Pat Letts

Senior project proposal

F17 CS 480

Therapist: a Brainfuck interpreter

**Language overview**

Brainfuck is a very simple esoteric language created by Urban Müller. Simple beyond the point of usefulness, with only 8, single-character commands: <>-+,.[]; these commands allow for the navigation and changing of an array and its contents, the unofficial standard being at least 30,000 cells in size, all initialized to zero. ‘<’ and ‘>’ respectively decrement and increment a data pointer to track a program’s point in the array. ‘+’ and ‘–‘ respectively increment and decrement the value in the current cell. ‘,’ and ‘.’ allow for single-byte input and output of the current cell. ‘[‘ and ‘]’ allow for looping. All other characters should be ignored.

Here’s a Hello World:

++++++++[>++++[>++>+++>+++>+<<<]>+>+>->>+[<]]>>.>---.+++++++..+++.>>..<.+++.------.--------.>>+.>++.

I’d like to build a decent interpreter for Brainfuck. Some extra command line options and a debug mode would be cool too.

**Challenges and considerations**

The first challenge that comes to mind for an utmost basic implentation of Brainfuck is looping, especially nested loops. My current plan is to use a stack to keep track of loops, storing the locations of left brackets for proper jumping.

There’s a lot of variations between implementations; how long should the array be? Urban Müller’s implementation was 30,000 cells. How many bits in a cell? Some support 8 bits, others would argue for 32. Signed or unsigned? Should I/O always be considered ASCII values instead of raw integers? And so on. Luckily, a number of common conventions have been established regarding these questions and more.

**Goals**

I plan for the majority of my code to use C/C++, though I may very well end up investing some code in a scripting language for tasks such as compilation and packaging; maybe also some inline assembly for maximum *fun*. Naturally, I hope to learn a deal about interpreter design, at least at a basic level, and I’m also really interested in learning how to package software for say, a linux repository, so it can be downloaded and setup directly through a package manager. In a more general sense, I also hope to become more familiar with the intracacies of C/C++, and get more experience with scripting languages.

**Features**

Basic interpreter – provide a source file for the program, or direct command line input.

Simple options – switch certain options for the interpreter with flags. Un/signed bytes, array size,

ASCII I/O, etc.

Debug mode – a particular feature that I haven’t seen yet would be a cool graphical debug mode, showing the states of particular stacks, and stepping through code, allowing the user to watch the array change, one command at a time. Maybe also more specific options for it. I’ll probably use the curses C library for this, I’d like to learn to do more with it.

**Grading**

Total points – 90

Basic interpreter setup – 70

-Without nested looping – 30

-With – 40

Debug mode – 20

Simple options – 10 ~ define ASCII I/O, un/signed cells, and cell size

More complex options – 10

Man page – 10

Auto-install setup – 10

Scale – out of 90 points

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **D** | **F** |
| >80 | >70 | >60 | >50 | <50 |