|  |  |  |
| --- | --- | --- |
| Entorno: Sintaxis | | |
| Java | Pseudocódigo | PseInt |
| BASICO | | |
| Apertura de programa | | |
| Public class <clase> {  Public static void main (String[] args) {  . . .  }  } | PROGRAMA <nombre>  . . .  FIN | Proceso <nombre>  . . .  FinProceso |
| Declaración de variables | | |
| final PI = 3.14;  final EDAD = 18;  int a, b;  double | float | long c;  boolean d;  <tipo> <tabla>[];  <tipo>[] <tabla>; | CTE  PI = 3.14  EDAD = 18  VAR  a, b : ENTERO  c : REAL  d: BOOLEANO  t: ENTERO [ ] | Definir <variable1>, <variable2> Como Entero  Definir <variable1>, <variable2> Como Caracter  Definir <variable1>, <variable2> Como Real  Definir <variable1>, <variable2> Como Logico  Definir <tabla> como Entero | Carácter | Real | Logico  Dimension <tabla>[<tamaño] |
| Asignación de variables | | |
| <variable> = <valor>;  <tabla> = new <tipo>[<tamaño>];  <booleano> = true | false; | <variable> = <valor>  <tabla> = new ENTERO [100]  <booleano> = true | false  <booleano> = verdadero | falso | <variable> <- <valor>  <tabla>[<índice>] <- <valor>  <booleano> <- verdadero | falso |
| Mostrar mensajes por pantalla | | |
| System.out.println (“Hola mundo”);  System.out.println (<variable);  System.out.println (“El número es: ” + <variable>); | ESCRIBIR (“Hola mundo”)  ESCRIBIR (<variable>)  ESCRIBIR (“El número es: ” + <variable>) | Escribir “Hola mundo”  Escribir <variable>  Escribir “El número es: ”, <variable> |
| Mostrar mensajes por pantalla sin saltar | | |
| System.out.print (Mensaje sin salto de línea); | ESCRIBIR\_SS (“Mensaje sin salto de línea”) | Escribir sin saltar “Mensaje sin salto de línea” |
| Leer del teclado | | |
| Scanner <variable> = new Scanner (System.in);  <variable> = sc.next<tipo> ();  sc.close (); | LEER (<variable>) | Leer <variable> |
| Incremento de variables | | |
| <contador> = <contador> + 1;  <contador>++;  <contador>--; | <contador> = <contador> + 1  <contador>++  <contador>-- | <contador> <- <contador> + 1  <contador> <- <contador> - 1 |
| Resto de una división | | |
| <resto> = <dividendo> % <divisor>; | <resto> = <dividendo> % <divisor>  <resto> = <dividendo> MOD <divisor> | <resto> <- <dividendo> % <divisor>  <resto> <- <dividendo> MOD <divisor> |
| Potencia de un número | | |
| <potencia> = Math.Pow (<base>, <exponente>); | <potencia> = <base> ^ <exponente> | <potencia = <base> ^ <exponente> |
| CONDICIONALES | | |
| Condicional Simple | | |
| if (<condición>) {  . . .  } | SI <condición>  . . .  FIN SI | Si <condición> Entonces  . . .  FinSi |
| Condicional Doble | | |
| if (<condición>) {  . . .  } else {  . . .  } | SI <condición>  . . .  SI NO  . . .  FIN SI | Si <condición>  . . .  SiNo  . . .  FinSi |
| Condicional Anidado | | |
| if (<condición>) {  . . .  } else if (<condición>) {  . . .  } | SI <condición>  . . .  SI NO SI <condición>  . . .  FIN SI  FIN SI | Si <condición> Entonces  . . .  SiNo Si <condición> Entonces  . . .  FinSi  FinSi |
| Condicional Múltiple | | |
| switch (<variable>) {  Case <valor>:  . . .  break;  Default:  } | SEGUN <variable>  CASO <valor> :  . . .  DEFECTO :  . . .  FIN SEGUN | Según <variable> Hacer  <valor1> :  . . .  De otro Modo :  . . .  FinSegun |
| BUCLES | | |
| Bucle Pre-evaluación | | |
| while (<condición>) {  . . .  } | MIENTRAS <condición>  . . .  FIN MIENTRAS | Mientras <condición> Hacer  . . .  FinMientras |
| Bucle Pos-evaluación | | |
| Do {  . . .  } while (<condición>); | REPETIR  . . .  MIENTRAS <condición> | Repetir  . . .  Hasta que <condición> |
| Bucle Incremental | | |
| For (<tipo> <variable> = 0 ; <condición>; <variable>++) {  } | VAR  i : ENTERO  PARA i = 1 HASTA 10 INC +2  . . .  FIN PARA | Para i <- 0 hasta 10 con paso 2 hacer  . . .  FinPara |
| FUNCIONES | | |
| Declaración de funciones y procedimientos | | |
| Public static <tipo> <nombre\_func> ([<parametro1>, . .]) {  [<tipo> <valorRetorno>;]  . . .  [return <valorRetorno>;]  } | FUNCION <nombre\_func> ([<parámetro1>:<tipo>]): [<tipo>]  VAR  [<valorRetorno> : <tipo>]  . . .  [DEVOLVER <valorRetorno>]  FIN FUNCION |  |
| Truncar un número | | |
| <parteEntera> (int) <numeroReal>; | <parteEntera> = TRUNC (<numeroReal>) | <parteEntera> <- TRUNC (<numeroReal) |
| Tablas (primitivas) | | |
| Declaración y definición de tablas | | |
| <tipo>[ ] <nombreTabla>;  <nombreTabla> = new <tipo> { valor1, valor2, . .}; | <nombreTabla> : <tipo>[ ]  <nombreTabla> = new <tipo>[<tamaño>] |  |