Southern New Hampshire University

IT 140 Writing Good Beginner Pseudocode Transcript

[00:00:01.63] Hello. I'm Professor Eric Cameron, and this video's going to discuss writing good beginner-level pseudocode for a programming course. Now, the question you probably ask yourself is, why do I need to use this pseudocode? And the short answer is probably going to be because somebody required it of you.

[00:00:20.59] Now, in terms of why we use this professionally, it's a very important thing for us to put down our requirements and get the customer to agree to them. And it's very difficult in an introductory class for us to explain this very well, because you write a 20-line, a 50-line program, and you could probably get away with that, honestly, without pseudocode.

[00:00:46.00] But the last project I was working on was a defense system for aircrafts, and we had about 700,000 lines of code. And as you can imagine, if we just kind of went off and did that, it would be very difficult to get that to work. So what we did was we worked with our customer, we developed requirements, put together our requirements in the form of pseudocode, flow charts, other types of documents. And then from those requirements that the customer agreed on, we would then go forward and create the program.

[00:01:23.94] Now, of course, the pseudocode, the idea is we're going to lay out the logic, because what ends up happening — and you'll see the third bullet point here is "don't do what I did." What ends up happening is if you don't have your logic down and then you start to program, you end up in this situation where you start making changes and you're not sure—is it the language, is it my logic? And so on and so forth.

[00:01:47.58] And at least if you have the pseudocode and if the customer has signed off on the pseudocode, your logic should be pretty solid. It should go through a nice review process. And then you're not left kind of hacking away at the code. And again, I made that mistake. I got through my first two years without doing any pseudocode, and I got into a course at the 300 level, and I found that it just didn't work anymore. So it's a good habit to get into—to jot down your requirements and kind of go through that.

[00:02:23.42] So what's going to make good pseudocode for you? First thing is, remember that the computer takes input from the user, it processes that input into output, takes data and turns it into information, and is able to output. So we do want to clearly try to outline those tasks for the programmer. And the other thing is, we want to keep this language-independent. So ideally, the pseudocode should not use any specific terminology from a language like C++, Python, whatever you might be talking about.

[00:03:01.72] Now, of course, if it's a huge project and everybody knows it's going to be in a specific language, you could do things a little differently. But generally, our idea is going to be, we want to put together a language-independent set of explanations. And then we could then implement that in whatever language you wanted. I could do it in Perl, you could do it in Python, somebody else does it in Java, and we are fine.

[00:03:30.04] So let's take a look at pseudocode that I put together for adding two numbers. So you'll notice this is going to be six lines of text. And it basically describes to the user, or to the programmer rather, what they should be doing. Notice we are starting with verbs generally.

[00:03:53.05] We are saying “input,” “output,” “add,” and so on. And we are also giving guidance as to what we should name the variables. You'll notice I've used first letter lowercased and then everything else uppercased, every new word. That is going to be what we consider “camelCase,” you may hear. Like, the animal the camel has a hump on its back.

[00:04:22.28] And you could see what we do here is we start by saying, “Input a name–” since this is a beginning pseudocode, you'll notice I'm very specific, “from the user.” It could be done from other spots. For example, you can input stuff from a file or whatever. But for the moment, you see I start with inputs—"a name from the user, save as userName.”

[00:04:45.32] Now, this is not going to be code that you can go and copy into any language and it would run. This would 100% not run. But you could see that I've outlined the logic here. And now this should be something I could take into any language and implement.

[00:05:01.78] Now, for example, in our CIS 108 course, we use Python. For input purposes, we do happen to use a command called “input.” However, in a programming language like C++, we would use a command called “cin.” You'll see “output.” In Python, we use a command called “print.” In C++, we use “cout.” So there will be different ways for us to implement this, and I'll show you that shortly.

[00:05:30.70] But the idea is we are basically saying, get the user's name, get two [numbers] from the user, add them together, output this phrase, and then output the total to the user. And basically, at this point, this would allow us to outline the steps needed to get a user's name, two numbers, add the [numbers] together, and print it out.

[00:06:00.45] And then the implementation in the real world would be left to our programmers. And typically, your managers, your systems analysts, those types of folks would be the ones developing the pseudocode and other requirements documents.

[00:06:18.63] Now, the way this would look in Python would be something like this. And let me make this a little bigger. Now, you'll notice what I did was, just to comment the program, I copied over my pseudocode and pasted it in. This is a little trick to help you to document the program. You could see it does make it a little easier to figure out what is going on. It doesn't always copy in line for line. But in this case, it does. And you'll see that I now implement this using that programming language.

[00:06:53.42] So in the case of Python, I don't need to declare my variables. In other languages, I would need to go and create this variable before we could use it. I would need to say, “this is an integer,” “this is a character,” et cetera. You'll notice the pseudocode, as we see here, says nothing about “eval.” “Eval” happens to be the way, in Python, you turn a number—or a string into a number. The default input is going to be bring things in as a number—or rather, I'm sorry, as a character.

[00:07:35.47] So you'll see what we now do is we convert, line by line, each of those things of our pseudocode into programming. And you'll see that this is good pseudocode, because it leaves very little to the programmer's imagination. And remember, we are trying to make sure we implement our customer's requirements as best as possible.

[00:07:58.23] So we do try to match it. We don't want to give the programmer leeway to go and do whatever they feel like, because that's when the customer says, “This is not what we wanted.” Now, again, you'll notice—“Output the phrase Your total followed by userName.” And this is the way that looks in Python. But again, it is the programmer's job to convert our pseudocode over to whatever language they are utilizing.

[00:08:26.18] So basically, ladies and gentlemen, when we talk about pseudocode, we are trying to take text, we are trying to take those requirements, we are turning it into something that can be read, followed, and programmed. And at least from an introductory level, this would be a good, simple set of pseudocode. Now, for those of you who might be more advanced, maybe you know what a loop is, you know what an if statement is, you can imagine this does present some more challenges as the program gets a little bigger.

[00:09:02.59] But this would be a very good pseudocode for a very simple program. And again, even if you are someone who is more advanced and you know the language you are programming in, I did not need to do pseudocode for me to write this program, but again, it is a good habit to get into have this ready to go. So good luck on your studies.