**Title: SOLVING A QUADRATIC EQUATION** Date: 10/29/2018

Team Members: Safwan Kadir, David Moussalli, Dakota Grant

**Requirements**

Language: C

Platform: Linux

Build System: Make File

**Host Website**

https://github.com/dmoussalli/QuadSolver

**Stories**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Description** | **Time** | **Risk (1-5)** | **% Complete** | **Total Time** |
| We expect the program to validate and have quad precision | 2 weeks | 4 | 100% | 2 weeks |
| We expect the system to warn us if there is any chance of the system rounding off the numbers and providing us with an inaccurate result | 6 weeks | 3 | 100% | 6 weeks |
| We will set our version control as Git and host it on GitHub. We also will have our version control for documents on OneDrive as it has appropriate Version Control properties. | 3 weeks | 2 | 100% | 3 weeks |
| We plan to have Unit Testing, Coverage Testing, Functionality Testing and Usability Testing. All Unit Testing will be done extensively for each module with annotations and will be done as module/functions are developed | 4 weeks | 4 | 80% | 4 weeks |
| We expect the program to run with –Wall –Wpedantic for make and expect no errors. | 4 weeks | 4 | 100% | 4 weeks |

**Pseudocode Snippet for QuadSolver:**

QuadSolver

Define double a, b, c, root1, root2, discriminant

Get a,b,c from user

discriminant = b\*b-4\*a\*c

If discriminant>0

Root1 = (-b+sqrt(discriminant))/(2\*a)

Root2 = (-b-sqrt(discriminant))/(2\*a)

Print root1 and root2

Else if discriminant ==0

root1 = root2 = -b/(2\*a)

Print root1 and root2 which are equal

End If

//code has not taken into factor of roots that are not real

End QuadSolver

**Program Snippets**

QuadSolver.c:

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h> |
| #include <math.h> |
| #include <ctype.h> |
|  |
|  |
| int main(int argc, char const \*argv[]) { |
| if(argc!=4){ |
| printf("Usage ./a.out input1 input2 input3\nProgram Exitted\n"); |
| exit(-1); |
| } |
| if(isalpha(argv[1][0])||isalpha(argv[2][0])||isalpha(argv[3][0])){ |
| printf("Usage ./a.out input1 input2 input3\nProgram Exitted\n"); |
| exit(-1); |
| } |
|  |
| float a= atof(argv[1]); |
| float b= atof(argv[2]); |
| float c= atof(argv[3]); |
|  |
|  |
|  |
| double discriminant=0; |
| discriminant = b\*b-4\*a\*c; |
| double root1=0; |
| double root2=0; |
| printf("a: %.2f\n",a ); |
| printf("b: %.2f\n",b ); |
| printf("c: %.2f\n",c ); |
| printf("Discriminant: %.2f\n",discriminant ); |
| if (discriminant>0){ |
| root1 = (-b+sqrt(discriminant) )/(2\*a); |
| root2 = (-b-sqrt(discriminant))/(2\*a); |
|  |
| } |
| else if (discriminant ==0){ |
| root1 = root2 = -b/(2\*a); |
|  |
| }else { |
| printf("Root is an imaginary number\n\n"); |
| exit(0); |
| } |
| printf("Root 1: %f\n",root1 ); |
| printf("Root 2: %f\n\n",root2 ); |
| return 0; |
| } |

UnitTest.c:

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

char assertEquals(int a, int b);

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

float a= atof(argv[1]);

float b= atof(argv[2]);

float c= atof(argv[3]);

double discriminant=0;

discriminant = b\*b-4\*a\*c;

double root1=0;

double root2=0;

printf("a: %.2f\n",a );

assertEquals(a,1);

printf("b: %.2f\n",b );

assertEquals(b,2);

printf("c: %.2f\n",c );

assertEquals(c,1);

printf("Discriminant: %.2f\n",discriminant );

if (discriminant>0){

root1 = (-b+sqrt(discriminant) )/(2\*a);

root2 = (-b-sqrt(discriminant))/(2\*a);

}

else if (discriminant ==0){

root1 = root2 = -b/(2\*a);

}else {

printf("Root is an imaginary number\n\n");

exit(0);

}

printf("Root 1: %f\n",root1 );

printf("Root 2: %f\n\n",root2 );

return 0;

}

char assertEquals(int a, int b){

if(fabs(a-b)<0.00001)

return printf("input == number read\n");

else

return printf("input != number read\n");

}

**Programming Standards**

-Standard used is c99

-Functions are to be defined as they are used

- Variable should be defined on top of methods

-If a variable is not being used, it should not be there

-Provide comments after every methods

- program layout should be prototype followed by main and then functions

- IEEE floating pointers will use standard of 754-2008

- gcc version we will be using is gcc 7.3