**Definición**

size\_t es un **tipo de dato entero sin signo** (es decir, no puede representar valores negativos) definido en las bibliotecas estándar <stddef.h> o <cstdlib> en C++, y <stdio.h> o <stdlib.h> en C.

Este tipo se utiliza principalmente para **representar el tamaño de objetos en bytes** o **índices de arreglos**. Es el tipo de retorno de la función sizeof, lo que significa que cada vez que se usa sizeof, el valor que devuelve es de tipo size\_t.

**Características principales**

1. **Es un entero sin signo**, lo que significa que solo puede almacenar valores positivos o cero.
2. **Depende de la arquitectura del sistema**:
   * En sistemas de **32 bits**, size\_t suele tener 4 bytes.
   * En sistemas de **64 bits**, size\_t suele tener 8 bytes.
3. Es **garantizado por el estándar** para poder almacenar el tamaño máximo posible de cualquier objeto en memoria.
4. Es **el tipo estándar** para manejar índices y conteos en funciones que trabajan con memoria o arreglos.

**Uso común de size\_t**

**1. En la función sizeof**

Como se mencionó, sizeof devuelve un valor de tipo size\_t, que representa cuántos bytes ocupa un objeto o tipo:

size\_t bytes = sizeof(double); // Devuelve 8 en la mayoría de sistemas

**2. En bucles con arreglos**

Dado que los arreglos tienen tamaños no negativos, size\_t es el tipo ideal para recorrerlos:

int numeros[] = {10, 20, 30, 40};

size\_t n = sizeof(numeros) / sizeof(numeros[0]);

for (size\_t i = 0; i < n; i++) {

cout << numeros[i] << " ";

}

**3. En funciones de la biblioteca estándar**

Muchas funciones del lenguaje utilizan size\_t para parámetros o valores de retorno, por ejemplo:

* strlen() → devuelve la longitud de una cadena (size\_t)
* malloc() → recibe un argumento de tipo size\_t
* fread() y fwrite() → utilizan size\_t para manejar tamaños de bloques

Ejemplo:

#include <cstring>

#include <iostream>

using namespace std;

int main() {

const char\* texto = "Hola Isabela";

size\_t longitud = strlen(texto);

cout << "La cadena tiene " << longitud << " caracteres." << endl;

return 0;

}