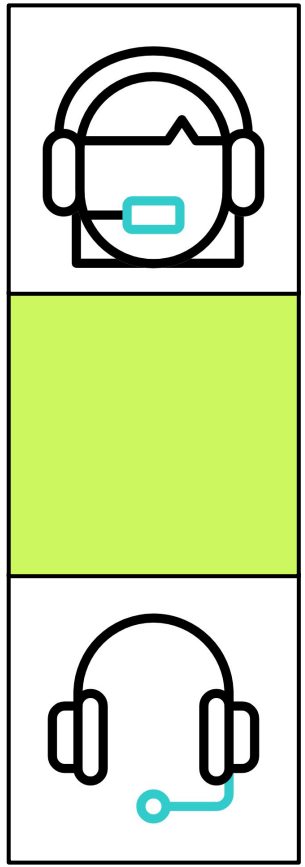
¿Como son las variables en C?



Laboratorio de Programación

En esta Clase aprenderemos sobre:

1. Las reglas para nombrar variables en C.
2. Los principales tipos de datos en C: int, float, double y char.
3. Declarar e inicializar tipos de datos en C y como cambiar su valor.
4. Conversión de un tipo de datos a otro y algunas limitaciones potenciales en C.

Comencemos con lo que es una variable:

Es una referencia utilizada para almacenar información para uso futuro.

Por ejemplo, se puede declarar:

int puntaje

Y más adelante en su código asignar, cambiar, comparar, etc. esta variable
La estructura básica de creación de variables (conocida como declaración) es:

type nombre_variable.

Como consejo adicional desde el principio, puede crear más de una variable del mismo tipo a la vez enumerandolas con comas entre sus nombres, como:

variable1, variable_2, variable_3, variable4.

Sintaxis a respetar en la declaración de variables:

1. Los nombres pueden estar compuestos por letras mayúsculas y minúsculas, números y guiones bajos.
2. El primer carácter debe ser una letra (mayúscula o minúscula).
3. No se permiten palabras clave como nombre completo (int no está permitido pero int_count funcionaría).

Puedes encontrar los errores en estas declaraciones?

```c
#include <stdio.h>

int main() {

    int numero!;
    int testeo valor;
    int 1ejemplo;
    float int;
    printf("Parece que todo anda bien!\n");

}
```

Tipos de Datos

El tipo de una variable indica qué tipo de información se puede almacenar en ella. C tiene una buena cantidad de tipos que puede usar, pero en esta lección, repasaremos algunos de los más comunes:

int, float, double y char.

En C, debe especificar el tipo de la variable cuando la declara. Una vez que se establece, ese el único tipo que puede incluirse en esa variable. Entonces, si crea una variable de tipo int, que solo puede almacenar números enteros, no podrá almacenar "a" ni 1.2 en ella. La siguiente tabla detalla estos tipos que estamos revisando y qué información puede contener cada uno.

Tipos de Datos

La siguiente tabla detalla estos tipos que estamos revisando y qué información puede contener cada uno.

Type	Description	Values
`int`	a whole number	-2,147,483,648 to 2,147,483,647
`float`	a number with possible decimals	6 decimal places
`double`	a number with possible decimals	15 decimal places
`char`	stores one character (letter or number)	a single character

Algunas preguntas

¿Cuál es la diferencia entre un *float* y un *double*?

char contiene una sola letra o número, pero...

¿qué sucede si desea almacenar el nombre de una persona, que tiene más de un carácter (en la mayoría de los casos)?

Podrias asignar los tipos de variables correspondientes?

```c
#include <stdio.h>

int main() {
    // Arregla los tipos de datos
  idEstudiante;
  promedioEstudiante;
  notaEstudiante;
    // Aqui no se necesita realizar ningun cambio
 idEstudiante = 1;
 promedioEstudiante = 90.56;
 notaEstudiante = 'A';

 printf("El id de estudiante es: %dst\n", idEstudiante);
 printf("El promedio total del estudiante es: %3.2f\n",
     promedioEstudiante);
 printf("La nota del estudiante es: %c\n", notaEstudiante);
    return 0;
}
```

Output de variables

```c
#include <stdio.h>

int main() {
   int dia = 3;
printf("Hola mundo! hoy es %drd! dia de la semana", dia);
    return 0;
}
```

Cuando el compilador ejecuta el código, reemplazará %d con el valor en la lista de variables, tomandolos en el orden que se encuentra en la cadena que coincide con el orden en que aparecen en la lista; primero en la cadena coincide con el primero en la lista de parámetros).

Tipos de salida

Hay muchos formatos y tipos de parámetros opcionales que se pueden usar, pero para nuestros propósitos aquí están algunos de los básicos.

symbol	type
%d or %i	int
%f	double or float
%c	char

Inicialización de variables

Hay muchos formatos y tipos de parámetros opcionales que se pueden usar, pero para nuestros propósitos aquí están algunos de los básicos.

Ahora que tenemos nuestra variable, sabemos su nombre y lo que puede contener (tipo)

¿qué hacemos con ella?

En ese momento está vacía y no tiene ningún propósito real. El poder de una variable proviene de su capacidad para establecerse, cambiarse y examinarse. Así que la pregunta es,

¿cómo hacemos eso?

Hay dos formas de establecer un valor, por ahora veamos cómo establecer el valor cuando crea la variable en sí:

Inicialización de variables

Hay dos formas de establecer un valor, por ahora veamos cómo establecer el valor cuando crea la variable de la siguiente manera:

```
int ejemplo = 3;
```

En este caso, no solo creó la variable, la nombró como *ejemplo* y la identificó para contener números enteros, sino que estableció su valor en 3.

¡Hemos ahorrado tiempo al hacer todo esto en una sola línea!

Inicialización de variables

Al declarar un carácter *char*, necesita comillas simples alrededor:

```
char letra = 'a';
char algo = 'x';
```

Inicialización de variables

```c
#include <stdio.h>

int main() {

    int numLibros;
    char letraApellido;
    char letraNombre;
    double valorTurron;

    printf("La cantidad de libros que lei es: %d\n", numLibros);
    printf("La letra de mi apellido es: %c\n", letraApellido);
    printf("La letra de mi nombre es: %c\n", letraNombre);
    printf("Esperas pagar $%.2f Por un turrón?.\n", valorTurron);
  return 0;
}
```

float y double, cual es la diferencia?

Un flotante tiene menos precisión que un doble, 6 frente a 15 decimales posibles respectivamente y, por lo tanto, ocupa menos memoria (4 frente a 8 bytes).

La otra cosa a tener en cuenta es que el sistema está redondeando los valores que almacena en cualquiera de los dos. Esto puede causar resultados inesperados, especialmente con un *float* ya que tienen menos precisión.

Esta es la razón por la que verá que el *double* se usa cada vez que la precisión es importante, como en aplicaciones científicas, médicas o financieras.

Actualización de variables

Acabamos de examinar cómo asignar valores en la declaración, pero... también se pueden asignar valores en cualquier punto del código. Como ejemplo:

```
 7  int total_units;
 8  …
 9  …
10  …
11  total_units = 3;
```

Asignación de variables

También puede asignar una variable para que sea igual a lo que está almacenado en otra variable, como

```
a = b;
```

Si cambia b después de esto, los valores ya no coincidirán (hay una manera de vincular las dos variables para que siempre coincidan, pero hablaremos de eso más adelante).

Ejercicio

```c
#include <stdio.h>

int main() {
    char categoriaLibro = 'A';
    char categoriaPelicula = 'B';
    double precioTicket = 150.50;
    double precioLibro  = 1900.99;
    printf("La categoria del libro es %c con un costo $%
        .2f\n", categoriaLibro, precioLibro);
    printf("La categoria del libro es %c y tiene un costo de
        $%.2f\n", categoriaPelicula, precioTicket);

}
```

Modifique este programa para que la categoría del libro sea igual a la categoría de la película y el precio del ticket sea igual al precio del libro. No modificar los valores asignados en la declaración.