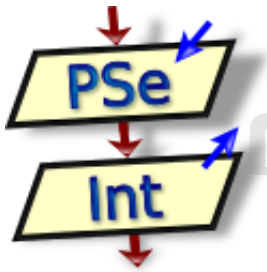


...una invitación a entrar en el maravilloso mundo de la programación...

Advertencia: Las sintaxis que aquí se presenta corresponde a la sintaxis en la cual se basó originalmente el proyecto. Desde hace varias versiones estoy buscando que en la palabra pseudocódigo pese más la parte pseudo que la de código. Para ello se han agregado posibles sinónimos, sintaxis alternativas o versiones incompletas de algunas construcciones en pos de desrigidizar el pseudolenguaje utilizado, siempre y cuando no se caiga en ninguna ambigüedad respecto a su interpretación. Actualmente, el intérprete puede configurarse para obligar al alumno a respetar completamente la sintaxis original, aceptar las variaciones, o algunos grados intermedios (ver perfiles para una comparación entre dos versiones de un mismo algoritmo).

El Pseudo-código

Las características de este pseudolenguaje fueron propuestas en 2001 por el responsable de la asignatura Fundamentos de Programación de la carrera de Ingeniería Informática de la FICH-UNL. Las premisas son:

- Sintaxis sencilla
- Manejo de las estructuras básicas de control
- Solo 3 tipos de datos básicos: numérico, caracter /cadenas de caracteres y lógico (verdadero-falso).
- Estructuras de datos: arreglos

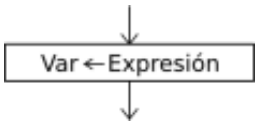
Todo algoritmo en pseudocógido tiene la siguiente estructura general:

```
Proceso SinTitulo
     accion 1;
     accion 1;
          .
          .
          .
     accion n;
FinProceso
```

Comienza con la palabra clave *Proceso* seguida del nombre del programa, luego le sigue una secuencia de instrucciones y finaliza con la palabra *FinProceso*. Una secuencia de instrucciones es una lista de una o más instrucciones, cada una terminada en punto y coma.

Las acciones incluyen operaciones de entrada y salida, asignaciones de variables, condicionales si-entonces o de selección múltiple y/o lazos mientras, repetir o para.

Asignación



La instrucción de asignación permite almacenar una valor en una variable.

```
<variable> <- <expresión> ;
```

Al ejecutarse la asignación, primero se evalúa la expresión de la derecha y luego se asigna el resultado a la variable de la izquierda. El tipo de la variable y el de la expresión deben coincidir.

Entradas

La instrucción Leer permite ingresar información desde el ambiente.

```
Leer <variable1> , <variable2> , ... ,
<variableN> ;
```

Esta instrucción lee N valores desde el ambiente (en este caso el teclado) y los asigna a las N variables mencionadas. Pueden incluirse una o más variables, por lo tanto el comando leerá uno o más valores.

Salidas

La instrucción Escribir permite mostrar valores al ambiente.

```
Escribir <expr1> , <expr2> , ... , <exprN> ;
```

Esta instrucción imprime al ambiente (en este caso en la pantalla) los valores obtenidos de evaluar N expresiones. Dado que puede incluir una o más expresiones, mostrará uno o más valores.

Dimensionamiento

La instrucción Dimension permite definir un arreglo, indicando sus dimensiones.

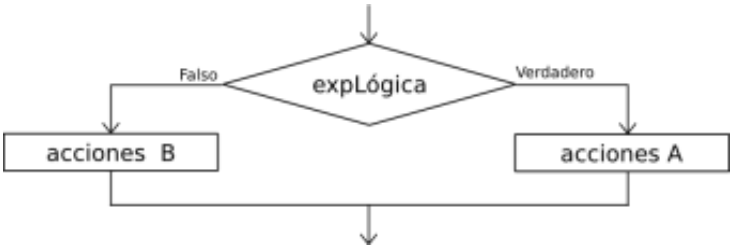
```
Dimesion <identificador> (<maxl>,...,<maxN>);
```

Esta instrucción define un arreglo con el nombre indicado en <indentificador> y N dimensiones. Los N parámetros indican la cantidad de dimensiones y el valor máximo de cada una de ellas. La cantidad de dimensiones puede ser una o más, y la máxima cantidad de elementos debe ser una expresión numérica positiva.
Se pueden definir más de un arreglo en una misma instrucción, separándolos con una coma (,).

```
Dimension <ident1> (<max11>,...,<max1N>),..., <identM> (<maxM1>,...,<maxMN>)
```

Es importante notar que es necesario definir un arreglo antes de utilizarlo.

Condicional Si-Entonces

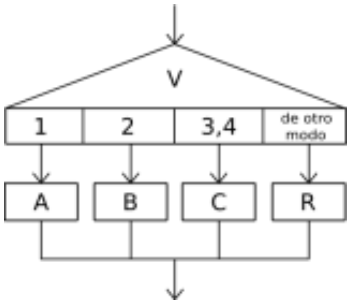


La secuencia de instrucciones ejecutadas por la instrucción Si-Entonces-Sino depende del valor de una condición lógica.

```
Si <condición>
      Entonces
           <instrucciones>
      Sino
            <instrucciones>
FinSi
```

Al ejecutarse esta instrucción, se evalúa la condición y se ejecutan las instrucciones que correspondan: las instrucciones que le siguen al *Entonces* si la condición es verdadera, o las instrucciones que le siguen al *Sino* si la condición es falsa. La condición debe ser una expresión lógica, que al ser evaluada retorna *Verdadero* o *Falso*.
La cláusula *Entonces* debe aparecer siempre, pero la cláusla *Sino* puede no estar. En ese caso, si la condición es falsa no se ejecuta ninguna instrucción y la ejecución del programa continúa con la instrucción siguiente.

Selección Multiple

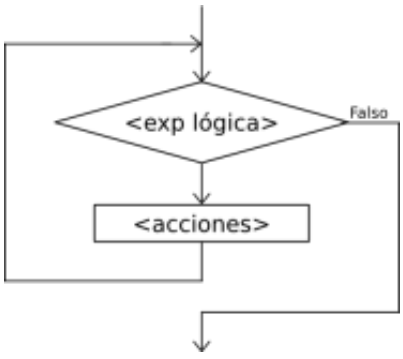


La secuencia de instrucciones ejecutada por una instrucción *Segun* depende del valor de una variable numérica.

```
Segun <variable> Hacer
      <número1>: <instrucciones>
      <número2>,<número3>: <instrucciones>
      <...>
      De Otro Modo: <instrucciones>
FinSegun
```

Esta instrucción permite ejecutar opcionalmente varias acciones posibles, dependiendo del valor almacenado en una variable de tipo numérico. Al ejecutarse, se evalúa el contenido de la variable y se ejecuta la secuencia de instrucciones asociada con dicho valor.
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.
Opcionalmente, se puede 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.

Lazos Mientras



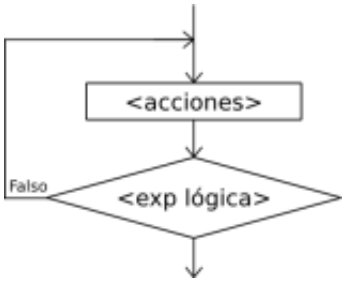
La instrucción *Mientras* ejecuta una secuencia de instrucciones mientras una condición sea verdadera.

```
Mientras <condición> Hacer
      <instrucciones>
FinMientras
```

Al ejecutarse esta instrucción, la condición es evaluada. Si la condición resulta

verdadera, se ejecuta una vez la secuencia de instrucciones que forman el cuerpo del ciclo. Al finalizar la ejecución del cuerpo del ciclo se vuelve a evaluar la condición y, si es verdadera, la ejecución se repite. Estos pasos se repiten mientras la condición sea verdadera.
Note que las instrucciones del cuerpo del ciclo pueden no ejecutarse nunca, si al evaluar por primera vez la condición resulta ser falsa.
Si la condición siempre es verdadera, al ejecutar esta instrucción se produce un ciclo infinito. A fin de evitarlo, las instrucciones del cuerpo del ciclo 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.

Lazos Repetir

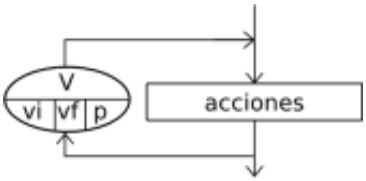


La instrucción *Repetir-Hasta Que* ejecuta una secuencia de instrucciones hasta que la condición sea verdadera.

```
Repetir
      <instrucciones>
Hasta Que <condición>
```

Al ejecutarse esta instrucción, la secuencia de instrucciones que forma el cuerpo del ciclo se ejecuta una vez y luego se evalúa la condición. Si la condición es falsa, el cuerpo del ciclo se ejecuta nuevamente y se vuelve a evaluar la condición. Esto se repite hasta que la condición sea verdadera.
Note que, dado que la condición se evalúa al final, las instrucciones del cuerpo del ciclo serán ejecutadas al menos una vez.
Además, a fin de evitar ciclos infinitos, el cuerpo del ciclo debe contener alguna instrucción que modifique la o las variables involucradas en la condición de modo que en algún momento la condición sea verdadera y se finalice la ejecución del ciclo.

Lazos Para



La instrucción *Para* ejecuta una secuencia de instrucciones un número determinado de veces.

```
Para <variable> <- <inicial> Hasta <final> ( Con Paso <paso> ) Hacer
```

```
        <instrucciones>
    FinPara
```

Al ingresar al bloque, la variable <variable> recibe el valor <inicial> y se ejecuta la secuencia de instrucciones que forma el cuerpo del ciclo. Luego se incrementa la variable <variable> en <paso> unidades y se evalúa si el valor almacenado en <variable> superó al valor <final>. Si esto es falso se repite hasta que <variable> supere a <final>. Si se omite la cláusula *Con Paso* <paso>, la variable <variable> se incrementará en 1.

Operadores y Funciones

Este pseudolenguaje dispone de un conjunto básico de operadores y funciones 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
Logicos		
& 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

La jerarquía de los operadores matemáticos es igual a la del álgebra, aunque puede alterarse mediante el uso de paréntesis.

A continuación se listan las funciones integradas disponibles:

Función	*Significado*
RC(X)	Raíz Cuadrada de X
ABS(X)	Valor Absoluto de X
LN(X)	Logaritmo Natural de X
EXP(X)	Función Exponencial de X
SEN(X)	Seno de X
COS(X)	Coseno de X
ATAN(X)	Arcotangente de X
TRUNC(X)	Parte entera de X
REDON(X)	Entero más cercano a X
AZAR(X)	Entero aleatorio entre 0 y X-1

Algunas Observaciones

- Se pueden introducir comentarios luego de una instrucción, o en líneas separadas, mediante el uso de la doble barra (*//*). Todo lo que precede a *//*, hasta el fin de la línea, no será tomado en cuenta al interpretar el algoritmo.
- Notese que no puede haber instrucciones fuera del programa, aunque si comentarios.
- Las estructuras no secuenciales pueden anidarse. Es decir, pueden contener otras adentro, pero la estructura contenida debe comenzar y finalizar dentro de la contenedora.
- Los identificadores, o nombres de variables, deben constar sólo de letras y números, comenzando siempre con una letra, y no pueden ser palabras reservadas (como para, mientras, y, no, etc...)
- Las constantes de tipo carácter se escriben entre comillas (").
- En las constantes numéricas, el punto (.) es el separador decimal.
- Las constantes lógicas son *Verdadero* y *Falso*.

Ejemplos

Ver sección ejemplos...