CSCI 200 - Fall 2023
Foundational Programming Concepts & Design

A2 - Complex Vector Math!

→This assignment is due by Tuesday, September 26, 2023, 11:59 PM.←

→ As with all assignments, this must be an individual effort and cannot be pair programmed. Any debugging assistance must follow the course collaboration policy and be cited in the comment header block for the assignment.←

→ Do not forget to complete the following labs with this set: L2A, L2B, L2C ←

→ Do not forget to complete zyBooks Assignment 2 for this set.←

Jump To: **Rubric Submission**

Concepts

When considering points in space, we can use negative numbers to correspond moving backwards along a heading. We can also use imaginary numbers to correspond to a rotation around the origin. The following article gives an excellent explanation and visualization of this process: **Better Explained: A Visual, Intuitive Guide to Imaginary Numbers**. The section **A Real Example: Rotations** explains the math of multiplying complex numbers to perform rotations.

We will use complex numbers to correspond to a point in space and a rotation to apply. Our program will then multiply these values together to determine the resultant point after rotation. Our rotation calculator can then continually spin a point around the origin

---

RMP-H

`main()` Pseudocode

Your program will prompt the user with a menu of choices. They will continue to enter choices until they opt to quit. Below are the menu options and what happens for each choice:

- Enter a point (x,y) coordinate
  - Prompts the user to enter an (x, y) coordinate and stores these values
- Rotate the current point by an angle
  - Prompts the user to enter an angle
  - Converts the angle into its corresponding vector
  - Normalizes the vector
  - Rotates the point by the vector
  - Updates the stored point and prints the new point location
- Rotate the current point by a vector
  - Prompts the user to enter a vector <x, y> coordinate
  - Normalizes the vector
  - Rotates the point by the vector
  - Updates the stored point and prints the new point location
- Quit
  - Gracefully ends the program

Be sure to handle the scenario if the user enters a selection that doesn't align to one of the above choices.

The Functions

We will use functions to implement each step of the above pseudocode. The specifications and descriptions for each function are given below. Your task is to create the appropriate functions and call them in the correct order to implement the expected pseudocode.

- `angle_input()`
  - **Input**:
    1. A `float` pointer corresponding to the angle
  - **Output**: None
  - **Description**: Prompts the user to enter an angle and updates the corresponding parameter. Be sure to specify to the user what units they are working with.
- `angle_to_vector()`
  - **Input**:
    1. A `float` corresponding to the angle from the positive X-axis

        2. A `float` pointer corresponding to the resultant vector's x coordinate
        3. A `float` pointer corresponding to the resultant vector's y coordinate
    - **Output**: None
    - **Description**: Converts an angle with the positive X-axis to the resultant vector on the unit circle
- `point_input()`
    - **Input**:
        1. A `float` pointer corresponding to the point's x coordinate
        2. A `float` pointer corresponding to the point's y coordinate
    - **Output**: None
    - **Description**: Prompts the user to enter the (x, y) coordinate and updates the corresponding parameters.
- `vector_input()`
    - **Input**:
        1. A `float` pointer corresponding to the vector's x coordinate
        2. A `float` pointer corresponding to the vector's y coordinate
    - **Output**: None
    - **Description**: Prompts the user to enter the <x, y> coordinate and updates the corresponding parameters.
- `vector_normalize()`
    - **Input**:
        1. A `float` corresponding to the input vector's x coordinate
        2. A `float` corresponding to the input vector's y coordinate
        3. A `float` pointer corresponding to the output normalized vector's x coordinate
        4. A `float` pointer corresponding to the output normalized vector's y coordinate
    - **Output**: None
    - **Description**: Normalizes the input vector setting the corresponding pointer parameters to the output normalized vector
- `vector_to_angle()`
    - **Input**:
        1. A `float` corresponding to the vector's x coordinate
        2. A `float` corresponding to the vector's y coordinate
    - **Output**: A `float` corresponding to the angle the vector forms with the positive X-axis
    - **Description**: Determines the angle the vector forms with the positive X-axis based on where it intersects the unit circle. Angle is computed by taking the arctan of y over x. Be sure to handle edge cases correctly.
- `rotate_point_by_vector()`
    - **Input**:
        1. A `float` corresponding to the input point's x coordinate

2. A `float` corresponding to the input point's y coordinate
3. A `float` corresponding to the input vector's x coordinate
4. A `float` corresponding to the input vector's y coordinate
5. A `float` pointer corresponding to the output rotated point's x coordinate
6. A `float` pointer corresponding to the output rotated point's y coordinate

- **Output**: None
- **Description**: Performs the rotation by multiplying complex numbers. Form a complex number using the point (p_x, p_y) coordinate: $p_x + p_y i$. Form a second complex number using the vector $<v_x, v_y>$ coordinate: $v_x + v_y i$. Multiply these two complex numbers to compute the resultant point:

  $$(p_x + p_y i) * (v_x + v_y i)$$

  which expands, via FOIL, to

  $$(p_x v_x - p_y v_y) + (p_x v_y + p_y v_x) i$$

  and can then be converted back to a point

  $$(p_x v_x - p_y v_y, p_x v_y + p_y v_x)$$

Public Tests

Below is a sample interaction that demonstrates the correct calculations and program flow.

```
Menu:
        1) Enter point (x, y) coordinate
        2) Rotate point by angle
        3) Rotate point by vector
        0) Quit
Choice: 1
Enter point x coordinate: 1
Enter point y coordinate: 2
Menu:
        1) Enter point (x, y) coordinate
        2) Rotate point by angle
        3) Rotate point by vector
        0) Quit
Choice: 2
Enter angle in degrees: 45
The point (1, 2) rotated by 45 degrees is now at (-0.707107, 2.12132)
Menu:
        1) Enter point (x, y) coordinate
```

```
        2) Rotate point by angle
        3) Rotate point by vector
        0) Quit
Choice: 3
Enter vector x coordinate: 1
Enter vector y coordinate: 1
The point (-0.707107, 2.12132) rotated by the vector <1, 1> (45 degrees) is now at (-2, 1)
Menu:
        1) Enter point (x, y) coordinate
        2) Rotate point by angle
        3) Rotate point by vector
        0) Quit
Choice: 2
Enter angle in degrees: -90
The point (-2, 1) rotated by -90 degrees is now at (1, 2)
Menu:
        1) Enter point (x, y) coordinate
        2) Rotate point by angle
        3) Rotate point by vector
        0) Quit
Choice: 3
Enter vector x coordinate: -1
Enter vector y coordinate: 0
The point (1, 2) rotated by the vector <-1, 0> (180 degrees) is now at (-1, -2)
Menu:
        1) Enter point (x, y) coordinate
        2) Rotate point by angle
        3) Rotate point by vector
        0) Quit
Choice: 5
Invalid selection, enter a valid option
Menu:
        1) Enter point (x, y) coordinate
        2) Rotate point by angle
        3) Rotate point by vector
        0) Quit
Choice: 0
```

Be sure to test additional values to ensure the program is correct!

Specification Requirements

Your code and program must meet the following specifications:

- The function prototypes must be placed in a separate header file. The function definitions must be placed in a separate implementation file.

- Function parameters need to be passed by value or by pointer as appropriate. While pass-by-reference can be used to perform equivalent operations, this assignment is assessing the proper usage of pass-by-value and pass-by-pointer.
- User input must be validated to prevent future errors (such as a menu item that does not exist). You can assume the user will always enter the correct data type.
- Feel free to make additional helper functions as appropriate to handle common tasks.

Extra Credit

The prior calculations always perform a rotation centered around the origin. Add another menu option that allows the user to rotate around an arbitrary point. An example is shown below.

```
Menu:
        1) Enter point (x, y) coordinate
        2) Rotate point by angle
        3) Rotate point by vector
        4) Rotate point by vector around point
        0) Quit
Choice: 1
Enter point x coordinate: -1
Enter point y coordinate: -1
Menu:
        1) Enter point (x, y) coordinate
        2) Rotate point by angle
        3) Rotate point by vector
        4) Rotate point by vector around point
        0) Quit
Choice: 3
Enter vector x coordinate: -1
Enter vector y coordinate: 0
The point (-1, -1) rotated by the vector <-1, 0> (180 degrees) is now at (1, 1)
Menu:
        1) Enter point (x, y) coordinate
        2) Rotate point by angle
        3) Rotate point by vector
        4) Rotate point by vector around point
        0) Quit
Choice: 1
Enter point x coordinate: -1
Enter point y coordinate: -1
Menu:
        1) Enter point (x, y) coordinate
        2) Rotate point by angle
        3) Rotate point by vector
        4) Rotate point by vector around point
        0) Quit
Choice: 4
Enter point x coordinate: 1
```

```
Enter point y coordinate: 1
Enter vector x coordinate: -1
Enter vector y coordinate: 0
The point (-1, -1) rotated by the vector <-1, 0> (180 degrees) around the point (1, 1) is
```

Grading Rubric

Your submission will be graded according to the following rubric:

Points	Requirement Description
0.5	Submitted correctly by Tuesday, September 26, 2023, 11:59 PM
0.5	Project builds without errors nor warnings.
2.0	**Best Practices** and **Style Guide** followed.
0.5	Program follows specified user I/O flow.
0.5	Public and private tests successfully passed.
2.0	Fully meets specifications.
6.00	**Total Points**

Extra Credit Points	Requirement Description
+1.0	Extra credit: Add another menu option to rotate around a point other than the origin.

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 `main.cpp, vectorMath.h, vectorMath.cpp, Makefile` files and name the zip file `A2.zip`. Upload this zip file to Canvas under A2.

→This assignment is due by Tuesday, September 26, 2023, 11:59 PM.←

→ As with all assignments, this must be an individual effort and cannot be pair programmed. Any debugging assistance must follow the course collaboration policy and be cited in the comment header block for the assignment.←

→ Do not forget to complete the following labs with this set: L2A, L2B, L2C ←

→ Do not forget to complete zyBooks Assignment 2 for this set.←

Last Updated: 09/15/23 09:48

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

[Jump to Top] [Site Map]