Reglas para pseudocodigo

Utiliza reglas basadas en el lenguaje C y C++.

Nota para exportar a PDF

- Para evitar que líneas distintas se peguen en el PDF, usa siempre:
  - listas con `-` o numeradas,
  - bloques de código con triple comilla invertida,
  - línea en blanco entre secciones.

Asignación

- Operador: `←` (en tus apuntes: `<—`)
- Ejemplo: `a <— b` ("a se le asigna b").

Comentarios

- Sintaxis: `// Comentario`

Expresiones booleanas

- Mayor que: `>`
- Menor que: `<`
- Mayor o igual que: `>=`
- Menor o igual que: `<=`
- Igualdad: `=`
- Desigualdad: `!=`
- AND: `^`
- OR: `v`
- NOT: `¬`
- Valor nulo: `NULL`

Expresiones matemáticas

- Suma: `+`
- Resta: `-`
- Multiplicación: `*`
- División: `/`
- Módulo (resto): `mod`
- Potencia: `^`

Tipos de datos

- Entero: `int`
- Flotante: `float`
- Cadena: `string`
- Carácter: `char`
- Arreglo: `array[rango] of Tipo`
- Matriz: `array[rangoFilas][rangoColumnas] of Tipo`
- Clase genérica: `Clase<Element>`
- Punteros: `pointer to Clase/Tipo`

Declaración

- Forma general: `Tipo: variable`

Ejemplos:

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

Parámetros:

- Entrada por valor: `Tipo: variable`
- Entrada por referencia: `Ref Tipo: variable`

Estructuras de control

Condicional simple

```
if condicion then
  bloque
endif
```

Condicional anidado

```
if condicion then
  bloque1
else
  bloque2
endif
```

Ciclos

```
for i <- inicio to fin do
  bloque
endfor
```

```
while condicion do
  bloque
endwhile
```

```
do
  bloque
while condicion
```

Funciones y procedimientos

Función

```
func NombreFuncion(parametros): TipoSalida
  Var
    bloqueVariables
  Begin
    bloqueCodigo
    return valor
endfunc
```

Procedimiento

```
proc NombreProcedimiento(parametros)
  Var
    bloqueVariables
  Begin
    bloqueCodigo
endproc
```

Llamadas

- Llamada de función: `nombreFuncion(parametros)`
- Llamada de método: `objeto.metodo(parametros)`
- Desreferencia de puntero: `puntero->metodo()`

Creación de memoria (nodos)

Para crear nodos se usa `new`, que devuelve un puntero al espacio en memoria.

Ejemplo (ver también [[Clases LPC Algoritmos]]):

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

Declaración de clases

Los métodos se declaran en la clase y se implementan afuera.

```
Class NombreClase
  Attributes:
    NivelAcceso:
      bloqueAtributos
  Methods:
    NivelAcceso:
      bloqueMetodos
endclass
```

Implementación de métodos de clase

```
func/proc NombreClase::NombreMetodo(parametros)[:TipoSalida]
  Var
    bloqueVariables
  Begin
    bloqueCodigo
endfunc/endproc
```