How to Learn Programming in the Least Possible Time

Walk into any reputable book store and scan the computer selections. You may very well be overwhelmed. To answer the question, "What's the best way to learn programming?" dozens of authors, for a small or not-so-small fee, in 500 pages or less, are eager to show you their way.

Daytime television is replete with ads from technical schools purporting to teach you programming in six weeks to two years (with financial and job placement assistance, of course).

For a teacher, the question looms larger since you not only need to know how best to learn, but also how best to teach.

Before you pick up a book or turn on a system or smudge your first floppy disk—stop. What would you like your computer to do? Start small. Maybe display your name in the center of the screen? Count from 1 to 100? Play "Yankee Doodle Dandy" on its speaker?

Perhaps a rousing game of Hangman or Tic-Tac-Toe would be more to your liking. Or figuring the mortgage payments if you get financing on that dream house. You might even be happy to have the computer help you balance your checkbook each month! (Don't go overboard here. Predicting weather patterns for the next six weeks for North America is *not* starting small!)

You *don't* have to read the book first. Just decide what you want the computer to do. Next, try to think like a computer. What would you do to accomplish the task you have chosen?

Say you chose the "name in the middle of the screen" application. You would probably want to erase everything on the screen first. Then you'd want to find a way to get the computer to write your name so many lines down and so many spaces over.

Of course, this assumes you already know how to turn on your computer and load the language in which you'll be programming. If you don't know how to boot the computer, then that should be your first goal—"Learn to turn on the computer and get it ready to program."

Once the computer is ready to go, it's time to hit the books. But read them as though you're compiling information for a term paper. Don't go chapter by chapter. Check the index. "How do I clear the screen?" That's all you want to know. Forget for-loops, if-thens, and print statements—none of this matters unless it addresses your most pressing problem: "How do I clear the screen?"

I know that a lot of you are moaning that you can't understand the books. If this is your problem, it's perfectly legal to ask those who have gone before. If you're lucky enough to know such a person, it's incredibly more efficient to ask for help than to try to wade through 500 pages to find one example of what you're looking for. But either way, stay with it. Don't focus on anything but the task at hand.

An amazing thing happens when you follow this procedure. You remember how to clear the screen for the moment (tips on how to keep track of what you've learned will appear in the next issue), and you don't clutter your mind with all that other stuff you don't need to know right now.

As a byproduct of this approach you will find yourself skimming all sorts of worthless goodies you'll need later. Don't get side-tracked from your prime directive. In spite of your single-mindedness these goodies will lodge in your brain with no conscious effort. When you need them later, they will come back to you.

Having conquered the "name in the middle of the screen" program, choose another application. Follow the same procedure.

About 20 programs into this self-styled how-to-learn-to-program course, it'll be time to read your first book on programming—cover to cover. Now you are ready to fill in the gaps that occurred because your own programs have not exhausted the capabilities of the computer or the language.

At this point your brain is ready to fill in those gaps—not when you're just beginning. By attacking only the applications in which you're really interested and focusing on the parts of the language that address those needs, you will have places to put each abstract piece of information as you assimilate it. You will not be trying to memorize the Chinese dictionary before you learn to say "Good morning" in Chinese.

This isn't the most structured approach (although it's exactly the foundation you need for structured code development).

However, I've personally found it to be the fastest way to learn and retain useful programming techniques.

You aren't a programmer if you merely type in someone else's code and see it work. When you get the computer to do *your* program, then you begin to feel like a programmer. That feeling of achievement will motivate you to improve your skills to the point that you feel like a real pro.—Dave Ruskjer. □

The author is President of Touch Talk Technologies, Takoma Park, Maryland.