Tipo de datos

Los valores de las variables se almacenan como 0's y 1's dentro de la memoria del computador. El programa no necesita saber el lugar exacto en dónde se almacena una variable; simplemente puede referirse a ella a través de su nombre.

Lo que el programa necesita saber es el "tipo de dato" que se almacena dentro una variable. No es lo mismo almacenar un entero que una letra o un número decimal. Si bien todos son representados como 0's y 1's, no son interpretados de la misma manera y, generalmente, no ocupan la misma cantidad de memoria.

Los tipos de datos fundamentales implementados por un lenguaje son:

- **Caracteres** : Almacenan letras o símbolos.
- **Enteros numéricos** : Almacenan números enteros. Pueden contener signo y ser de diferentes tamaños.
- **Punto flotante** : Almacenan valores reales (una parte entera y otra decimal) con diferentes niveles de precisión.
- **Booleanos** : Almacenan valores `true` o `false`.

Listado de tipos fundamentales en C++

1. Tipos de caracter

Tipo	Notas sobre Tamaño / Precisión
char	Exactamente 1 byte de tamaño. Al menos 8 bits.
char16_t	No más pequeño que char. Al menos 16 bits.
char32_t	No más pequeño que char16_t. Al menos 32 bits.
wchar_t	Representa el set de caracter más largo soportado.

2. Tipos enteros (con signo)

Tipo *	Notas sobre Tamaño / Precisión
signed char	Igual que char. Al menos 8 bits.
signed **short** *int*	No más pequeño que char. Al menos 16 bits.
signed **int**	No más pequeño que short. Al menos 16 bits.
signed **long** *int*	No más pequeño que int. Al menos 32 bits.
signed **long long** *int*	No más pequeño que long. Al menos 64 bits.

> * El nombre de *algunos* enteros puede abreviarse sin la parte *signed* e *int*. Sólo la parte que aparece en

> **negrita** es requerida.

3. Tipos enteros (sin signo)

Tipo *	Notas sobre Tamaño / Precisión
unsigned char	Mismos tamaños que su contraparte con signo.
unsigned short *int*	
unsigned int	
unsigned long *int*	
unsigned long long *int*	

> * El nombre de *algunos* enteros puede abreviarse sin la parte *int*. Sólo la parte que aparece en **negrita** es requerida.

4. Tipos de punto flotante

Tipo	Notas sobre Tamaño / Precisión
float	
double	Precisión no menor que float.
long double	Precisión no menor que double.

5. Tipo booleano

Tipo	Notas sobre Tamaño / Precisión
bool	Usa la unidad más pequeña de memoria asignable.

Dentro de cada uno de los grupos anteriores, la *diferencia entre tipos es sólo su tamaño* (es decir, cuánto ocupan en memoria). El primer tipo dentro de cada grupo es el más pequeño y el último es el más grande, con cada tipo al menos igual de grande como el que lo precede. Aparte de eso, los tipos de un grupo tienen las mismas propiedades.

Nótese que a excepción de `char`, ninguno de los tipos de datos fundamentales tiene un tamaño específico. Sus tamaños pueden variar dependiendo del compilador y la arquitectura del computador en donde se ejecuta el programa.

El tamaño de los tipos de datos se expresan en *bits*. Mientras más bits tiene un tipo de datos, **más valores puede representar**, pero al mismo tiempo, **más memoria ocupará**.

Tamaño	Valores únicos representables	Notas
8-bit	256	$= 2^8$
16-bit	65.536	$= 2^{16}$
32-bit	4.294.967.296	$= 2^{32}$ (~4 billones)
64-bit	18.446.744.073.709.551.616	$= 2^{64}$ (~18 billones billones)

Para el caso de los enteros, mientras **mayor bits** tenga el tipo de dato, **más valores** podrá representar.

> Ejemplo:
>
> Un entero sin signo de 16-bit (`unsigned int`) puede representar entre `0` y `65.535` valores, mientras que su contraparte con signo (`signed int` o `int`) puede representar, en la mayoría de los casos, valores entre `-32768` y `32767` . Nótese que el rango de valores positivos del entero con signo es *aproximadamente* la mitad de los valores que tiene el entero sin signo, esto debido a que uno de los 16 bits es utilizado para el signo).

En el caso de los tipos de punto flotante, el tamaño de bits afecta a su precisión.

Los tipos recientemente descritos (caracteres, enteros, punto flotante, booleanos) son conocidos también como "**Tipos aritméticos**".

Existen también 2 tipos fundamentales adicionales:

- `void` : Vacío. Identifica una carencia de tipo de dato.
- `nullptr` : Tipo especial de puntero.