Tyler Tackett

tjtackett

CS1300 AA Fall 2019

Program 2

Baseball Programming Problem

11/1/19

Baseball is known as America's pastime. It involves taking a bat and hitting a ball across a field that is in the shape of a diamond. Often times, baseball is regarded as a game of numbers, where the best teams can be built by using statistics collected over the careers of various players. The role numbers play in baseball can be taken even further to the velocity the ball travels and the angle it takes off at once the batter hits it. Write a program to calculate the distance the ball travels, how long it stays in the air, the maximum height the ball reaches, whether or not it is a home run, and whether or not it is hit out of the park.

A standard baseball field has a distance of 400 ft between the fence marking the edge of the field and home plate, where the batter is. The fence is usually 10 ft high. A guaranteed homerun will pass the edge of the field at a height greater than the fence, where an outfielder may intercept it. For this particular stadium, for the ball to make it out of the park it must clear a distance of 200 feet beyond the field's edge while additionally having a height exceeding the 30ft catch net.

A chronometer is used to measure the horizontal velocity of the baseballs being hit by the various players. A high-speed camera is used to determine the angle with respect to the horizontal plane that the ball is struck at. The typical strike zone where the ball contacts the bat or misses is between 1.6ft and 3.6 ft.

Test your program using the data below:

Player	**Initial Horizontal Velocity** (feet/sec)	**θ** (degrees)	**Bat Contact height** (feet)
99	161	20	3.5
42	176	15	2.7
37	139	27	1.9
68	150	22	3.1

In order to write the program we must consider first the parameters that need to be set to qualify a home run and knocking the ball out of the park.

From the problem statement, it can be said that if the ball height at 400 ft is greater than 10 ft, it is a home run.

It can also be said that if the ball height at 400ft plus an additional 200 ft is greater than 30 ft, it is knocked out of the park.

So, we may define parameters as follows:

$$height\ of\ fence = y_w$$

$$height\ of\ catchnet = y_p$$

$$height\ of\ ball\ at\ the\ time\ when\ it\ reaches\ fence = y_{tw}$$

$$height\ of\ ball\ at\ the\ time\ when\ it\ reaches\ edge\ of\ park = y_{tp}$$

$$if\ y_{tw} > y_w\ then\ it\ is\ a\ home\ run$$

$$if\ y_{tp} > y_p\ \ then\ it\ is\ out\ of\ the\ park$$

*Note the last 2 parameters cannot be programed until there is either a numerical or mathematical definition for y_{tw} and y_{tp}.

The height of the ball is obviously a function of time, but in order to determine how it varies with time, we must define the input variables.

The input variables that alter between players will be the horizontal velocity of the hit ball, the angle the ball is hit at, and the height that the ball makes contact with the bat.

We can define these variables as:

$$initial\ horizontal\ velocity\ of\ the\ ball = v_{ix}$$

$$angle\ of\ impact\ with\ respect\ to\ horizontal\ plane = \theta$$

$$height\ the\ ball\ makes\ contact\ with\ the\ bat = y_i$$

In order to calculate our desired quantities, we will use the simple kinematics equations from physics that describe projectile motion. We can make the assumption that wind and air resistance are negligible for a baseball.

We also know that horizontal and vertical motion behave independently, so given a velocity and an angle, we may break our velocity into horizontal and vertical vector components.

Because the chronometer only measures, horizontal velocity, we already know the horizontal velocity of the ball. The initial vertical velocity of the ball can be given by the following equation:

$$initial\ vertical\ velocity = v_{iy} = v_{ix} * tan\theta$$

The ball will experience the force of gravity acting on it in the vertical direction, but should not experience any forces acting on it in the horizontal direction. So v_{ix} will be constant.

This means that the time it takes the ball to reach the fence is simply the distance from homeplate to the fence divided by the initial horizontal velocity of the ball.

Given by:

$$time\ for\ ball\ to\ reach\ fence = t_w = \frac{400ft}{v_{ix}\left(in\,\frac{ft}{sec}\right)}$$

And the time for the ball to leave the park is given by:

$$time\ for\ ball\ to\ reach\ edge\ of\ park = t_p = \frac{400ft + 200ft}{v_{ix}\left(in\ \frac{ft}{sec}\right)}$$

Knowing the time it takes for the ball to either reach the fence or leave the park allows us to calculate the height of the ball when it does reach the fence or the edge of the park, because the height of the ball is a function of time.

From the basic equations of projectile motion:

$$\Delta y = v_{iy}t + \frac{1}{2}at^2$$

Where:

$$\Delta y = change\ in\ height$$

$$v_{iy} = initial\ vertical\ velocity$$

$$t = time$$

$$a = acceleration$$

In our case the baseball is being hit on Earth, where the acceleration due to gravity is –g, the gravitational constant.

Define:

$$acceleration\ due\ to\ gravity = g = 32.10\ {}^{ft}/_{sec^2}$$

So, finally, y_{tw} can be given the mathematical definition as follows:

$$y_{tw} = v_{iy} * t_w - \frac{1}{2} * g * t_w^2 + y_i$$

And for y_{tp}

$$y_{tp} = v_{iy} * t_p - \frac{1}{2} * g * t_p^2 + y_i$$

We can now determine whether or not we reached a home run and/or knocked the ball out of the park.

Next we can determine how high the ball travels.

Using the equation of motion:

$$v_{fy}^2 = v_{iy}^2 + 2a\Delta y$$

It can be assumed at the maximum height the ball travels, its final vertical velocity is 0 due to it decelerating due to gravity, so we can write the equation:

$$0 = v_{iy}^2 - 2g * (y_{max} - y_i)$$

Which simplifies to

$$maximum\ height = y_{max} = y_i + \frac{v_{iy}^2}{2 * g}$$

Next we can determine the balls travel time from the following equation of motion:

$$\Delta y = v_{iy} * t + \frac{1}{2}a * t^2$$

Where Δy is the change in height, assuming a level surface, the final height of the ball will be 0 ft.

So we can write this as

$$0 - y_i = v_{iy} * t - \frac{1}{2} * g * t^2$$

Where *t* is the travel time of the ball. And all other values have previously been specified

Noticing that solving for *t* is the same as solving the roots of a quadratic equation, we can rearrange this and use the quadratic formula to determine the travel time:

$$travel\ time\ in\ seconds = t_{trav} = \frac{-v_{iy} - \sqrt{v_{iy}^2 + 2 * g * y_i}}{-g}$$

The final value that needed to be calculated was the horizontal distance the ball travelled, which is simply calculated by multiplying the horizontal velocity by the travel time as follows:

$$distance\ traveled\ in\ feet = x_{trav} = v_{ix} * t_{trav}$$

Design

Restated Problem: The goal of this program is to see where the ball lands after it's hit by the player. The ball can either land in the park, be a homerun, or be completely knocked out of the park.

Inputs -> Initial horizontal velocity, impact angle, ball contact height

Outputs-> Distance traveled, airtime, max height, and where the ball lands

In order to get outputs, the inputs are run through a series of given formulas.

Variables:

double gravity

int distanceOfFence

int distanceOfEdge

double initialHorizontalVelocity

double impactAngle

double contactHeight

bool verifiedInput

double initialVerticalVelocity

double timeToFence

double timeToEdge

double ballHeightAtFence

double ballHeightAtEdge

double maxBallHeight

double ballTravelTime

double distanceBallTraveled

int result

Functions:

getInput

- ➔ Collects Initial horizontal velocity, impact angle, ball contact height from a user. Must return all inputs in a way to the other functions can use them fenceTime

- ➔ NEEDS: Initial horizontal velocity
- ➔ Calculates the time it takes the ball to reach the fence
    - o distanceOfFence / initialHorizontalVelocity

parksEdgeTime

- ➔ NEEDS: Initial horizontal velocity
- ➔ Calculates the time it takes the ball to reach the edge of the park
    - o distanceOfEdge / initialHorizontalVelocity

heightAtFence

- ➔ NEEDS: Vertical velocity, timeToFence, ball contact height
- ➔ Calculates the balls height when it reaches the fence
    - o initialVerticalVelocity * timeToFence – .5 * GRAVITY * timeToFence^2 + contactHeight

heightAtEdge

- ➔ NEEDS: Vertical velocity, timeToEdge, ball contact height
- ➔ Calculates the balls height when it reaches the edge
    - o initialVerticalVelocity * timeToEdge – .5 * GRAVITY * timeToEdge^2 + contactHeight

maxHeight

- ➔ NEEDS: Vertical velocity, ball contact height, gravity
- ➔ Calculates the balls maximum height
    - o contactHeight + (initialVerticalVelocity^2 / 2 * GRAVITY)

airTime

- ➔ NEEDS: Vertical velocity, ball contact height, gravity
- ➔ Calculates the total time the ball is in the air
    - o (-initialVerticalVelocity – sqrt (initialVerticalVelocity^2 + 2 * GRAVITY * contactHeight)) / -GRAVITY

distanceTraveled

- ➔ NEEDS: ballTravelTime, horizontal Velocity
- ➔ Calculates the total distance the ball travels
    - o initialHorizontalVelocity * ballTravelTime

hitResult

- ➔ NEEDS: height AT Fence, height AT Edge, height OF fence, height OF edge, distance traveled
- ➔ Calculates where the ball lands
    - o If ballHeightAtEdge > 30 && distanceBallTraveled > distanceOfEdge
        - ▪ Out-Of-Park
        - ▪ result = 2

    - If ballHeightAtFence > 10 && distanceBallTraveled > distanceOfFence
        - Homerun
        - result = 1
    - Otherwise it's in the park
        - result = 0

Output

- ➔ NEEDS: distanceBallTraveled, ballTravelTime, maxBallHeight, result
- ➔ Outputs Distance traveled, airtime, max height, and where the ball lands

Source Code

```
#include <iostream>

#include <fstream>

#include <cmath>

#include <iomanip>


using namespace std;

//function prototypes

void getInput (double& initialHorizontalVelocity, double& impactAngle, double&
contactHeight, bool& verifiedInput);

double fenceTime (double horizontalVelocity);

double parksEdgeTime (double horizontalVelocity);

double heightAtFence (double verticalVelocity, double fenceTime, double

        baseHeight);

double heightAtEdge (double verticalVelocity, double edgeTime, double

        baseHeight);

double maxHeight (double verticalVelocity, double baseHeight, double gravity);

double travelTime (double verticalVelocity, double gravity, double baseHeight);

double distanceTraveled (double travelTime, double horizontalVelocity);

int hitResult (double heightAtFence, double distanceToFence, double

        heightAtEdge, double distanceToEdge, double distanceTraveled);

void output (double distanceTraveled, double airTime, double maxHeight, int

        hitResult);


//constants
```

```
const double GRAVITY = 32.10;
const double DISTANCEOFFENCE = 400;
const double DISTANCEOFEDGE = 600;

int main ()
{
    double initialHorizontalVelocity;
    double impactAngle;
    double contactHeight;
    bool verifiedInput;

    //Calls input function
    getInput(initialHorizontalVelocity, impactAngle, contactHeight, verifiedInput);

 //Checks to see if the user input is valid
    if(verifiedInput)
    {
        //Calls functions with actual parameters
        double initialVerticalVelocity = initialHorizontalVelocity *
                        tan(impactAngle    * M_PI /180);
        double timeToFence = fenceTime (initialHorizontalVelocity);
        double timeToEdge = parksEdgeTime (initialHorizontalVelocity);
        double ballHeightAtFence = heightAtFence (initialVerticalVelocity,
                        timeToFence, contactHeight);
        double ballHeightAtEdge = heightAtEdge (initialVerticalVelocity,
                    timeToEdge, contactHeight);
        double maxBallHeight =  maxHeight (initialVerticalVelocity, contactHeight,
                    GRAVITY);
        double ballTravelTime = travelTime (initialVerticalVelocity, GRAVITY,
                    contactHeight);
        double distanceBallTraveled = distanceTraveled (ballTravelTime,
                    initialHorizontalVelocity);
        int result = hitResult (ballHeightAtFence, DISTANCEOFFENCE,
                    ballHeightAtEdge, DISTANCEOFEDGE, distanceBallTraveled);
        output (distanceBallTraveled , ballTravelTime, maxBallHeight , result);
        return 0;
    }
return 0;
}


//Collects user input, validates it, and assigns it to variables
void getInput (double& initialHorizontalVelocity, double& impactAngle, double&
contactHeight, bool& verifiedInput)
{
    cout << "Please input an Intitial Horizontal Velocity, Impact Angle, and Contact
            Height of the bat and ball" << endl;
    //tests input to make sure the input is a number
    if(!(cin >> initialHorizontalVelocity))
    {
        cout << "Please don't enter letters of words, only numbers" << endl;
        verifiedInput = false;
    }
    else if(!(cin >> impactAngle))
    {
        cout << "Please don't enter letters of words, only numbers" << endl;
        verifiedInput = false;
    }
    else if(!(cin >> contactHeight))
    {
        cout << "Please don't enter letters of words, only numbers" << endl;
        verifiedInput = false;
    }
    else
    {
        //checks to make sure the contaact height is within range
        if((1.6 <= contactHeight) && (contactHeight <= 3.6))
        {
            verifiedInput = true;
        }
        else
        {
            verifiedInput = false;
            cout << "Make sure your contact height is between 1.6 and 3.6" << endl;
        }
    }
```

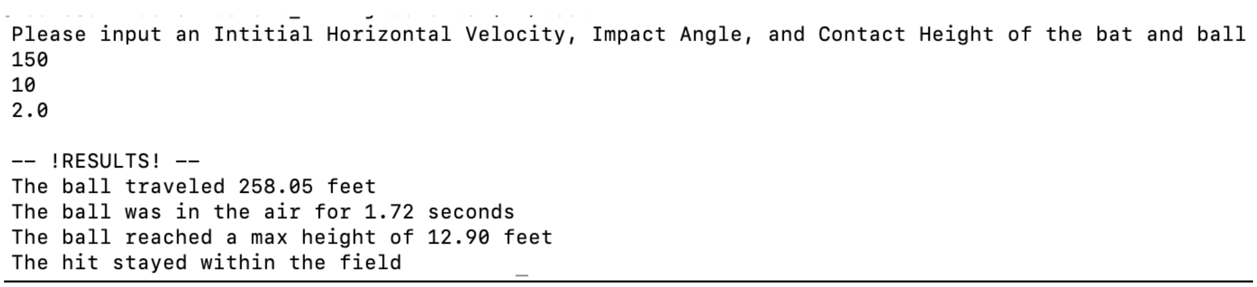
```
}


//finds the time it takes for the ball to reach the fence
double fenceTime (double horizontalVelocity)
{
   double timeToFence = DISTANCEOFFENCE / horizontalVelocity;
   return timeToFence;
}


//finds the time it takes for the ball to reach the edge of the park
double parksEdgeTime (double horizontalVelocity)
{
   double timeToEdge = DISTANCEOFEDGE / horizontalVelocity;
   return timeToEdge;
}


//finds the height of the ball at the fence
double heightAtFence (double verticalVelocity, double fenceTime, double baseHeight)
{
   double ballHeightAtFence = (verticalVelocity * fenceTime) - (0.5 * GRAVITY
         * pow(fenceTime, 2) + baseHeight);
   return ballHeightAtFence;
}


//finds the balls height at the edge of the park
```

```
double heightAtEdge (double verticalVelocity, double edgeTime, double baseHeight)
{
   double ballHeightAtEdge = (verticalVelocity * edgeTime) - (0.5 * GRAVITY *
         pow(edgeTime, 2) + baseHeight);
   return ballHeightAtEdge;
}

//finds the maximum height that the ball reaches
double maxHeight (double verticalVelocity, double baseHeight, double gravity)
{
   double maxBallHeight = baseHeight + (pow(verticalVelocity, 2) / (2 *
         GRAVITY));
   return maxBallHeight;
}

//finds the total time the ball is in the air
double travelTime (double verticalVelocity, double gravity, double baseHeight)
{
   double ballTravelTime = ( (- 1 * verticalVelocity) - sqrt(pow(verticalVelocity,2)
         + (2 * GRAVITY * baseHeight))) / (-1 * GRAVITY);
   return ballTravelTime;
}

//finds the horizontal distance that the ball travels
double distanceTraveled (double travelTime, double horizontalVelocity)
```

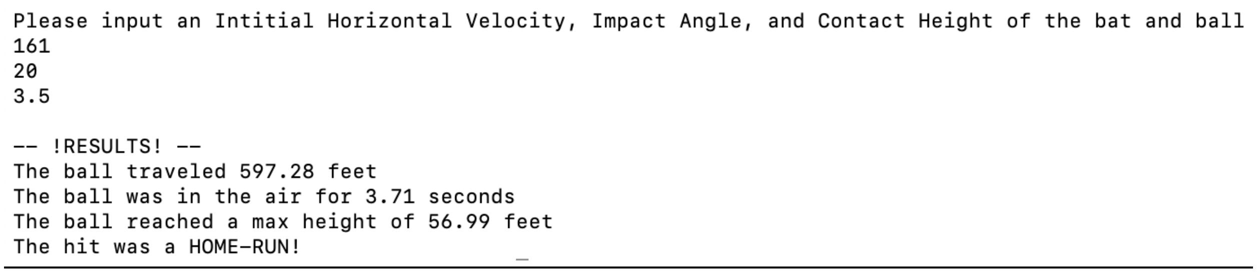
```
{
   double distanceBallTraveled = travelTime * horizontalVelocity;
   return distanceBallTraveled;
}


//compares values and finds out where the ball lands
int hitResult (double heightAtFence, double distanceToFence, double heightAtEdge, double
distanceToEdge, double distanceBallTraveled)
{
   int hitResult;
   if(heightAtEdge > 30 && distanceBallTraveled > distanceToEdge)
   {
      hitResult = 2;
   }
   else if(heightAtFence > 10 && distanceBallTraveled > distanceToFence)
   {
      hitResult = 1;
   }
   else
   {
      hitResult = 0;
   }
   return hitResult;
}
```

```
//outputs all results in two different ways
void output (double distanceTraveled, double airTime, double maxHeight, int hitResult)
{
    ofstream outData;
    outData.open("Output.txt");

    outData << fixed << showpoint;
    outData << setprecision(2);

    cout << fixed << showpoint;
    cout << setprecision(2);

    outData << "The ball traveled " << distanceTraveled << " feet" << endl;
    outData << "The ball was in the air for " << airTime << " seconds" << endl;
    outData << "The ball reached a max height of " << maxHeight << " feet" <<
            endl;

    cout << endl;
    cout << "-- !RESULTS! --" << endl;
    cout << "The ball traveled " << distanceTraveled << " feet" << endl;
    cout << "The ball was in the air for " << airTime << " seconds" << endl;
    cout << "The ball reached a max height of " << maxHeight << " feet" << endl;

    if(hitResult == 0)
    {
        outData << "The hit stayed within the field" << endl;
        cout << "The hit stayed within the field" << endl;
    }
    else if(hitResult == 1)
    {
        outData << "The hit was a HOME-RUN!" << endl;
        cout << "The hit was a HOME-RUN!" << endl;
    }
    else
    {
        outData << "The hit was OUT OF THE PARK!" << endl;
        cout << "The hit was OUT OF THE PARK!" << endl;
    }
    outData.close();
}
```

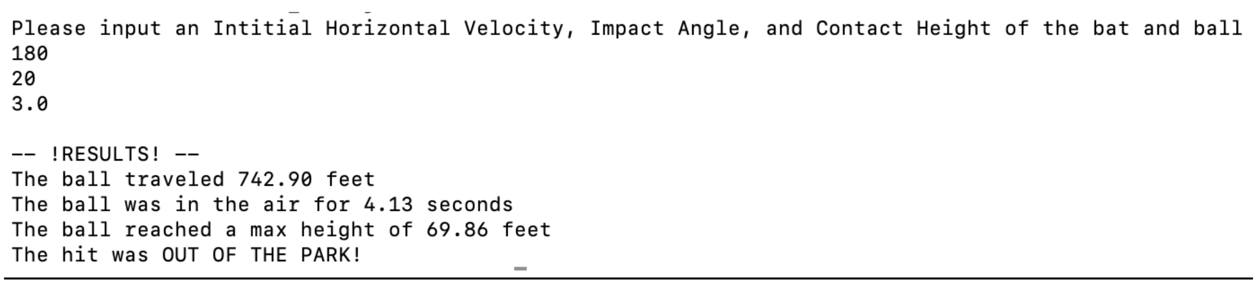
Outputs



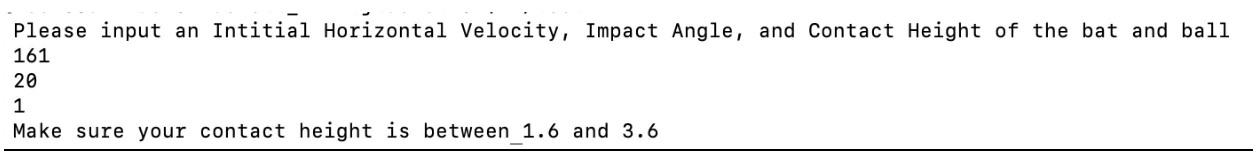
```
Please input an Intitial Horizontal Velocity, Impact Angle, and Contact Height of the bat and ball
150
10
2.0

-- !RESULTS! --
The ball traveled 258.05 feet
The ball was in the air for 1.72 seconds
The ball reached a max height of 12.90 feet
The hit stayed within the field
```



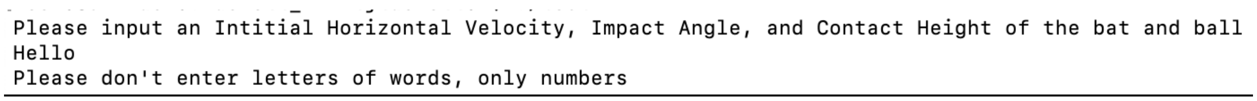
```
Please input an Intitial Horizontal Velocity, Impact Angle, and Contact Height of the bat and ball
161
20
3.5

-- !RESULTS! --
The ball traveled 597.28 feet
The ball was in the air for 3.71 seconds
The ball reached a max height of 56.99 feet
The hit was a HOME-RUN!
```



```
Please input an Intitial Horizontal Velocity, Impact Angle, and Contact Height of the bat and ball
180
20
3.0

-- !RESULTS! --
The ball traveled 742.90 feet
The ball was in the air for 4.13 seconds
The ball reached a max height of 69.86 feet
The hit was OUT OF THE PARK!
```



```
Please input an Intitial Horizontal Velocity, Impact Angle, and Contact Height of the bat and ball
161
20
1
Make sure your contact height is between 1.6 and 3.6
```



```
Please input an Intitial Horizontal Velocity, Impact Angle, and Contact Height of the bat and ball
Hello
Please don't enter letters of words, only numbers
```

Reflection

Project Summary

This project was a baseball program; it required 3 inputs and would output results after running the inputs through the given formulas. The inputs required were initial horizontal velocity, impact angle, and impact height. The primary output is where the ball landed, but other values such as the distance the ball traveled, how long the ball was in the air and the maximum height of the ball.

This project provided two learning points for me: get into a habit of planning before writing any code and how to use a reference parameter. Preplanning a program smooths out the actual programming process 100-fold and it’s an important practice that should always be done. Reference parameters are a surprisingly new concept that got overlooked in my previous programming class; they make input functions a lot simpler and are very effective.

Challenges

While programming this project I ran into a couple of issues collecting inputs and verifying said inputs. At first, the input function only had value parameters so the user inputs would only be saved inside of my input method and couldn’t be used for other functions. Once the value parameters were changed to reference parameters everything worked as expected. The next problem was verifying the inputs, at first I just tried to check to see if the inputs were numbers but that didn’t work very well since the program would error out if anything, but a number was entered anyway. The solution to this was to just check for the errors; if the program runs into an error while assigning the inputted value to a double then it asks for new input.

Lessons Learned

Takeaways from this program include: always remember to plan a program before you start coding, reference parameters are very useful in particular situations, and that sometimes certain ideas need to be abandoned for a better more efficient route.