Title: Command Line Quadratic Equation Solver

Date: January 17, 2017

Team: Danny Kiel

Introduction:

The customer is a wind turbine manufacturer. They have had some issues with engineers solving the quadratic formula by hand that have resulted in out of control turbines. They need a simple way to solve complex quadratic formulas that is reliable and accurate.

Stories:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Release 1 | Time Estimate | Risk | Time Spent | Percent Complete |
| 1. The engineers need some way to input A, B, and C in decimal notation. | 20 min | 1->1 | 12 min | 100% |
| 1. Input on separate lines after the program starts. | 20 min | 2->1 | 2 min | 100% |
| 1. Output only real solutions and omit complex solutions. | 2 hours | 4->1 | 2 hours | 100% |
| 1. Single precision floating point output truncated to 40 decimal places. | 10 min | 2->1 | 5 min | 100% |
| Release 2 |  |  |  |  |
| 1. The engineers want to be able to input A, B, and C in scientific notation. | 1 hour | 5->1 | 1 min | 100% |

Spikes:

1. Conversion to double for internal calculations

scanf("%lf",&a);

scanf("%lf",&b);

scanf("%lf",&c);

This ended up being easy enough using scanf.

1. Handling overflow and underflow

strtof(buffer,NULL);

This was the hardest one to figure out but I found a good solution. This change mean that I was no longer sure about scientific notation but, a simple test proved that I had nothing to worry about.

1. Scientific notation

After some simple testing, it turns out scanf support scientific notation so this ended up being easier than I thought.

Other Requirements:

Hardware: Computer

Software: Ubuntu Linux and Bash

Compiler: GCC

Code:

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <errno.h>

#include <string.h>

int main(int argc, char \*argv[]){

float a = 0,b = 0,c = 0;

double duba,dubb,dubc,negb,disc,sqr,root1,root2;

char \*buffer = calloc(4096,sizeof(char));

char \*temp=buffer;

fprintf(stdout, "Enter a value for a:\n");

fgets(buffer,sizeof(buffer),stdin);

temp = buffer;

a = strtof(temp,NULL);

if(errno){

fprintf(stderr, "ERROR: %s\nEXITING...\n", strerror(errno));

exit(errno);

}else if(a == 0){

fprintf(stderr, "ERROR! a cannot equal becuase of divided by 0 error.\n");

exit(0);

}

fprintf(stdout, "Enter a value for b:\n");

fgets(buffer,sizeof(buffer),stdin);

temp = buffer;

b = strtof(temp,NULL);

if(errno){

fprintf(stderr, "ERROR: %s\nEXITING...\n", strerror(errno));

exit(errno);

}

fprintf(stdout, "Enter a value for c:\n");

fgets(buffer,sizeof(buffer),stdin);

temp = buffer;

c = strtof(temp,NULL);

if(errno){

fprintf(stderr, "ERROR: %s\nEXITING...\n", strerror(errno));

exit(errno);

}

duba = (double)a;

dubb = (double)b;

dubc = (double)c;

negb=0-b;

disc = dubb\*dubb-4\*duba\*dubc;

if(disc < 0){

fprintf(stdout, "Solution: 2 complex roots.\n");

exit(0);

}

sqr = sqrt(disc);

root1 = (negb+sqr)/(duba\*2);

root2 = (negb-sqr)/(duba\*2);

fprintf(stdout, "Root 1: %.40f\nRoot 2: %.40f\n", root1,root2);

//fprintf(stderr, "a: %.40f\nb: %.40f\nc: %.40f\n",a,b,c);

return 0;

}