Beginning C++ Programming

Pointer Arithmetic

At around 7:25 in this video I say that the dereference operator and the post-increment operator have the same precedence – this is incorrect.

Let me take this one step at a time, since it can be confusing at first.

`*score_ptr++` has 2 operators, `*` and `++`, the **dereference** operator and the **postfix increment** operator.

Since we have 2 operators with **different** precedence, we use precedence **NOT** associativity.

So, we know the precedence of the postfix increment operator is greater than that of the dereference operator.

So, this becomes equivalent to: `*(score_ptr++)`

Now, this means that we are dereferencing `score_ptr`, but we also know that we are incrementing `score_ptr` .
Since the increment is a post-increment, then the effect of `*score_ptr++` is

1. *score_ptr
2. increment score_ptr

Hope that makes sense.

see
https://stackoverflow.com/questions/18481740/pointer-expressions-ptr-ptr-and-ptr

```
*ptr++   // effectively dereferences the pointer, then increments the pointer
*++ptr   // effectively increments the pointer, then dereferences the pointer
++*ptr   // effectively dereferences the pointer, then increments dereferenced value
(*ptr)++ // effectively forces a dereference, then increments dereferenced value
```

Now, how would you apply this to the following?

`*++score_ptr`

In this case, the **dereference** operator and the **pre-increment** operator have the **SAME** precedence.
So, now we use **associativity** to determine what binds with what.

Pre-increment AND dereference associate **right-to-left**.

The rightmost operator is ++, so it binds to score_ptr first then the dereference.
So, this becomes equivalent to: *(++score_ptr)

And using what we know about pre-increment the effect is

1. increment score_ptr
2. *score_ptr

Thanks to Aditya, Clem, and Francisco for asking and pointing this out!

Best regards,
Frank Mitropoulos