- They are defined using the template keyword followed by template parameters enclosed in angle brackets (<>).
- Template parameters can be type parameters or non-type parameters.
- Syntax:

```
template <typename T>
T functionName(T parameter) {
    // Function body
}
```

- Example:

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

// Function template for adding two values of the same type
template <typename T>
T add(T a, T b) {
    return a + b;
}

int main() {
    // Instantiation for int
    int result1 = add(5, 3);
    cout << "Result 1: " << result1 << endl;

    // Instantiation for double
    double result2 = add(3.5, 2.7);
    cout << "Result 2: " << result2 << endl;

    return 0;
}
```

In this example, the function template add can add two values of any data type (int, double, etc.).

2.3 Class Templates

- Definition: Class templates allow the creation of generic classes that can work with any data type. They are instantiated to create specific classes for each data type when used.
- They are defined similar to function templates, but instead of functions, entire classes are templated.
- Syntax:

```
template <typename T>
class ClassName {
    // Class members and methods
};
```

- Example:

```
#include <iostream>
```

```
using namespace std;

// Class template for a generic Pair
template <typename T1, typename T2>
class Pair {
private:
    T1 first;
    T2 second;
public:
    Pair(T1 f, T2 s) : first(f), second(s) {}
    T1 getFirst() const { return first; }
    T2 getSecond() const { return second; }
};

int main() {
    // Usage of Pair class template with different types
    Pair<int, double> pair1(5, 3.14);
    cout << "First: " << pair1.getFirst() << ", Second: " <<
pair1.getSecond() << endl;

    Pair<char, string> pair2('A', "Hello");
    cout << "First: " << pair2.getFirst() << ", Second: " <<
pair2.getSecond() << endl;

    return 0;
}
```

In this example, the class template Pair represents a generic pair of two values of potentially different types (int, double, char, string, etc.).

Class Templates vs. Function Templates

Aspect	Class Template	Function Template
Purpose	Class templates are used to create generic classes	Function templates are used to create generic functions
Syntax	template <typename T> class ClassName	template <typename T> returnType functionName
Usage	Class templates are instantiated with specific data types when creating objects	Function templates are instantiated with specific data types when calling the function
Example	Class templates define structure and behavior of generic classes	Function templates define logic of generic functions