

# Coding is Communication

Bill McNeill

18 October 2006

## A Program's Audience

Writing code is like writing prose—you have to have your audience in mind. Each program will usually have to simultaneously accommodate three different audiences.

- Computer . . . “I am stupid. I will run your source code exactly as written.”
- User . . . “I don't want to know anything about your source code.”
- Programmer . . . “I will grudgingly look at your source code when necessary.”

## Programming Goals

The obvious goal of writing a program is to make a computer do something. However, there are other goals which are less obvious but equally important.

### Usability

- The program has to work (be bug free).
- The program has to do what the user expects. (No one reads the README.).

### Clarity

- A program is not just a set of instructions for a computer; it is also a way of precisely communicating an idea.
- As in prose, you want to take a complicated concept and express it as simply as possible.
- Another human being is going to use your program. Be kind to them.

## Flexibility

- Programs are living things. They only stop changing when they die.
- Design your programs so that they are easy to change (expand, reuse, fix bugs in).

## Programming Style

### Modularity

You achieve flexibility by writing modular code.

- Break a task into small interacting sub-tasks/data structures.
- When the subtasks and data structures are small enough, implement them.
- Each piece should keep its internals hidden (information hiding).
- Standard subtasks
  - Algorithms—the unique work the program does
  - Input/output—interaction with the user while the program is running (command line arguments, printing to the screen)
  - Persistence—storage of information after the program has exited (changing files, updating databases)
- Keep special case information separate (e.g. don't put file names and configuration parameters in the source)
- Design in problem space not implementation space. Be able to draw boxes and arrows without writing code.

## Documentation

Documentation is the most effective way to communicate with your human audience and so should never be an afterthought.

- Every structural piece (e.g. function, object) should be documented in clear, carefully-written prose.
- More people will read your documentation than your source.
- Write in problem space, not implementation space.
- Documentation as discipline—if you have a hard time describing a function, it may be poorly written.

## Source Aesthetics

- If it hurts your eyes, it'll hurt your brain.
  - Functions, objects, etc. should not be too long—a screenful should tell a story.
  - Deep indentation is a sign of a problem
- Write boring code

## Techniques

### The Development Cycle

Programming is an ongoing duel between your ingenuity (which you use to implement *features*) and your natural capacity for error (which ensures your implementation will contain *bugs*).

1. Implement a set of features.
2. Test the resulting program.
3. Fix the bugs you found in step (2).
4. Repeat steps (2), (3) and (4) to fix the bugs you introduced by fixing other bugs.
5. When you can't find any more bugs, repeat step (1).

The more of this process that you can automate, the better.

## Source Management

- Use a good code editor (**emacs**, **vi**, commercial integrated development environment (IDE))
  - You need syntax highlighting and automatic indenting.
  - An integrated source-level debugger is really nice.
- Use source control (CVS or Subversion).

## Testing

- All code is guilty (buggy) until proven innocent.
- After every change you should run tests.
  - Informal test—run on a fragment of the input and eyeball the output
  - Formal test—write code to test your code
- For larger projects you should develop a test suite in parallel with the code

## Debugging

- Programming is 10% composing and 90% debugging.
- Every change you make introduces the risk of a bug.
- Be aggressive about finding bugs. Don't rely on wishful thinking. The earlier you locate a problem, the better.
- **USE THE DEBUGGER!**
- Step through new code in the debugger.