1. Detalle de los Comandos:

Este es un resumen puede servir como guía rápida para consultas, para mas detalles remitirse a la Ayuda de PSeInt. Observar que cada línea que termina una sentencia debe tener punto y coma (;).

1.1. Inicio y Fin de un Proceso

Para comenzar a escribir un programa es necesario iniciar con el comando Proceso seguido del nombre del **proceso** (Ej: Proceso Sumador), el nombre no debe llevar espacios y es recomendable que sea lo más representativo a lo que hace el código. Al finalizar se debe colocar el comando **FinProceso**, esto cierra el código del programa.

Ejemplo:

Proceso Sumador

..

..

FinProceso

1.2. Asignación

Esta instrucción permite almacenar el valor en una variable, ya sea resultado de una operación o bien el valor de otra variable

El comando es **<variable> = <expresión>**; primero evalúa la expresión de la derecha y luego asigna el resultado a la variable de la izquierda. Es importante que el tipo de variable que involucra a la expresión coincida con el tipo de variable de asignación.

C = A + B; J = 8;

1.3. Definición de Variable

La instrucción Definir nos permite explicitar el tipo de una o más variables que se utilizarán en el programa. Antes de usar una variable en el proceso, es necesaria indicar que va a existir, esto es definirla y a la vez es necesario indicar de que tipo va a ser, para ello luego de los nombres de las variables, se coloca el indicador **como**, y luego el tipo de variable Los tipos de variables permitidos son : **REAL ENTERO LOGICO CARACTER**

Ejemplo:

Definir A **como ENTERO**;

1.4. Operadores

Este pseudolenguaje dispone de un conjunto básico de operadores que pueden ser utilizados para la construcción de expresiones más o menos complejas. Las siguientes tablas exhiben la totalidad de los operadores de este lenguaje reducido:

Operador	*Significado*	*Ejemplo*
Relacionales		
>	Mayor que	3>2
<	Menor que	'ABC'<'abc'
=	Igual que	4=3
<=	Menor o igual que	'a'<='b'
>=	Mayor o igual que	4>=5
<>	Distinto que	'a'<>'b'
Lógicos		
& ó Y	Conjunción (y).	(7>4) & (2=1) //falso
| ó O	Disyunción (o).	(1=1 | 2=1) //verdadero
~ ó NO	Negación (no).	~(2<5) //falso
Algebraicos		
+	Suma	total <- cant1 + cant2
-	Resta	stock <- disp - venta
*	Multiplicación	area <- base * altura
/	División	porc <- 100 * parte / total
^	Potenciación	sup <- 3.41 * radio ^ 2
% ó MOD	Módulo (resto de la división entera)	resto <- num MOD div

Figura 1: Operadores

La jerarquía de los operadores matemáticos es igual a la del álgebra, aunque puede alterarse mediante el uso de paréntesis. Para el caso de los operadores & y |, la evaluación se realiza en cortocircuito. Esto significa que si dos expresiones están unidas por el operador ? y la primera se evalúa como Falso, o están unidas por el operador | y la primera se evalúa como Verdadero, la segunda no se evalúa ya que no altera el resultado.

1.5. Ingreso de Datos

La instrucción **Leer <variable>**; permite ingresar datos cuando se ejecute el programa, se puede ingresar una o más variables en el mismo comando, estos datos se ingresan desde el teclado en el caso de una computadora.

Para ingresar datos a una variable es necesario que la misma esté definida con nombre y tipo de variable.

Ejemplo:

Definir A **como ENTERO**;

Leer A;

En la representación de Leer , se puede ver un Paralelogramo con una flechita hacia adentro.

1.6. Salida de Datos

La instrucción **Escribir <variable>** nos permite mostrar en pantalla un valor, ya sea el contenido de una variable, el resultado directo de una expresión o bien un texto.

El comando **podría ser:**

Escribir <variable>;

Escribir <expresión>;

Escribir "TEXTO";

O una combinación de estas separadas por **coma (,)** donde se evalúa cada una de las expresiones y muestra en pantalla el contenido, en el caso de mostrar un texto siempre debe estar entre comillas dobles.

Ejemplo:

Definir A **como ENTERO**;

Leer A; **Escribir** "La Variable ", A, " es un entero";

En la representación de Escribir , se puede ver un Paralelogramo con una flechita hacia afuera.

1.7. Condición Si-Entonces (if)

La ejecución de estas sentencias depende del valor de la condición lógica. Al evaluar la expresión lógica, devuelve un Verdadero o un Falso. Si la expresión lógica es Verdadera se realizan las instrucciones que están bajo la indicación. La sintaxis sería:

Si <expresión> **Entonces**

Veamos un Ejemplo:

```
1 Algoritmo if
2 definir a, b como entero;
3 leer a;
4 leer b;
5 Si a>b Entonces
6         Escribir "El primer número ", a, " es mayor que el Segundo ", b
  , " ingresado";
7 FinSi FinAlgoritmo
```

Notar que la acción se toma cuando se sale por Verdadero.

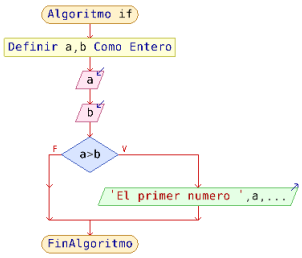

Figura 2: if

1.8. Condición Si-Entonces Sino (if - else)

Existe una alternativa que es que el if aparte de tener una acción por el Verdadero, TAMBIEN , tenga una alternativa por el Falso, (PERO SIEMPRE debe estar al menos una por el verdadero).

Si la condición es Falsa, se ejecutan las instrucciones que están bajo la indicación **Sino**, esta indicación puedo no estar, en este caso no se ejecutará nada.

Ejemplo:

Veamos como sería la representación de esto.

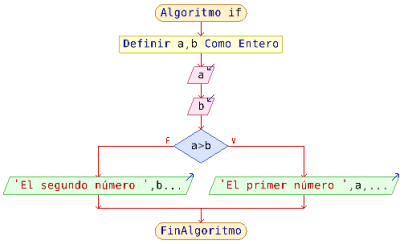

Figura 3: if - else

```
1  Algoritmo if
2  definir a, b como entero;
3  leer a;
4  leer b;
5  Si a>b Entonces
6          Escribir "El primer número ", a, " es mayor que el Segundo ", b
   , " ingresado";
7  Sino
8          Escribir "El segundo número ", b, " es mayor que el primer número", a ,
9  FinSi
10 FinAlgoritmo
```

1.9. Selección Múltiple (switch)

La ejecución de esta sentencia depende del valor de una variable, que va seguido del comando **Segun**, esto evalúa la variable numérica y finaliza con **FinSegun**.

Esta instrucción permite ejecutar opcionalmente varias acciones posibles, al ejecutarse, se evalúa el contenido de la variable y se ejecuta la secuencia de instrucciones asociada con dicho valor que se detallan luego del comando Hacer.

Es importante destacar que NO HAY OPERACIÓN DE NINGÚN TIPO , dentro luego de **Segun** , si no que solamente evalúa una variable.

Cada opción está formada por uno o más números separados por comas, dos puntos y una secuencia de instrucciones.

Si una opción incluye varios números, la secuencia de instrucciones asociada se debe ejecutar cuando el valor de la variable es uno de esos números.

Se debe agregar una opción final, denominada **De Otro Modo**, cuya secuencia de instrucciones asociada se ejecutará sólo si el valor almacenado en la variable no coincide con ninguna de las opciones anteriores.

Este sentencia se utiliza normalmente para la creación de Menúes.

```
1  Algoritmo switch
2  Definir a Como Caracter;
3  Escribir "Ingrese una opción: ";
4  Escribir "1-Primer Alternativa";
5  Escribir "2-Segunda Alternativa";
6  Escribir "3-Tercer Alternativa";
7  Escribir "4-Salir o Finalizar";
8  Leer a;
9          Segun(a)
10                 '1': Escribir "Ud. eligió la primer opción";
11                 '2': Escribir "Ud. eligió la segunda opción";
12                 '3': Escribir "Ud. eligió la tercera opción";
13                 'S': Escribir "Ud. eligió salir ";
14                 De otro Modo: Escribir "Ud. eligió una opción no valida";
15         FinSegun
16 FinAlgoritmo
```

El equivalente en Diagrama de flujo sería:

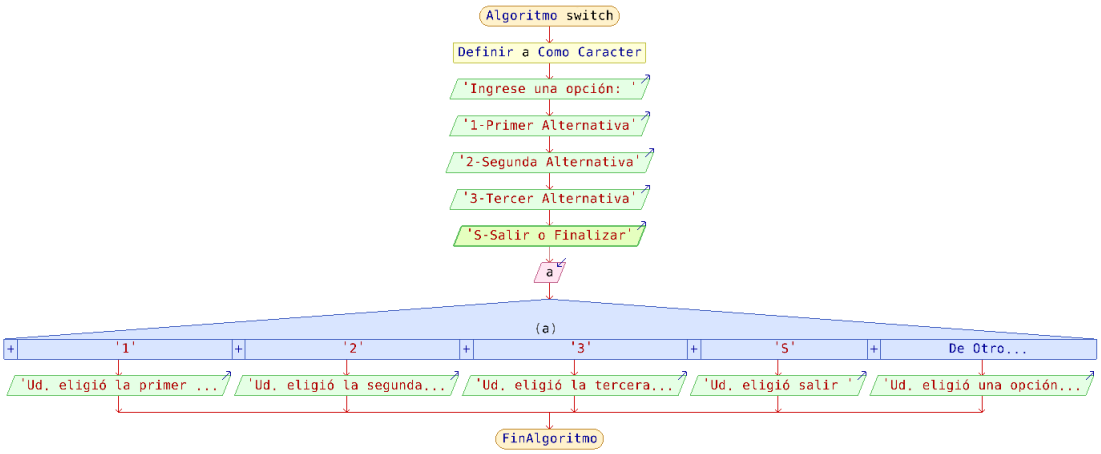

Figura 4: switch

1.10. Lazo Para

La instrucción **Para** ejecuta una secuencia de instrucciones un **número determinado de veces**, definido por el valor seguido del comando **Hasta**.

Al ingresar al bloque, la variable de control recibe el valor inicial y se ejecuta la secuencia de instrucciones que forma el cuerpo bucle. Luego se incrementa la variable de control en un paso de unidades determinado y se evalúa la condición de fin. Se repite el ciclo hasta que de falso la condición.

Si se omite la cláusula **Con Paso <paso>**, la variable de control se incrementará en 1.

Ejemplo:

Definir i como ENTERO;
Escribir “Los primeros números pares son: “;
Para i<-0 Hasta 10 Con Paso 2 Hacer Escribir i, “; “;
FinPara

Veamos otro ejemplo:

```
1 Algoritmo do_while
2 Definir i , x como ENTERO;
3 Escribir "Ingrese un numero: ";
4 Leer x;
5 Escribir "Motraremos los primeros pares menores a ",x ;
6         Para i=0 Hasta x Con Paso 2  Hacer
7         Escribir i;
8         FinPara
9 FinAlgoritmo
```

El equivalente en Diagrama de flujo sería:

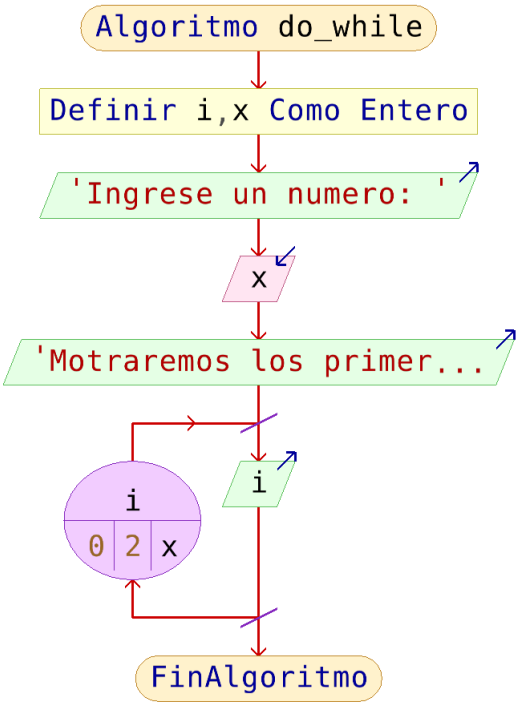

Figura 5: for

Existen varias alternativas que se pueden usar en la sentencia **Para**, se recomienda ver la ayuda del PSeInt. Notar que se evalúa i<x , pero solamente se escribe **Hasta x**, no va la expresión de comparación.

1.11. Lazo Mientras (while)

La instrucción Mientras ejecuta una secuencia de instrucciones seguidas del comando **Hacer** mientras una condición sea verdadera. Al ejecutarse esta instrucción, primero se evalúa la condición. Si la condición resulta **verdadera**, se ejecuta una vez la secuencia de instrucciones que forman el cuerpo del ciclo.

Luego se vuelve a evaluar la condición y, si es verdadera, la ejecución se repite. Caso contrario de que sea **falsa, sale del ciclo** y el programa continúa abajo del comando **FinMientras**.

Tener en cuenta que las instrucciones del cuerpo del ciclo pueden no ejecutarse nunca, si al evaluar por primera vez la condición resulta ser falsa. Por otro lado, si la condición siempre es verdadera, al ejecutar esta instrucción se produce un ciclo infinito. A fin de evitarlo, las instrucciones del cuerpo del bucle deben contener alguna instrucción que modifique la o las variables involucradas en la condición, de modo que ésta sea falsificada en algún momento y así finalice la ejecución del ciclo.

Veamos un ejemplo:

La representación sería:

```
1 Algoritmo while
2 Definir A, L como ENTERO;
3 L=0;
4         Mientras L<250 Hacer
5         Leer A; L<-L+A;
6         FinMientras
7 FinAlgoritmo
```

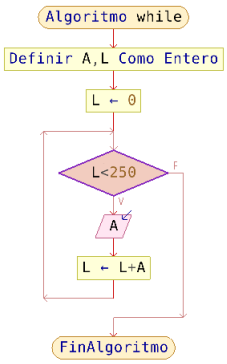

Figura 6: while

1.12. Lazo Repetir Mientras (do- while)

La sentencia Repetir entra a un bucle, donde se ejecuta las instrucciones que le proceden, y luego evalúa una condición determinada por el comando **Mientras Que <condición>**; si esta condición es **verdadera** el ciclo sigue ejecutándose, **sale** del mismo una vez que la condición evaluada es **falsa**.

Se debe tener en cuenta que el cuerpo del ciclo se ejecuta **al menos una vez**, ya que primero ejecuta las instrucciones y luego evalúa la condición de fin. Al igual que el Lazo Mientras, hay que modificar en el cuerpo del bucle la o las variables que intervienen en la condición, de manera tal que en algún momento obtengamos un falso, y no quedemos en un bucle infinito.

Veamos un ejemplo:

```
1 Algoritmo do_while
2 Definir suma,x Como Entero;
3 suma = 0;
4         Repetir
5         Escribir "Ingrese un número del acumulador: ";
6         Leer x;
7         suma=suma +x;
8         Mientras Que suma <100 ;
9 FinAlgoritmo
```

La representación sería:

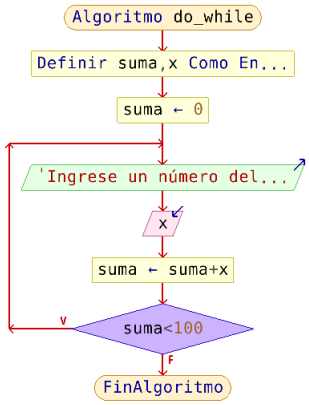

Figura 7: do-while

1.13. Declaración de Arreglos

Para declarar un arreglo es necesario primero declarar la variable con el comando **Definir** indicando también el **tipo de dato** (indicados en 1.3).

Luego en otra línea se introduce el comando **Dimension** seguido del **nombre del arreglo** ya declarado como variable y la dimensión entre corchetes.

Se pueden definir arreglos de una dimensión o de varias dimensiones.

La sintaxis es:

Dimension <variable>**[<elementos>,. . . ,<elementos>]**;

Ejemplo:

Definir Vec, Mat **como REAL Dimension** Vec[25], Mat[5,5];

1.14. Borrado de Pantalla

Se puede utilizar una instrucción para limpiar la pantalla en la que se ejecuta el código y posiciona al cursor en la esquina superior izquierda.

La sintaxis es **Borrar Pantalla**; y es útil cuando se necesita refrescar la pantalla.

1.15. Espera o Pausa

Sirve para pausar el programa, continuando con el ingreso de cualquier tecla apretada por el usuario, o bien se puede colocar la instrucción de esperar tantos segundos o milisegundos. Las sintaxis se pueden observar en el siguiente ejemplo.

Ejemplo:

Esperar Tecla;
Esperar 4 Segundos;
Esperar 115 Milisegundos;