ANKARA UNIVERSITY
COM102B
Fall 2017-18 (Spring)
Programming Assignment 4

Submission Deadline: 20/05/2018, 23:55

In this assignment, you will implement a simple graphics system using an ASCII implementation, where a point is a character position and you will write to the graphics system by placing a * character in a position.

DisplayWindow(n,m) dynamically creates an area of size ***n*** by ***m*** on the screen. **Points** on the screen are addressed using **(x,y)** Cartesian Coordinates.

A ***DisplayWindow w*** has a current position ***w.current()***. Initially, current is ***Point(0,0)***. The current position can be set by ***w.current(p)*** where ***p*** is a ***Point***.

A point is specified by a coordinate pair ***Point(x,y)***.

A Line is specified by a pair of points: ***Line(w.current(), p2);***

class **Shape** is the common interface to *Dots, Lines and Rectangles*, and will be defined as an abstract class with the following pure virtual function:

virtual void draw(DisplayWindow& w) = 0;

A **Point** is not a **Shape**. A Dot, ***Dot(p)***, can be used to represent a **Point p** on the screen.

A **Shape** is invisible unless it is **draw()** n. For example: ***w.draw(Dot(w.current()));***

Every ***Shape*** has 9 contact points***: e(east), w(west), n(north), s(south), ne, nw, se, sw and c (center)***. For example: ***Line(x.c(), y.nw())***; creates a line from x's center to y's top left corner. Note that **south** and **north** directions are defined according to the display window; hence, north of a shape is on the upper part, while south is on the lower part on the display window (Figure 1, Figure 2).

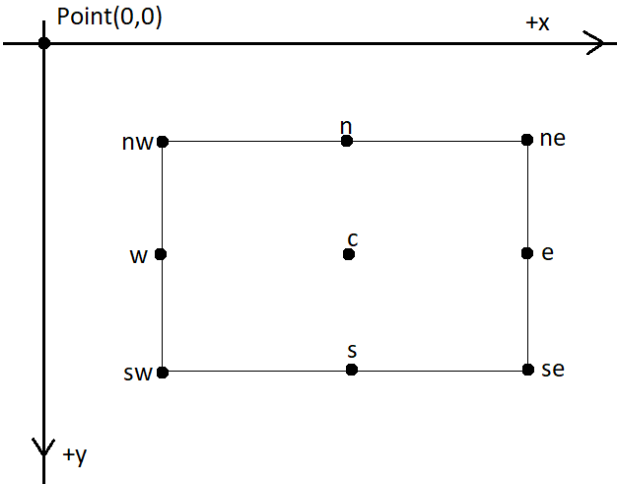

Figure 1: Display Coordinates and contact points of a Rectangle shape

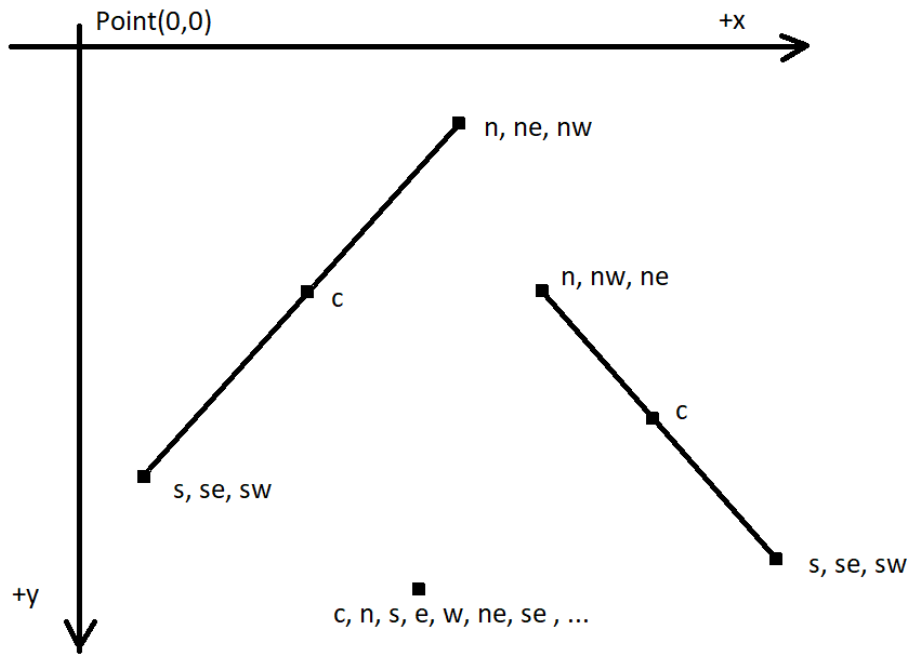

Figure 2: Display Coordinates and contact points of two sample lines and a Dot shape. All the contact points of a Dot shape are the same as the center point.

After **draw()** ing a **Shape,** the current position is the Shape's **se()**, except for the Line shapes; the current position is set to the **second end point of a line**. For example: ***w.draw(Line(Point(10,10), Point(20,20)));*** Shape's current position becomes ***Point(20,20)***.

A Rectangle is specified by its bottom left and top right corner: ***Rectangle(w.current(), Point(10,10));***

After drawing is complete, you can **show()** the content of your **DisplayWindow w** using ***w.show();***

Display buffer's **origin**, i.e. **Point(0,0)**, is depicted on the top left corner of the window. So, the Cartesian x coordinate is increasing in the right direction, while y-coordinate is increasing in the downward direction. Note that, in this display convention, bottom left corner of a rectangle is displayed as the top-left corner on the screen, although in the Cartesian coordinates it's still bottom left corner.

A cartesian coordinate, **Point(x,y)** can be modified by an amount ***n*** using: ***right(n), left(n), up(n), down(n)***. For **up()** and **down()** functions, ***up()*** sets the point to the upper side of **the screen** (by reducing the y-coordinate) and ***down()*** in the opposite manner. when no argument is passed to these functions, n is 1, i.e. related coordinate is modified only by 1. For example: ***p.right();*** increments the x coordinate **by 1**. As a result, these functions return the **Point** with the new coordinates.

Task:

According to the specifications detailed above, please create **PA4.h** and **PA4.cpp** files and write all the necessary classes into these files. Basically, you will implement these classes: DisplayWindow, Point, Shape, Dot, Rectangle, Line. You are free to extend your class definitions anyway you like; you can include additional functions that will help you to operate well with these classes. Make sure that

your classes are working correctly with the sample Main.cpp file, hence provide the necessary interfaces for the specifications discussed above. For Point class, overloading some operators may help you a lot for computing the Cartesian coordinates for shape objects.

You will need to make **forward declarations**, hence, it's wise to separate your class declaration and member-function definitions in .**h** and .**cpp** files; forward declarations do not work otherwise.

Suggestion for the Development Environment:

Use ***Visual Studio Express Edition*** (freely available for students), to develop and test your codes. Visual Studio provides a Visual Debugger, which may be very helpful when you want to investigate what is going on in your computations. After completing your classes, you can test your codes on Ubuntu before submission.

Testing:

We provide 4 sample output text files for you to test your codes at Ubuntu. These text files look in a different format in Windows. If you want to see and use them in Windows operating system, you need to change the line endings to Windows. You can do this easily on Windows by using Notepad++ program, using *Edit/EOF Conversion Menu (Click the Windows submenu there)*.

Compile your codes using:

>g++ PA4.cpp Main.cpp -o PA4

We recommend you to use *input redirection* mechanism of your operating system to test your programs. For example, if your executable is called as **PA4** (generated after compiling comment given above), redirect your outputs to a file using **>** operator such as:

> ./PA4 >output.txt

You are provided with a sample Main function. The basic functionality of your DisplayWindow and Shape classes are tested with 3 functions, for all these functions expected outputs are provided as ***output_dots.txt, outputs_lines.txt*** and ***output_rectangles.txt***. After you tested your basic Shape functions, uncomment the call **drawHome()** function to test all functions simultaneously; the expected output of this function is provided to you in ***output_home.txt*** file. Note that **DisplayWindow()** size has to be allocated ***dynamically***, since it can change to different amounts in different scenarios.

Submission: Submit PA4.h and PA4.cpp files, archived as <student_id>.rar file. Ex: if your student id is 112600, you will send a rar file named as: 112600.rar

Warning: Any form of code copying, including the copies from the internet, is strictly prohibited. If we determine similarities between your codes with any other students in the class, it will be treated as cheating and will be punished. So, you would increase the risk of cheating when you see somebody else's code directly or see a solution in the internet (i.e. somebody else might have also copied the same code from the internet like you, so both of these codes will be evaluated as copies,

since they both copy from an external source. Such attempts will always be considered as cheating). You are supposed to write the solution by yourselves.

Please test your programs with the given output files before submission.

Ask any questions related with the homework specs. to the course news forum in Moodle.
Follow further announcements about this homework from Moodle.

have fun ☺