Problem Solving

How do you get from a set of requirements to a working program? There is no cut-and-dried recipe to follow. If there were, someone would write a program to do it, and computer programmers would be obsolete. However, there are general principles you can follow that will help to guide you toward a solution. (Based on *How to Solve it* by George Pólya.)

1. Make sure you have a clear understanding of all the requirements. In the real world, this is sometimes the most challenging part.
2. Plan tests you can use to check potential solutions for correctness.
3. Design your approach. One of the most commonly used methods for this is *stepwise refinement*. This is a process of breaking up the original problem into some number of smaller sub-problems. Next, you take each sub-problem and break it up into even smaller sub-problems. This is repeated until you get sub-problems that are small enough to be translated into code.
4. Translate your design into computer code (checking it with the tests from #2).
5. Reflect on how you arrived at your solution. What worked and what didn't? How can this experience help you solve future problems?

As an example of stepwise refinement, consider the problem of wanting to paint your house, which is a big job. It might seem overwhelming at first, but you can break that top-level problem into the following sub-problems:

- buy paint
- paint house
- clean up

Those sub-problems can in turn be broken down further:

- buy paint
  - choose colors
  - choose brand of paint
- paint house
  - tape over parts that shouldn't get painted
  - purchase/borrow supplies (ladders, rollers, etc.)
  - bribe friends with pizza
  - schedule a day/time
  - actually paint the house
- clean up
  - return/put away supplies
  - un-tape the taped bits
  - finish any fine detail work

This process continues until you've defined everything to a level of detail where you feel comfortable that you know everything that needs to happen. For programming, this process often involves pseudocode or flowcharts, which embody the steps required and which (once you've refined them to a sufficient level of detail) can then be translated into a computer language. See the Pseudocode and Flowchart Guidelines in this module.