Since Liberty BASIC is considered a “structured programming language” I like to use the following guide to help me develop a program in Liberty BASIC or any other programming language.

Before I get started with my development steps let me explain a little bit about “structured programming” and what it is. Structured programming is based around the statement that modularization will gain you better programs. Modularization simply means the grouping of statements with relation to each other into modules (functions/subroutines). By breaking up your program in to logical sections it makes your code easier to understand, write, and debug.

So, when I set out to write a new program in Liberty BASIC or any other language that uses “structured programming” style, I try and make sure that the modules that I write a no more than a page in size. To me, it makes it easier to view the entire module at a glance and it makes it much easier to understand a series of smaller modules than it does one big huge program.

Depending on the size of the project that I am trying to create, after the initial designing of the modules and breaking them up, I am ready to go. With larger programs I need to write, I may have to break the modules up even further. At any rate, I continue to break the modules down until I have it in small enough chunks where it describes the problem that I am trying to solve. (You need to be able to understand the problem very thoroughly.) Some like to call this process a “top-down” design where you start with a general description of the problem you are trying to solve and work to a more specific one through module refinement.

After I feel like I have the organization in order that best describes the problem and solution path, I can begin designing the overall structure of my program. This could take a considerable amount of time depending on the size and nature of your project. Be careful here and make sure that you have everything defined and that you understand everything fully. (This area usually gets allot of people into trouble later on in the process.)

I am now ready to write my program.

So in summary, I try to follow the outline below doing each step along the way:

1. Plan your program first. Don't get in a rush to start writing code. Try to ask yourself the following questions:
2. What is the overall purpose you would like your completed program to achieve?
3. Does the program need input? What kind? How will I process it?
4. What kind of output does the program need and how will I produce it? Where will I save it?
5. How do I want to display my input prompts and my output to the users of my program?
6. How is the best way to break my program up into the simplest, easy-to-follow modules?
7. How is the best way to fit all those modules into a program so they can communicate together forming a complete program solution?
8. Can I break the modules up even further?
9. Write the main program leaving place holders for the modules. By doing this you can test the logic and flow of the main program as you write it.
10. Write each module seperately, test and debug each one before adding it to the main program.

11. Did I place enough comments in my code for better understanding when I look at it later?
12. Debug and test the entire program.

Enjoy!