



Ejercicio 5

Docente: Jimmy Nataniel Requena Llorentty

Materia: Programación III

Carrera: Ingeniería En Sistemas

Estudiantes: Joaquin Marcos Maita Flores

Santa Cruz – Bolivia

2025

UN MOSTRAR () PARA GOBERNARLOS A TODOS

El Problema: Queremos una función que pueda mostrar diferentes tipos de datos en la consola (enteros, flotantes, texto, incluso objetos más complejos) pero queremos llamarla siempre mostrar(), por simplicidad.

```
#include <iostream>
#include <string>
#include <vector>

// Declaraciones de las funciones 'mostrar' (prototipos)
void mostrar(int valor); // un entero
void mostrar(double valor); // un decimal
void mostrar(const std::string& valor);
/* una cadena de texto en donde <const> indica que la función no modificará la
cadena recibida; <std::string> es el tipo de dato;
<&> significa que la cadena se pasa por referencia, no por copia*/
void mostrar(char valor); // nos da un caracter
void mostrar(const std::vector<int>& miVector); // es un vector de enteros

int main() {
    std::cout << "--- Demostracion de 'mostrar' sobrecargado ---" << std::endl;
    mostrar(100);
    mostrar(3.14159);
    mostrar(std::string("Hola desde Programacion III!")); // le valor equivale a
los dicho en "Hola desde Programacion III!"
    mostrar('Z');

    std::vector<int> numeros = {10, 20, 30, 40, 50}; // este es el vector de
enteros en que se mostrara
    mostrar(numeros);

    // Llamada con un literal de cadena de C (const char*)
    // El compilador puede convertirlo a std::string si no hay otra sobrecarga
mejor
    mostrar("Esto es un literal de C-string.");
    std::cout << "\nJoaquin Marcos Maita Flores" << std::endl;

    return 0;
}

// Implementaciones de las funciones 'mostrar'
// aqui van los valores que ya fueron tomados o puestos en inr main mostrar
void mostrar(int valor) {
```

```
    std::cout << "Tipo Entero (int): " << valor << std::endl;
}

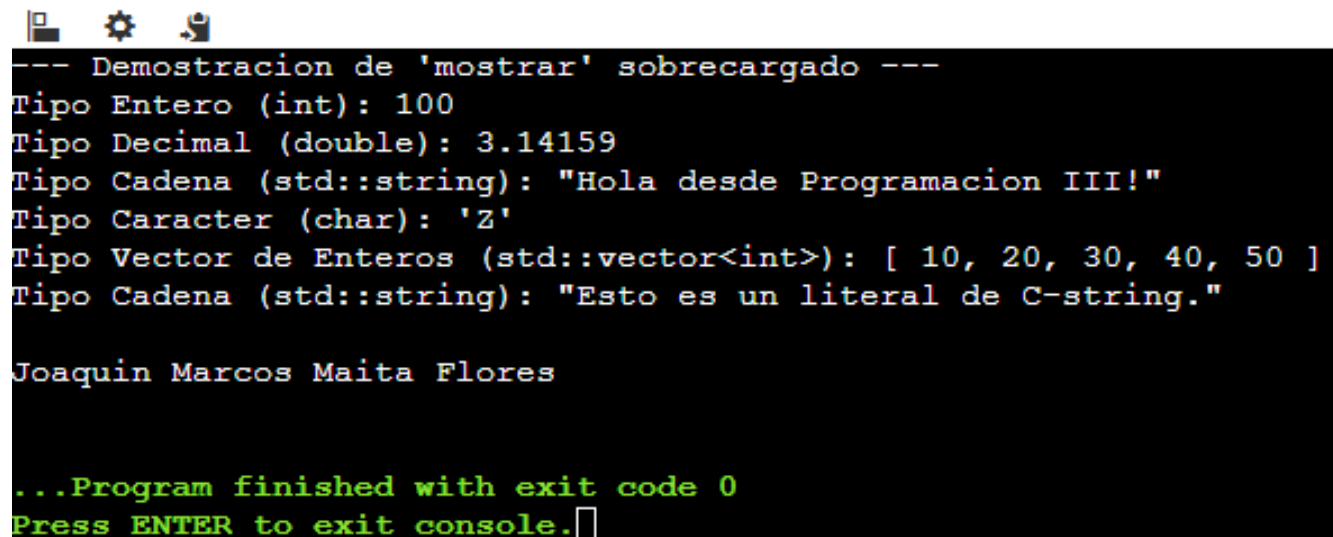
void mostrar(double valor) {
    std::cout << "Tipo Decimal (double): " << valor << std::endl;
}

void mostrar(const std::string& valor) {
    std::cout << "Tipo Cadena (std::string): \"" << valor << "\"" << std::endl;
}

void mostrar(char valor) {
    std::cout << "Tipo Caracter (char): '" << valor << "'" << std::endl;
}

void mostrar(const std::vector<int>& miVector) {
    std::cout << "Tipo Vector de Enteros (std::vector<int>): [ ";
    for (size_t i = 0; i < miVector.size(); ++i) {
        std::cout << miVector[i] << (i == miVector.size() - 1 ? "" : ", ");
    }
    std::cout << " ]" << std::endl;
}
```

CÓDIGO EJECUTADO



```
--- Demostracion de 'mostrar' sobrecargado ---
Tipo Entero (int): 100
Tipo Decimal (double): 3.14159
Tipo Cadena (std::string): "Hola desde Programacion III!"
Tipo Caracter (char): 'Z'
Tipo Vector de Enteros (std::vector<int>): [ 10, 20, 30, 40, 50 ]
Tipo Cadena (std::string): "Esto es un literal de C-string."

Joaquin Marcos Maita Flores

...Program finished with exit code 0
Press ENTER to exit console.
```

Este ejemplo enseña:

- Qué es la **sobrecarga de funciones**.
- Cómo manejar **diferentes tipos de datos** con el mismo nombre de función.
- Cómo usar **`std::string` y `std::vector`**.
- Cómo el compilador selecciona la función correcta según el **tipo de argumento**.