Programming Work Habits and Best Practices for C++

Document origin: ITGM 315 – Dean Lawson
Modified for VSFX 375 – Deborah R. Fowler

- Start working on an assignment as soon as possible.
    - This will give you more time to ask questions both inside and outside of class
    - Sometimes you will hit a roadblock and your brain will need time to figure things out. If it's too close to the deadline you won't have time for this process to happen.
    - Plan on working frequently but in small pieces, if things go well keep working, if not so well, then you won't waste much time.
- Write code in small pieces.
    - This works well with working frequently as well.
    - Compile and test each piece to make sure it's working before moving on to writing more code. ie. start with a program that compiles and build on it.
    - If you have a problem since you're only working on a small amount of code, the solution is usually much easier to find.
- Conform to the coding standards at all times.
    - The Coding Standards are designed to help you write code in a clear and understandable way.
    - Check your code before you hand in an assignment to make sure that the coding standards are followed. Don't give away points by not double-checking your code against the standards.
- Ask a lot of questions
  This works well with starting early and working in small pieces.
    - Ask questions in, before or after class.
    - Email and come by during office hours.
    - Email questions.
        - I usually answer email within 24 hours
        - Please include all the code you are working on
        - Conform to the coding standards!
    - **When looking for help make sure your code at least mostly conforms to the coding standards! If it's too hard to read then I'll ask you to bring your code up to the standards before I will help you with a problem!**
    - tutoring is available for this class
- Use the debugger!
    - You should set a breakpoint and trace the execution of all the code you write. Even if your project seems to be working well, you will find many problems this way.
- Errors and warnings
    - **When you get compile errors always start with the top-most error** as many C++ errors have a cascade effect. Often, fixing one error will eliminate many others.
    - Many warnings can be ignored but should not be ignored. In particular it is important to not ignore the following warnings:
        - Not all execution paths return a value
        - Variable is being used without being initialized
        - Etc.
        - You will be expected to have warning free code whenever possible
    - Double click error messages to bring up the line of code that has an error
    - Linker errors can't be double clicked on. The most common error is:
        - Unresolved external reference – it means that you are using a function/class/global variable that isn't defined properly.

- Most compiler errors are caused by missing semi-colons and mismatched {}'s – following the coding standards will help here.
- Unknown identifier errors are very common. It means you are trying to use a variable or function that hasn't been defined first. Very often these are caused by misspelled names. Check the spelling and capitalization of your variable and function names. Following the coding standards is helpful here as well.
- Watch out for **= instead of ==** in condition statements. The first is the assignment operator, the second a comparison