Practical Exercise — Generic Data Store

Objective

Implement a **Generic Data Store** in C++ that combines function templates, class templates, and template specialization to demonstrate the practical application of generic programming

This exercise will demonstrate how to:

- *Create **generic classes** using class templates.*
- *Create **generic function** using function templates.*
- *Use **template specialization** to customize behavior for specific data types.*
- *Store and retrieve data of different types*

Requirements

- Create a class template named **DataStore.**
- Implement both **Default** constructor and **Parameterized** constructor.
- Define public method – ***setData (), getData ()** and **display ().***
- Implement a function template named ***printValue ()*** that works with any data type.
- Provide a **specialized** version of the ***display ()*** function for the **string data type**.

Solution

```
#include <iostream>
#include <string>
using namespace std;

// Class Template
template <typename T>
class DataStore {
    private:
        T data;

    public:
        // Default Constructor
        DataStore () {
        data = T ();
        }

        // Parameterized Constructor
        DataStore (T value) {
            data = value;
        }

        // Setter Method
        void setData (T value) {
            data = value;
        }

       // Getter Method
        T getData () {
            return data;
        }

        // Display Method
        void display () {
        cout<<"Stored Value: "<<data<<endl;
        }
};
```

```cpp
// Function Template
template <typename T>
void printValue (T value) {
    cout << value << endl;
}

// Template Specialization for string
template <>
class DataStore<string> {
    private:
    string data;

    public:
        // Default Constructor
        DataStore () {
            data = "";
        }

        // Parameterized Constructor
        DataStore (string value) {
            data = value;
        }

        // Setter Method
        void setData (string value) {
            data = value;
        }

        // Getter Method
        string getData () {
            return data;
        }

        // Specialized Display Method
        void display () {
            cout<<"Stored Text: "<<data<<endl;
        }
};
```

```
int main () {

    // Integer Data Store
    DataStore<int> intStore (100);

    // Float Data Store
    DataStore<float> floatStore (25.75f);

    // String Data Store
    DataStore<string> stringStore ("Database Systems");

    cout << "Displaying Stored Data" << endl;
    cout << "----------------------" << endl;

    intStore.display();
    floatStore.display();
    stringStore.display();

    cout << "\n Using Function Template" << endl;
    cout << "----------------------" << endl;

    printValue (intStore.getData());
    printValue (floatStore.getData());
    printValue (stringStore.getData());


    return 0;
}
```