Learning to Read C Code

Another Useful Skill: Reading Code

- How to express types of problems.
- How to properly use application programming interfaces (APIs) for networking, mathematics, graphics, sound, animation, user interfaces, and so forth.
- How to make code easy to read (style).

It’s Often Necessary to Read Code to Understand It

We try to make you write plenty of comments.
When we give you code for class assignments, it will be well-commented (DISCLAIMER: THIS IS NOT A WARRANTY!)
In the real world...
  - You will be lucky to find comments.
  - Remember the Big Screw award?
  - You will be really lucky to find comments in a language that you understand.

Let’s Do an Exercise in Code Reading Together

Our next example has no topical comments and uses one-letter variable names.
Let’s figure out what it does.
For more exercises of this type,
  - use the ECE120 C Analysis tool.
  - But note that the tool
    - has only 14 examples.
  - Type an answer before you press ‘Check Answer.’
Structure is Similar to Previous Examples

Take a look at the program.

Basic structure is similar to previous examples:
- print a prompt,
- wait for input,
- check input for correctness,
- compute something, and
- print a result.

What Input is Expected?

Look at the following:
- the scanf format,
- the arguments (types must match),
- the error check and the error message.

As input, the program requires...
- two 2's complement numbers (%d) (variables A and C)
- separated by an ASCII character (%c) (variable B)

Now Look at the Computation

if-else structure with five cases.
- The last case is an error condition.
- The other four are ways of calculating variable D.

Notice that variable D is used for the final output.

When Does the Computation Print an Error?

The last case is reached when...
- B is NOT a '+', AND
- B is NOT a '-', AND
- B is NOT a '/', AND
- B is NOT a '*'.

In other words, the code generates an error unless the user enters +, -, /, or * as the character between two integers.
### How is D Computed?

First case: when \( B \) is `+`, \( D \) is \( A + C \).

Second case: when \( B \) is `-`, \( D \) is \( A - C \).

Third case: when \( B \) is `/`, \( D \) is \( A / C \).

Fourth case: when \( B \) is `*`, \( D \) is \( A * C \).

So ... the program is doing what?

**computing the value of an expression with one arithmetic operator**