Project 1

Basic I/O

Overview

In this project you will begin to explore the basics of input and output from a C program. In order to perform error checking and to provide a robust, interactive program, you will also get to grips with looping and conditionals. Additionally, you will get some practice using Makefiles and the compiler.

All this will be accomplished by building a simple ballistic calculator which can determine for how long and how far a projectile would fly when launched with a given launch height, angle, and initial velocity.

Submission Instructions

This and all other assignments will be submitted through BBLearn. Look for the submission link in the same place you found this assignment.

Submit all your .c and .h files along with your Makefile, but do not zip them!

Technical Description and Instructions

Your program will take some inputs from the user and use them to perform a ballistic calculation, that is, it will calculate how long it would take for a launched object to hit the ground and how far the launched object went. Specifically, it will prompt the user for a starting height above the ground for the object, the angle of launch, and the initial launch velocity. Once good inputs have been collected, your program will then display the time and distance results to the user.

Note that your program may be evaluated by a program I compose for just this purpose. As such, when the instructions indicate that something should be printed in a certain way, failing to do so may cause my evaluator to reject your program!

Your program should do the following:

1. Display a welcome message.
2. Prompt for a height above the ground *in meters*[1]. Your program should accept numbers with decimal parts such as 16.037 for this input. Any number less than 0 or something that is not a number should be rejected with this message: "Input Error: Expected a positive number". *All error messages should end with at least one newline (\n).* If you reject an input, reprint the initial prompt and wait for input again.
3. After receiving acceptable input for the height, prompt the user for a launching angle. The angle given will be *in degrees* and should be between 90°and -90°. The launch angle may have a decimal part. An invalid angle or something that is not a number should be rejected with this message: "Input Error: Expected an angle between 90 and -90". Like the previous item, re-prompt for input until a correct input is received.

[1] Note that the user will not specify the units in their input (like 10.5m) but that you should inform the user that the number they enter will be interpreted as being in meters. This is the case for all inputs your program takes in.

4. Your program should then prompt for an initial launching velocity. The launch velocity should be greater than 0 and should be in meters per second. Any negative number or something that is not a number should be rejected with this message: "Input Error: Expected a positive number". Like before, re-prompt until correct input is received.
5. Calculate the initial vertical and horizontal velocity from the launch angle and launch velocity[2].
6. Calculate the time it will take for the projectile to reach the ground. For a positive launch angle, first calculate how long until the projectile returns to the launch height $2 \times \frac{verticalvelocity}{g}$. For the fall time from the launch height to the ground, use $\frac{height \times 16}{verticalvelocity \times 15}$ which is an approximation. Add these two times together, if applicable, to obtain the total flight time.
7. To find the horizontal distance traveled, simply multiply the horizontal velocity by the flight time.

```
launch height: <user's input>
launch angle: <user's input>
launch velocity: <user's input>
time to impact: <result>
distance travelled: <result>
```

Things to Remember and Helpful Hints:

- You should compile and run your program many times as you write it so that you can fix bugs as they arise
- You can break out of an enclosing loop with the `break` keyword, for instance the following code snippet loops until the user inputs '7':

```
while(1) { // This loops forever since 1 is always true
  int guess = 0;
  printf("Guess a number between 1 and 10\n\t");
  scanf("%d", &guess); // %d is used to retrieve integers (no decimal point)
  if(guess == 7) {
    break;
  }
}
```

- You can use `<condition>&& <condition>` inside an `if` or `while` statement to require that both conditions be fulfilled and `<condition>| | <condition>` to require that one, the other, or both conditions are fulfilled (&& == and, || == or)

Grading Specification

For your submission to receive a grade of 'pass', it must fulfill the following criteria:

- **It must be submitted correctly**.
- I must be able to compile your program simply by invoking "make basic_io".
- The program must compile with no warnings or errors.
- The program should run with no errors when given the correct inputs and should produce a plausibly correct result without crashing.
- The program should reject most bad inputs at each step without crashing. I'm not requiring perfection here, but failing to do any error checking is unacceptable.

[2] This should only require some simple geometry. We'll discuss the use of the necessary math functions in class.