CSCI 200 - Fall 2023
Foundational Programming Concepts & Design

Lab 2C - Coordinate Conversion By Pointer

This lab is due by Tuesday, September 26, 2023, 11:59 PM.
As with all labs you may, and are encouraged, to pair program a solution to this lab. If you choose to pair program a solution, be sure that you individually understand how to generate the correct solution.

Jump To: **Rubric Submission**

Pass-by-Value & Pass-by-Pointer

One of the limitations of functions is that they can only return a single value. A workaround to this limitation is to pass parameters by pointer. When the function completes, the arguments corresponding to these parameters will contain the modified values. A generic function prototype would match the following template:

```
void function_name( const dataType IN_PARAM_1, const dataType IN_PARAM_2, // input to the
                    dataType* pOutParam1, dataType* pOutParam2 );         // output from t
```

This usage of constant pass-by-value input and pass-by-pointer output reinforces the procedural programming style.

```
int value1, value2, value3, value4;              // no variables are initialized
cin >> value1 >> value2;                          // user enters value1 and value2
```

```
function_name(value1, value2, &value3, &value4); // function call populates value3 and val
cout << value1 << " " << value2 << endl;         // all variables now have a value
```

Instructions

We will create two functions called `polar_to_cartesian` and `cartesian_to_polar` that match the above format. Begin by reviewing the **Polar to Cartesian Conversion** equations.

Your program should first prompt the user which direction they wish to convert, either

`(r, θ) -> (x, y)`

or

`(x, y) -> (r, θ)`

Prompt the user to input the values on the left hand side and then call the corresponding function to compute the values on the right hand side. Display these resultant values to the user.

Your functions must match the following specifications:

1.
   - **Function Name**: `polar_to_cartesian`
   - **Input**:
     1. `double` passed by constant value corresponding to the `radius`
     2. `double` passed by constant value corresponding to the `angle`
     3. `double` passed by pointer corresponding to the `xCoordinate`
     4. `double` passed by pointer corresponding to the `yCoordinate`
   - **Output**: None
   - **Description**: Converts polar `(r, θ)` to cartesian `(x, y)`.

2.
   - **Function Name**: `cartesian_to_polar`
   - **Input**:
     1. `double` passed by constant value corresponding to the `xCoordinate`
     2. `double` passed by constant value corresponding to the `yCoordinate`
     3. `double` passed by pointer corresponding to the `radius`
     4. `double` passed by pointer corresponding to the `angle`
   - **Output**: None
   - **Description**: Converts cartesian `(x, y)` to polar `(r, θ)`.

Refactor To Multiple Files

Once your solution is working, refactor your code to use multiple files. Your project should consist of the following files with respective contents:

- `coordinate_conversion.h` : Contains the function declarations
- `coordinate_conversion.cpp` : Contains the function definitions
- `main.cpp` : Contains all the User I/O and function calls

Be sure to update your `Makefile` appropriately to work with the additional files.

Grading Rubric

Your submission will be graded according to the following rubric:

Points	Requirement Description
0.70	Fully meets specifications
0.15	Submitted correctly by Tuesday, September 26, 2023, 11:59 PM
0.15	**Best Practices** and **Style Guide** followed
1.00	**Total Points**

Lab Submission

Always, **always**, **ALWAYS** update the header comments at the top of your `main.cpp` file. And if you ever get stuck, remember that there is LOTS of **help** available.

Zip together your `coordinate_conversion.h`, `coordinate_conversion.cpp`, `main.cpp`, `Makefile` files and name the zip file `L2C.zip` . Upload this zip file to Canvas under L2C.

This lab is due by Tuesday, September 26, 2023, 11:59 PM.
As with all labs you may, and are encouraged, to pair program a solution to this lab. If you choose to pair program a solution, be sure that you individually understand how to generate the correct solution.

Last Updated: 08/31/23 11:09

Any questions, comments, corrections, or request for use please contact jpaone {at} mines {dot} edu.

[Jump to Top] [Site Map]