Goals of this assignment

In this assignment you will practice the use of overloading, multidimensional arrays as well as the principle of incremental design. You will be creating, in stages, a simple, ASCII version of the Seven Seas game which was shown in class and can be found at popcap.com.

You may use Pair Programming if you wish. Instructions are on the website (under Assignments).

Game description

Seven Seas presents the player with an $M \times N$ grid, representing the sea. The grid is populated by the player’s ship, several pirate ships, a number of islands, and possibly shipwrecks that are the results of ship collisions.

The player’s goal is to destroy the pirate ships by causing them to collide with islands, existing shipwrecks, or one another. The game ends when all the pirate ships have sank, or when the player has been destroyed. The player’s ship may sink due to a collision with another ship, an island or a wreck. All ships involved in a collision are destroyed and a shipwreck appears at that location.

The player can move in any direction, one “cell” at a time. Pirate ships automatically move one step closer to the player. Islands and wrecks are stationary.

The SevenSeas class

This is the main class of your project and represents the game universe. Ultimately, it should:

- Create a 10 $\times$ 10 board
- Place the player, pirate ships and islands at random locations on the grid
- Display the board
- Control the game

You do not need separate player or ship objects. All you need to keep track of is their locations on the grid.
The Coordinates class

Create a coordinates class that will represent the position $(x,y)$ of an element on the grid. You will need to provide appropriate accessors and mutators as well as an overloaded operator $==$ that can be used to check whether two locations are identical (in other words, whether there is a collision).

The main function

The main function should be very small. All it needs to do is print the instructions and then start the game. Make sure you write well-defined functions for any actions here. For example, write a `print_instructions` function instead of filling `main()` with `cout`s.

Designing your project

Start simple!

First have the class create an empty grid, and print it. By default, you should create a $10 \times 10$ grid, but you should write your class so that the size may be easily changed.

Then, add the player at a random position. Add the pirates and islands incrementally. Make sure you don’t try to place any new element on top of something else. The only allowed “values” for a grid element are WATER, PIRATE, ISLAND, WRECK and PLAYER. Have a random number of islands appear every time the game is played (make sure you place some limit though) and at least three pirate ships.

Run the game a few times, to test the placement of all the elements on the grid.

Now you can write the `play()` function that will control the game. Its operation is simple. It should repeatedly move the player (by asking the user for the coordinates of the new position) and then the ships. Write separate, helper functions for the movements. Make sure you update the board and ship positions correctly. After every movement, your program should check whether the player has lost or won.

You will need several member functions for SevenSeas. Feel free to add extra features if you want. For example, you can have the user select a level of difficulty, or have the game print random comments after every move or whenever a ship sinks.

Run the provided executable to see how your game is expected to behave.

Makefile

You should include a makefile in your project.

What to submit

Create a tar archive with all your source files. Do NOT include any object or executable files. Email your archive to b11@cs.northwestern.edu.