

6 Where are we now?

We have covered much of Chapters 1-4.

Things we have not (yet?) covered from those chapters are as follows.

- We have not really covered the “preprocessor” as such.

Lines that start with a splat (pound sign) (**#**) are not really part of the C language. They are translated by the C preprocessor BEFORE the C compiler does the compilation on the code. For the moment, though, since we have used only (**#include**) and (**#define**), you can almost think of these as being part of C.

We will get to the preprocessor later. What the preprocessor lets you do is write one program that the preprocessor uses to generate more than one different version of a program. That is, you can “program” what it is that your program will be.

- We have not discussed formatted input or output (pp. 54ff). That will come later.

- We did cover redirection of input and output with the `<` and `>`, but we did not cover the **FILE** data type (pp. 59ff). That will come later.
- We did not cover user-defined functions (pp. 100ff). That will come later.
- I have not gone into great detail on operator precedence for either logical expressions (pp. 118ff) or arithmetic expressions. I believe it is harmful to know too much about this, because it then leads to writing code that is unclear. If you don't learn the rules, you will be forced to parenthesize properly, which is a better thing.
- I have not covered “simple” versus “compound” statements (until right now). I generally use only compound statements on control structures, because it seems to me to be very rare that a simple statement suffices, and the compound statements are harder to screw up. (See page 133 for how to screw up.)
- I have not gone into excruciating detail on the details of syntax for nested **if** statements. If you use compound statements exclusively, then you don't need to know these rules, because the rules for compound statements are pretty obvious.
- I have not covered the **switch** statement (pp. 146ff) (until now).

An issue to think about

Many of you are having problems dealing with the fact that you are actually running programs and tools on multiple computers.

More on numerical things

The notion of “not a number.”

Numerical errors with floating point numbers.

Library functions (see page 98)

Compiler switches.