Project #11 (50 points): Battle Boats

Submit Your Solution Using Angel by 10 PM, Tuesday April 23,2013

(A late submission drop box will be available until 11:59PM on 4/23/2013)

Obtaining Project 11 Input Order and Output:

Run the provided sample solution to see what information needs to be output by the program. You may **RUN** the sample solution **Project_11_solution** by typing the following at a command prompt in a terminal window that has a current working directory of **~/CPE112_SPR13/Project_11 (where ~ represents your home directory)** and that the sample input files are present in that directory. **(Note, the input files must be in the directory from which the command below is run)**

/home/work/cpe112/Executables/Project_11/Project_11_solution

Also, run the comparison script before submitting your program to verify your output versus the sample solution output

/home/work/cpe112data/Project_11/CompareSolution.bash Project_11.cpp

Input is to follow the same order as that illustrated by the provided solution executable. Output is to be the same as that shown by the provided solution.

Project 11 Restrictions

You cannot use structures. All other C++ techniques covered in Chapters 1 through 11 are allowed. ***You are not allowed to use any global variables.*** If necessary, global constants may be used.

Project 11 Directions:

- Open a terminal window and move (cd) into the Project_11 directory created in the CPE112_SPR13 directory (This is the directory structure created in project 1.) The command for this is: **cd CPE112_SPR13/Project_11 (typing cd ~/CPE112_SPR13/Project_11 works as well)**. You will need to modify the names as necessary to match your capitalization style for the two directories.
- Download all needed files from ANGEL into this directory. Using a Text Editor (i.e. gedit), edit a file for project 11 (i.e. g**edit Project_11.cpp**). Write your complete program so that it **produces the desired output as shown by the provided solution executable.**
- Test the operation of your program using the sample data provided and compare results from your program to those of the sample solution.
- Once you are satisified with your solution, submit Project_11.cpp to the dropbox for project 11.
- Header files required: **iomanip, string, fstream and iostream**.

Project 11 Description

Input is to follow the same order as that illustrated by the provided solution executable. Output is to be similar to that shown by the provided solution.

.

For this project, your program is to read an input file that contains information on the placement of 4 boats into a 10x10 character array. Each boat consists of three locations that must be hit (three elements of an array), and each boat is placed vertically or horizontally only. After placing the boats on the game board, the user is prompted for a row and column number to see if it is a hit on one of the 4 boats or a miss. Play continues until the user sinks all 4 boats. After sinking the last boat, the percentage of shots that were hits is displayed followed by the final game board. **Run the sample solution using the first input file to see how the program runs**.

For the game board, use a 10x10 character array. This array is to be initialized to all periods ('.'). After initializing the array, open an input file and use the contents of the file to place the boats into the array. The input file contains 2 integer values and then a character on each line. The first integer is the row index value, the second integer is the column index value and the character is the boat number (1,2,3 or 4). Note: the third value on the line is an integer representing the boat number; however, it should be read into a character variable. The boat number read is then placed into the array at the row and column index specified.

After the entire input file has been read and all the boats placed, the user is prompted for a guess on which row and column to fire at. If a boat is present at the spot selected, then the phrase ".....Hit!!!" is printed out followed by the board with an H present in the spot hit. If a boat is not present at the spot selected, then the phrase ".....Miss!!!" is printed out followed by the board with an X present in the spot selected.

After all three parts of a boat have been hit, the message to the screen indicates the third hit and the phrase "You sank boat #X" where X will be 1, 2, 3 or 4. After all 4 boats have been sunk; the congratulation message and statistic message on the percentage of shots that were hits are printed followed by the game board.

Project 11 Requirements

- Output of the program shall match that of the sample solution. Running the sample solution will help with understanding the requirements for the program and providing information that may be missing from this description.

- This project is to introduce the concept of arrays and how to use arrays with functions. Therefore, **your solution is to have at least 3 valid functions (void or value returning) other than main**. **Some or all of the functions you write shall contain arrays as parameters.**

- The program is to play one game

- **Your program must use a function to initialize the game board to all periods**. After initializing the game board to all periods, a function to place the boats is called. This function reads the placement information from the input file and places the boat numbers as characters into the array.

- **The code for printing out the game board is provided in the file printBoard.cpp.** In this code, the variable board is a 2 dimensional array of characters, and the boats have been designated with characters 1, 2, 3 and 4 (note they look like integers, but the have to be saved in the array as characters). The statement board[0][0] = 1; stores the character with integer value 1 in the board array. The statement board[0][0] = '1'; stores the character 1 (integer value of 49) in the array

- All values read from the keyboard (using standard input stream cin) are to be echo printed (run the sample solution with redirected input and input file in1.txt – see the README.txt file included with the zipped input files). There are two content problem messages for the input file: 1) The input file is empty – failure to read the header line in the input file and 2) File open error when trying to open the file

- After reading the user's choice for the row and column index value, the index values are tested. There are three possible error message outcomes for the index values: 1) Invalid Row, 2) invalid column, 3) invalid row and column. Run the sample solution with these three possibilities to see the messages. If a row and/or column index is invalid, the error message is displayed and the board is reprinted followed by the prompt for a row index value.

- For a valid row and column index, the board location is checked to see if it has been previously selected. If it has, then the appropriate error message is printed and a new selection must be made.

- Possible functions are one to open the input file, place the boats, print out the game board, obtain a valid move(valid row and column index and spot has not been previously selected) and check a valid move to see if it is a hit or a miss (this function would also keep track of the number of hits on each boat),.

- After obtaining and placing a valid move, the message of hit or miss is printed followed by the game board. Then the next selection is made unless the game has completed.

Project 11 Assumptions

You can assume the following facts:

- If a file contains a header line, then the data is all present and correct
- First line of the input file is a header line that can be read when testing for an empty input file
- Each row of the input file will be row_index col_index boat_number
- Integers only are entered by the user for row and column index values.

Project 11 Helpful Information

- Run the sample solution to make sure you understand how the program works, what the inputs are and what the output looks like.

- Use the extraction operator to read the row, column and boat number information from the input file – two integers and a character

- Use the getline function to read the header line from the input file.

- Read the boat numbers as characters and place them in the game board array

- Use static variables to keep track of the total number of hits on each boat. Once that count reaches three for a boat, it is sunk

- Use a variable to keep track of the total number of hits. Once it reaches 12 hits, the game is over.

- For a position selected by the user, if the position in the array has a boat placed in it (character 1, 2, 3 or 4), change the position in the array to the character 'H'. If there is not a boat at the location, change the period to an X

- Remember that arrays are always passed by reference unless const is used before the data type of the array.

- Use the sample solution to understand the flow of the program. .

- Using functions is required and make it easier to write the program. The function to obtain a valid move verifies that the row and column index entered are valid indices and the spot selected has not been previously selected. Another function will take a valid move and see if it is a hit or a miss.

- Each input file has a header line in it indicating the values contained in the columns of the input file. This line can be read to test for empty input files

- Use a loop to read the input file.

- A valid move is one that selects a spot that has not already been selected.

- A function used to check a valid move could use a switch statement to see if the spot selected contains a '.' (a miss), a 'H' (a previous selected spot – print out message), a 'X' (a previous selection that is a miss) or a '1', '2', '3', or '4' if the spot contains a boat.

- Only shots fired at unchecked positions are counted as a shot fired. In other words, if the user enters in a row and column index that has previously been selected, the count for the number of shots fired is not incremented. Therefore there are at most 100 shots that can be counted and a maximum of 12 hits.

- The code that prints out the board has to recognize the boat numbers so that a period can be output instead of the boat number – this is done in the code provided.

<Project 11 C++ Concepts Explained>

The following C++ concepts are included or introduced in this project:

Arrays - Structured collection of components (called array elements) that can be accessed individually by specifying the position of a component with an index value(s).

ArrayDeclaration (for 2 dimensions) syntax template

DataType ArrayName[ConstIntExpression] [ConstIntExpression]

Where DataType describes what type of values are stored in each component of the array.

Accessing Array Elements (2 dimensions) syntax template

ArrayName[indexExpression][indexExpression]

where index expression is an arbitrarily complicated expression that evaluates to an integer value.

Passing Arrays to functions – arrays are always passed by reference; therefore C++ has been designed so that no & is needed when dealing with arrays. To pass arrays into a function as a constant (no changes allowed), the reserved word **const** must be used before the data type of the array.

Sample function heading using arrays and the defined global constant NUM_COL.
void myFunction(const int image[ ][NUM_COL], int filteredImage[ ][NUM_COL])

typedef statements – typedef statements are extremely useful when used with arrays. A typedef statement allows for declaration of an array DataType that can then be used as indicated below.

```
const int ROW = 10; const int COL = 10; // placed in global area of the program
typedef int ImageArray[ROW][COL]; // placed in global area of the program.  Creates a new
                                  // DataType called ImageArray.  Variables declared of
                                  // DataType ImageArray will be 10x10 integer arrays.
```

```
ImageArray myArray; // declares a variable named myArray to be of DataType ImageArray
                    // which makes myArray a 10x10 integer array
```

Sample function heading using the above array DataType created by the typedef statement. Note that dimension brackets are no longer required with the array variable.

void myFunction(const ImageArray image, ImageArray filteredImage)

Allowed Aggregate Operations on Arrays

Aggregate Operation	Allowed on Arrays?
Input/Output	No
Assignment	No
Arithmetic	No
Comparison	No
Argument Passage	Yes, by reference (default) or as a constant
Return as a function's return value	No