C and C C have the concept of macros. In modern programming we try to stay away from macros as much as possible because they are not really part of the codes, they are processed in a very different way. However, you are very likely to see macros and so we'll go over some of the basic macros as part of it. So there is a set of variables that are already defined by the compiler. And when there is a one of these variables is present at the compilation time, the text is replaced by the value, whatever the value is into the code and then the code is compiled.

So macros are done in the pre processing stage and they are used as a kind of a find and replace model of code. So looking at the predefined macros, and these are just a few of the ones that's in C, we can have a look at the date, which is June 17th for the recording of this video, the compilation time. So this is at compilation the time as was inserted into where this file was and into where this time variable was. Similarly, the file name can can be used and the line number as part of it, what function we are part of and even what the C C standard version we are using is looking more into macros. A lot of times macros are used to define constants.

A better way to define a constant is, for example, if you want to define PI, you might want to use constants, floats or constant double or constant unsigned, long or whatever is the appropriate data type. When we define PI like this, what it means is that every time the compiler sees this variable, not variable, this macro PI, it's going to substitute this string to wherever it sees it. So before even the compilation starts, the text find and replace operation, the pre processing stage and we'll replace this PI with this number that's over there. So now we can say, hey, when we print it we get value of PI is 3.14159, but we have actually given it a much longer one. The reason it's not printing it is because of the defaults about how.

See how it is working. So if we set the precision to print 20 digits after the decimal points, we get some more of it. We can also define macros so that it does some kind of calculation. For example, and this is a bad macro, don't, don't repeat it and decide that we use the square operation a lot. So, so square of anything is going to be X times X.

You will notice that this macro doesn't have any data types. It is not C code. It is find and replace type of code. So when we see square five, this square five is going to get replaced with open parentheses five times five. And so that is done and then the compiler takes care of the rest.

So in this case, square of five gives us 25, which is great, but this is where the danger of macros come. So we want to calculate the square of 5 plus 1. Well, 5 plus 1 is 6, the square should be 36. But when we use square in parentheses 5 plus 1, what happens is in the text processing, the pre processing stage, this gets replaced with five plus one times five plus one. So we have five plus one times five plus one dot and because multiplication takes precedence over addition, we are going to end up multiplying the ones first and then doing the addition.

And as a result, now we get square of five plus one is 11. And this can be very difficult to debug. So there is a better way to do the square. This involves using parentheses. So instead of just writing x times x, we have in parentheses X times in parentheses X.

So now the better square of two plus one gives us nine, as expected. Still, macros are notoriously difficult to debug and you don't give the compiler a chance to catch mistakes. And, and they are very likely to introduce bugs. So try to stay away from macros as much as possible. But one macro that you still see in some code, sometimes legacy code, is something like a logging function.

So here's a define log and the macro has to be all in one line. So because of so for that we have a backspace here. And anytime we see a log and something in parentheses, we are going to replace it with cout file name colon, line name, debug colon and then X, which means the variable, the name of the variable rather than the value of the variable equals the value of the variable X. So now if we have integer total equals 100, and we said log total, and we would get on the output line the file name colon36 because it was on line 36. Debug colon total.

The variable name that we used equals 100. So people still use it as part of debugging. You will still see it in some code. In general, try to stay away from the macros, but sometimes it's difficult to avoid. One area that you do end up using macros is for the H files for the header guards.

If not defined 3H. If not define 3H, you say define it and then you have a whole bunch of code and then you, you say endif. That way if the H file is included multiple times because of the way pre processing and text substitution works. It's only get going to get included once and that's it for macros.