SYSC 2006 Foundations of Imperative Programming

Assignment 2

Filenames must be exactly as shown for the submit program. Read and follow the instructions.

Email submissions will not be accepted by TAs or instructor.

Objective:

Write a program called **assign2.c** two-dimensional arrays and file I/O.

Introduction

You will write a single-player version (i.e. play against the computer) of Battleship[1]. Although you should write code so that you can easily change limits (i.e. use constants), you may make a small playing area and use only two boats, one of length two and one of length three.

The requirements can be met by any students in this class, but they are laid out in a gradient, from minimal to bonus. Minimal requirements will earn you a maximum of 80% (assuming full marks for style and testing) whereas meeting all bonus will earn you the full 100% (assuming full marks for style and testing).

Program Requirements

1. Minimal (Up to 80% of the mark) : A user can play a game of Battleship, with the program repeatedly:
   (1) displaying the current view of the playing area, using '.' to represent an unknown space, a 'w' to represent a miss, 'r' to represent a hit but not the whole boat which turn to 'R' when the whole boat has been hit, and
   (2) prompting the user to enter a <row-column> coordinate pair (from 1 to max) for their next guess

   The game should quit once all boats have been discovered, or if the user types 'q' instead of a coordinate. The program shall print out some game statistics, such as the number of guesses, misses and hits.

   For a minimal version, two boats are required, one of length two and one of length three. The playing area can be anything but is recommended to be reasonably small (otherwise you'll be forever playing without ever hitting a boat).

   The game shall randomly locate the two boats upon startup, either vertically or horizontally, with no intersection of the boats.

2. Additional Requirements (Up to 90% of the mark)

[1] Those with deprived childhoods: Look on the web to learn how to play battleship

   a. Command-line arguments are used to provide the dimensions of the boards, as well as the number of boats. Boats are created so that they have a length of 2 if they are the 2^{nd}, 4^{th}, 6^{th} …. boat created, whereas they have a length of 3 if they are the 1^{st}, 3^{rd}, 5^{th} … boat created.

3. Bonus Requirements (Up to 100% of the mark)
   a. Use a configuration file to setup the game. The file format is of your choosing but would describe the number of boats as well as the size of each boat. Ultimately, the file should describe the actual Battleship boats.
   b. Display the game graphically. See Chapter 3.6 of the textbook (Introduction to Computer Graphics (Optional))

Universal Requirements

1. The program shall not send any extraneous outputs to the console. If you use printf() to debug, these extraneous outputs must be removed before submission.
2. The program should use functions to organize the activities.
3. Boats shall be randomly located.

Please! Learn to write your program in incremental steps. You can submit half-written assignments with partial functionality. A partially working program is better than a complete non-functioning program, in school and in life. After each step, compile and run (the execution may not do much at first, but at least you know it does not hang or crash)

   a. Declare your variables and initialize them where appropriate.
   b. Writing to a file is easier and less error-prone than reading from a file. So, do the easy part first: Write out the histogram to a file. The histogram will contain all zeros, but at least you know you have written a file.
   c. Now do the hard part. Read in the data from the test file.
   d. Now do any remaining work to link the input data to the output data.

Testing Requirements

How are you going to test it? How do you know where the boats are, if they are hidden and not displayed to the user? How about, during testing, the program prints out its version of the board once, for reference sake, and then continues on.

Your test should work in two "modes" – testing mode or game mode. In testing mode, the answer board is shown once. In game mode, the answer board is hidden. If you are using command line argument, then add another command line argument to indicate the mode; otherwise, use a prompt-and-input to query the user which mode to run.

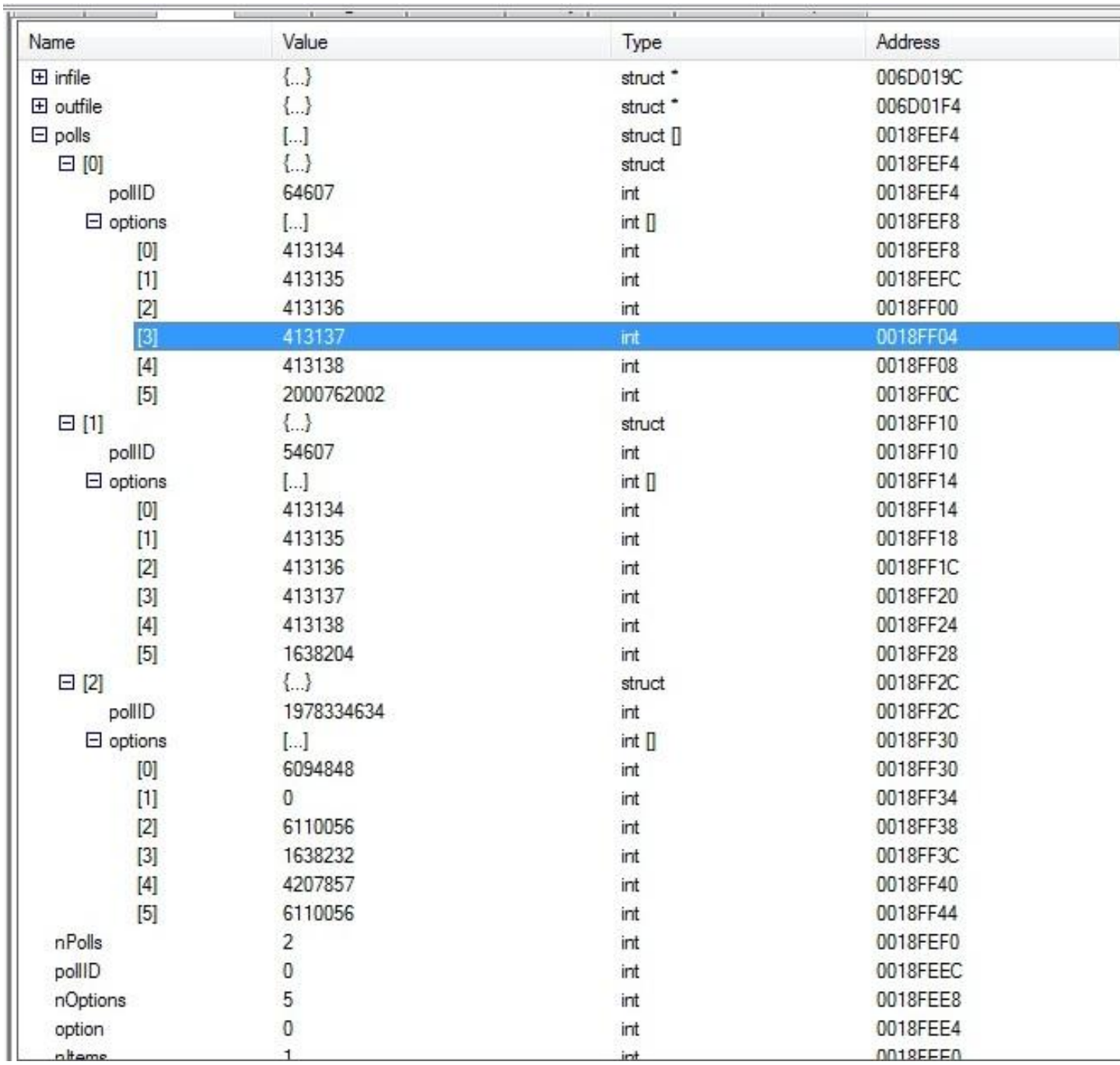
Submission

The following files shall be submitted through the SUBMIT program that is available for download from a link on the Course Resources page, before the deadline.

At the top of your program, in a header comment, write which level of requirements your programs runs at. Help the TA understand what parts you have written and which parts you haven't.

assign2.c: The program

Sample Data Structures



Name	Value	Type	Address
⊞ infile	{...}	struct *	006D019C
⊞ outfile	{...}	struct *	006D01F4
⊟ polls	[...]	struct []	0018FEF4
⊟ [0]	{...}	struct	0018FEF4
pollID	64607	int	0018FEF4
⊟ options	[...]	int []	0018FEF8
[0]	413134	int	0018FEF8
[1]	413135	int	0018FEFC
[2]	413136	int	0018FF00
[3]	413137	int	0018FF04
[4]	413138	int	0018FF08
[5]	2000762002	int	0018FF0C
⊟ [1]	{...}	struct	0018FF10
pollID	54607	int	0018FF10
⊟ options	[...]	int []	0018FF14
[0]	413134	int	0018FF14
[1]	413135	int	0018FF18
[2]	413136	int	0018FF1C
[3]	413137	int	0018FF20
[4]	413138	int	0018FF24
[5]	1638204	int	0018FF28
⊟ [2]	{...}	struct	0018FF2C
pollID	1978334634	int	0018FF2C
⊟ options	[...]	int []	0018FF30
[0]	6094848	int	0018FF30
[1]	0	int	0018FF34
[2]	6110056	int	0018FF38
[3]	1638232	int	0018FF3C
[4]	4207857	int	0018FF40
[5]	6110056	int	0018FF44
nPolls	2	int	0018FEF0
pollID	0	int	0018FEEC
nOptions	5	int	0018FEE8
option	0	int	0018FEE4
nItems	1	int	0018FEE0

Name	Value	Type	Address
nPolls	2	int	0018FEF0
pollID	0	int	0018FEEC
nOptions	5	int	0018FEE8
option	0	int	0018FEE4
nItems	1	int	0018FEE0
⊟ count	[...]	int [] []	0018FE98
⊟ [0]	[...]	int []	0018FE98
[0]	0	int	0018FE98
[1]	0	int	0018FE9C
[2]	0	int	0018FEA0
[3]	0	int	0018FEA4
[4]	0	int	0018FEA8
[5]	1638156	int	0018FEAC
⊟ [1]	[...]	int []	0018FEB0
[0]	0	int	0018FEB0
[1]	0	int	0018FEB4
[2]	0	int	0018FEB8
[3]	0	int	0018FEBC
[4]	0	int	0018FEC0
[5]	0	int	0018FEC4
⊟ [2]	[...]	int []	0018FEC8
[0]	7143692	int	0018FEC8
[1]	5570644	int	0018FECC
[2]	6109944	int	0018FED0
[3]	65536	int	0018FED4
[4]	6109872	int	0018FED8
[5]	0	int	0018FEDC
p	2	int	0018FE94
v	5	int	0018FE90
⊞ line	"64607 413134\n"	char []	0018FE40
poll	64607	int	0018FE3C
vote	413134	int	0018FE38
pollIndex	0	int	0018FE34
optionIndex	0	int	0018FE30