A Guide to Program Development

By Teoryn (Kevin S.)
teoryn@gmail.com
http:\\hiddenuniverse.blogspot.com

Introduction

Since I entered the TI community there was always one issue that bothered me the most. That issue was the existence of so many programs that didn't work, or if they did, they didn't work correctly. It wasn't until recently in my study of Java I believe that I understand the cause of that problem, which is in my opinion a lack of any standard development cycle. Many new programmers, especially those who know only basic, may think that programming is only writing code, when that is actually only a small part of the actually process of programming. In this document I will present a simple development cycle that will aid programmers in designing their programs and hopefully lead to more meaningful content within the TI community.

This process is broken down into 5 parts: Analysis, Design, Implementation, Testing, and Deployment.

Analysis

The analysis phase is when you decide what your program will do. Obviously, setting out with the goal only of writing a program will not get you very far. You need to use this phase and decided what exactly your program will do. This is not the time to worry about how things will be done. For example, if you are creating a program do display an e-book, you will want to decide on a menu system, and a layout for the text. This is also an excellent time for community involvement. Visit a TI community website (see links [1]-

[3]) and ask what people would expect from your program. This communication is vital because it ensures your program has what people want from the beginning.

This phase should be properly documented with detail. If done correctly, a documentation of the analysis phase should be a near complete operation manual for your program, telling the user what to press, and how to interpret the output.

Design

The design phase is when you map out how the program will work. However, this will be difficult when programing for the calculators since there aren't any structured languages for the calculator. In most cases the best solution is modularization, in which individual components are made independently. Each module has two major parts, dependencies and the interface. The dependencies are variables and other modules that the are required for function. The interface describes how the module interacts with other modules (input/output). To make this easiest on yourself it is best to design a module, and then continue through the development cycle with that module to make sure it is properly working, and then return to the design phase for the next module.

Implementation

Finally, this is what you were waiting for, the implementation phase, when all the code gets written. However, if you properly executed the design phase, you will discover that actually coding the program is very simple.

I should mention at this point that even if you code is very easy to read now, it won't be long before you look back and completely forget what you were trying to do. So please, get in the habit of adding comments. Lots of comments. Even if it seems too simple to forget, comment. Now, for assembly programmers this is easy, but while programming in

basic this is slightly different. The solution is to start a line with quotes so that the interpreter reads it as a string. (WARNING: this will destroy your ans var, so be careful if your using around ans) Of course, for a public release you should remove these comments for size and speed.

However, once you finished one section of the code, you should not move on to the next. Make sure to test each part (see next phase) which will return you to the design phase. Rinse and repeat this cycle until your program is complete.

Testing

This is in my opinion the single most important phase of development. Within the testing phase you will spend time making sure that the program works, and when I say works, I mean flawlessly. Even if it seem like something will work for sure, test it. Also, if it seems like a stupid idea, be sure to test it. You can be guaranteed that somebody will do something stupid while using your program. Be sure to use abstract ideas for testing, don't just do what seems obvious to you because most likely, nothing will be obvious to the user.

Once you find a problem during testing, it falls under one of two categories: design error or programming error. Programming errors are much easier to fix, they are simply code that wasn't properly written. To fix these errors all you must do is pinpoint the error and rewrite the code. The challenging errors are the design errors. These are faults with the underlying design of the program, and in worst case could demand a completely new design for all or a large part of the program. The solution to these is to return to the design phase and correct the mistake there. Of course, from there rewrite the code.

Of course, once you have fixed an error, be sure to test anything that could be effected by it, because new code within the program could easily cause problems.

To some this step may seem like to much work or not even worth it, however it is vital that you devote time to testing. It is possible to get some beta-testers, however do not consider this an excuse to not test it yourself. Without proper attention to testing your program could end up like many programs on ticalc.org that don't work. Do not let this be you.

Deployment

The deployment phase is the time when you put everything together and show everybody the result of all your time and hard work, but it's not all fun and games here. You must be sure that your program is presented to the public in a way that makes people want to download it. Make sure that you use proper English (or any language of your choice) and prevent your description from looking 'ugly'. By ugly I mean improper use of capital letters along with excessive exclamation marks. Also, don't be dishonest, nobody wants to download a program only to discover that it's nothing like it's description. Up loading screen-shots is also a way to show your program to people before they download it.

Besides the 'marketing' aspect of deployment, you should also make sure that you include everything that a user will need. This means including a user manual with your program, which if you properly followed the first phase should be near complete other than some contact information and anything else you want to add, a credits section for example.

Conclusion

Hopefully after reading this you already have plans for how to better develop your next or current program. Don't forget that these step can also apply for later programming you do on computers if you go on to that.

I hope that this was helpful for you, now get developing.

Links

[1] http://www.maxcoderz.com

[2] http://unitedti.org/

[3] http://wateringelaan.demon.nl:8080/index.jsp

Credits

I'd like to thank Cay Horstmann for his description of "The Software Life Cycle" found in his book 'Big Java' which I borrowed the five phase concept from.

Special thanks to Vincent Jünemann, for his help with concepts of this paper.

Special thanks to Patai Gergely for his advice concerning the design phase.

Finally, thanks to the whole TI community for introducing me to programming.