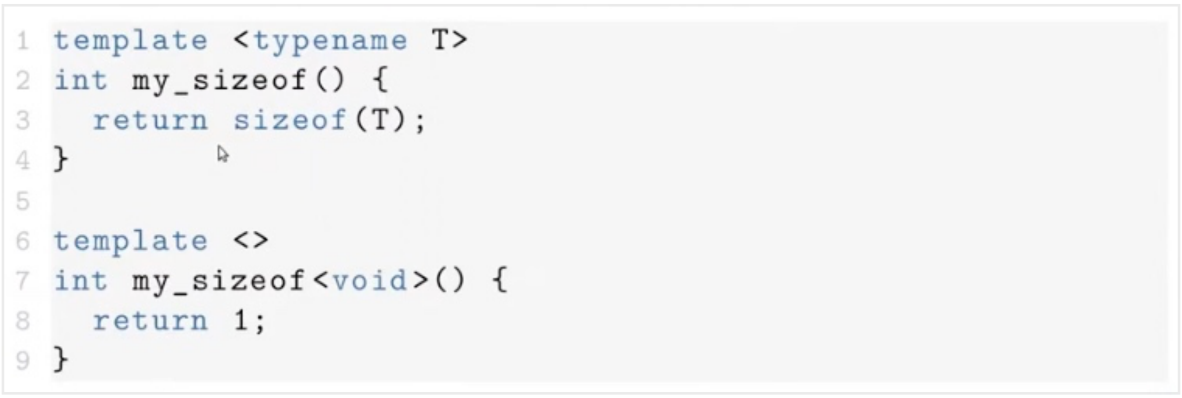
Template Specialization

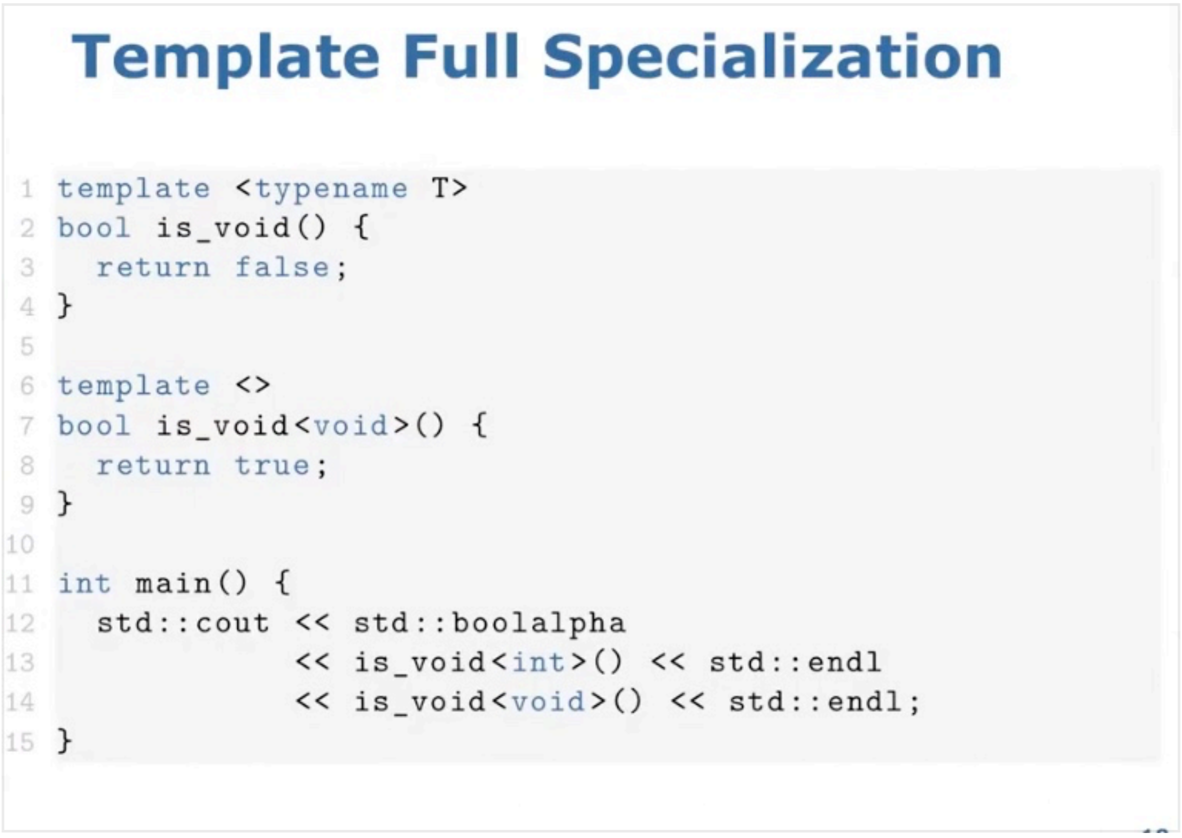
Template Full Specialization

- Prefix the definition with template<>
- Then write the function definition
- Usually means you don’t need to write any more angle brackets at all
- Unless T can’t be deducted



```cpp
template <typename T>
int my_sizeof() {
  return sizeof(T);
}

template <>
int my_sizeof<void>() {
  return 1;
}
```



Template Full Specialization

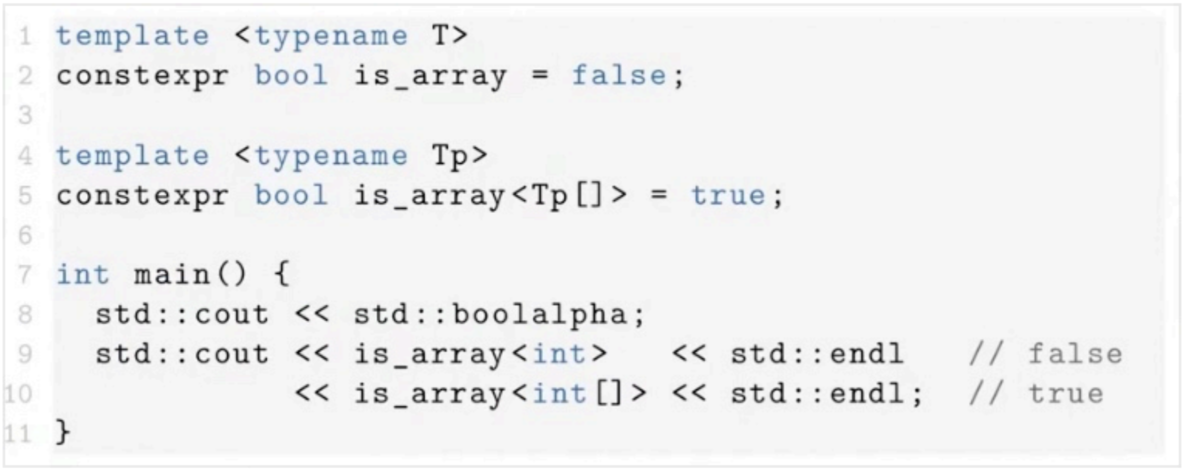
```cpp
template <typename T>
bool is_void() {
  return false;
}

template <>
bool is_void<void>() {
  return true;
}

int main() {
  std::cout << std::boolalpha
            << is_void<int>() << std::endl
            << is_void<void>() << std::endl;
}
```

Template Partial Specialization

A partial specialization is any specialization that is, itself, a template. It still requires further “customization” by the user before it can be used.



```cpp
template <typename T>
constexpr bool is_array = false;

template <typename Tp>
constexpr bool is_array<Tp[]> = true;

int main() {
  std::cout << std::boolalpha;
  std::cout << is_array<int>   << std::endl   // false
            << is_array<int[]> << std::endl;  // true
}
```