Variables y Tipos de datos en C

Diego Iván Oliveros Acosta

Tipos de datos C

- Las variables son contenedores para almacenar valores de datos, como números y caracteres.
- En C, hay diferentes tipos de variables (definidas con diferentes palabras clave), por ejemplo:
- int- almacena números enteros (números completos), sin decimales, como 123o-123
- float- almacena números de punto flotante, con decimales, como 19.99o-19.99
- char- almacena caracteres individuales, como 'a'o 'B'. Los caracteres están entre comillas simples

Declaración (creación) de variables

- Para crear una variable, especifique el **tipo** y asígnele un **valor** :
- *type variableName = value;*

- Donde tipo es uno de los tipos de C (como int) y variableName es el nombre de la variable (como x o myName).
- El signo igual se utiliza para asignar un valor a la variable.

- Entonces, para crear una variable que debería almacenar un número, observe el siguiente ejemplo:
- Crea una variable llamada minum de tipo int y asígnale el valor 40 :

- `int minum = 40;`

- También puedes declarar una variable sin asignar el valor y asignar el valor más tarde:

```
// Declare a variable
int myNum;

// Assign a value to the variable
myNum = 15;
```

Variables de salida

- En muchos otros lenguajes de programación (como Python , Java y C++), normalmente se usaría una **función de impresión** para mostrar el valor de una variable.
- Sin embargo, esto no es posible en C, Para generar variables en C, debes familiarizarte con algo llamado **" especificadores de formato "**

```
printf("Hello World!");




int myNum = 15;
printf(myNum);  //
Nothing happens
```

Especificadores de formato

Los especificadores de formato se utilizan junto con la printf() función para indicar al compilador qué tipo de datos almacena la variable. Básicamente, se trata de un marcador de posición para el valor de la variable.

Un especificador de formato comienza con un signo de porcentaje %, seguido de un carácter.

Por ejemplo, para generar el valor de una variable int, utilice el especificador de formato %d rodeado por comillas dobles (""), dentro de la función printf().

```c
#include <stdio.h>

int main() {
  int myNum = 15;
  printf("%d", myNum);
  return 0;
}
```

Para imprimir otros tipos, utilice %c for char y %f for float:

```
// Crear variables
int myNum = 15;               // Integer (whole number)
float myFloatNum = 5.99;      // Floating point number
char myLetter = 'D';          // Character

// Imprimir variables
printf("%d\n", myNum);
printf("%f\n", myFloatNum);
printf("%c\n", myLetter);
```

Variables de salida

- Para combinar texto y una variable, sepárelos con una coma dentro de la función printf()
- Para imprimir diferentes tipos en una sola función printf(), puede concatenarlos

```
int myNum = 15;
printf("My favorite number is: %d", myNum);
```

```
int myNum = 15;
char myLetter = 'D';
printf("My number is %d and my letter is %c", myNum, myLetter);
```

¿Imprimir valores sin variables?

- También puedes simplemente imprimir un valor sin almacenarlo en una variable, siempre que uses el especificador de formato correcto.
- **Sin embargo** , es más sostenible utilizar variables, ya que se guardan para más adelante y se pueden reutilizar en cualquier momento.

```c
#include <stdio.h>

int main() {
  printf("My favorite number is: %d\n", 15);
  printf("My favorite letter is: %c", 'D');
  return 0;
}
//¿Cuál es la utilidad?
```

Cambiar valores de variables

Si asigna un nuevo valor a una variable existente, sobrescribirá **el** valor anterior.

```
int myNum = 15;  // myNum is 15
myNum = 10;  // Now myNum is 10

int myNum = 15;

int myOtherNum = 23;

// Assign the value of myOtherNum (23) to myNum
myNum = myOtherNum;

// myNum is now 23, instead of 15
printf("%d", myNum);
```

También puedes asignar el valor de una variable a otra o copiar valores a variables vacías.

```
// Create a variable and assign the value 15 to it
int myNum = 15;

// Declare a variable without assigning it a value
int myOtherNum;

// Assign the value of myNum to myOtherNum
myOtherNum = myNum;

// myOtherNum now has 15 as a value
printf("%d", myOtherNum);
```

Agregar variables juntas

Para agregar una variable a otra variable, puede utilizar el + operador:

```c
#include <stdio.h>

int main() {
  int x = 5;
  int y = 6;
  int sum = x + y;
  printf("%d", sum);
  return 0;
}
```

Ejercicio:

- Muestra la suma de 5 + 10, utilizando dos variables: x, y.

C Declarar múltiples variables

Para declarar más de una variable del mismo tipo, utilice una lista **separada por comas :**

```c
#include <stdio.h>

int main() {
  int x = 5, y = 6, z = 50;
  printf("%d", x + y + z);
  return 0;
}
```

También puedes asignar el **mismo valor** a múltiples variables del mismo tipo:

```c
#include <stdio.h>

int main() {
  int x, y, z;
  x = y = z = 50;
  printf("%d", x + y + z);
  return 0;
}
```

Nombres de variables en C

- Todas **las variables** C deben **identificarse** con **nombres únicos** .
- Estos nombres únicos se llaman **identificadores** .
- Los identificadores pueden ser nombres cortos (como x e y) o nombres más descriptivos (edad, suma, volumen total).
- **Nota:** Se recomienda utilizar nombres descriptivos para crear un código comprensible y fácil de mantener:

- Las reglas generales para nombrar variables son:

- Los nombres pueden contener letras, dígitos y guiones bajos.
- Los nombres deben comenzar con una letra o un guión bajo (_)
- Los nombres distinguen entre mayúsculas y minúsculas (myVary myvarson variables diferentes)
- Los nombres no pueden contener espacios en blanco ni caracteres especiales como !, #, %, etc.
- Las palabras reservadas (como int) no se pueden utilizar como nombres

Ejemplos:

- A menudo, en nuestros ejemplos, simplificamos los nombres de las variables para que coincidan con su tipo de datos (myInt o myNum para inttipos, myChar para chartipos, etc.). Esto se hace para evitar confusiones.

- Sin embargo, para un ejemplo práctico del uso de variables, hemos creado un programa que almacena diferentes datos sobre un estudiante universitario

```
// Good variable name
int minutesPerHour = 60;

// OK, but not so easy to understand what m actually is
int m = 60;

// Student data
int studentID = 15;
int studentAge = 23;
float studentFee = 75.25;
char studentGrade = 'B';

// Print variables
printf("Student id: %d\n", studentID);
printf("Student age: %d\n", studentAge);
printf("Student fee: %f\n", studentFee);
printf("Student grade: %c", studentGrade);
```

Calcular el área de un rectángulo

En este ejemplo de la vida real, creamos un programa para calcular el área de un rectángulo (multiplicando el largo por el ancho)

```c
#include <stdio.h>

int main() {
  // Create integer variables
  int length = 4;
  int width = 6;
  int area;

  // Calculate the area of a rectangle
  area = length * width;

  // Print the variables
  printf("Length is: %d\n", length);
  printf("Width is: %d\n", width);
  printf("Area of the rectangle is: %d", area);

  return 0;
}
```