**INFORMAL CPP NOTES**

Ternary operators:

<something A> ? <something B> : <something C>

The question mark and the colon both achieve something. It is essentially a “if-elif” shorthand.

Basically is says if “something A” is true, then we return “something B”. If not, then we return “something C”

Ex.

#define MAX(a, b) (((a) > (b)) ? (a) : (b))

So the example I found uses a pre-processing macro literally means before the compiling starts, the compiler will check out the code and replace every single instance of your macro with what you defined it as. So in this case, we defined our macro as MAX(a, b), so whenever we use MAX(a, b), we will actually be using the macro!

Anyways, back on topic. So for our ternary operators we have ((a) > (b)) which is our Boolean statement that we see if it is true or not.

If it IS true, then we return (a).

If false, then we return (b)

So in practice:

Int x = MAX(4, 5);

cout << x; // outputs to 5!

One thing to note about ternary operators. While it is a bit…nuanced and we have to get use to the syntax, it is pretty convenient to use for simple one-offs/for loops.

HOWEVER for debugging, it will NOT stop at the sub-expressions. So if the ternary operator is giving you an error, and you are depending on the debugger, it will not break it up for you and help you catch the bug.

Pointers.

Pointers are strange in the fact that they are like variables but not. It is essentially a lower lever form of variables. It contains the address of the thing you want to point at.

So.

int \*cool = &q

cool = the MEMORY ADDRESS OF “q”. So in essence it is a variable, since a variable points to a value.

To get the value from the pointer, we simply add the “\*” to dereference the memory address and gets the value from there.

This is useful because we can point at things that may be dynamic/will change over time! So we can continually use the pointer name to refer to the dynamic object. Just think of pointers as addresses to houses. The variable/value = house and the pointer is the address.

Now WHY use pointers? Instead of the just the reference to the memory directly? Instead of doing \*cool = &q, why not just &q?

Like with most things, it really depends on your situation. But as you get deeper and deeper into C++, pointers seem to be indispensable. Unfortunately I am not equipped to deal with that explanation, because I’m still trying to understand the use of them/the practical side of it.

It seems, like with most intermediate programming things, you need to play with it and break things before understanding everything. Because theory and bytes are nice, but nothing beats playing around.

References:

Friend:

Virtual Function:

**ABSOLUTE C++ (5th ed)**

**Chapter 1.**

Variables.

Don’t start your variables out with an underscore (“\_”). Those are (informally) reserved for system identifiers and standard libraries. Just a style thing to help make your code more readable/identify what is actually being declared or used. Also capital letters MATTER. You can make them NOT matter, but by default, they matter

Keywords.

There are keywords in C++ that are hard-wired so you can’t change it. There are also pre-defined identifiers, like “cout” or “cin” which are changeable but HIGHLY RECOMMENDED NOT TO CHANGE. This is because it is used often enough that it will lead to horrible no good problems if you do change the meaning/value of them.

On Style

Above all, keep consistent. I mean try to incorporate the most clear and logical style, but keep consistent. Easier to fix if needed. Like camelCasing variable names, instead of under\_scoring variable names.

Libraries and namespaces.

Libraries are essentially pre-defined functions and operators. The libraries store their definitions in namespaces. Basically, it’s hard to tease out the difference between the two, since they are interconnected. Just realize that libraries need namespaces to work, and namespaces are simply a collection of definitions.

We use “include” to use a library. And use “using namespace” to use a specific namespace.

Such as “using namespace std;”

Or if you only want a specific names in the namespace:

“using namespace std::cout;”

And you can use “cout” without the “std::” when using it.