

- Solution

The solution to the type traits exercise of the previous lesson.

WE'LL COVER THE FOLLOWING ^

- Solution
- Explanation
- Further information

Solution

```
// typeModifications.cpp
#include <iostream>
#include <type_traits>

int main(){

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

    std::cout << "std::is_const<std::add_const<int>::type>::value: " << std::is_const<std::add_const<int>::type>::value << std::endl;
    std::cout << "std::is_const<std::remove_const<const int>::type>::value: " << std::is_const<std::remove_const<const int>::type>::value << std::endl;

    std::cout << std::endl;
    typedef std::add_const<int>::type myConstInt;
    std::cout << "std::is_const<myConstInt>::value: " << std::is_const<myConstInt>::value << std::endl;
    typedef const int myConstInt2;
    std::cout << "std::is_same<myConstInt, myConstInt2>::value: " << std::is_same<myConstInt, myConstInt2>::value << std::endl;

    std::cout << std::endl;

}
```



Explanation

- In line 7, due to the flag `boolalpha` in line 10, the program displays either `true` or `false` instead of 1 or 0.

- In line 9, we used `std::add_const<int>` to add `const` to `int` and checked it using `std::is_const`.
- In line 10, we used `std::remove_const<const int>` to remove `const` from `const int` and checked it using `std::is_const`.
- In lines 13-14, we defined a `const int myConstInt` using `std::add_const<int>::type` and checked it using `std::is_const`.
- In lines 15-16, we defined a `const int myConstInt2` using `const int` keyword and checked to see that it is the same as `MyConstInt` using `std::is_same`.

Further information

- [Type_traits](#)
 - [Variations of gcd algorithm](#)
-

This concludes our discussion on utilities. In the next chapter, we will learn about smart pointers.