CISC 5300—Programming in C++
Fall, 2018

Programming Project #2: Solving Quadratic Equations

Date Due: Wednesday 26 September 2018

Write a program that prompts the user to enter the coefficients of a quadratic polynomial $ax^2 + bx + c$. It should then solve the polynomial equation

$$ax^2 + bx + c = 0.$$

You should use the quadratic formula to solve this equation; if you don't remember it, you'll find it later on in this handout.

Assuming that this is a genuine quadratic equation (i.e., the leading coefficient a is nonzero), there are three possible situations:

- Two distinct real solution, such as for the equation $x^2 - 3x + 2 = 0$, which has the solutions $x = 1$ and $x = 2$.
- Only one real root[1] as in the case $x^2 - 2x + 1 = 0$, which has the solution $x = 1$.
- No real roots, as with $x^2 + 1 = 0$.

You can distinguish between these by checking the discriminant $b^2 - 4ac$.

Of course, you can't control the input your users are giving you, which means that $a = 0$ is a possibility; in this case the quadratic equation has now degenerated to a linear equation. So your code should handle the case $a = 0$. Note that when $a = 0$, we have two subcases:

- When $b \neq 0$, the equation $bx + c = 0$ can be solved for x. For example, $2x + 5 = 0$ has the solution $x = -2.5$.
- When $b = 0$, you're now left with something of the form $c = 0$ to "solve". For example, $1 = 0$ is a contradiction, whereas $0 = 0$ is an identity.

A few considerations:

1. You are to do this using only the techniques found in Chapters 1 through 4 of the text.
2. You should do this project, in a subdirectory `proj2` of your `~/private/programming-c++` directory. See the Project 1 handout for further discussion, and adjust accordingly.
3. I am providing you with the following goodies, to help you get started.
   - An executable version of this program for you to try out, named `proj2-agw`.
   - A "stub version" of the C++-code, named `proj2-stub.cc`. I'll also explain this further on.

   You should copy these files into your working directory, via the `cp` command. Assuming that you're logged into one of our machines (i.e., on `erdos` or a lab machine) and that the working directory described in (2) above is your current directory, the command

[1]More precisely, it's a repeated real root of multiplicity two.

```
cp ~agw/class/programming-c++/share/proj2/* .
```

will do this. **The dot is part of the command; it means "current working directory".** Once you've copied it over, *please* try it out, so you can see what I consider to be good input/output behavior, not to mention how I handle various strange data values (as mentioned above).

4. Of course, you'll need to compute the square root of the discriminant. The standard library has a function `sqrt()` that will be pretty helpful here. Its prototype (which you get for free) is

   ```
   double sqrt(double x);
   ```

   The return value of `sqrt(blah)` is (not surprisingly) the square root of `blah`, assuing that `blah` is non-negative.[2]

5. As you know by now, it's a good idea to let a function carry out a well-encapsulated subtask. When done carefully, this can shorten the `main()` function; my `main()` is only 21 lines long. I used two subsidiary functions:

   - The function having prototype

     ```
     void solve_linear(double b, double c);
     ```

     solves a linear equation $bx + c = 0$, distinguishing between the cases $b \neq 0$ and $b = 0$. Please make sure that this function does not divide by zero! For the record, my version was 14 lines long.

   - The function having prototype

     ```
     void solve_quadratic(double a, double b, double c);
     ```

     solves a genuine quadratic equation $ax^2 + bx + c = 0$, "genuine" meaning that $a \neq 0$. Mine was 21 lines long. This function should not take the square root of a negative number. Also, note that since the discriminant is zero for a repeated root, the quadratic formula can be used without the square root of the discriminant.

   The main reason for using functions here (other than simply getting practice) is that each function (including the `main()` function) is small enough to understand easily; in particular, each function takes up less than a screenful of space.

   At this point, we can talk about `proj2-stub.cc`. This file

   - takes care of the overall logic flow,
   - declares the two functions mentioned above, and
   - gives "stub" implementations of these functions, i.e., minimal implementations for the program to compile.

[2]More honestly, the `sqrt()` function doesn't really return the exact square root of its argument, but an approximation to same. But it's a *very* good approximation.

Your first step will be to make a copy of `proj2-stub.cc`, which you'll call `proj2.cc`. For fun, you should compile this file "as is," and see if you understand why it works the way it does, in its limited way. If you now fill in the blanks (which means implementing the input section and completing the implementation of the two functions mentioned above), you'll now have a working program.

Big hint: Do one thing at a time. First, take care of the input phase (this should be pretty easy). Run the program, to make sure that this piece works right. Now, implement one of the two functions, after which you should run the program and test that the program works right so far. Finally, implement the other function, and do yet more testing.

6. Here is a good collection of sample data to use.

a	b	c	Equation	Explanation
1	-3	2	$x^2 - 3x + 2 = 0$	Two real roots ($x = 1$ and $x = 2$)
2	-6	4	$2x^2 - 6x + 4 = 0$	Two real roots ($x = 1$ and $x = 2$)
1	-2	1	$x^2 - 2x + 1 = 0$	One double root ($x = 1$)
1	-2	4	$x^2 - 2x + 4 = 0$	No real roots
0	2	4	$2x + 4 = 0$	Linear equation, root $x = -2$
0	0	0	$0 = 0$	An identity
0	0	1	$1 = 0$	A contradiction

7. Use the `photo` program so you can have something to turn in. *Don't do this until the program is working!* The commands you should issue are as follows:

```
photo
cat proj2.cc
g++ -std=c++17 -o proj2 proj2.cc
proj2
proj2
proj2
proj2
proj2
proj2
proj2
exit
```

You'll be running `proj2` once for each test dataset. Once again, see the Project 1 handout if you're still unclear about `photo`.

Good luck!

Here is a listing of `proj2-stub.cc`:

```
/**********************************************************************
 *
 * Project 2: Quadratic Equation Solver
 *
 * Say something here.
 *
 * Author: Harry Q. Bovik, PhD
 * Date:   17 September 2018
 *
 **********************************************************************/

#include <bjarne/std_lib_facilities.h>

// function declarations
void solve_linear(double b, double c);
void solve_quadratic(double a, double b, double c);

int main()
{
   // input the coefficients of the polynomial
   double a, b, c;                      // coefficients of the polynomial
   // figure out the rest yourself!!

   // handle degenerate case (linear equation) and quit
   if (a == 0)  // linear equation, not quadratic
      solve_linear(b, c);
   else  // genuine quadratic equation ... forge ahead
      solve_quadratic(a, b, c);
}

// solve the linear equation b*x + c == 0
void solve_linear(double b, double c)
{
   cout << "Trying to solve linear equation "
        << b << "*x + " << c << " == 0\n";
   // figure out the rest yourself!
}

// use classical quadratic formula to solve a genuine quadratic equation
// a*x^2 + b*x + c ==0, with a != 0
void solve_quadratic(double a, double b, double c)
{
   cout << "Trying to solve the quadratic equation "
        << a << "*x*x + " << b << "*x + " << c << " == 0\n";
   // figure out the rest yourself!
}
```