

Tipos de Variables y Operadores Aritméticos

Variable

En C, las **variables** son espacios de memoria reservados para almacenar datos, los cuales **pueden cambiar** a lo largo de la ejecución del programa. Cada variable tiene un **tipo de dato**, que define el tamaño en memoria y las operaciones que se pueden realizar sobre ella.



Variable

El nombre que se le da a una variable es muy importante; intenten usar siempre nombres significativos que, de alguna forma, identifiquen el contenido.

Vamos a escribir los nombres de variables en formato lowerCamelCase. La primera letra se escribe en minúscula y, a continuación, si se utiliza más de una palabra, cada una de ellas empezaría con mayúscula. Por ejemplo, **edadMin**. Los nombres de las variables:

Empiezan con una letra o guión bajo.

No puede comenzar con un número.

Puede contener letras, números y guión bajo. No se puede usar palabras reservadas(como `int`, `float`, `return`, `if`, `while`, etc.)

El nombre de la variable es sensible a mayúsculas y minúsculas



Declaración

En C **todas** las variables **deben ser declaradas** antes de usarse.

```
int edad; // Declaración de una variable llamada "edad" de tipo entero
```

También se puede definir una variable asignándole un valor inicial:

```
int edad = 33; // Definición con inicialización
```

El orden para hacerlo es: tipoDato nombreVariable



Tipos de datos en C

C proporciona varios tipos de datos primitivos:

1. **Enteros** (**int**, **short**, **long**, **long long**, **unsigned**)
2. **Flotantes** (**float**, **double**, **long double**)
3. **Caracteres** (**char**)
4. **Tipos Modificados** (signed, unsigned, short, long)

Hay más! Pero los vemos más adelante!!



Tipos Enteros

Tipo	Tamaño en bytes	Rango de valores
short	2	-32,768 a 32,767
Int	4	-2,147,483,648 to 2,147,483,647
long	4 u 8	-9,223,372,036,854,775,808 a 9,223,372,036,854,775,807
long long	16	-9,223,372,036,854,775,808 a 9,223,372,036,854,775,807

Tipos Sin Signo y modificados

Si usas **unsigned**, solo vas a almacenar valores positivos:

```
unsigned int u = 4294967295; // Máximo para un unsigned int
```

Tipos Modificados (**short, long, signed, unsigned**)



Tipos Flotantes

Se usan para almacenar números decimales (números con coma)

Tipo	Tamaño en bytes	Precisión
float	4	6-7 decimales
double	8	15-16 decimales
long double	10/16	19+ decimales



Tipo Caracter

Almacena un solo carácter (1 byte).

```
char letra = 'A';
```

Puede representarse con comillas simples 'a'.

Internamente, se guarda como un número ASCII (ejemplo: 'A' = 65).



Y las cadenas de caracteres?

En C, una **cadena de caracteres** es una secuencia de caracteres almacenados en un arreglo (lo veremos más adelante).

```
char saludo[ ] = "Hola";
```

En este caso, la cadena "Hola" se guarda como un arreglo de caracteres: {'H', 'o', 'l', 'a', '\0'}.

En C no existen las cadenas de caracteres como un tipo de dato primitivo



También existen constantes...

Las **constantes no son variables** en el sentido tradicional, porque su valor no puede cambiar después de ser definido.

Las constantes son útiles porque:

- **Sirven para evitar modificaciones accidentales** en valores críticos.
- **Hacen que el código sea más legible** al usar nombres descriptivos en lugar de valores "mágicos".
- **Facilitan el mantenimiento del código**, ya que si necesitas cambiar un valor, solo lo haces en un lugar.

```
const float PI = 3.14159;
```

```
const int MAX_EDAD = 100;
```

También se puede usar la directiva `#define`...



Operadores Aritméticos

Los operadores aritméticos en **C** son aquellos que permiten realizar operaciones matemáticas básicas como suma, resta, multiplicación, división y módulo. Son fundamentales para cualquier cálculo en el lenguaje.

$a + b$ toma los valores de a y b y los suma. SUMA +

$a - b$ resta b de a .

RESTA -

$a * b$ multiplica los valores.

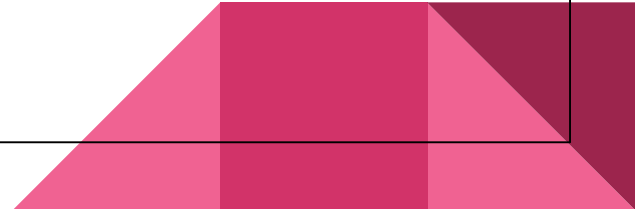
MULTIPLICACIÓN *

a / b realiza la división.

DIVISIÓN /

*****Si a y b son enteros, el resultado será entero (truncará la parte decimal).

*****Qué pasa si divido por cero??



El módulo %

Devuelve el residuo de una división. El resto.

- Si hacemos $10 \% 3$ devuelve el residuo de $10 / 3$, que es 1.

Importante: Solo funciona con números enteros



Operadores de Incremento y Decremento (++ , --)

Estos operadores incrementan o decrementan el valor de una variable en 1.


En lugar de hacer $x = x - 1$

podemos hacer $x--$ y ahí tenemos el mismo resultado

Pre-incremento (++x) vs. Post-incremento (x++)

- **++x**: Incrementa antes de usar la variable.
- **x++**: Usa la variable y luego la incrementa.

**prueben post y pre incremento y decremento en sus compus, impriman las variables a ver que tienen



Combinación operadores de asignación

El operador de asignación es el =

Podemos combinar los operadores aritméticos con = para simplificar código.

Operador	Ejemplo	Expresión equivalente
+=	x += 5	x = x + 5
-=	x -= 6	x = x - 6
*=	x *= 8	x = x*8
/=	x /= 2	x = x/2
%=	x %= 3	x = x%3

Orden de Precedencia de los Operadores en C

Los operadores en **C** tienen una **jerarquía** que determina qué operaciones se realizan primero en una expresión. La precedencia es similar a la de la matemática convencional.

Prioridad	Operador	Descripción	Asociatividad
1 (Mayor)	()	Paréntesis (agrupación)	De izquierda a derecha
2	++, --	Incremento/Decremento o	De derecha a izquierda
3	*, /, %	Multiplicación, División, Módulo	De izquierda a derecha
4	+, -	Suma y Resta	De izquierda a derecha
5 (Menor)	=	Asignación	De derecha a izquierda