

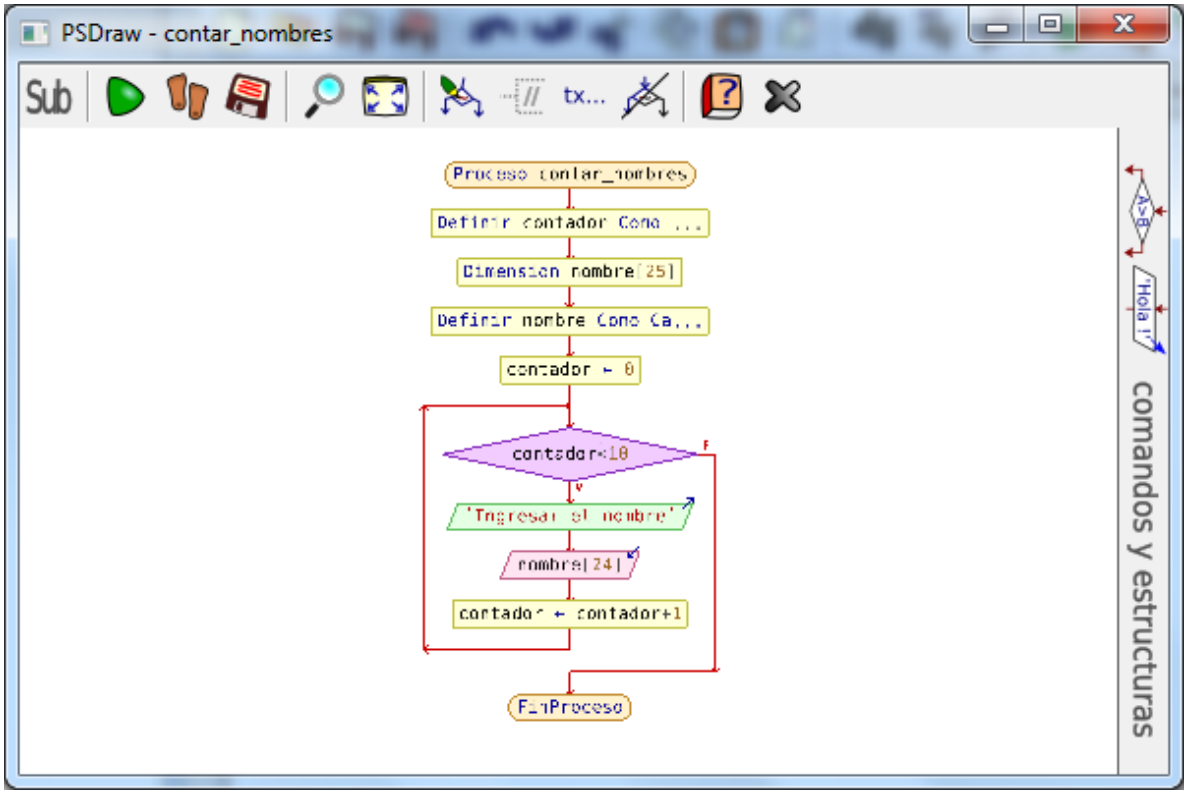
Mientras condición Hacer
    instrucciones;
FinMientras

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
```

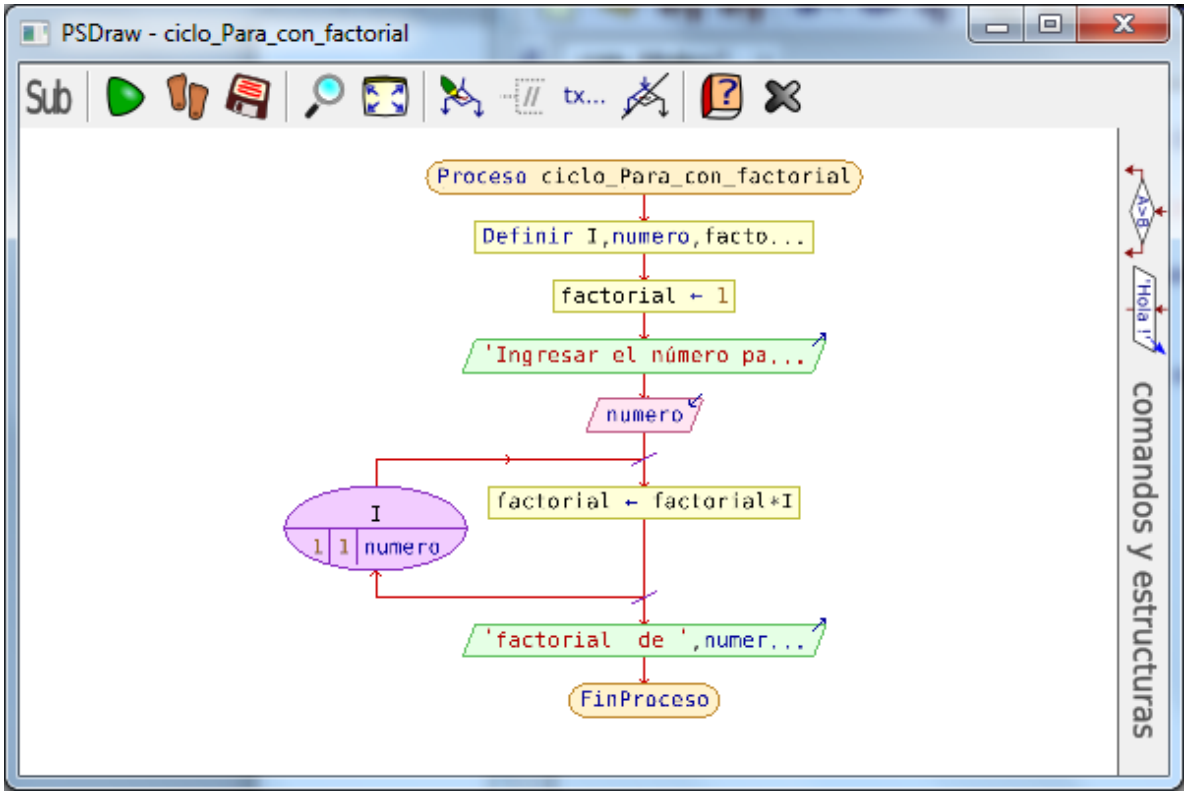
Ahora el programa se vuelve más dinámico, nosotros podemos indicar el número de veces que se repetirá el ciclo, usando una variable entera para indicar el final del ciclo.

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
```

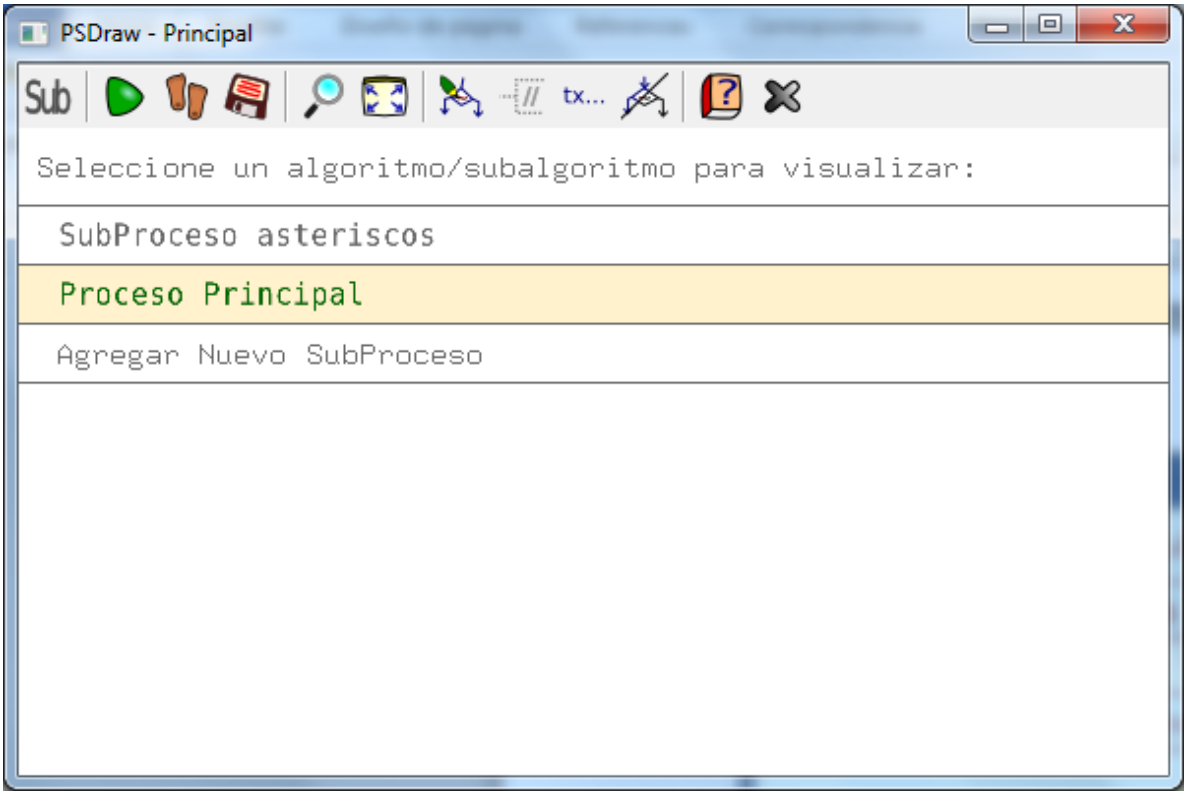
En este programa usamos un subproceso (función -palabra equivalente, PSeInt también la toma-, o procedimiento) para escribir 5 asteriscos, si no lo hubiéramos hecho de esta manera donde se encuentra la instrucción asteriscos; tendríamos que escribir el ciclo, y lo haríamos dos veces, de la forma en que 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:

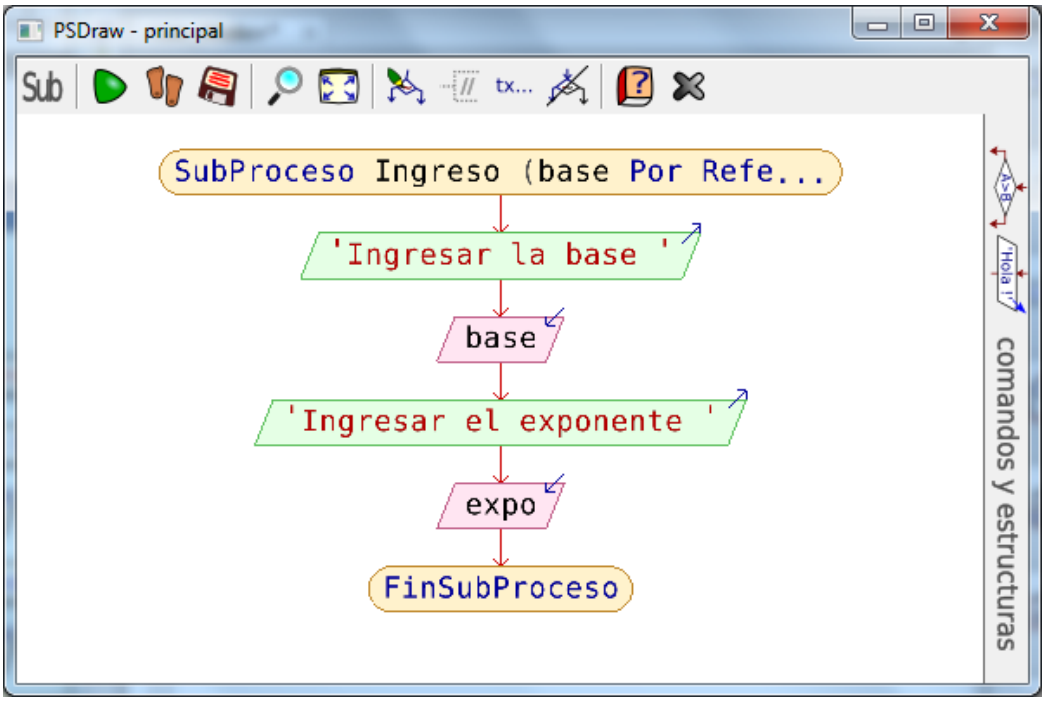
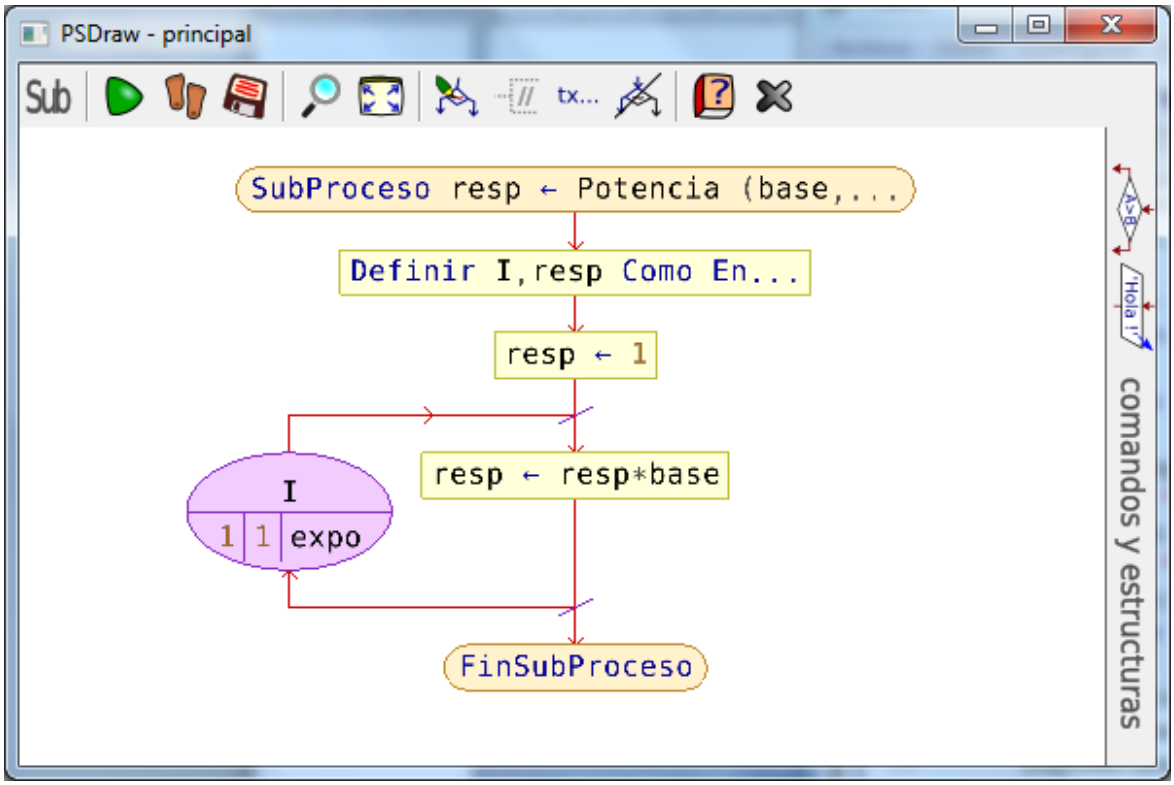


Diagrama de flujo del procedimiento potencia:



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.