

Callable Objects Solutions

std::function

- What is the purpose of std::function?
 - std::function is a generic function object which can represent any type of callable object
 - std::function allows us to pass callable objects as arguments without having to write an overload for each kind of callable object
 - std::function also allows us to create containers of callable objects and populate them with assorted types of callable objects

Limitations of `std::function`

- Are there any limitations to `std::function`?
 - The type of the callable object is erased
 - The parameter must exactly match the signature of the callable object
 - Invoking a callable object through `std::function` has runtime overhead, because the call is made indirectly
 - `std::function` may allocate memory on the heap, if the callable object is too large to be stored inside the `std::function` object
- What alternatives are there to using `std::function`?
 - For storing a callable object in a variable, use `auto`
 - This retains the type information
 - Invoking the callable object has no runtime overhead, because the call is made directly

count_strings

- Write a `count_strings` function similar to `std::count_if` which
 - Takes a vector of `std::string` and a function pointer
 - Calls the function on each element
 - If the function returns true, increments a counter
 - Returns the final value of the counter
- Write a program to test your code. It should work correctly with this function

```
bool match(const string& test) {  
    return test == "cat";  
}
```

std::function

- Modify `count_string()` to use `std::function` instead of a function pointer. Check that it still works
- Modify the program to pass a functor to `count_string()`. Check that it still works
- Modify the program to pass a lambda expression to `count_string()`. Check that it still works

std::bind

- Using this version of match, modify the program to use an object returned from calling std::bind()

```
bool match(const string& text, const string& value) {  
    return text == value;  
}
```