

Lecture 2A – Advice on Effective Programming

1. **Design from the top down.** Start by considering all of the elements you need for your application. For this class, you might want to think about the following: (1) data acquisition, (2) data analysis and (3) data storage and retrieval. Optionally, you might want to include a simulation facility. List the requirements for the front panel including the types of controls and indicators together with schemes for presenting the data in graphical format. Create mock-up front panels you can show to prospective users and obtain their input.

Resist the inclination is to write some critical piece of code first and then expand in all directions. This will lead you to many rewrites.
2. **Create a hierarchical program.** Divide the task into manageable, logical pieces. Develop a flow chart that illustrates how the task is to be carried out and how the subtasks communicate with each other.
3. **Make liberal use of subVIs.** This occurs where a logical division of labor exists or where there is a potential for code reuse. It is much easier to debug a small VI of well-defined scope than a great sprawling mess. Often you can tell your VI is getting too large when you can't view it all on the computer screen.
4. **Document, document and document.** The first thing you should do when creating a new subVI is to fill out the description box. Don't leave documentation as the last task. Document everything. Remember - *there is no possibility of over documentation.*
5. **Test your VI thoroughly.** Remember that *all* non-trivial programs have bugs. Be critical when you examine the output of your program. On the other hand, remember that no programming project is ever finished - it is abandoned. Budget your time carefully.
6. **Make your program idiot proof.** All of your controls should have default values, so that if the program is run without any changes to the control values it will execute and give a reasonable output. Also, use the Data Range option to specify the range of reasonable values for input.
7. **Do not write programs by experimentation.** A programming language like LabVIEW us a self-consistent logical entity. You should be able to predict what output will occur for a given input. All programs break. If you don't understand how your program worked in the first place, how will you fix it?