Sintaxis / convenciones de Sintaxis

|  | Convención para  Pseudocódigo | Convención para  PHP |
| --- | --- | --- |
| **Tipos de Datos** | En el Pseudocódigo el tipo de dato se declara estáticamente, antes de utilizar la variable debemos declarar su tipo.  Otros lenguajes que declaran los tipos explícitamente son JAVA, C#, C  **ENTERO**  **BOOLEAN**  **FLOAT** (= DECIMAL. Se utiliza el punto como separador de decimales)  **STRING** (= TEXTO. Para delimitar la cadena de caracteres usaremos comillas dobles: " ") | En PHP, el tipo de dato de una variable se establece dinámicamente cuando se ejecuta el programa, y depende del valor que se le asigne a la variable  **int**  **boolean** (valores TRUE o FALSE)  **float** (se utiliza el punto como separador de decimales)  **string** (para delimitar la cadena de caracteres usaremos comillas dobles: " ")  (http://php.net/manual/es/language.types.php) |
| **Variable** | Identificador: utilizaremos notación [lowerCamelCase](https://es.wikipedia.org/wiki/CamelCase), letras mayúsculas para inicializar las palabras, excepto la primera letra. No utilizaremos guión bajo (\_).  Los identificadores deben ser nombres significativos del problema que estamos resolviendo.  ejemplos:  ladoMenor  ladoMayor  perimetroRectangulo  **Declaración de variable** significa: indicar el nombre y tipo de dato de la variable.  **Inicializar variable** significa: asignar el primer valor a la variable. A partir de la inicialización decimos que la variable está definida. | Identificador (con la misma convención [lowerCamelCase](https://es.wikipedia.org/wiki/CamelCase) que el Pseudocodigo) al que antepondremos el signo $  ejemplos:  $ladoMenor  $ladoMayor  $perimetroRectangulo |
| **comentarios** | Comentarios de varias líneas:  **(\*comentario**  **varias lineas \*)** | Comentarios de varias líneas:  **/\* comentario**  **varias lineas \*/**  **//comentario 1 linea** |
| **Instrucción de Asignación** | nombreVariable 🡨 **Expresion**  Donde **Expresion** puede ser:   * un valor * una variable con valor * combinación de operandos y operadores * funciones que retornen valores (tema de unidad 4)   Ejemplo:  ladoMenor 🡨 20 | $nombreVariable = **Expresion ;**  Donde **Expresion** puede ser:   * un valor * una variable con valor * combinación de operandos y operadores * funciones que retornen valores (tema de unidad 4)   Ejemplo:  $ladoMenor = 20 ;  (observación: en php las instrucciones terminan con ; ) |
| **Instrucción de Entrada** | Ejemplo:  LEER (ladoMenor) | Ejemplo:  $ladoMenor = trim(fgets(STDIN)) ;  (observación: en php las instrucciones terminan con ; ) |
| **Instrucción de Salida** | Ejemplo:  **ESCRIBIR**(**“** El perímetro del rectángulo es **“ ,** perimetroRectangulo)  (observación:  Para la operación de concatenación de string se utiliza coma ",") | Ejemplo:  **echo** **“** El perímetro del rectángulos es **“ .** perimetroRectangulo;  (observación: en php las instrucciones terminan con ;  Para la operación de concatenación de string se utiliza punto ".") |
| **Separación de instrucciones** | Una instrucción por renglón, identando adecuadamente las instrucciones. | Utilizando punto y coma "**;**"  (Por prolijidad siempre conviene 1 instrucción por renglón, identando adecuadamente las instrucciones) |
| **Estructura de control Secuencial**  **(se ejecuta una instrucción debajo de la otra)** | PROGRAMA NombrePrograma  (\*descripción del programa ¿Qué problema resuelve?\*)  Declaración de valiables: Tipo nombresVariables  instruccion1  ...  instruccionN  FIN PROGRAMA  Ejemplo: | **<?php**  /\*nombrePrograma\*/  /\* Declaración de variables: Tipo nombresVariables \*/  instruccion1;  ...  instruccionN;  **?>**  Obs: En lenguajes como PHP, donde el tipo de las variables es dinámico, no hay declaración de tipos. Por eso comentaremos la declaración o no la incluiremos. Podemos decir que es el tipo que esperamos tenga una variable por los valores que serán asignados a dichas variables. |
| **Operadores**  **para expresiones** | | **Concepto** | **Pseudocódigo** | **php** | | --- | --- | --- | | Valor absoluto: |a| | abs(a) | abs(a) | | Potencia: ab | potencia(a,b) | pow(a,b) | |  |  |  |   Uso del operador ternario:  **PROGRAMA** MayoriaEdad  (\*determinar si una persona es mayor de edad\*)  Boolean esMayor, Entero edad, **String** nombre,mensaje  ESCRIBIR( "Ingrese su nombre" )  LEER(nombre)  ESCRIBIR( "Ingrese su edad" )  LEER(edad)  esMayor <- (edad >= 18)  mensaje <- SI **esMayor** ENTONCES **nombre," es mayor de edad"** SINO **nombre," es menor de edad "**  ESCRIBIR (mensaje)  **FIN PROGRAMA** | |
| **Modularización: Funciones** | (\*\*  \*Descripcion del módulo sin retorno (¿qué hace?)  \*)  **MODULO** nombre(tipo1 pf1, tipo2 pf2,..,tipoN pfN) RETORNO ∅  |instruccion1  |...  | instruccionN  **FIN** **MODULO**  (\*\*  \* Descripcion del módulo con retorno (¿qué hace?)  \*)  **MODULO** nombre(tipo1 pf1, tipo2 pf2,..,tipoN pfN) RETORNO tipo  |instruccion1  |...  | instruccionN  |**RETORNO**( E)  **FIN** **MODULO**  (obs.: tipo puede ser boolean, integer, float, string, etc.) | /\*\*  \*Descripcion del modulo sin retorno (¿qué hace?)  \* @param tipo $pf1  \* @param tipo2 $pf2  \* ...  \* @param tipo $pfN  \*/  **function** nombre($pf1, $pf2,..,$pfN){  instruccion1;  ...  instruccionN;  }  /\*\*  \*Descripcion del modulo con retorno (¿qué hace?)  \* @param tipo1 $pf1  \* @param tipo2 $pf2  \* ...  \* @param tipoN $pfN  \* @return tipo  \*/  **function** nombre($pf1, $pf2,..,$pfN){  instruccion1;  ...  instruccionN;  **return** E;  } |
| **Alternativa:**  **Instrucción IF** | **SI** (condición) **ENTONCES**  |instruccion1  |instruccionN  **FIN SI**  **SI** (condición) **ENTONCES**  |instruccionA1  |instruccionAN  **SINO**  |instruccionB1  |instruccionBN  **FIN SI**  **SI** (condición) **ENTONCES**  |instruccionA1  |instruccionAN  **OTRO-SI** (condiciónB) **ENTONCES**  |instruccionB1  |instruccionBN  **OTRO-SI** (condiciónM) **ENTONCES**  |instruccionM1  |instruccionMN  **FIN SI**  **SI** (condición) **ENTONCES**  |instruccionA1  |instruccionAN  **OTRO-SI** (condiciónB) **ENTONCES**  |instruccionB1  |instruccionBN  **OTRO-SI** (condiciónM) **ENTONCES**  |instruccionM1  |instruccionMN  **SINO**  |instruccionN1  |instruccionNN  **FIN SI** | **if** (condición)**{**  Instrucción1;  InstrucciónN;  **}**  **if** (condición)**{**  InstrucciónA1;  InstrucciónAN;  **}** **else{**  InstrucciónB1;  InstrucciónBN;  **}**  **if** (condiciónA)**{**  InstrucciónA1;  InstrucciónAN;  **} elseif** (condiciónB)**{**  InstrucciónB1;  InstrucciónBN;  **} elseif** (condiciónM)**{**  InstrucciónM1;  InstrucciónMN;  **}**  **if** (condiciónA)**{**  InstrucciónA1;  InstrucciónAN;  **}** **elseif** (condiciónB**) {**  InstrucciónB1;  InstrucciónBN;  **} elseif** (condiciónM**) {**  InstrucciónM1;  InstrucciónMN;  **} else{**  InstrucciónN1;  InstrucciónNN;  **}** |

| **Alternativa:**  **Instrucción IF anidada** | **Las siguientes instrucciones son a modo de ejemplo, hay tanta combinaciones como problemas a resolver**  **SI** (condicion) **ENTONCES**  **SI** (condición) **ENTONCES**  instruccion\_a1  instruccion\_aN  **FIN SI**  instruccionN  **FIN SI**  **SI** (condicion) **ENTONCES**  **SI** (condición) **ENTONCES**  instruccion\_a1  instruccion\_aN  **FIN SI**  instruccionN  **SINO**  instruccionB1  instruccionBN  **FIN SI**  **SI** (condicion) **ENTONCES**  **SI** (condición) **ENTONCES**  instruccion\_a1  instruccion\_aN  **FIN SI**  instruccionN  **SINO**  instruccionB1  **SI** (condición) **ENTONCES**  instruccion\_b1  instruccion\_bN  **SINO**  instruccion\_c1  instruccion\_cN  **FIN SI**  **FIN SI**  **SI** (condicion1) **ENTONCES**  **SI** (condicion\_a) **ENTONCES**  instruccion\_a1  instruccion\_aN  **FIN SI**  instruccionN  **OTRO-SI** (condicion2) **ENTONCES**  instruccion\_c1  **SI** (condición) **ENTONCES**  instruccion\_b1  instruccion\_bN  **SINO**  instruccion\_c11  instruccion\_c1N  **FIN SI**  instruccion\_c3  **SINO**  instruccionB1  instruccionBN  **FIN SI** | **Las siguientes instrucciones son a modo de ejemplo, hay tanta combinaciones como problemas a resolver**  **if** (condición)**{**  **if** (condición)**{**  Instrucción\_a1;  Instrucción\_aN;  **}**  InstrucciónN;  **}**  **if** (condición)**{**  **if** (condición)**{**  Instrucción\_a1;  Instrucción\_aN;  **}**  InstrucciónN;  **}** **else {**  InstrucciónB1;  InstrucciónBN;  **}**  **if** (condición)**{**  **if** (condición)**{**  Instrucción\_a1;  Instrucción\_aN;  **}**  InstrucciónN;  **}** **else {**  instrucciónB1;  **if** (condición)**{**  Instrucción\_a1;  Instrucción\_aN;  **}else {**  Instrucción\_b1;  Instrucción\_bN;  **}**  **}**  **if** (condición)**{**  **if** (condición)**{**  Instrucción\_a1;  Instrucción\_aN;  **}**  InstrucciónN;  **}** **elseif**( condicion2 ) **{**  instrucción\_c1;  **if** (condición)**{**  Instrucción\_b1;  Instrucción\_bN;  **}else {**  Instrucción\_c11;  Instrucción\_c1N;  **}**  instrucción\_c3;  **}**  **else {**  InstrucciónB1;  InstrucciónBN;  **}** |
| --- | --- | --- |
| **Repetitiva:**  **Instrucción WHILE** | **MIENTRAS (condición) HACER**  **|instruccion1**  **|instruccionN**  **FIN MIENTRAS**  ***Repite 0 o más ciclos*** | **while (condición) {**  **instruccion1;**  **instruccionN;**  **}** |
| **Repetitiva:**  **Instrucción DO..WHILE** | **REPETIR**  **|instruccion1**  **|instruccionN**  **MIENTRAS (condición)**  ***Repite 1 o más ciclos*** | **do{**  **instruccion1;**  **instruccionN;**  **} while (condición);** |
| **Repetitiva:**  **Instrucción FOR** | **ENTERO j , n**  **PARA j <- 0 HASTA n PASO 1 HACER**  **|instruccion1**  **|instruccionN**  **FIN PARA**  **ENTERO j , n**  **PARA j <- 0 HASTA n+1PASO 1 HACER**  **|instruccion1**  **|instruccionN**  **FIN PARA** | **for($j=0; $j<$n; $i++){**  **instruccion1;**  **instruccionN;**  **}**  **for($j=0; $j<=$n; $i++){**  **instruccion1;**  **instruccionN;**  **}** |
|  |  | |
| **VARIABLE ESPECIALES** | **CONTADOR: siempre es de tipo ENTERO, se inicializa antes de la instrucción repetitiva (en general en 0, pero puede ser cualquier valor), dentro de la repetiva se incrementa en un valor constante**  **ACUMULADOR: puede ser**   * **de SUMA (+), se inicializa antes de la instrucción repetiva (en general en 0), dentro de la repetitiva acumula un valor VARIABLE de tipo float o entero** * **de PRODUCTO (\*), se inicializa antes de la instrucción repetiva (en general en 1, pero puede ser cualquier valor <> 0), dentro de la repetitiva multiplica por un valor VARIABLE de tipo float o entero** * **de STRING (,), se inicializa antes de la instrucción repetiva (en general en “”), dentro de la repetitiva concatena un valor VARIABLE de tipo String**   **BANDERA: siempre es de tipo BOOLEAN**  **SELECTORA: depende de la variable que queremos seleccionar. (ver ejercicio 21 del TP 6, ingresar los datos de Año Nac, altura y peso de los alumnos)** |  |
| **TIPOS DE CICLOS** | **Definido**  **Indefinido** |  |