Reglas para pseudocodigo

Utliza relgas basadas en el lenguaje C y C++

Asignación: <— ej: a <— b "**a** se le asigna **b**"

Comentarios: //`Comentario`

Expresiones Booleanas: mayor que: **>** menor que: **<** mayor igual que: **>=** menor igual que: **<=** igualdad: **=** desigualdad: **!=** and: **^** or: **v** not: **¬** valor nulo: **NULL**

Expresiones matemáticas: Suma: **+** Resta: **-** Multiplicación: **asterisco "*"** division: **/** modulo/resto de una division: **mod** potencia: **^**

Tipos de datos:

entero: **int** flotante: **float** cadena de texto: **string** caracter: **char** arreglo: **array[range] of** `Tipo` matriz: **array[rangoFilas][rangoColumnas] of** `Tipo` Clase de elemento generico: **Clase**`<Element>` Punteros: **pointer to** `Clase/Tipo`

Declaración: `Tipo`: `Variable` ej: `int: valor1`, `float: valor2`, `string: cadena1`, `Lista<Element>: result`, `array[1..N] of int: valores` `pointer to int: punteroEntero`

Para parametros de una funcion: entrada por valor: **Tipo: Variable** entrada por referencia: **Ref Tipo: Variable**

Condicional simple: if **condicion** then `bloque1` endif

Condicional anidado: if **condición** then `bloque1` else `bloque2` endif

Ciclos: for `Variable`<— `valor` to `valor final` do `bloque1` endfor

while **condicion** do `bloque` endwhile

do `bloque` while **condicion**

Funciones y procedimientos: func **Nombre de la funcion**(**parametros**):**salida** Var `bloque de declaracion de variables` Begin `bloque de funcion` return `variable` endfunc

proc **Nombre de la funcion**(**parametros**) Var `bloque de declaracion de variables` Begin `bloque de funcion` endfunc

Llamada de funciones: `nombre funcion`(**parametros de entrada**)

Llamada de metodos: `nombre objeto`.`nombre funcion`(**parametros de entrada**) *Desreferencia de un punter:* `nombrePuntero`->`metodo`

Creacion de memoria: para crear nodos se usa la funcion `new` que devuelve el puntero del espacio en memoria, se utiliza ese puntero para insertarle la informacion usando setters

ej: creación de un nodo en algún método de la clase lista (revisar referencia) [[Clases LPC Algoritmos]]

```
Var
  Element: e
  pointer to Nodo<Element>: pointerNew
Begin
  pointerNew <- new(Nodo<Element>)
  pointerNew->.setInfo(e)
```

Declaracion de Clases: Los metodos se declaran unicamente en la clase, la implementación se hace afuera. Class `Nombre Clase` Attributes: **nivel de acceso**: `bloque de declaracion de atributos/variables` Methods: **nivel de acceso**: `bloque de declaracion de metodos` endclass

Implementacion de metodos de clases: func/proc `Nombre Clase`::`Nombre Funcion`(`Parametros`)**salida si es funcion** Var `bloque variables` Begin `bloque codigo` endfunc/proc