1 Design Basics

Designing/Specifying a program is a difficult task!

Why?

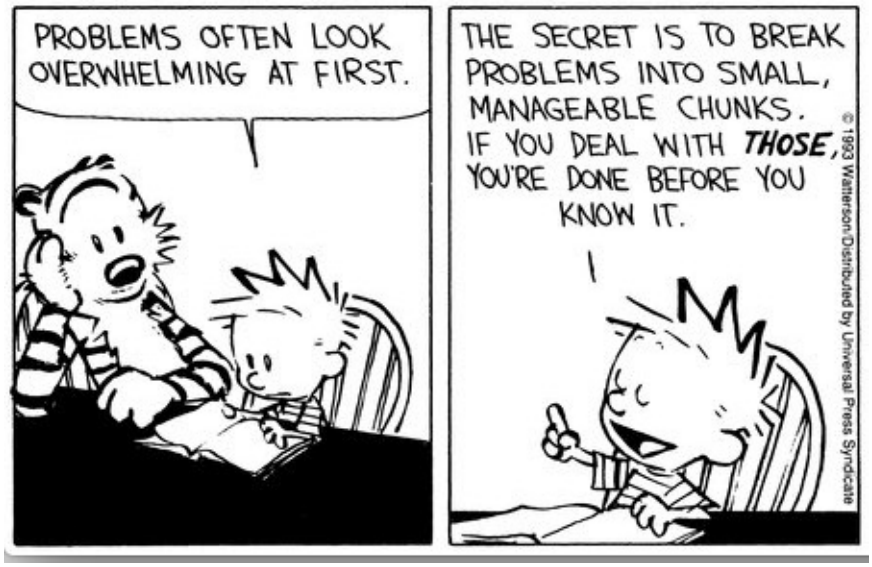
Designing/Specifying a program is a difficult task!

Why?

Have to define—in detail!—what is being done.

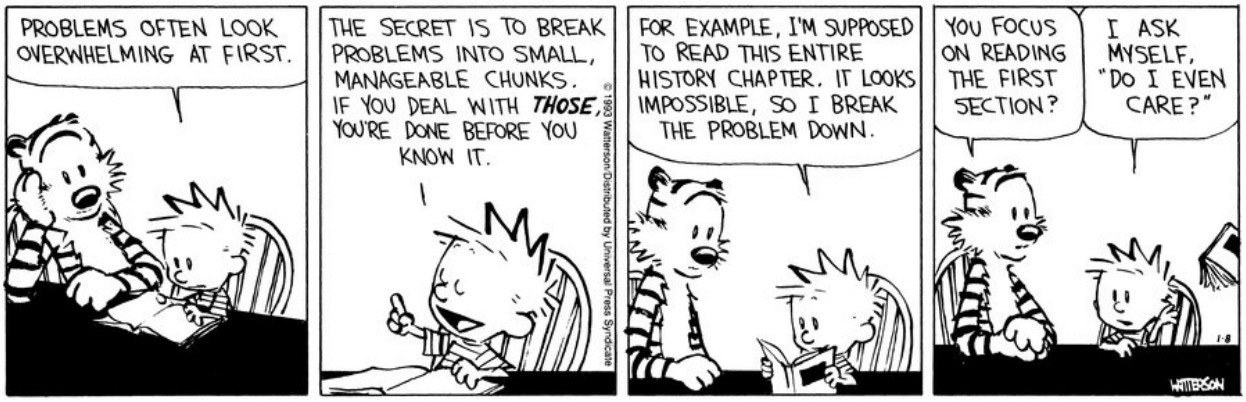
Can be tedious.

Design wisdom from Calvin:



`https://twitter.com/Calvinn_Hobbes/status/1085222787692646400/photo/1`

Real design wisdom from Calvin:



1.1 General Program Operation

The operation of almost all programs can be defined by the following sequence of operations:

```
Input/              -->    Calculation/    -->    Output/
Initialization             Work                   Cleanup
```

where:

Input: files or interactive
Output: files or graphical

1.2 Program Development Process

1. Identify required input(s) and output(s).
2. Develop *Test cases*
3. Perform *Sample Calculations*
4. Decide how output will be displayed (files or graphically).
5. Make an overall design of the program.
   Include the general method—the algorithm(s)—by which the program computes the output.
6. Refine the overall design by specifying more detail.
7. Write the program code.

1.3 Pseudocode

Given an overall design, including input, output, and any algorithm(s). How do we turn this into a program?

A common technique is to use *pseudocode*. Pseudocode is an informal program, in which there are few details, and there is no formal syntax.

Reading an integer

For example, a function to read an integer from a file in C++, might look like:

```
int ReadInteger( ifstream& fIn )
{
   int iTmp;

   fIn >> iTmp;

   return iTmp;
}
```

might be written in pseudocode as:

```
readInt
```

Some programmers convert their pseudocode into comments for the program they are writing. This provides a mechanism to look back at the design process when things go wrong/need to be changed.

Looping through a collection

We can iterate over an *array* using

```
for element in array { }
```

or we could repeatedly cycle through an array

```
loop:
while true {
    for element in array {
        // do something

        if someCondition {
            leave (break) loop
        }
    }
}
```

1.4 Flow Charts

Flow charts are a detailed diagram of how a program/function is to operate. However, many people find them tedious to draw. Fortunately, there are applications for drawing them.

Many older texts use a flow charts to describe various operations.