C Program Style Requirements
August 2011 (McCann)

Style Principle
Structure and document your program the way you wish other programmers would.

Programs written to fulfill the programming requirements of this course will be expected to adhere to the following rules of programming style. If you fail to satisfy these requirements, you will lose points; it’s that simple. To see these rules applied to an example program, please refer to the document "Toward Developing Good Programming Style" at http://www.cs.arizona.edu/people/mccann/style.html

Formatting of Block (Internal and External) Comments: To create consistency in your documentation, and to encourage you to be complete, I have prepared templates of the Internal and External documentation block comments that you can copy from http://www.cs.arizona.edu/people/mccann/templates/c/. There are versions with explanations (internal.c and external.c) and without (int.c and ext.c). You are not required use these, but doing so is a good way to ensure that you aren’t forgetting anything.

“External” Documentation: In programming classes, the comprehensive set of documents that detail the design, development, and structure of a program are usually condensed into a comparatively brief ‘block comment’ at the top of the source code. This “external” documentation will minimally include:

1. Your name, the course name, assignment name/number, instructor’s name, and due date.
2. Description of the problem the program was written to solve.
3. Approach used to solve the problem. This should always include a brief description of the major algorithms used, if any, or their names if they are common algorithms.
4. The program’s operational requirements: Which language system you used, special compilation information, where the input can be located on disk, etc.
5. Required features of the assignment that you were not able to include.
6. Known bugs should be reported here as well. If a particular feature does not work correctly, it is in your best interest to be honest and complete about your program’s shortcomings.

Internal Documentation: The details of the code are explained by comments placed within the code. Your internal documentation will minimally include the following:

1. A ‘block’ comment should be placed at the head of every user-defined function. This will include the function name; the purpose of the function; a list of all parameters, including direction of information transfer (into this routine, out from the routine back to the calling routine, or both), and their purposes; and a description of the function’s return value.
2. Meaningful variable names. In a nod to tradition, simple loop variables may have single letter variable names, but all others should be meaningful. Never use nonstandard abbreviations.
3. Each variable and constant must have a brief comment next to its declaration that explains its purpose. This applies to all variables, as well as to fields of record or struct declarations.
4. Complex sections of code and any other parts of the program that need some explanation should have comments just ahead of them or embedded in them.

Miscellaneous Requirements:

1. Global variables and unconditional branching (e.g., GOTOs) should never be used in your programs for my classes unless I have approved their use for your particular situation.
2. Be generous with your use of “white space” (blank lines) to set off logically related sections of code.
3. Indent bodies of function, loops and If statements, and do so with a single, consistent style. See http://www.cs.arizona.edu/mccann/indent.c.html for some examples.
4. If you have a choice between writing code that runs quickly and writing code that is easy to understand, make it easy to understand. Avoid pursuing efficiency at the expense of clarity.
5. Insert documentation as you write your program; never wait until after the program is working. If you wait until it works, you may have already forgotten the important details you need to mention, and you may be too eager to stop working to devote enough time to do a good job.
6. When creating multifile programs, include a block comment at the top of each file that explains its content and purpose.