Tutorial 4.1
Class and Object Manipulation

Overview

- You will be doing this tutorial in a collaborative coding session in groups.

Breakout Sessions

- I will split the main meeting room on Webex into several rooms. Each room will have 4 members.
- You will be assigned to a room. See the attached file to find which room you belong to.

Collaborative Coding

- In each room, appoint one member to be the host. The host member will initiate a session for collaborative coding using Live Share.
- Other members will be invited as collaborators.
- All members (host and collaborators) will need to open VS Code.
- The host member will also need to share his/her screen via Webex.
- Each group is required to present the code at the end of this session.

Problem

Write a C++ program that declares a class named **Point** to model a point. A point is represented by its coordinates *x* and *y*. Separate the class definition from declaration within the same file. Do the following tasks to accomplish the program.

1. Define several constructors for the class as follows. Use constructor initializer and default arguments whenever possible.
    a. A constructor that accepts two parameters to initialize the coordinates x and y respectively.
    b. A copy constructor that accepts another **Point** object. Use a constructor initializer to invoke constructor (a)
    c. A default constructor that sets the coordinates x and y to 0. Use a constructor initializer to invoke any of the above constructor s.

2. Define a **constant** method named **print()** that prints the coordinates x and y of the point. Explain what is a constant method and what is used for? Why is this method better to be declared as a constant method? Write your answers as comments in the program.

3. Define a method named **input()** that asks the user for the coordinates x and y from the keyboard. Can you declare this method as a constant method? Justify your answer. Write your answer as comments in the code.

4. Define an operation to add two points using three different approaches (a) to (c) below. Addition of two points is done as follows. Given two points, for example, ***p1*: (1, 2)** and ***p2*: (3, 3)**, thus adding ***p1*** and ***p2*** results in a new point, **(3, 5).**
    a. with a method of the class, named **add()**
    b. with an overloaded **operator +** of the class
    c. with a friend function named **addPoints()**

5. Define an operation that performs a division of point to a number using three different approaches (a) to (c) below. The divisopm operation is done as follows. Given a point, for example, ***p*: (1, 2)**, thus ***p* / 2.0** results in a new point, **(0.5, 1.0).**
    a. with a method of the class, named **divide()**
    b. with an overloaded **operator /** of the class
    c. with a friend function named **dividePoint()**

6. In the main function,
    a. Declare an array to hold a list of points
    b. Read a list points from user inputs and store them into the array. Use an appropriate method from the class Point to accomplish this task.
    c. Print the list of points onto the screen. Use an appropriate method from the class Point to accomplish this task.
    d. Calculate the middle point from the list, using three different approaches.
        i. with the methods **add()** and **divide()** from the class Point.
        ii. with the operator + and / from the class.
        iii. with the friend functions.

The middle point is calculated by the average of the coordinates *x* and *y* of the points, respectively. For example, if the points are ***p1:*** **(1,2)**, ***p2:*** **(2,4)** and ***p3:*** **(3,3)**, then the middle point is calculated as:

$$(p1 + p2 + p3) / 3 =$$

$$\left(\frac{1+2+3}{3}, \frac{2+4+3}{3}\right) = (2, 3)$$

Figure 1 shows an example run of the program

```
How many points you want to enter => 4

Enter the coordinates (x and y) => 1  6
Enter the coordinates (x and y) => 2  7
Enter the coordinates (x and y) => 3  8
Enter the coordinates (x and y) => 4  9

Printing all points
(1,6)
(2,7)
(3,8)
(4,9)

Printing the middle point with different approaches
(2.5,7.5)
(2.5,7.5)
(2.5,7.5)
```

Figure 1: An example run of the program with user inputs (indicated by the bold texts) and the screen output.