Lab 6 – Compile & Debug

Introduction

This lab further explores debugging, both at compile-time and during program execution. You'll be given two separate programs to debug, one for compile-time errors and the other for run-time errors. Using the provided documentation of an RPN calculator program you will test and fix various defects with it.

Reading

• None

Provided files

• CircularBuffer.c, CircularBuffer.h – These files will be used for the compilation portion of this lab. There are several errors in each file that will need to be fixed with the goal being no errors or warnings upon compilation.
• debug.c, FloatStack.h, FloatStack.c – These files will be used for the run-time debugging portion of the lab. debug.c contains a much more complete version of the RPN calculator that was introduced in lab 3, albeit with several errors that will require fixing.

Assignment requirements

• Compile: Edit CircularBuffer.c and CircularBuffer.h so that they adhere to the style guidelines and have no compilation errors or warnings.
  o Be sure that your edits do not affect their functionality as they will be tested during grading.
  o If you are compiling just CircularBuffer.c within a project by itself the project will fail to build with an error during the link step that says that the function main isn’t found. That error is expected and just fine (it’s actually a linking error and not a compilation error).
• Debug: Edit debug.c and FloatStack.c to fix the various run-time errors within the RPN calculator program. Proper operation is specified within this lab manual.
  o Comments must also be added in both files to document its functionality.
Also document all of the errors that you fixed. Specify what the error was, how you found it, and how you fixed it.

- Add the following to the top of every file as comments:
  - Your name
  - The names of colleagues who you have collaborated with

- Submit CircularBuffer.c, CircularBuffer.h, debug.c, and FloatStack.c via eClassrooms before the due date (double check that it was accepted, again BEFORE the due date).

Grading

This assignment again consists of 10 points:

- 2 points – Fixing CircularBuffer.[ch] to compile correctly
- 2 points – Fixing CircularBuffer.[ch] to follow the formatting guidelines
- 6 points – Fixing all of the changes found in debug.c and FloatStack.c

You will lose points for the following:

- -4 points: Compile code doesn’t compile
- -6 points: Debugging code doesn’t compile
- -2 points: any compiler warnings within Debugging code
- -2 points: the files you submit aren’t named as described in this document or you submit more than just the required documents
- -2 points: Not using the #define constants for return values
- -2 points: if gotos were used
- -2 points: Debugging code has sparse commenting (should be about 1 inline comment per 5-8 lines)

Debugging

The debug.c and FloatStack.[ch] files comprise an RPN calculator program. This program is a bit more sophisticated than the one you wrote for lab 3. It has been revised to act more like production RPN calculators and to therefore be useful in real-world situations. This also means it is more complex and has better error handling than the earlier versions.

The largest change to it is that it allows for iterative calculation. The stack is preserved between input prompts (which look like ">") and so calculations persist as well. Input can be written in space-delimited form on a single line as it could for Lab 3, but there is no longer the requirement that the stack size be 1 after execution. Backspaces are also supported within the user input so that the program input matches the terminal display.

The original arithmetic operations have been extended with more complicated mathematical operations and new stack manipulation operations. The first set perform raw computation while the second complete stack management tasks such as rearranging elements or displaying stack elements. All operations with the exclusion of "help" are checked for only the first character so that an operation such as "+6" will be
interpreted simply as addition. All operations take advantage of the full precision of C's float data type.

Error handling within the RPN calculator is handled on a per-operation basis. So any misunderstood operations are reported as invalid while the rest of the valid operations are calculated as normal. If an operation does not have enough operands then an error will be printed, although other operations in the sequence will still be attempted.

The calculator's internal stack can hold a maximum of 16 values before it's full. At this point an error will be displayed, but as before the program will attempt to continue execution of its input.

**Supported mathematical operations:**

help "help": Prints out program help, which is similar to this operation listing.

Addition "+": Pops two values, adds them together, pushes the result

Subtraction ";": Pops two values, subtracts the first popped from the second and pushes the result

Multiplication "*": Pops two values, multiplies them together, pushes the result

Division "/": Pops two values, divides the second value popped from the first, and pushes the result

Modulus "%": Pops two values, returns the remainder from the division "/" operation, and pushes the result

Exponential "^": Pops two values, raising the second value popped to the power of the first, pushes the result

Square root "v": Pops a single value, calculates the square root, and pushes the result

**Supported stack operations:**

Clear stack "c": Empties the current stack.

Duplicate "d": Duplicates the value at the top of the stack and pushes it on top.

Reverse "r": Reverses the ordering of the two top elements in the stack.

Print "p": Prints out the value at the top of the stack without altering the stack.

Stack trace "s": Prints out a complete representation of stack, its current size, item values, and their position within the stack without altering the stack.