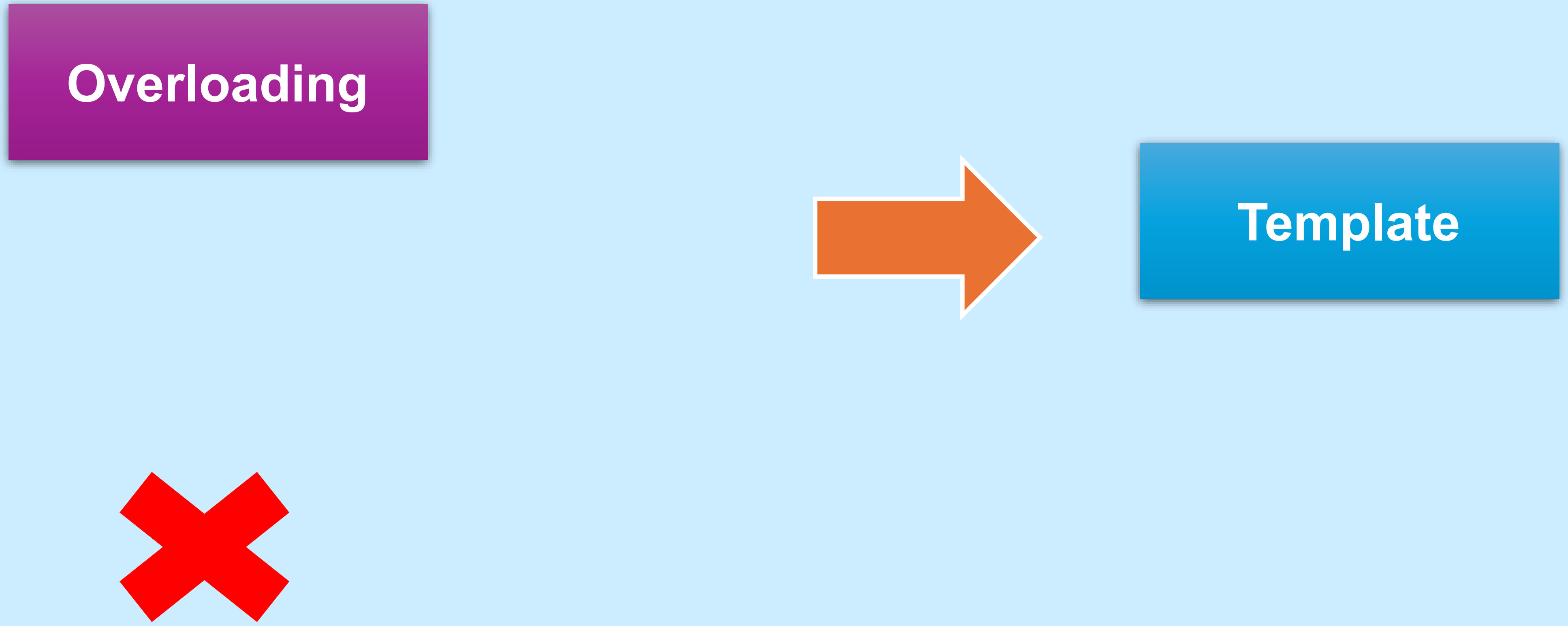
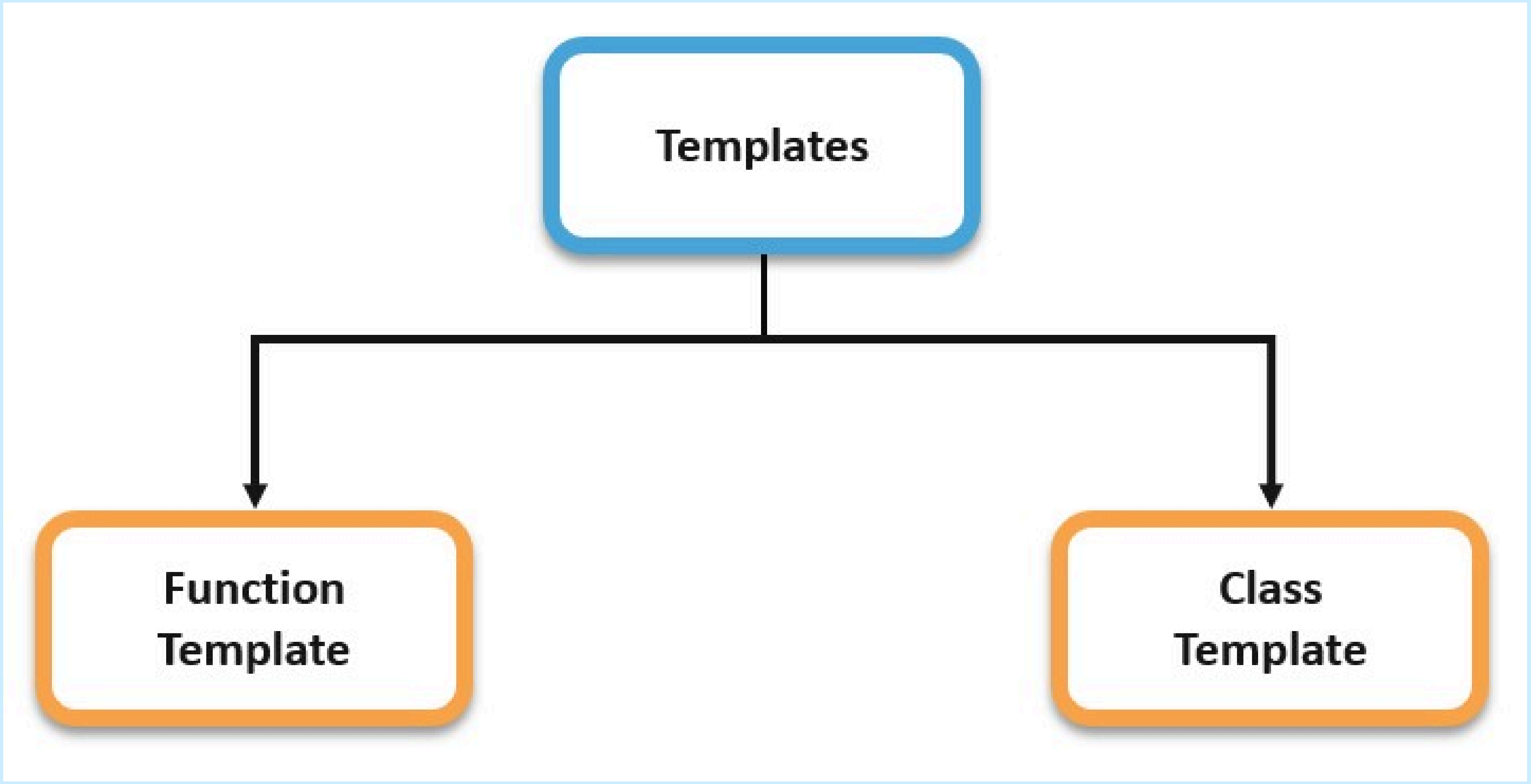
Introduction to Template in C++

What are Templates?

- Definition: Templates allow writing generic and reusable code
- Purpose: Write once, use with different data types
- Types:
  - Function Templates
  - Class Templates
- Benefits:
  - Code reusability
  - Type safety
  - Performance (compile-time resolution)

Why Templates?




Templates
Function Template
Class Template

1] Function Templates

- Function templates define a blueprint for functions that enables a function to operate on different data types without rewriting the same logic.
- The syntax for the general form of a template function definition is:

```
template <typename T>

return_type function_name(T parameter) {

  // code

}
```

1] Function Templates (cont.)

- Code Example:

```
template <typename T>
T maximum(T a, T b) {
    return (a > b) ? a : b;
}
```

1] Function Templates (cont.)

- Code Example:

```
// Usage
int main() {
    cout << maximum(5, 10) << endl;      // int
    cout << maximum(3.14, 2.71) << endl; // double
    cout << maximum('a', 'z') << endl;   // char
}
```

2] Class Templates

- Similarly, class templates also define a blueprint for creating classes that can work with any data type.

```
template <typename T>

class ClassName {

  // members and methods using T

};
```

2] Class Templates (cont.)

- Code Example 1:

```cpp
template <typename T>
class Box {
  public:
    T value;
    Box(T v) {
      value = v;
    }
    void show() {
      cout << "Value: " << value << "\n";
    }
};

int main() {
  Box<int> intBox(50);
  Box<string> strBox("Hello");

  intBox.show();
  strBox.show();
  return 0;
}
```

2] Class Templates (cont.)

- Code Example 2:

```cpp
template <typename T1, typename T2>
class Pair {
  public:
    T1 first;
    T2 second;

    Pair(T1 a, T2 b) {
      first = a;
      second = b;
    }

    void display() {
      cout << "First: " << first << ", Second: " << second << "\n";
    }
};

int main() {
  Pair<string, int> person("John", 30);
  Pair<int, double> score(51, 9.5);

  person.display();
  score.display();

  return 0;
}
```