**Course: ENSF 614** - Fall 2023  
**Lab #:** Lab 1  
**Instructor:** Mahmood Moussavi  
**Student Name:** Christian Valdez  
**Submission Date:** September 20, 2023

**Exercise B:**

/\*

\* Filename: lab1exe\_B.cpp

\* Assignment: Lab 1 Exercise B

\* Section: B01

\* Completed by: Christian Valdez

\* Submission date: Sep 20, 2023

\*/

#include <iostream>

#include <iomanip>

#include <cmath>

using namespace std;

const double G = 9.8; /\* Gravitational acceleration 9.8 m/s^2 \*/

const double PI = 3.141592654;

const int MAX\_ANGLE\_DEG = 95; // Maximum angle in degrees to be used in the table

/\*\*

\* Create a table displaying the time and distance traveled by a projectile

\* at varying angles for a given velocity.

\*

\* @param v - Initial velocity of the projectile.

\*/

void create\_table(double v);

/\*\*

\* Calculate the time of flight for a projectile.

\*

\* @param a - Gravitational acceleration.

\* @param v - Initial velocity of the projectile.

\* @return Time in seconds the projectile remains in the air.

\*/

double Projectile\_travel\_time(double a, double v);

/\*\*

\* Calculate the horizontal distance traveled by a projectile.

\*

\* @param a - Gravitational acceleration.

\* @param v - Initial velocity of the projectile.

\* @return Distance in meters the projectile covers horizontally.

\*/

double Projectile\_travel\_distance(double a, double v);

/\*\*

\* Convert a given angle from degrees to radians.

\*

\* @param d - Angle in degrees.

\* @return Equivalent angle in radians.

\*/

double degree\_to\_radian(double d);

double angleDeg = 0; // Angle in degrees for the projectile

double angleRad; // Angle in radians for the projectile

int main(void) {

double velocity;

// Prompt user for projectile velocity

cout << "Please enter the velocity at which the projectile is launched (m/sec): ";

cin >> velocity;

// Check for valid input

if (!cin) {

cout << "Invalid input. Bye...\n";

exit(1);

}

// Ensure the velocity is non-negative

while (velocity < 0) {

cout << "\nplease enter a positive number for velocity: ";

cin >> velocity;

if (!cin) {

cout << "Invalid input. Bye...";

exit(1);

}

}

// Generate the table for projectile data

create\_table(velocity);

return 0;

}

void create\_table(double v) {

double t, d; // time and distance

// Table header

cout << "Angle" << "\t\t t" << "\t\t d" << endl;

cout << " (deg)" << "\t\t (sec)" << "\t\t (m)" << endl;

cout << fixed << setprecision(6);

// Calculate and display data for each angle until MAX\_ANGLE\_DEG

while (angleDeg < MAX\_ANGLE\_DEG) {

angleRad = degree\_to\_radian(angleDeg);

t = Projectile\_travel\_time(G, v);

d = Projectile\_travel\_distance(G, v);

cout << angleDeg << "\t " << t << "\t " << d << endl;

angleDeg += 5; // Increase angle by 5 degrees

}

}

double degree\_to\_radian(double d) {

return angleDeg \* (PI / 180);

}

double Projectile\_travel\_time(double a, double v) {

return (2 \* v \* sin(angleRad)) / a;

}

double Projectile\_travel\_distance(double a, double v) {

return (v \* v \* sin(2 \* angleRad)) / a;

}

**Sample output:**

**A screenshot of a computer

Description automatically generated**

**Exercise D2**

A diagram of a bar code

Description automatically generated

**Exercise E**

/\*

\* Filename: lab1exe\_E.cpp

\* Assignment: Lab 1 rxercise E

\* Section: B01

\* Completed by: Christian Valdez

\* Submission date: Sep 20, 2023

\*/

#include <iostream>

using namespace std;

/\*\*

\* Converts milliseconds to minutes and seconds.

\* @param ms\_time - Time in milliseconds to convert.

\* @param minutes\_ptr - Pointer to store the resulting minutes.

\* @param seconds\_ptr - Pointer to store the resulting seconds.

\*/

void time\_convert(int ms\_time, int\* minutes\_ptr, double\* seconds\_ptr);

// Constants for time conversion

const int MS\_PER\_SECOND = 1000;

const int SECONDS\_PER\_MINUTE = 60;

int main(void) {

int millisec;

int minutes;

double seconds;

// Prompt the user to input time in milliseconds

cout << "Enter a time interval as an integer number of milliseconds: ";

cin >> millisec;

// Check if input conversion to integer was successful

if (!cin) {

cout << "Unable to convert your input to an int.\n";

exit(1);

}

// Check for non-negative time input

if (millisec < 0) {

cout << "Please enter a non-negative value.\n";

exit(1);

}

// Convert the input milliseconds to minutes and seconds

cout << "Doing conversion for input of " << millisec << " milliseconds ... \n", millisec;

time\_convert(millisec, &minutes, &seconds);

cout << "That is equivalent to " << minutes << " minute(s) and " << seconds << " second(s).\n";

return 0;

}

void convert\_time(int ms\_time, int\* minutes\_ptr, double\* seconds\_ptr) {

// Calculate the number of whole minutes

\*minutes\_ptr = ms\_time / (MS\_PER\_SECOND \* SECONDS\_PER\_MINUTE);

// Calculate the remaining seconds

\*seconds\_ptr = (double)(ms\_time % (MS\_PER\_SECOND \* SECONDS\_PER\_MINUTE)) / MS\_PER\_SECOND;

}

**Sample output:**

A screenshot of a computer

Description automatically generated