Tecnológico de Monterrey

zencode

a peaceful introduction to code

Quick Reference Manual

¡BIENVENIDO A ZENCODE!

Bienvenido al extraordinario mundo de ZenCode, donde el viaje de la programación comienza con un toque de magia y mucha alegría. Entra en un reino donde la complejidad se domestica, y el poder de la codificación se aprovecha con una sonrisa.

¿Demasiado? Lo sentimos.

En ZenCode, creemos que la programación debe ser una aventura llena de emoción y asombro. Di adiós por un tiempo a la jerga intimidatoria y a los tecnicismos, y saluda a un lenguaje diseñado para hacer de tu experiencia en la pantalla, más sencilla. Es un lugar donde principiantes y entusiastas experimentados pueden reunirse y embarcarse en un increíble viaje de descubrimiento.

Prepárate para explorar el arte de la programación con facilidad y gracia. ZenCode adopta la simplicidad sin comprometer la capacidad. Tanto si estás empezando como si buscas un nuevo lenguaje para ampliar tus horizontes, ZenCode es tu brújula, guiándote hacia la magia de crear tus propias maravillas de software.

Veamos rápidamente cómo se utiliza cada función de ZenCode.

ASIGNACIONES

En ZenCode, la asignación de variables es el acto de asignar un valor a una variable, lo que le permite almacenar y manipular datos a través de su programa con un sentido de propósito y dirección.

La forma de hacerlo es utilizando estas dos “<”:

```
a << 5;
c << a + 1;
```

En el código anterior, la variable a va a guardar el valor 5, mientras que la variable “c” va a guardar 5, que es el valor que tiene “a”, más 1; es decir, 6. Pero antes de poder dar valores a nuestras variables, tenemos que crearlas.

DEFINICIÓN DE VARIABLES

Cuando defines una variable en ZenCode, estás declarando su existencia y especificando su tipo, proporcionando la base necesaria para almacenar y acceder a los datos a través de tu código.

En ZenCode, lo hacemos escribiendo “set”, luego el tipo de contenido que queremos que nuestra variable guarde y, finalmente, el nombre que queremos darle

```
set int b;
set dec a;
set char f;
set bool x;
```

Entonces, la variable “b” va a guardar enteros, la variable “a” va a guardar decimales, la variable “f” guarda caracteres y la variable “x” guarda booleanos; es decir o “True” (verdadero) o “False” (falso). El nombre no tiene que ser una sola letra; puede ser el que prefieras, siempre y cuando comience con una minúscula y solo tenga valores alfanuméricos y/o guiones bajos, como:

```
set int diasProgramandoZen, xMedia2, y_Mediana;
```

¡No olvides el punto y coma al final!

INTERACCIÓN CON LA CONSOLA

La interacción de consola en ZenCode se refiere al intercambio dinámico de información entre el programa y el usuario a través de la consola. Permite a los usuarios proporcionar entradas y recibir salidas, creando una experiencia atractiva e interactiva.

Esto se hace por medio de dos comandos. Si queremos que se muestre en la consola, escribimos “cwrite”:

```
cwrite << c + d * a;
```

Con cwrite también podemos mandar texto para que imprima la consola:

```
cwrite << "¡Hola, mundo!";
```

Y si queremos que la consola lea lo que nosotros escribimos, agregamos “cread” a nuestro programa:

```
cread << f;
```

Entonces, lo que escribamos a la consola, se guardará en la variable “f” si el tipo es compatible.

OPERACIONES COMPARATIVAS

Con las variables, podemos realizar operaciones comparativas que se pueden guardar en nuestras variables booleanas (verdadero/falso).

```
set int a, c;
set bool x;

a << 3;
c << 2;
x << {a > c};
```

En este caso, la variable “x” guardará “True”.

Así, podemos realizar diferentes comparaciones: mayor que (>), menor que (<), igual (=), diferente (><), mayor o igual que (>=) o menor o igual que (<=).

```
mayorQue << {a > c};
menorQue << {a < c};
mayor_o_igualQue << {a >= c};
menor_o_igualQue << {a <= c};
igualQue << {a = c};
diferenteDe << {a >< c};
```

OPERACIONES LÓGICAS

ZenCode tiene una variedad de operaciones lógicas, como AND (&&), OR (||) y NOT(~), permitiéndote realizar evaluaciones condicionales y tomar decisiones basadas en la verdad o falsedad de expresiones lógicas.

```
cwrite << {a = c || a = b};
x << {~ a = c && a = b};
```

IF-ELSE

Con los estatutos if-else, puedes controlar el flujo de tu programa basándote en ciertas condiciones. Puedes especificar un bloque de código que se ejecutará si una condición es verdadera y un bloque alternativo que se ejecutará si la condición es falsa.

```
if (b >< 5) {
    cwrite << "b >< 5";
    b << 6;
  } else
    cwrite << "b = 5";
```

Si solo tienes una instrucción, no son necesarios los corchetes, como en el ejemplo para el else. Pero, si vas empezando, te recomendamos que los mantengas por seguridad.

WHILE Y DO-WHILE

El ciclo while en ZenCode te permite ejecutar repetidamente un bloque de código mientras una condición dada permanezca verdadera, proporcionando una forma flexible y eficiente de iterar y realizar tareas hasta que se cumpla una determinada condición.

```
while (a < 5) {
    a << a + 1;
    cwrite << a;
  }
```

Similar al while, el ciclo do-while en ZenCode ejecuta primero un bloque de código y luego comprueba la condición. Garantiza que el bloque de código se ejecute al menos una vez, independientemente de si la condición es inicialmente verdadera o falsa.

```
do
  a << a - 1;
while (a > 0);
```

LOOP

El loop en ZenCode ofrece un enfoque estructurado a la iteración, permitiéndote especificar una inicialización, una condición y una sentencia de iteración. Simplifica el proceso de ejecución de un bloque de código repetidamente durante un número predeterminado de veces.

Este tiene tres partes: la primera, muestra qué valor quieres para iniciar. La segunda, es cuánto quieres que avance con cada iteración. Y, la tercera, es qué quieres que suceda para que deje de correr el loop.

```
loop a in range (b - 2 : 1 : a > 5)
    cwrite << a;

loop c in range(0 : 1 : c = 4) {
    loop f in range(0 : 1 : f = 3) {
      cwrite << delta[c,f];
    }
  }
```

FUNCIÓN main

La función `main` es la función principal del programa. Todo lo que quieras ejecutar va dentro de esta función. La función main siempre debe terminar con un `end`.

```
main {
  set int x, y;
  x << 2;
  y << 3;

  cwrite << x + y;

  end
}
```

OTRAS FUNCIONES

Si no quieres escribir lo mismo muchas veces, las funciones son para ti. Las funciones son partes de código a las que puedes ponerle un nombre y volverlas a usar cuantas veces quieras.

Hay dos tipos de funciones: Funciones con tipo, que son funciones que tienen que generar un valor cuando terminen de ejecutarse; y funciones vacías o “void”, que solamente ejecutan el procedimiento.

Aquí te dejamos un ejemplo de ellas:

```
function int rojo() {
  set int a, b;
  a << 3;
  cread << b;
  return a + b;
}

function void azul() {
  set int a;
  set dec x;
  a << 4;
  x << 2.4 + rojo();
  cwrite << x;
}

main {
  azul();

  end
}
```

En este caso, se manda a llamar la función azul(), la función azul() manda a llamar a la función rojo() en el proceso y utiliza su valor de retorno para hacer un cálculo y guardarlo en la variable “x”.

LISTAS Y MATRICES

Las listas y matrices en ZenCode son estructuras de datos que te permiten almacenar colecciones de valores del mismo tipo. Proporcionan almacenamiento eficiente y fácil acceso a los elementos, lo que te permite organizar y manipular datos relacionados con facilidad y precisión.

Primero, se definen así:

```
set int list alpha[3];
set int matrix delta[4,3];
```

Y el número que le des es la cantidad de elementos que quieres que tenga guardados.

Y después, puedes guardar cosas en ellos, así:

```
alpha[c-3] << 4;
```

DATATABLES

Finalmente, nuestro “chile del que pica”. Cuando tengas más experiencia, podrás practicar introducirte a otro mundo de las tecnologías: las bases de datos. Data funciona como una ligera base de datos, con la que puedes comenzar a imaginarte cómo funcionan las bases de datos en la vida diaria. Estas solo se pueden declarar una vez por función:

```
create data table[
    {text:"name"}, {dec:"un"}, {dec:"deux"}, {int:"trois"}
  ][30];
```

Una vez que la tengas hecha, puedes escribir en sus filas, de esta manera:

```
write in table[12] values (pass, a, 2 * 4, b + 8);
```

O ahorrarte el trabajo y dejar que nosotros llenemos las columnas por ti.

```
ndist fit table:"deux" with (0.25+b, a);
binomial fit table:"un" with (b, 0.6);
```

Esto es solo un poco de lo que puedes hacer con la programación. Lo más importante es que no te pongas límites para lo que puedes hacer con una computadora. Si te has propuesto a empezar, te motivamos a que lo hagas, y aquí estamos en ZenCode para ayudarte. No lo olvides: Si tienes un sueño, persíguelo, ¡y no olvides el **end** al final de tu código!