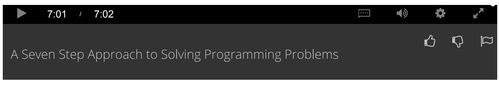
Which of the following is the correct sequence for the seven step approach?

- 1) Work example by hand. 2) Check by hand. 3) Write down what you did. 4) Translate to code. 5) Find patterns. 6) Run test cases. 7) Debug failed test cases.
- 1) Check by hand. 2) Work example by hand. 3) Find patterns. 4) Translate to code. 5) Run test cases. 6) Debug failed test cases. 7) Write down what you did.
- 1) Work example by hand. 2) Write down what you did. 3) Find patterns. 4) Check by hand. 5) Translate to code. 6) Run test cases. 7) Debug failed test cases.

Correct

- 1) Write down what you did. 2) Check by hand. 3) Work example by hand. 4) Translate to code. 5) Find patterns. 6) Run test cases. 7) Debug failed test cases

Continue




# A Seven Step Approach to Solving Programming Problems

Have a question? Discuss this lecture in the week forums.

## Interactive Transcript

Search Transcript English

0:03
Now we're going to talk a little bit more about solving programming problems. In the last lesson, we worked through the example of doing the green screen problem. We took a specific problem we wanted to solve, taking images and applying the green screen technique to them. And worked through step-by-step how to go from that problem statement to a working piece of code.

0:25
Now, what we're going to do in this lesson is we're going to take a closer look at solving programming problems. We're going to teach you the seven step approach that you're going to use to solve programming problems through the rest of this course and the rest of this specialization. As well as a tool that you can use to solve programming problems in general, whenever they come up in the rest of your life.

0:47
In general, solving a programming problem starts with a problem statement. I want to be able to take a foreground image and a background image and combine them using the green screen technique or anything else you might want to do. And ends when you have working code that solves your problem. Now, doing this is a big leap. It takes a lot of work to go from a problem statement to a working piece of code. And you may not be able to just do this in your head all at once. Sometimes, when a problem becomes easy for you as you become more skilled, you'll be able to just do this and it will happen naturally and you won't need to worry about it. However, the seven step approach we're going to teach you is always going to be a good way to approach any problem where you can't just see the answer right away.

1:31
So these are the seven steps that you're going to use to solve programming problems. And we're going to look at each of them and see how they work. You've seen this already with the green screen example. We just didn't talk about the details of each step, we just did the process.

1:45
The first thing to do is to solve a small instance by hand. If you remember from the green screen lesson that we just did, we took a four-pixel image. We didn't want to start with the millions of pixels that are in a real image, we just worked through for a very small instance. Something manageable.

2:04
If you have trouble with this step, maybe the problem is unclear. The problem statement doesn't actually tell you how to do things. In which case, you need to find out what you're supposed to do before you can proceed. In a classroom situation, this may involve asking your teacher or teaching assistant for clarification. In the real world, this may involve talking to your technical lead, your customer, or refining the problem yourself to figure out exactly what it is you need to do.

2:30
The other possibility if step 1 is difficult, is that you might need domain knowledge. If you're trying to solve a programming problem related to physics but you don't know the physics equations involved, then no amount of programming expertise will help you. You need to consult a physics textbook, a Wikipedia article on physics, or a physics professor to understand the physics before you can try to solve your programming problem.

2:55
Once you've completed step 1, you're ready to proceed to step 2. In step 2, you're going to write down the exact steps of what you did in step 1. That is, you're going to give directions to someone to solve just that instance of the problem. Not to solve the more general case. Not to do it for any image. But just how you came up with your answer for that the particular four-pixel image or whatever your problem might be.

3:20
The tricky part here is that there are a lot of things that we as humans do without consciously thinking about them. We just kind of naturally do them, and we have to think really carefully because the computer doesn't have common sense. We need to be very precise. If we're a little sloppy with our steps here, it's going to make things more difficult for us later on because we're going to be missing crucial parts of our algorithm.

3:45
In step 3, we need to find patterns. We want to write an algorithm for any instance of the problem, not just the particular instance that we worked.

3:55
And so what we're going to do is look at the steps that we wrote down in step 2, and we're going to find patterns. We're going to think about repetition. What things are we doing over and over again? How many times are we doing them? These will lead to looping constructs. Conditions, sometimes we do something, sometimes we don't do it. Under what conditions does this happen? This will lead us to conditional constructs, like if else, and well have values. Maybe we used a particular number because it was part of our input, or related to our input. We need to think about why we used that particular number.

4:30
If we have difficulties in this step, we need to go back to steps 1 and 2 and do them again for different inputs. If we do these over and over again, we'll come up with more information to look for patterns from. We'll have different sets of steps that we come up with for step 2 that tell us how we work multiple instances of the problem and be able to find patterns more easily.

4:55
Once we finish step 3, we want to check our algorithm by hand. We just came up with what we think is a general algorithm, but we may have made a mistake. We may not have realized that something happened in a particular case. Or we may have left a value that was particular to the parameters that we chose. If we did one of these mistakes, we'd like to find this now before we translate our algorithm into code. So we're going to check this with one or more different inputs which are again, going to be of sizes that are manageable to do by hand. We're not going to try it on a million pixels but we might try it on a very small other image or some other small input.

5:37
Once we're confident in our algorithm, we're ready to translate it into code. This is where the particular programming language that we're working in is going to come into play. So far, we've been looking at JavaScript. So we would take our algorithm and express it in the syntax of JavaScript.

5:54
Once we've written this down, we want to run test cases. That is, we want to execute our program and see, did we get the right answer with our program based on what we expect the right answer to be, knowing what problem we're trying to solve? Every time we run a test case, if it passes, that is we get the right answer, we're more confident in our program. If it fails, we know something is wrong with our program and we move on to step 7, debugging failed test cases. For this, we're going to use the scientific method, which we're going to talk about a lot in the next lesson. And this is going to help us understand what the problem is and give us guidance on what we need to do to fix the problem.

6:37
Ultimately, what's going to happen is if we have an algorithmic problem, we're going to return to Step 3 and fix our algorithm. And if we have an implementation problem, we're going to return to Step 5 and correct our translation of our algorithm to code.

6:53
So these are the seven steps that you can use to solve any programming problem, and we're going to use them throughout our examples in the rest of this course and specialization.

## Downloads

Lecture Video mp4
Subtitles (English) WebVTT
Transcript (English) txt
Lecture Slides pdf

Would you like to help us translate the transcript and subtitles into additional languages?