ENSF 614

Advanced System Analysis and Software Design

LAB 1

Author:

Steven Duong
(30022492)

Affiliation
Department of Electrical and Computer Engineering
University of Calgary
Calgary, Alberta

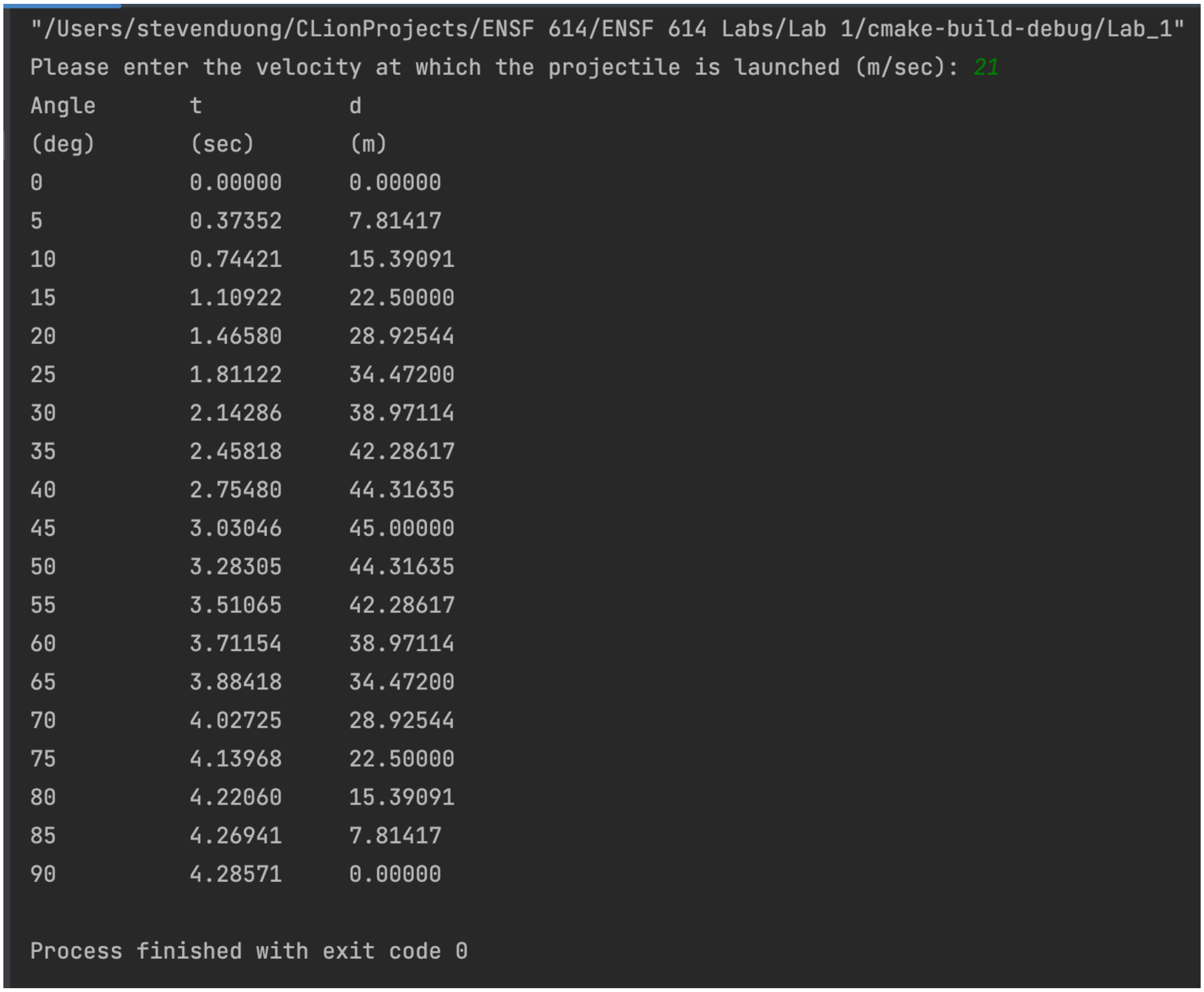
Lab Block: B01
Date of Report: Jan 20, 2023

Exercise B

C++ code to be executed:

```
/*
 *  File Name: lab1exe_B.cpp
 *  Assignment: ENSF 614 Lab 1 Exercise B
 *  Lab Section: Lab B01
 *  Created by: Mahmood Moussavi
 *  Completed by: Steven Duong (30022492)
 *  Submission Date: Jan 20, 2023
 */

#include <iostream>

#include <cmath>

#include <iomanip>

using namespace std;

const double G = 9.8; /* gravitation acceleration 9.8 m/s^2 */
const double PI = 3.141592654;

void create_table(double v);
double Projectile_travel_time(double a, double v);
double Projectile_travel_distance(double a, double v);
double degree_to_radian(double d);

int main(void) {
  double velocity;

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

  if (!cin) // means if cin failed to read
  {
    cout << "Invalid input. Bye...\n";
    exit(1);
  }

  while (velocity < 0) {
    cout << "\nPlease enter a positive number for velocity: ";
    cin >> velocity;
    if (!cin) {
      cout << "Invalid input. Bye...";
      exit(1);
    }
  }

  create_table(velocity);
  return 0;
}

// Creates a table with 3 columns which represents the angle in
degrees,
// time in seconds and distance in meters.
void create_table(double v) {
  int size = 90 / 5;
  int angles_d[size];

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

  for (int i = 0; i <= size; i++) {
    angles_d[i] = 5 * (i);

    cout << fixed;
    cout << setprecision(5);

    cout << angles_d[i] << "\t\t\t" <<
Projectile_travel_time(angles_d[i], v) << "\t\t" <<
      Projectile_travel_distance(angles_d[i], v) << endl;
  }
}

// Calculates the travel time for the projectile.
double Projectile_travel_time(double a, double v) {

  double rad = degree_to_radian(a);
  return (2 * v * sin(rad)) / G;
}

// Calculates the travel distance for the projectile.
double Projectile_travel_distance(double a, double v) {

  double rad = degree_to_radian(a);
  return abs((pow(v, 2) / G) * sin(2 * rad));
}

// Converts degrees to radians.
double degree_to_radian(double d) {

  return d * (PI / 180);
}
```

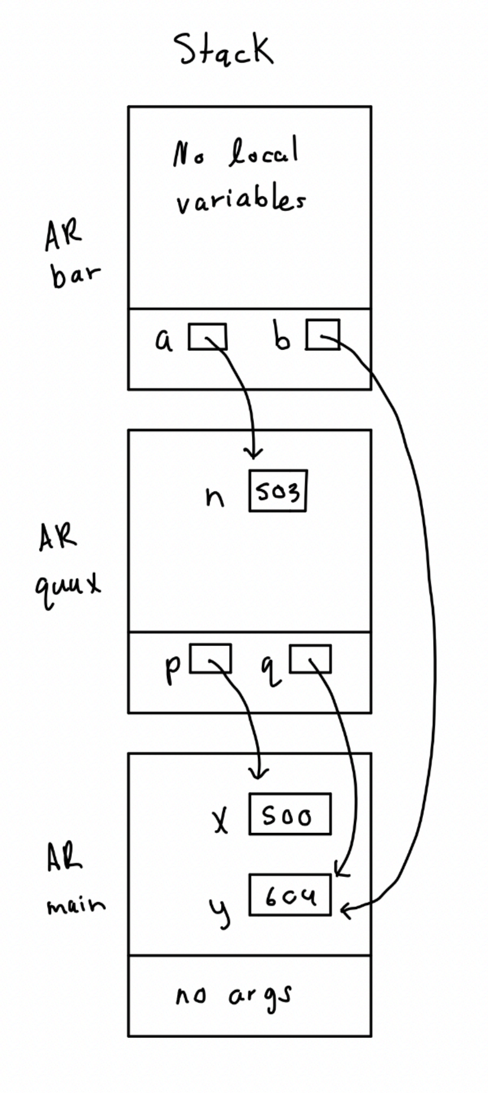
Program Output for Exercise B:



```
"/Users/stevenduong/CLionProjects/ENSF 614/ENSF 614 Labs/Lab 1/cmake-build-debug/Lab_1"
Please enter the velocity at which the projectile is launched (m/sec): 21
Angle       t           d
(deg)       (sec)       (m)
0           0.00000     0.00000
5           0.37352     7.81417
10          0.74421     15.39091
15          1.10922     22.50000
20          1.46580     28.92544
25          1.81122     34.47200
30          2.14286     38.97114
35          2.45818     42.28617
40          2.75480     44.31635
45          3.03046     45.00000
50          3.28305     44.31635
55          3.51065     42.28617
60          3.71154     38.97114
65          3.88418     34.47200
70          4.02725     28.92544
75          4.13968     22.50000
80          4.22060     15.39091
85          4.26941     7.81417
90          4.28571     0.00000

Process finished with exit code 0
```

Exercise D

AR Diagram for lab1_exeD2 at point one:

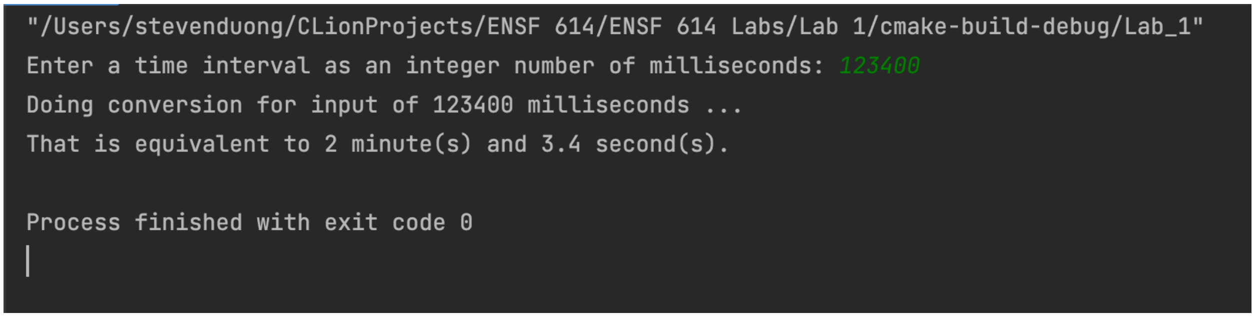


Exercise E

C++ Code to be executed:

```
/*
 *  File Name: lab1exe_E.cpp
 *  Assignment: ENSF 614 Lab 1 Exercise E2
 *  Lab Section: Lab B01
 *  Created by: Mahmood Moussavi
 *  Completed by: Steven Duong (30022492)
 *  Submission Date: Jan 20, 2023
 */

#include <iostream>

#include <cmath>

using namespace std;

void time_convert(int ms_time, int * minutes_ptr, double *
seconds_ptr);
/*
 * Converts time in milliseconds to time in minutes and seconds.
 * For example, converts 123400 ms to 2 minutes and 3.4 seconds.
 * REQUIRES:
 *    ms_time >= 0.
 *    minutes_ptr and seconds_ptr point to variables.
 * PROMISES:
 *    0 <= *seconds_ptr & *seconds_ptr < 60.0
 *    *minutes_ptr minutes + *seconds_ptr seconds is equivalent to
 *    ms_time ms.
 */

int main(void) {
  int millisec;
  int minutes;
  double seconds;

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

  // printf("Enter a time interval as an integer number of
milliseconds: ");
  cin >> millisec;

  // This allows the user to continuously input a number until the
number is
  // positive.
  while (millisec < 0) {
    cout << "Enter a time interval as an integer number of
milliseconds: ";

    // printf("Enter a time interval as an integer number of
milliseconds: ");
    cin >> millisec;
  }

  if (!cin) {
    cout << "Unable to convert your input to an int.\n";
    exit(1);
  }
  cout << "Doing conversion for input of " << millisec << "
milliseconds ... \n", millisec;

  /* MAKE A CALL TO time_convert HERE. */
  time_convert(millisec, & minutes, & seconds);
  cout << "That is equivalent to " << minutes << " minute(s) and " <<
seconds << " second(s).\n";
  return 0;
}

/* PUT YOUR FUNCTION DEFINITION FOR time_convert HERE. */
void time_convert(int ms_time, int * minutes_ptr, double *
seconds_ptr) {

  // Local variable mins.
  int mins = floor(ms_time / 60000);

  // Modulus ensures that:
  // 0 <= *seconds_ptr & *seconds_ptr < 60.0
  // Local variable secs.
  double secs = (ms_time % 60000) / 1000.0;

  // Ensures that the minutes and seconds pointers have
  // values that add up to the total original ms_time
  // before re-assigning.
  if ((mins * 60000 + secs * 1000) == ms_time) {
    * minutes_ptr = mins; // points to local variable mins
    * seconds_ptr = secs; // points to local variable secs
  }
}
```

Program Output for Exercise E:



```
"/Users/stevenduong/CLionProjects/ENSF 614/ENSF 614 Labs/Lab 1/cmake-build-debug/Lab_1"
Enter a time interval as an integer number of milliseconds: 123400
Doing conversion for input of 123400 milliseconds ...
That is equivalent to 2 minute(s) and 3.4 second(s).

Process finished with exit code 0
```