Using if In A Search Statement

Let's test the 'if' statement feature we discussed in the last lesson.

We learned that C++17 allows us to specify a new variable which can be used in the condition. This variable is inside the `if` condition scope:

```
if (auto val = GetValue(); condition(val))
  // on success
else
  // on false...
```

Let’s see if this is useful. Say you want to search for a few things in a string:

Method 1 | Method 2 | Error

```
#include <iostream>
using namespace std;

int main() {
  // your code goes here
  const std::string myString = "My Hello World Wow";
  const auto pos = myString.find("Hello");
  if (pos != std::string::npos)
    std::cout << pos << " Hello\n";
    const auto pos2 = myString.find("World");
  if (pos2 != std::string::npos)
    std::cout << pos2 << " World\n";
}
```

As you can see, you have to use different names for `pos` or enclose it with a separate scope, otherwise the code will fail as it does in the “Error” tab above.

The new if statement will make that additional scope in one line:

```
#include <iostream>
using namespace std;
```

```
int main() {
  const std::string myString = "My Hello World Wow";
  if (const auto pos = myString.find("World"); pos != std::string::npos)

    std::cout << pos << " World\n";
  else
    std::cout << pos << " not found!!\n";
}
```

As mentioned before, the variable defined in the if statement is also visible in the `else` block. So you can write:

```
if (const auto pos = myString.find("World"); pos != std::string::npos)
    std::cout << pos << " World\n";
else
    std::cout << pos << " not found!!\n";
```

Plus, you can use it with structured bindings (following Herb Sutter code):

```
// better together: structured bindings + if initializer
if (auto [iter, succeeded] = mymap.insert(value); succeeded) {
  use(iter); // ok
  // ...
} // iter and succeeded are destroyed here
```

In the above example, you can refer to `iter` and `succeeded` rather than `pair.first` and `pair.second` that is returned from `mymap.insert`.

As you can see, structured bindings and tuples allow you to create even more variables in the init section of the if-statement. But is the code easier to read that way?

For example:

```
string str = "Hi World";
if (auto [pos, size] = pair(str.find("Hi"), str.size()); pos != string::np
os)
    std::cout << pos << " Hello, size is " << size;
```

We can argue that putting more code into the init section makes the code less readable, so pay attention to such cases.

> *Extra Info:* The change was proposed in: P0305R1.

In the next lesson, we will discuss inline variable initialization.