TIPOS DE DATOS Y VARIABLES

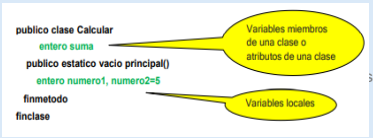
Tipos de datos

```
T01 texto nombre = "Maria"          // Variable tipo texto
T02 entero edad = 17                // Variable tipo entero
T03 real alto = 1.67                // Variable tipo real
T04 boleano hijoUnico = falso       // Variable tipo boleano
T05 caracter inicial = 'A'          // Variable tipo caracter
```

NOTA: Es obligatorio especificar el tipo de dato al declarar la variable

Tipos de variables

T06



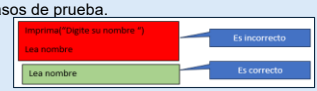

```
                                    /*Las variables de tipo bloque
                                    se especifican en la sección de ciclos*/
```

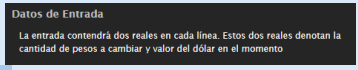
T07

ENTRADA Y SALIDA

Entrada de datos (lectura)

Logicode obtiene los datos de entrada desde unos casos de prueba de prueba. No se debe imprimir mensajes para ingresar datos de entrada. Se debe seguir el orden estricto, como se indica en los casos de prueba.



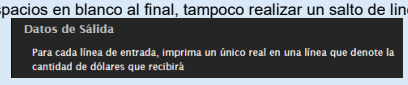


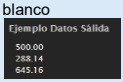

```
T08 lea nombre                      /*Para leer datos se usa
    lea identificacion              el comando lea en
    lea salario                     líneas separadas*/
```

NOTA: La variable a leer debe estar declarada anteriormente

Salida por pantalla (imprima o muestre)

Los datos de salida deben ser en estricto orden como se indica en los casos de prueba, incluso con su respectivo formato (cantidad de decimales) **CUIDADO:** la salida debe ser exacta, no se debe generar espacios en blanco al final, tampoco realizar un salto de linea en blanco






```
T09 imprima("Hola mundo")           // Imprime Hola mundo
T10 entero edad = 8                 // Imprime 8
    muestre(edad)
T11 imprima("Hola" + "\n")          /*Imprime Hola
    imprimaln("Mundo")                    Mundo (con salto de linea)*/
```

Concatenación

```
T12 texto nombre = "Maria"          //Imprime Hola Maria
    imprima("Hola" + nombre)
    texto nombre = "Juan"
T13 entero edad = 18                //Imprime Juan 18 123
    entero id = 123
    imprima(nombre +" "+ edad + " " + id)
```

Casting de Valores

En Logicode, al dividir dos enteros, el resultado es un entero. Para obtener un valor real, se debe asignar directamente el resultado, Ejemplo: real num= 0.2. La otra opción es realizar un **cast**, es decir convertir el resultado en un real, Ejemplo real num= (real) 2/10.

```
T14 entero div = 2/10
    real castDiv = (real)div          //Cambia el tipo de dato a real
```

Formatear (imprimir con formato)

En Logicode la sintaxis de formatear un número es **formatear(variable,2)**, donde 2 significa la cantidad de decimales que se desea formatear en este ejemplo es 2. Al momento de formatear dejando un espacio entre datos se puede realizar de varias maneras.

```
T15 imprima(formatear(num1,2) + " "+formatear(num2,2))
T16 imprima(formatear(num1,2) + " ")
    imprima(formatear(num2,2))     //NO imprime con salto de linea
T17 imprima(formatear(num1,2) + formatear(num2,2))
```

ESPECIFICADORES DE ACCESO

```
T52 publico clase ejemplo          // Permite el acceso desde cualquier clase.
T53 privado num1, num2             //Restringe el acceso solo dentro de la misma clase.
T54 protegido num1                 //Permite el acceso dentro del mismo paquete
                                   y a través de herencia. */
T55 clase abstracta                /* Restringe la creación de instancias y obliga a las
                                   clases derivadas a implementar sus métodos abstracto
```

FUNCIONES

```
T56 longitud(cadena)               // Devuelve la longitud de la cadena
    longitud(vector)               // Devuelve la longitud del vector
T57 seno(valor)                    // seno(60) Devuelve 0.86
T58 coseno(valor)                  // coseno(60) Devuelve 0.5
T59 tangente(valor)                // tangente(60) Devuelve 1.73
T60 cadena.obtenercaracter(indice) // Devuelve el caracter en el indice indicado
T61 cadena.obtenerindice(caracter) // Devuelve el indice donde se encuentra la letra
T62 //Este es un comentario de una sola linea
T63 /* Este es un comentario de
    multiples lineas */
```

MÉTODO CONSTRUCTOR

```
    publico clase Persona
       texto nombre
       entero edad

       // Método constructor
       publico metodo Persona(texto n, entero e)
          esteobjeto.nombre = n
          esteobjeto.edad = e
       finmetodo

       // Método para mostrar los datos de la persona
T67    publico vacio metodo mostrarDatos()
          imprima("Nombre: ", nombre)
          imprima("Edad: ", edad)
       finmetodo
    finclase

    publico clase Principal
       publico estatico vacio principal()
          Persona p = nuevo Persona("Ana", 25)
          p.mostrarDatos()
       finmetodo
    finclase
```

OPERADORES

Operadores Aritméticos

```
T18    +          // Suma, Ejm: 5 + 5 = 10
T19    -          // Resta, Ejm: 5 - 5 = 0
T20    *          // Multiplicación, Ejm: 5 * 5 = 25
T21    /          // División, Ejm: 5 / 5 = 1
T22    mod        // Residuo, Ejm: 5 mod 5 = 0
T23 potencia(base, exp)   // Potencia, Ejm potencia(5, 2) = 25
T24 raiz(numero)          // Raiz, Ejm raiz(25) = 5
```

Operadores Relacionales

```
       entero x = 10     // Declaracion de variable
T25    <                 // Menor que, Ejm: x < 11 verdadero
       >                 // Mayor que, Ejm: x > 9 verdadero
T26    <=                // Menor o igual que, Ejm: x <= 9 falso
       >=                // Mayor o igual que, Ejm: x >= 12 falso
T27    ==                // Igual que, Ejm x == 12 falso
T28    <>                // Diferente, Ejm x <> 8 verdadero
```

NOTA: Las operaciones con operadores relacionales, devuelven un valor boleano

Operadores Lógicos

```
T29    ^          // y (verdadero si los dos son verdadero)
T30    v          // o (verdadero si al menos uno es verdadero)
```

CONDICIONALES

Simples

```
       si(condición(es))          /* si (x > 10)
T31      proceso                    imprima("es mayor")
       finsi                      finsi */
```

Compuestos

```
       si(condición(es))          /* si(x < 10)
         proceso                    imprima("es menor")
T32    sino                       sino
         proceso                    imprima("es mayor")
       finsi                      finsi */
```

Anidados

```
       si(condición(es))          /* si(x mod 2 == 0)
         proceso                    imprima("es par")
       sino                       sino
         si(condición(es))          si(x < 10)
T33        proceso                    imprima("impar menor")
         sino                       sino
           proceso                    imprima("impar mayor")
         finsi                      finsi
       finsi                      finsi */
```

Multicondicionales (segun)

```
       segun(variable)            /* segun(edad)
         caso valor1:               caso 8:
           proceso                    imprima("es un nino)
         salto                      salto
         caso valor2:               caso 15:
           proceso                    imprima("es un adolescente")
T34      salto                      salto
         caso valor3:               caso 18
           proceso                    imprima("es un joven mayor de edad")
         salto                      salto
         en otro caso               en otro caso
           proceso                    imprima("es un adulto")
         salto                      salto (opcional)
       finsegun                   finsegun */
```

Operador Ternario

```
       (condición) ?              /* texto mensaje = (edad >= 18) ?
T35    expresión_si_verdadero:                  "Eres mayor de edad":
       expresión_si_falso                       "Eres menor de edad" */
```

ENVIAR Y RETORNAR PARÁMETROS

El método recibe parámetros y retorna un valor

T64

```
publico clase Operaciones
   publico estatico vacio principal()
      real nota1, nota2, nota3, notadef
      lea nota1
      lea nota2
      lea nota3
      notadef = calcular(nota1, nota2, nota3)
      imprima(notadef)
   finmetodo

   publico estatico real metodo calcular(real n1, real n2, real n3)
      real nd
      nd = (n1 + n2 + n3) / 3
      retorne nd
   finmetodo
finclase
```

El método recibe parámetros pero no retorna un valor

T65

```
publico clase Operaciones
   publico estatico vacio principal()
      real numero1, numero2
      lea numero1
      lea numero2
      calcular(numero1, numero2)
   finmetodo

   publico estatico vacio metodo calcular(real n1, real n2)
      imprima("La suma es: ", n1 + n2)
   finmetodo
finclase
```

El método no recibe parámetros pero retorna un valor

T66

```
publico clase Operaciones
   publico estatico vacio principal()
      entero sumaNros
      sumaNros = calcular()
      imprima("La suma es: ", sumaNros)
   finmetodo

   publico estatico entero metodo calcular()
      entero nro1, nro2
      lea nro1
      lea nro2
      retorne nro1 + nro2
   finmetodo
finclase
```

CICLOS

Ciclo Para

T36
```
para(entero i=0; i<=10; i=i+1)
  proceso
finpara
```
```
/* para(entero i= 1; i<5; i=i+1)
   imprimaln("Hola")
finpara */
```

NOTA: En el ciclo para, **i** es una variable de tipo bloque

Ciclo Mientras

T37
```
entero i=0
mientras(i<=10)
  proceso
  i=i+1
finmientras
```
```
/* entero i=2
   mientras(i<=20)
   imprimaln("Hola")
   i=i+2
finmientras */
```

NOTA: En el ciclo mientras se pueden usar operaciones lógicas: x > 0 ^ y < 10

Ciclo Mientras que

T38
```
entero i=0
haga
  proceso
  i=i+1
mientrasque(i<=10)
```
```
/* entero i=5
   haga
   imprimaln("Hola")
   i=i-1
mientrasque(i>=1) */
```

NOTA: En el ciclo mientrasque tambien se pueden usar operaciones lógicas

Ciclo Para cada (for each)

T39
```
entero edades[] = {20,31,50,21}
para(entero edad : edades)
imprima(edad)
finpara
```
```
/* entero numeros[] = {1,2,3,4}
   para(entero numero : numeros)
   imprima(numero)
finpara */
```

VECTORES

Creación

T40 `Tipo <nombre>[] = nuevo tipo[tamaño]`

T41 `entero edad[]=nuevo entero[20]`

T42
```
entero edad[]
edad= nuevo entero[20]
```

T43 `entero edad[] = {20,30,26,15,34}`

/* Estas son las tres diferentes formas de crear los vectores */

NOTA: Los corchetes deben ir delante del nombre del vector, ejemplo edad[ ] o al principio del nombre del vector, ejemplo [ ]edad

Leer e imprimir

T44
```
para(i=0; i<20; i=i+1)
  lea vector[i]
finpara
```

T45
```
para(i=0; i<longitud(vec); i=i+1)
  imprima(vec[i])
finpara
```

/* Puedes usar el número del tamaño del arreglo, o si deseas puedes usar la funcion longitud(vector) en caso de que desconozcas el tamaño del vector */

MATRICES

Creación

T46 `Tipo <nombre>[][] = nuevo tipo[filas][columnas]`

T47 `entero edades[][]=nuevo entero[20][20]`

T48
```
entero edad[][]
edad= nuevo entero[20][30]
```

T49 `entero edad[][]= {{20,30},{26,15},{20,34}}`

/* Estas son las tres diferentes formas de crear las matrices */

NOTA: Los corchetes deben ir delante del nombre de la matriz, ejemplo edad[ ][ ] o al principio del nombre del la matriz, ejemplo [ ][ ]edad

Leer e imprimir

T50
```
para(f=0; f<20; f=f+1)
 para(c= 0; c<20; c=c+1)
  lea mat[f][c]
 finpara
finpara
```

T51
```
para(f=0; f<longitud(mat); f=f+1)
 para(c= 0; c<longitud(mat); c=c+1)
  imprima mat[f][c]
 finpara
finpara
```

/* Puedes usar el número del tamaño del arreglo, o si deseas puedes usar la funcion longitud(matriz) en caso de que desconozcas el tamaño de la matriz */

INSTRUCCIONES EN LA POO

Logicode permite la utilización de algunas características de la programación orientada a objetos, como:

Encapsulación

```
publico clase viaje
   publico estatico vacio principal()
      real pesos, dolares, unidad_cambiaria
      lea pesos
      lea unidad_cambiaria
      calcular cal = nuevo calcular() // Aquí se instancia con nuevo
      cal.establecerpesos(pesos)
      cal.estableceruc(unidad_cambiaria)
      dolares = (cal.obtenerpesos() / cal.obteneruc())
      imprima(formatear("%.2f", dolares))
   finmetodo
finclase

publico clase calcular
   real pesos, uc

   // Método constructor
   publico metodo calcular(real p, real u)
      esteobjeto.pesos = p
T68   esteobjeto.uc = u
   finmetodo

   // Métodos establecer y obtener (get y set)
   publico vacio metodo establecerpesos(real pesos)
      esteobjeto.pesos = pesos
   finmetodo

   publico vacio metodo estableceruc(real uc)
      esteobjeto.uc = uc
   finmetodo

   publico real metodo obtenerpesos()
      retorne pesos
   finmetodo

   publico real metodo obteneruc()
      retorne uc
   finmetodo
finclase
```

Herencia

T69 publico clase perro hereda animal

NOTA: Se coloca la palabra hereda en la clase