Hacettepe University
Department of Computer Science & Engineering

BBM104 Introduction to Programming Laboratory
Experiment 1

Subject : Dynamic Arrays, Structs, File I/O
Submission Date : 28.02.2018
Due Date : 14.03.2018
Environment : ANSI C, CentOS 7.4.1708, GCC 4.8.5 20150623 (Red Hat 4.8.5-16)
Advisors : Dr. Gönenç Ercan, Dr. Öner Barut, Dr. Cumhur Yiğit Özcan, Dr. Ali Seydi Keçeli, R.A. Özge Yalçınkaya

INTRODUCTION

In this experiment, you are expected to implement a game which is runned with commands from an input file. This game is based on a fantastic adventure board game and runs on a given multi-dimensional array. There will be two sides (monsters, heroes) and they will attack to each other. After given commands are executed, your program should print related outputs into a text file such as current situation of the map and status of the characters. Details are explained below.

BACKGROUND INFORMATION

Adventure board games involve fantastical elements like heroes which explore the given map and the monsters they encounter during this exploration. Both heroes and monsters have different properties such as special attacks, health and they fight with each other accordingly. In general, game is finished after given quest is accomplished.

Monsters are placed on some specified points and heroes can attack them if they adjacent to their square. This combat takes place in an order like first all monsters attack to their adjacent heroes in their spawn order and then all heroes attack to their adjacent monsters in their spawn order vice versa. Therefore, if there is more than one monster adjacent to a hero, the first spawned monster will attack first and so on. Similarly, heroes attack in their spawn order to an adjacent monster.

Health is usually measured in hit points or health points, shortened to “HP”. When the HP of a character reaches zero, it become incapacitated or die. Each hero and monster have different attack powers and they give damage each other according to these attack powers.

Consequently, monster's or hero's HP is decreased according to attack's "damage" value. If a hero kills a monster, that hero gains one experience point, shortened to "XP".

EXPERIMENT

1) **Loading Characters**

You will read 2 input files in order to start the game. First one is named "chars_<input_number>.txt" and it includes properties of heroes and monsters such as "name", "HP" and "damage value" in the given order, separated by comma. In each line there will be a character in the format of <TYPE>,<NAME>,<HP>,<DAMAGE>:

HERO,DRIZZT,8,3
MONSTER,MINDFLAYER,10,2
HERO,CATTIE,6,3
MONSTER,TROLL,4,3
MONSTER,GOBLIN,1,3

There can be any number of heroes and monsters. The order of attacking can be determined according to the order of spawn of the characters. Therefore, while your program is reading lines of this file, in each line a character will be spawn. For instance, in the given example above, when its heroes turn and they are going to attack, the order of attacking will be like this: "1.DRIZZT, 2.CATTIE". Moreover, for the monsters it will be like this: "1.MINDFLAYER, 2.TROLL, 3.GOBLIN".

2) **Playing the Game**

The second input file, which is named as "commands_<input_number>.txt", includes the commands in order to play a scenario. The following commands will be fixed at the start of each file. Because, we need to load the map to start the game and we should put characters in specified places. Therefore, there is no need to print error messages for this part.

LOADMAP 5 5
PUT HERO DRIZZT 0 0 CATTIE 4 1
PUT MONSTER TROLL 1 1 GOBLIN 4 0 MINDFLAYER 0 3

All the other commands can be given in any order. Each line will contain only one command and there will not be an empty line anywhere in the file. The format of the

commands and the error messages that you should give will be explained below in detail. All commands and command parameters will be separated by a one space.

- **LOADMAP:** Loads a map with the given row and column sizes. These size values will be integers and can be any number. This map is basically a 2 dimensional array and should be created dynamically according to given row and column sizes.

  Input
  LOADMAP <row_size> <column_size>

  Output
  There will not be any messages for this command.

  Error
  There will be no incorrect entry for this command.

- **PUT:** Puts the given characters on to the map. <type> will be only "HERO" or "MONSTER". Characters can be given in any order for this command.

  Input
  PUT <type> <hero1_name> <row_no> <column_no> <hero2_name> <row_no> <column_no> <hero3_name> <row_no> <column_no> ...
  PUT <type> <monster1_name> <row_no> <column_no> <monster2_name> <row_no> <column_no> <monster3_name> <row_no> <column_no> ...

  Output
  There will not be any messages for this command.

  Error
  There will be no incorrect entry for this command.

- **SHOW (20 points):** Prints the current situation of the heroes, monsters or the map. <type> will be only "HERO", "MONSTER" or "MAP". Characters should be printed in the spawn order. While printing the map, you should use the first letter of the character names and put '.' for empty squares.

  Input
  SHOW <type>

Output

<type> STATUS

<char1_name> HP: <char1_hp> XP: <char1_XP>

<char2_name> HP: <char2_hp> XP: <char2_XP>

Note: "XP" should not be printed for MONSTER type.

MAP STATUS

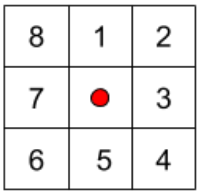
```
D M . . .
. T . . .
. . . . .
. . . . .
G C . . .
```

Error

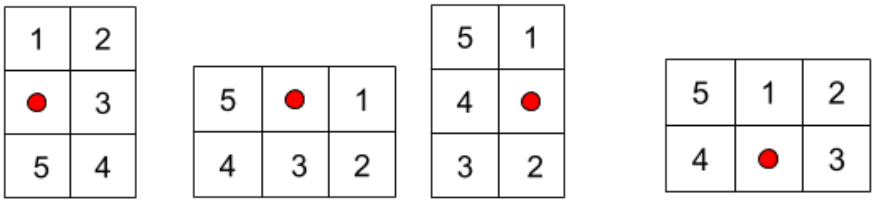
There will be no incorrect entry for this command.

- **ATTACK (25 points):** Executes the attack command for the given <type>. After this command is executed, the characters for the given <type> should attack the others and a combat will start. There are some rules you should consider:

    - First, characters can only attack adjacent enemies in the spawn order. For instance, for the ATTACK MONSTER command, all monsters start to attack in the order of: 1.MINDFLAYER, 2.TROLL, 3.GOBLIN.
    - Characters can attack any adjacent square and there are 8 directions in total. In addition, this directions have some attack priorities as shown below. Therefore, if there are more than one enemies adjacent to a hero or monster (red dot in the middle), it should attack the enemy in square 1 first, then square 2, square 3 and so on.



For the example above, if the command is ATTACK HERO, Drizzt will attack monsters before Cattie, and Drizzt will attack Mindflayer first and then Troll. This rule valid for the below conditions as well:



- Same type of characters can not attack each other.
- In order to simulate the combat, you should decrease the HP of damaged character in the amount of damage value of the character which is attacking. So, for the example above, after a monster attack, Drizzt will have HP=3 (8-2-3) and Catie will have HP=3 (6-3).
- A character's HP can not be smaller than "0". If a character reaches HP=0, it can not attack or can not be attacked and should be removed from the map. However, the status of this character should be printed after each SHOW <type> command.
- If a hero kills a monster, it's XP should be increased 1 point. If a monster kills a hero, do nothing.

Input
ATTACK <type>

Output
<type>S ATTACKED

Error
There will be no incorrect entry for this command.

- **MOVE (20 points):** This command will be only for heroes and inputs will be given only for hero type, there is no need to check for type. However, you should check if given hero's new position is out of the map or occupied by another character. Also, a dead hero can not be moved. Therefore, you should give error messages for these type of inputs. A hero will always move one unit and inputs will be given for one unit move. Heroes can be given in any order for this command.

Input
MOVE HERO <hero1_name> <row_no> <column_no> <hero2_name> <row_no> <column_no> <hero3_name> <row_no> <column_no> ...

Output
HEROES MOVED

Error

- If a hero or a monster is in the next position of a given hero, give this message:

  <hero_name> can't move. Place is occupied.

- If hero's next position is exceeds map boundaries, give this message:

  <hero_name> can't move. There is a wall.

- If hero is dead, give this message:

  <hero_name> can't move. Dead.

3) Finishing the Game (5 points)

After each ATTACK command is executed, you should check if all the heroes or all the monsters are dead. If one of these cases is true, print a message and terminate the program even there are more commands in the commands file. Message is:

ALL <type>S ARE DEAD!

Other than that, the game may not be finished according to given scenario from the input file (all monsters or heroes can be alive). In this case, just execute the commands and write the outputs accordingly.

RULES

- You should use dynamic arrays and structs in order to save given characters and their properties. Other methods will not be accepted.
- The game map should be created as 2D dynamic array. Other methods will not be accepted.
- Follow C coding standards and pay attention to code readability. Use comments. **(5 points)**
- You should comply with given output formats in order to get points from evaluation. **Note that it is not possible to get sufficient points without implementing the scenarios properly.**

INPUT & OUTPUT

The paths of the input and output files will be provided as command line arguments in the order of "chars_<input_number>.txt commands_<input_number>.txt". You will be able to find example input and output files on Piazza. You can also create your own inputs. Remember that except the mentioned commands, all commands can be in any order and in any amount. Output file name should be in the format of "output_<input_number>.txt"

REPORT (25 points)

- In the report, you are expected to determine and explain the problem: What is the main goal of the assignment and what points need to be understood. **(5 points)**
- Then, give your solution in detail: How do you approach this problem and how do you implement it, what do you use. **(5 points)**
- Explain your data structure: What structures do you use to store characters or the map etc. (dynamic arrays, structs) **(5 points)**
- Explain your code. For example, the purpose of the functions, but do not copy your code, just mention important parts. **(5 points)**
- Give your detailed algorithm step by step. **(5 points)**

NOTES

- Your experiments will be executed in DEV machine, please make it work on dev before submitting. Otherwise your submission will not be evaluated. Any library does not exist in DEV is forbidden. Compile your code with "-ansi" option.
- Do not forget to free allocated memory and check memory leaks by using "valgrind" on DEV machine.
- Input and output file names will not be fixed, so you should not hardcode them into your program.
- Save all your work until the assignment is graded.
- You can ask your questions about the experiment by sending your messages to Piazza Group. Any other method will not be accepted.
- Also, you are responsible to read discussions about the assignment on Piazza.

- You have to submit your experiment files in the given directory structure below:

```
<student id>
    <src>
        *.c
        *.h
    <report>
        report.pdf
```

- You have three days for late submission. You will lose 10 points from maximum evaluation score for each day (your submitted study will be evaluated over 90, 80 and 70 for each late submission day). You have to submit your solution in deadline date + three days, otherwise it will not be evaluated.

ACADEMIC INTEGRITY

All work on assignments must be done individually. You are encouraged to discuss with your classmates about the given assignments, but these discussions should be carried out in an abstract way. That is, discussions related to a particular solution to a specific problem (either in actual code or in the pseudocode) will not be tolerated. In short, turning in someone else's work, in whole or in part, as your own will be considered as a violation of academic integrity. Please note that the former condition also holds for the material found on the web as everything on the web has been written by someone else.

You can get punishment points if you don't follow the rules defined in this document.

REFERENCES

- A rule book of an adventure board game http://dnd.wizards.com/sites/default/files/media/LoD_Rulebook_EN.pdf
- A good reference for C libraries: www.cplusplus.com
- The whole ANSI C standard library reference: http://www.csse.uwa.edu.au/programming/ansic-library.html