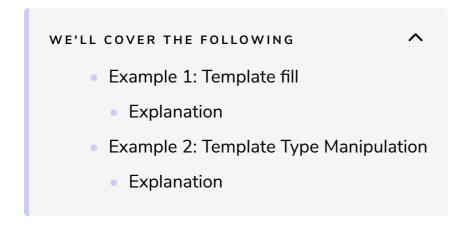# - Examples

Let's have a look at a couple of examples of type-traits.

Example 1: Template fill

```cpp
// templatefill.cpp

#include <cstring>
#include <chrono>
#include <iostream>
#include <type_traits>

namespace my{

  template <typename I, typename T, bool b>
  void fill_impl(I first, I last, const T& val, const std::integral_constant<bool, b>&){
    while(first != last){
      *first = val;
      ++first;
    }
  }

  template <typename T>
  void fill_impl(T* first, T* last, const T& val, const std::true_type&){
    std::memset(first, val, last-first);
  }

  template <class I, class T>
  inline void fill(I first, I last, const T& val){
    typedef std::integral_constant<bool,std::is_trivially_copy_assignable<T> ::value && (size
    fill_impl(first, last, val, boolType());
  }
}

const int arraySize = 100000000;
char charArray1[arraySize]= {0,};
```

```
char charArray2[arraySize]= {0,};

int main(){

  std::cout << std::endl;

  auto begin= std::chrono::system_clock::now();
  my::fill(charArray1, charArray1 + arraySize, 1);
  auto last=  std::chrono::system_clock::now() - begin;
  std::cout <<  "charArray1: " << std::chrono::duration<double>(last).count() << " seconds" <

  begin= std::chrono::system_clock::now();
  my::fill(charArray2, charArray2 + arraySize, static_cast<char>(1));
  last=  std::chrono::system_clock::now() - begin;
  std::cout <<  "charArray2: " << std::chrono::duration<double>(last).count() << " seconds" <

  std::cout << std::endl;


}
```

Explanation

In line 26, `my::fill` make the decision as to which implementation of `my::fill_impl` is applied. To use the optimized variant, the elements should have a compiler generated copy assignment operator `std::is_trivially_copy_assignable<T>` and should be 1 byte large: `sizeof(T) == 1`. The function `std::is_trivially_copy_assignable` is part of the type-traits library.

If the expression `boolType()` in line 26 is true, the optimized version of `my::fill_impl` in the lines 17 - 20 will be used. This variant fills in opposite of the generic variant `my::fill_impl` (line 10-16) the entire memory area - consisting of 100 million entries - with the value 1. `sizeof(char)` is 1.

What's about the performance of the program? We compiled the program with full optimization. The execution of the optimized variant is about 3 times faster on windows; about 20 times faster on Linux.

Example 2: Template Type Manipulation

```
// templateTypeManipulation.cpp

#include <iostream>
#include <type_traits>
```

```cpp
template <typename T>
struct RemoveConst{
    typedef T type;

};

template <typename T>
struct RemoveConst<const T>{
    typedef T type;
};


int main(){

    std::cout << std::boolalpha << std::endl;

    std::cout << "std::is_same<int, RemoveConst<int>::type>::value: " << std::is_same<int, Re
    std::cout << "std::is_same<int, RemoveConst<const int>::type>::value: " << std::is_same<i

    std::cout << std::endl;

}
```

Explanation

The code above uses the function `std::is_same` from the `type-traits` library. `std::is_same` compares the type passed in the function and the type given in the function defined by us, and it returns `true` only when both types are the same.

In the next lesson, we'll solve exercise on type-traits.