**Course: ENSF 694** – Summer 2024

**Lab #:** Lab 1

**Instructor:** Mahmood Moussavi

**Student Name:** Ryan Baker

**Submission Date:** June 27, 2024

# Exercise A

## lab1exe\_A.cpp

/\*

 \*  lab1exe\_A.cpp

 \*  ENSF 694 Lab 1, exercise A

 \*  Created by Mahmood Moussavi

 \*  Completed by: Ryan Baker

 \*  Development Date: June 26, 2024

 \*/

#include <iostream>

#include <cmath>

#include <iomanip> // included for table formatting

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);

/\* REQUIRES

 \*   v (double) the velocity to base the table on

 \* PROMISES

 \*   prints a table of angle (0-90 degrees) with corresponding times and distances

 \*/

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

/\* REQUIRES

 \*   0 < a < 90.

 \* PROMISES

 \*  return value is projectile travel time when maximum horizontal distance is reached (in seconds).

 \*/

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

/\* REQUIRES

 \*   0 < a < 90.

 \* PROMISES

 \*  return value is maximum projectile distance (in metres).

 \*/

double degree\_to\_radian(double d);

/\* REQUIRES

 \*  Angle in degrees between 0 and 360.

 \* PROMISES

 \*  returns value is angle in radians.

 \*/

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 << "Invlid input. Bye...\n";

        exit(1);

    }

    while (velocity < 0 )

    {

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

        cin >> velocity;

        if(!cin)

        {

            cout << "Invlid input. Bye...";

            exit(1);

        }

    }

    create\_table(velocity);

    return 0;

}

void create\_table(double v){

    cout << "Angle\t\t\tt\t\t\td\n(deg)\t\t\t(sec)\t\t\t(m)\n";

    cout << std::fixed << std::setprecision(6); // format digits of table

    for(double i = 0; i <= 90; i +=5){

        cout << i << "\t\t" << Projectile\_travel\_time(i, v) << "\t\t" << Projectile\_travel\_distance(i, v) << "\n";

    }

}

double degree\_to\_radian(double d){

    return (d / 180 \* PI);

}

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

    return abs(2 \* v \* sin(degree\_to\_radian(a)) / G);

}

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

    return abs(v \* v \* sin(degree\_to\_radian(2\*a)) / G);

}

## Code output:

QWE+RyanB@ryanb-pc /cygdrive/c/Users/RyanB/OneDrive - Quick Way Electrical (1999) Ltd/ENSF 694/Labs/ensf-694-a01

$ ./lab1exe\_A

Please enter the velocity at which the projectile is launched (m/sec): 100

Angle t d

(deg) (sec) (m)

0.000000 0.000000 0.000000

5.000000 1.778689 177.192018

10.000000 3.543840 349.000146

15.000000 5.282021 510.204082

20.000000 6.980003 655.905724

25.000000 8.624862 781.678003

30.000000 10.204082 883.699392

35.000000 11.705642 958.870021

40.000000 13.118114 1004.905870

45.000000 14.430751 1020.408163

50.000000 15.633560 1004.905870

55.000000 16.717389 958.870021

60.000000 17.673988 883.699391

65.000000 18.496077 781.678003

70.000000 19.177400 655.905724

75.000000 19.712772 510.204081

80.000000 20.098117 349.000146

85.000000 20.330504 177.192018

90.000000 20.408163 0.000000

# Exercise B

## Part I

* No submission required

## Part II

A diagram of a bar code

Description automatically generated

# Exercise C

## lab1exe\_C.cpp

 /\*

 \*  lab1exe\_C.cpp

 \*  ENSF 694 Lab 1 Exercise C

 \*  Completed by: Ryan Baker

 \*  Development Date: June 29, 2024

 \*/

#include <iostream>

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;

  if (!cin) {

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

    exit(1);

  }

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

  /\* 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){

  \*minutes\_ptr = ms\_time/6000;

  \*seconds\_ptr = (double)(ms\_time % 6000) / 1000.0;

}

## lab1exe\_C.cpp output

QWE+RyanB@ryanb-pc /cygdrive/c/Users/RyanB/OneDrive - Quick Way Electrical (1999) Ltd/ENSF 694/Labs/ensf-694-a01

$ ./lab1exe\_C

Enter a time interval as an integer number of milliseconds: 7450

Doing conversion for input of 7450 milliseconds ...

That is equivalent to 1 minute(s) and 1.45 second(s).

# Exercise D

## Part I

**Memory Diagram – Point One**

A graph paper with a diagram

Description automatically generated

**Memory Diagram – Point Two**

**A graph paper with writing on it

Description automatically generated**

**Memory Diagram – Point Three**

**A graph paper with a diagram

Description automatically generated**

**Part II**

**lab1exe\_D.cpp**

/\*/

 \*  lab1exe\_D.cpp

 \*  ENSF 694 Lab 1 Exercise D

 \*  Completed by: Ryan Baker

 \*  Development Date: June 29, 2024

 \*/

#include <iostream>

#include <iomanip>

using namespace std;

const int COL\_SIZE = 3;

const int ROW\_SIZE = 3;

void try\_to\_change(double\* dest);

void try\_to\_copy(double dest[], double source[]);

double add\_them (double a[5]);

void print\_matrix(double matrix[][COL\_SIZE], int rows);

/\*

 \* PROMISES: displays the values in the elements of the 2-D array, matrix,

 \* formated in rows columns separated with one or more spaces.

 \*/

void good\_copy(double \*dest, double \*source, int n);

/\* REQUIRES: dest and source points to two array of double numbers with n to n-1 elements

 \* PROMISES: copies the values in each element of array source to the corresponding element

 \* in array dest.

 \*/

int main(void)

{

    double sum = 0;

    double x[4];

    double y[] = {2.3, 1.2, 2.2, 4.1};

    double matrix[ROW\_SIZE][COL\_SIZE] = { {10, 20, 30}, {40, 50, 60}, {70, 80, 90}};

    cout << " sizeof(double) is " << (int) sizeof(double) << " bytes.\n";

    cout << " size of x in main is: " << (int) sizeof(x) << " bytes.\n";

    cout << " y has " << (int) (sizeof(y)/ sizeof(double)) << " elements and its size is: " <<  (int) sizeof(y) << " bytes.\n";

    cout << " matrix has " << (int) (sizeof(matrix)/ sizeof(double)) << " elements and its size is: " <<  (int) sizeof(matrix) << " bytes.\n";

    try\_to\_copy(x, y);

    try\_to\_change(x);

    sum = add\_them(&y[1]);

    cout << "\n sum of values in y[1], y[2] and y[3] is: " << sum << endl;

    good\_copy(x, y, 4);

    cout << "\nThe values in array x after call to good\_copy are expected to be:";

    cout << "\n2.30, -8.25, 2.20, 4.10\n";

    cout << "And the values are:\n";

    for(int i = 0; i < 4; i++)

        cout << fixed << setprecision(2) << x[i] << "  ";

    cout << "\nThe values in matrix are:\n";

    print\_matrix(matrix, 3);

    cout << "\nProgram Ends...\n";

    return 0;

}

void try\_to\_copy(double dest[], double source[])

{

    dest = source;

    /\* point one\*/

    return;

}

void try\_to\_change(double\* dest)

{

    dest [3] = 49.0;

    /\* point two\*/

    cout << "\n sizeof(dest) in try\_to\_change is "<< (int)sizeof(dest) << " bytes.\n";

    return;

}

double add\_them (double arg[5])

{

    \*arg = -8.25;

    /\* point three \*/

    cout << "\n sizeof(arg) in add\_them is " << (int) sizeof(arg) << " bytes.\n";

    cout << "\n Incorrect array size computation: add\_them says arg has " << (int) (sizeof(arg)/sizeof(double)) <<" element.\n";

    return arg[0] + arg[1] + arg[2];

}

void good\_copy(double \*dest, double \*source, int n)

{

    for(int i = 0; i < n; i++){

        dest[i] = source[i];

    }

}

void print\_matrix(double matrix[][COL\_SIZE], int rows)

{

    for(int i = 0; i < rows; i++){

        for(int j = 0; j < COL\_SIZE; j++){

            cout << matrix[i][j] << " ";

        }

        cout << endl;

    }

}

**Output**

QWE+RyanB@ryanb-pc /cygdrive/c/Users/RyanB/OneDrive - Quick Way Electrical (1999) Ltd/ENSF 694/Labs/ensf-694-a01

$ g++ -w lab1exe\_D.cpp -o lab1exe\_D

QWE+RyanB@ryanb-pc /cygdrive/c/Users/RyanB/OneDrive - Quick Way Electrical (1999) Ltd/ENSF 694/Labs/ensf-694-a01

$ ./lab1exe\_D

sizeof(double) is 8 bytes.

size of x in main is: 32 bytes.

y has 4 elements and its size is: 32 bytes.

matrix has 9 elements and its size is: 72 bytes.

sizeof(dest) in try\_to\_change is 8 bytes.

sizeof(arg) in add\_them is 8 bytes.

Incorrect array size computation: add\_them says arg has 1 element.

sum of values in y[1], y[2] and y[3] is: -1.95

The values in array x after call to good\_copy are expected to be:

2.30, -8.25, 2.20, 4.10

And the values are:

2.30 -8.25 2.20 4.10

The values in matrix are:

10.00 20.00 30.00

40.00 50.00 60.00

70.00 80.00 90.00

Program Ends...

# Exercise E

A graph paper with writing and a diagram

Description automatically generated

# Exercise F

## MyArray.cpp

/\*

 \*  MyArray.cpp

 \*  ENSF 694 Lab 1 Exercise F

 \*  Completed by: Ryan Baker

 \*  Development Date: July 2, 2024

 \*/

#include "MyArray.h"

int search(const MyArray\* myArray, int obj){

// Students are supposed to complete the implementation of the this function

    for(int i = 0; i < myArray->list\_size; i++){

        if(myArray->array[i] == obj){

            return i;

        }

    }

    return -1;

}

void initialize(MyArray\* myArray) {

    // Students are supposed to complete the implementation of the this function

    myArray->list\_size = 0;

}

int retrieve\_at(MyArray\* myArray, int pos){

    // Students are supposed to complete the implementation of the this function

    return myArray->array[pos];

}

int count(MyArray\* myArray, int obj ){

    // Students are supposed to complete the implementation of the this function

    int count = 0;

    for(int i = 0; i < myArray->list\_size; i++){

        if(myArray->array[i] == obj) count++;

    }

    return count;

}

void append( MyArray\* myArray, int array[], int n ) {

    // Students are supposed to complete the implementation of the this function

    if((myArray->list\_size + n) <= (int)(sizeof(myArray->array)/sizeof(int))){

        for(int i = 0; i < n; i++){

            myArray->array[myArray->list\_size] = array[i];

            myArray->list\_size++;

        }

    }

}

void insert\_at(MyArray\* myArray, int pos, int val) {

    // Students are supposed to complete the implementation of the this function

    myArray->list\_size++;

    for(int i = myArray->list\_size - 1; i > pos; i--){

        myArray->array[i] = myArray->array[i-1];

    }

    myArray->array[pos] = val;

}

int remove\_at(MyArray\* myArray, int pos ) {

    // Students are supposed to complete the implementation of the this function

    int val = myArray->array[pos];

    for(int i = pos; i < myArray->list\_size; i++){

        myArray->array[i] = myArray->array[i+1];

    }

    myArray->list\_size--;

    return val;

}

int remove\_all(MyArray\* myArray, int value ) {

    // Students are supposed to complete the implementation of the this function

    int count = 0;

    for(int i = 0; i < myArray->list\_size; i++){

        if(myArray->array[i] == value){

            remove\_at(myArray, i);

            count++;

        }

    }

    return count;

}

// You can modify this function however you want:  it will not be tested

void display\_all(MyArray\* myArray) {

    // Students are supposed to complete the implementation of the this function

    for(int i = 0; i < myArray->list\_size; i++){

        cout << myArray->array[i] << " ";

    }

    cout << endl;

}

bool is\_full(MyArray\* myArray){

    // Students are supposed to complete the implementation of the this function

    return ((sizeof(myArray->array) / sizeof(int)) == myArray->list\_size);

}

bool isEmpty(MyArray\* myArray){

    // Students are supposed to complete the implementation of the this function

    return myArray->list\_size == 0;

}

int size(MyArray\* myArray){

    // Students are supposed to complete the implementation of the this function

    return myArray->list\_size;

}

## output.txt

Starting Test Run. Using input file.

Line 1 >> Passed

Line 2 >> Passed

Line 3 >> Passed

Line 4 >> Passed

Line 5 >> Passed

Line 6 >> Passed

Line 7 >> Passed

Line 8 >> Passed

Line 9 >> Passed

Line 10 >> Passed

Line 11 >> Passed

Line 12 >> Passed

Line 13 >> Passed

Line 14 >> Passed

Line 15 >> Passed

Line 16 >> Passed

Line 17 >> Passed

Line 18 >> Passed

Line 19 >> Passed

Exiting...

Finishing Test Run

Showing Data in the List:

101 200 100 500

Program Ended ....