How to Program

Michael Gogins
http://michaelgogins.tumblr.com

Irreducible Productions
New York

6 March 2017

What is Programming?

- Programming is the art of programming a universal Turing machine, more commonly known as a "programmable computer," to implement a specific Turing machine, more commonly known as the "program."
- This term is confusing , because "program" can mean either the *algorithm* that the program is supposed to implement, or the *code* that is used to implement the algorithm.
- Algorithms are a branch of theoretical computer science, i.e. of mathematics.
- Code is a branch of software engineering, which isn't much of a kind of engineering, it is more of a craft like carpentry.

How to Program

1. Define the algorithm. Nobody knows how to do this, but this is the most important part of the job, usually because people haven't really decided what the program is for. Just try to write down what the program is supposed to do. Sometimes a little flow chart is helpful. You will have to come back and do this over several times.
2. Write the code to implement the algorithm. This is too easy, because there are way too many ways to implement the same algorithm.
3. Prove that the code implements the algorithm. Nobody knows how to do this, and in fact it has been proved to be impossible (this is called the "halting theorem"). But if you don't try, your program will never work properly.

The Steps I

1. You *must* write code that is simple and easy to read and understand. It should read like an example in a programming text.
2. Don't use little-known languages. Use standard languages, as that will make the code easier to read. For music today, these are C++, C, maybe Java, and JavaScript. These are also the most in-demand languages for general purpose programming, for some reason.
3. Don't use comments. Write the code so it explains itself.
4. Don't abbreviate, spell everything out.
5. Don't get fancy with pointers, anonymous functions, and so on.

The Steps II

6. Don't use special libraries if the language itself already provides facilities that you need. In other words, don't use Boost, use the standard C++ library. Don't use operating system calls, use the runtime library.
7. If you must use special libraries, use the best ones, these are usually the ones other people are already using. For music programming, use libsndfile and PortAudio.
8. Write code to test your code. If you don't do this, your code will never work properly.
9. If either the code is still not easy to read, or it still does not work properly, then there is something wrong either with your algorithm, or with your understanding of your algorithm. Clarify the algorithm, and then go back to the first step.

The Steps III

10. Do not debug until you have done all of the above. Debugging usually takes longer than just re-reading the code to make sure you really understand it.
11. Do not try optimizing your code until you have done all the above. Then, run a profiler on your code, and see if there are any slow places where the computer is spending too much time. Probably, the only way of speeding up these places is to find a better algorithm and go back to step 1.
12. The basic principle is that if code is hard to read, you can't even *know* if it implements the algorithm; while on the other hand, if the algorithm itself is inconsistent or unclear, you will *never* be able to write clear code for it.