Hacettepe University

Computer Science and Engineering Department

Spring 2018

Name and Surname ***: Tolgahan DİKMEN***

ID Number ***: 21327929***

Course ***: BBM104 Programming Lab.***

Experiment ***: Assignment 1***

Subject ***: Dynamic Arrays, Structs, File I/O***

Due Date ***: 14.03.2018***

Advisor ***: R.A Özge YALÇINKAYA***

E-mail ***: tolgahandikmen95@gmail.com***

1. INTRODUCTION

In this assignment, a basic game design is done using the C programming language. The design consists of two different parts. The first part starts creating character structures (heroes and monsters) with dynamically which are given by user via .txt file, then the process continue with creating a multi dimensional array structure which is going to be map. The second part is to make the desired operations using these structures. The implementation has to be created dynamically because the sizes of the map and the characters are changeable according to the input file.

According to design, there must be two sides which are monsters and heroes, they will attack and try to kill each other sequentially. The game can end in three different situations. All monsters can be killed, all heroes can be killed or all commands may be executed even if the heroes or monsters are alive.

2. SOFTWARE USING DOCUMENTATION

2.1. Software Usage

The program needs three different inputs to work; input for characters, for commands and for the output. The characters input file contains information about the monsters and the heroes. Each line of the input file contains race of character information, name of character information, health of character information and attack damage of character information. The second input file contains execution commands for the program. The execution commands will be examined below. The last input file is only needed to write output results.

There are four different operation commands in the program:

- **Load Map:**

  Format: ’LOADMAP’ ’Row Number’ ’Column Number’

  In this command, the boundaries of the map are determined by the numbers given.

  There is no output message and no incorrect entry for this execution command.

- **Put:**

  Format: ’PUT’ ’Type’ ’HeroName’ ‘RowNumber’ ‘ColumnNumber’...

  Format:’PUT’ ’Type’ ’MonsterName’‘RowNumber’ ‘ColumnNumber’

  In this command, given characters are placed on the map.

  There is no output message and no incorrect entry for this execution command.

- **Show:**

  Format: ’SHOW’ ’Type’

  In this command, the instant status of the map, heroes or monsters are printed on the output file. Type can be only “HERO”, “MONSTER” or “MAP”.

  There is no incorrect entry for this execution command.

- **Attack:**

  Format: ’ATTACK’ ’Type’

  This command executes the attack command for the given type. After is executed this command, a combat will be start.

  There is no incorrect entry for this execution command.

- **Move:**

  Format: ’MOVE’ ’HERO’ ‘heroName’ ’rowNumber’ ‘columnNumber’

  This command is only for heroes. After is executed this command, a hero can change its position if the place to go is convenient otherwise prints an error message.

2.2. System Bugs

The program may stop the execution according to some situations.

The boundaries of the map are not checked when the characters are placed on the map. If you try to place a character outside the map or if an attempt is made to place it where another character is placed, the program closes.

3. SOFTWARE DESIGN NOTES

3.1. Definition About The Values

Variables and arrays which has been used:

- **char *input = argv[1]:** Gets the name of the first input file which is going to be character file (or commands file).
- **char *input2 = argv[2]:** Gets the name of the second input file which is going to be commands file (or character file).
- **char *output = argv[3]:** Gets the name of the output file path and name.
- **int *heroCount:** Holds heroes number.
- **int *monsterCount:** Holds monsters number.
- **Map **map:** It is a two dimensional array which holds map information.
- **Hero *heroes:** It is an array which holds hero information on each element.
- **Monster *monsters:** It is an array which holds monsters information on each element.

3.2. Definition About The Functions and Structures

The program contains three structure of struct type.

```
typedef struct {
        char charName[100];
} Map;
```

That structure form contains a string (char array) to keeps the names of the characters on the map separately for each cell.

```
typedef struct {
        char name[100];
        int hp;
        int attackDamage;
        int xp;
} Hero;
```

That structure form stores all information about heroes.

```
typedef struct {
        char name[100];
        int hp;
        int attackDamage;
} Monster;
```

That structure form stores all information about monsters.

Functions about the program:

void getPlayerCounts () :

The function counts hero and monster numbers in characters input file.

void setChars () :

This function fills the character lists created for monster and hero.

void runCommands () :

This function separates the commands in the execution file and it follows which commands are handled respectively. It executes the necessary procedures for each game command and move on to next command. The game continues till all commands will be finished as long as a monster or a hero is alive.

int attack () :

This function manages the attack function of the characters according to which player is going to attack. After this command is executed, the given type of characters will attack to the enemies of them. Characters can only attack clockwise adjacent enemies based on their position on the map. Characters can damage to the enemies as much as their attack damage. This command also reduces health points (hp's) of characters that the attacked according to the taken damage.

This function returns '0' or '1'. If all monsters or heroes are dead, function will return '1', otherwise '0'.

This assignment is very educational for ‘struct’ structures and dynamic memory management. Also pointers need to be used for structure. In a given sense this assignment comes up with the time and difficulties in a parallel way.