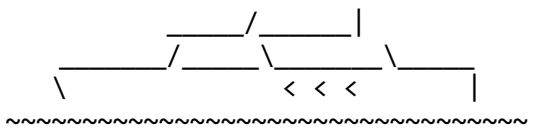
COMP1000 Assignment Report

Bharath Sukesh

19982634

May 17, 2020



Battleships

1 Overview

1.1 battleships.c

This file is the entry point of the program by connecting the terminal with the program and contains the main() method for the game of battleships. This file takes in two command-line arguments that specify what file names are to be read in as settings, over in the **fileio.c**. It also checks whether file names have been entered appropriately and if not, alerts the user. If the settings have been read in validly (handled in fileio.c), then the main menu is displayed from this file which prompts the user a selection of options including playing the game, viewing missiles or creating custom settings files of their own. I chose to create two linked lists in this file, which are used to store specific contents from the settings files. These linked lists are to be used in the game itself. Lastly, this file does any last-minute heap-freeing to prevent memory errors.

1.1 fileio.c

This file is called by the main method in **battleships.c**. This file is given the task of reading both settings files for ships and missiles as well as validating them to ensure they fit the requirements provided by the assignment specification. This file is also associated with assigning **game.c** with the appropriate function for each missile. A missile modifies the game board in a unique manner depending on what missile is used. I placed this functionality in this file as it was related to reading in from the missile settings file itself. It does this by using a function pointer *funcPtr* created in the corresponding header file *fileio.h*. Lastly, this file is responsible for saving user-defined custom settings files (both board & missile files), that can be reloaded into the program for use.

1.2 game.c

This file handles everything regarding the game itself. This file manages tasks such as creating and initialising the game board, printing the board each turn, and all functionality associated with each turn itself. This includes firing missiles, assessing the condition of ships (hit or destroyed), all win conditions, and game ending conditions. My missile and ship structures are created in *game.h* and are stored in linked lists (created back in the main method). This file calls **fileio.c** to read in missiles from the file and using the function field located in each Missile structure, calls its corresponding missile function that modifies the board in a specific, unique manner.

1.3 linkedlist.c

This file contains implementation for a generic doubly linked, double ended linked list. This class is vital for this program as it stores my Missile and Ship structures collectively in their own respective 'linked list'. Most of the code from this file was completed previously for a practical. The structure of my linked list allows me to iterate through them and check the properties of the structures within them. For missiles specifically, I am able to access one missile per turn (by using functions located within this file) which is necessary for the flow of the game.

1.4 macros.c

A small file that contains methods that are used all throughout the program. These include handy methods such as; converting case for specific datatypes like characters and strings; a simple printError method than appends "Error: ", to the given string; a spacer method which is extremely useful for readability within the terminal and lastly a printString method that prints out the contents of a linked list – a data structure used in my program; specifically useful for the menu system. It would have been messy repeating this code in multiple methods, so the best alternative was to implement a file that contains methods common to many files in the program. Its header file $macros.h$ simply contain a few macros I also used all throughout the program (similar idea to the .c file) e.g. TRUE, FALSE.

2 Storing ships & board.

To start off, I stored ships into its own structure, which contained its specific fields/information unique to each ship. I then stored this into a linked list so that it would group them into one data structure (and put them under one roof). This allows me to simply iterate through them when playing the game without any desire for what order they were in inserted in. In the game itself, I fire a missile and then check against the locations in the board where the ships occupy by importing in the length of the ship as well as the location of its head as x, y coordinates that represent its position in the board. If there is a missile present at the same $'x, y'$ as a ship, the program will overwrite the tile to present a different symbol to represent that a ship has been hit. This means that much like a real game of Battleships, the ships are always there, but they are just hidden to the player and only revealed when the user hits a part of it/the entirety of a ship.

The board was implemented using a 2D-array, this made sense to me as the board was representative of a matrix with a specific width and height. What I found difficult was implementing the display for the board to the terminal; as it consisted of a series of for loops that required a bit of thinking to get your head around the order of what was to be printed. This caused a complication which took a few days for me to overcome. An issue I had experienced with the board was accounting for double-digit numbers on the y-axis. I had constructed an approach which ended up looking messy, forcing me to rethink my design.

3 Program demonstration

The following text boxes will be indicative of either the file contents or what is displayed/entered in the terminal.

3.1 Contents of valid input files

board.txt – used as a settings file for the board and ships.

```
5 4
d4 E 3 NullByte Sub
A1 N 3 SSASR Popoto
C2 W 2 CPN "Rogers" Steve
```

test.txt – used as a settings file for the missiles.

```
V-LINE
h-line
V-LINE
H-LINE
H-LINE
```

3.2 Command-line used for program execution

```
./battleships <name of board settings file> <name of missile settings file>
```

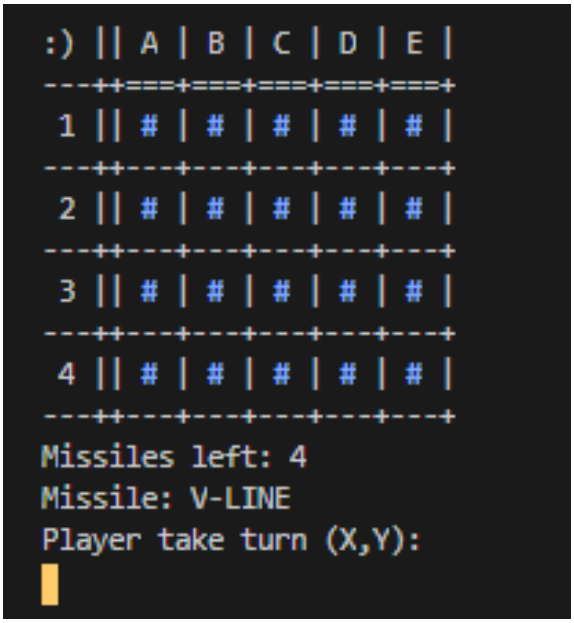
This is the format for running the program.

E.g.

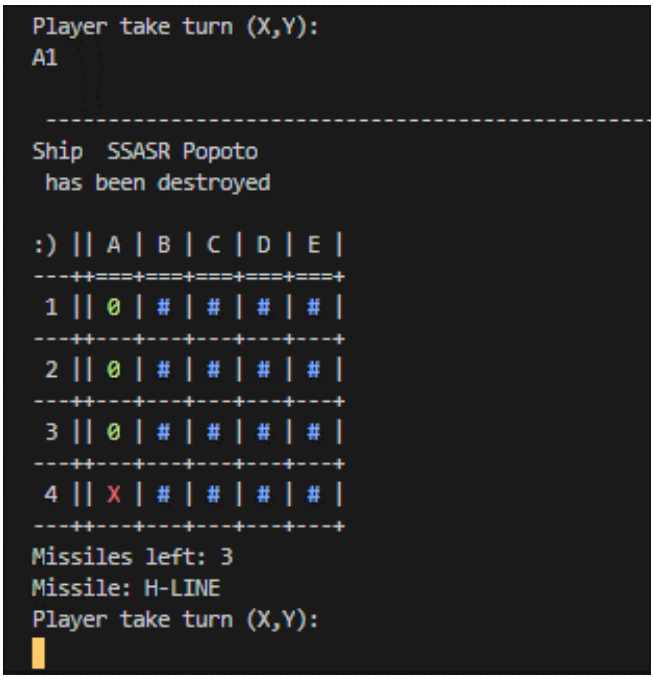
```
./battleships board.txt missiles.txt
```

The format is always the same. The board settings file **must** be entered in before the missiles settings file. You must also specify the file extension at the end **<u>if any</u>.**

3.3 Board output as shown to Terminal

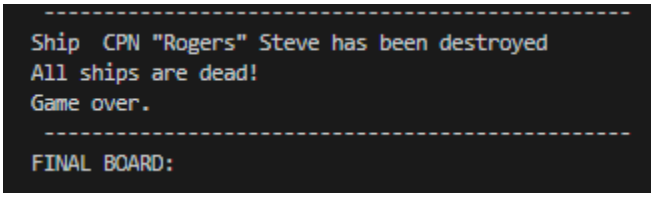


After a coordinate has been entered (also shows the indication of a ship being destroyed).



Game ending message:

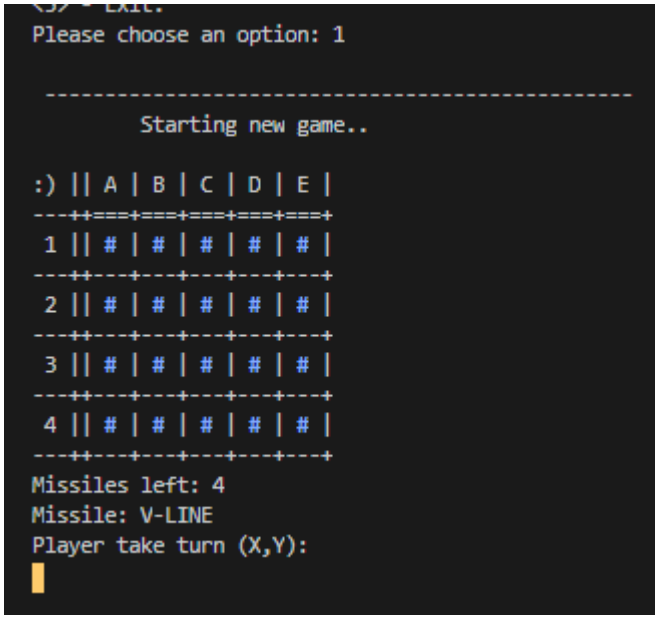
Alternate game ending message:



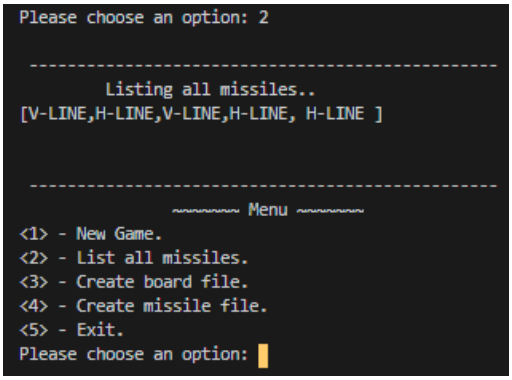
3.4 User input to interact with the main menu

Option #:

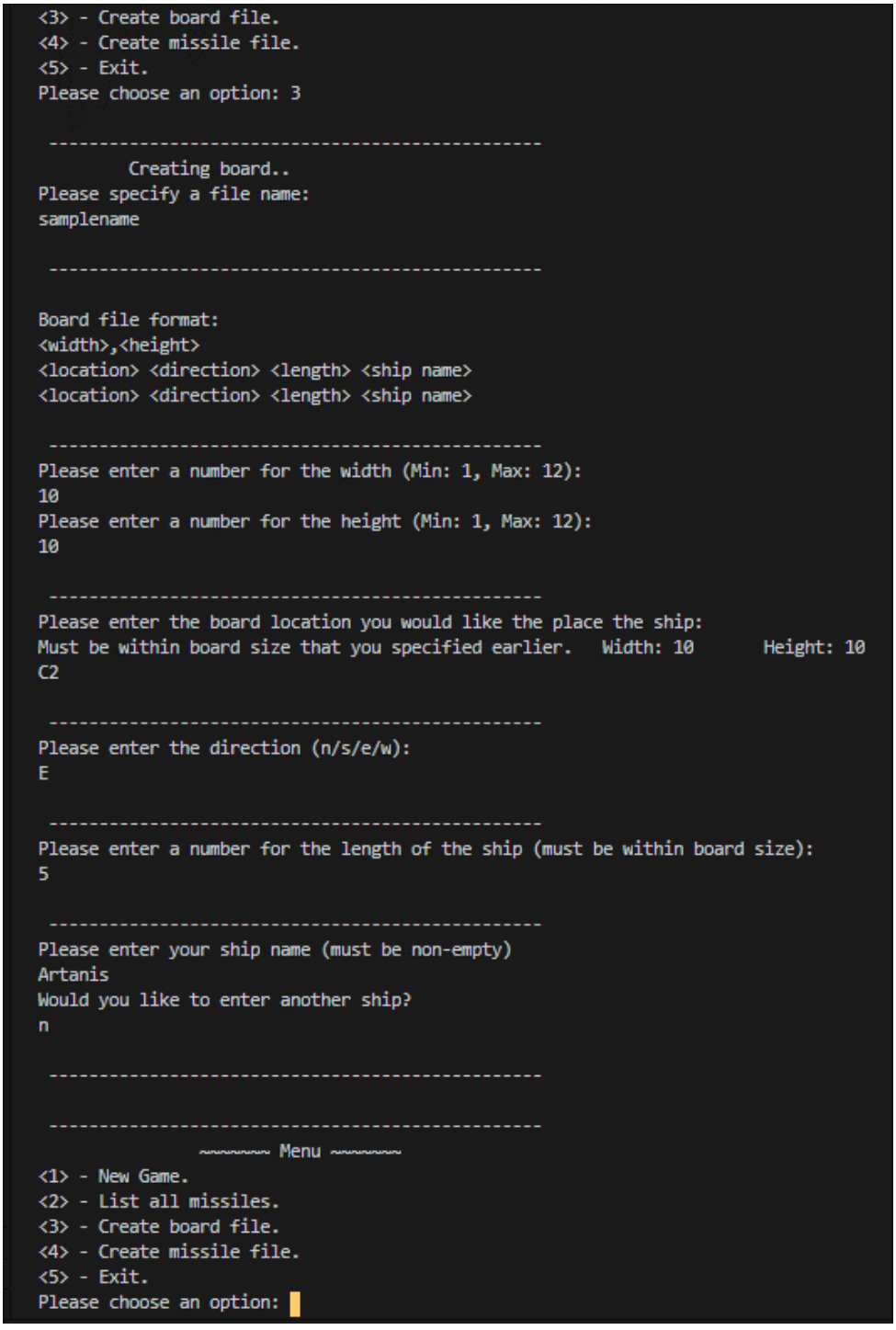
(1)



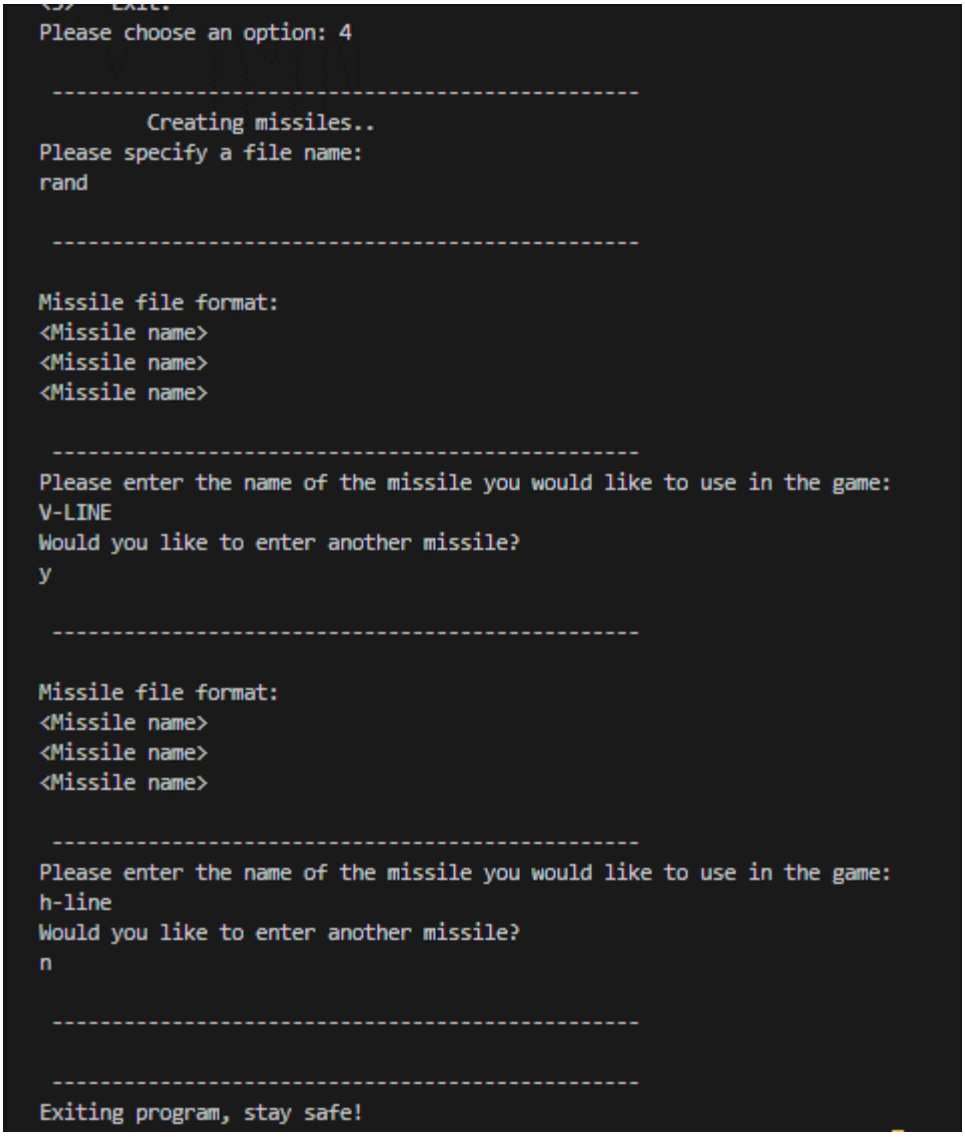
(2)


Please choose an option: 2
Listing all missiles..
[V-LINE,H-LINE,V-LINE,H-LINE, H-LINE]
Menu
<1> - New Game.
<2> - List all missiles.
<3> - Create board file.
<4> - Create missile file.
<5> - Exit.
Please choose an option:

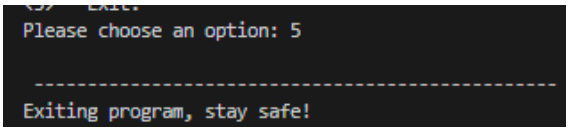
(3)


<3> - Create board file.
<4> - Create missile file.
<5> - Exit.
Please choose an option: 3
Creating board..
Please specify a file name:
samplename
Board file format:
<width>,<height>
<location> <direction> <length> <ship name>
<location> <direction> <length> <ship name>
Please enter a number for the width (Min: 1, Max: 12):
10
Please enter a number for the height (Min: 1, Max: 12):
10
Please enter the board location you would like the place the ship:
Must be within board size that you specified earlier. Width: 10 Height: 10
C2
Please enter the direction (n/s/e/w):
E
Please enter a number for the length of the ship (must be within board size):
5
Please enter your ship name (must be non-empty)
Artanis
Would you like to enter another ship?
n
Menu
<1> - New Game.
<2> - List all missiles.
<3> - Create board file.
<4> - Create missile file.
<5> - Exit.
Please choose an option:

(4)


Please choose an option: 4
Creating missiles..
Please specify a file name:
rand
Missile file format:
<Missile name>
<Missile name>
<Missile name>
Please enter the name of the missile you would like to use in the game:
V-LINE
Would you like to enter another missile?
y
Missile file format:
<Missile name>
<Missile name>
<Missile name>
Please enter the name of the missile you would like to use in the game:
h-line
Would you like to enter another missile?
n
Exiting program, stay safe!

(5)


Please choose an option: 5
Exiting program, stay safe!