Assignment 3

Design rEPORT

Mustafa Ahmed

2015

Table of Contents

[**1 Introduction** 2](#_Toc419204538)

[**2 Main.cpp** 2](#_Toc419204539)

[**3 Classes** 3](#_Toc419204540)

[Zombie Class 3](#_Toc419204541)

[Variables 3](#_Toc419204542)

[Functions 3](#_Toc419204543)

[Combat Class 5](#_Toc419204544)

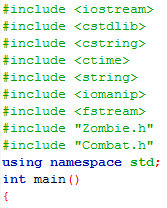
[Variables 5](#_Toc419204545)

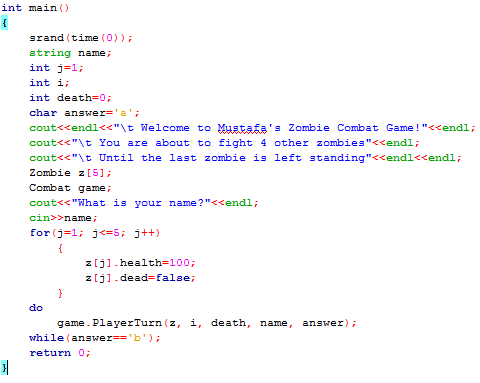
[Functions 5](#_Toc419204546)

# **1 Introduction**

This program has been created as per the requirements mentioned in the Assignment 3 Question PDF. This is a program for a Zombie Combat Game made using Object Orientated Programming concepts.

# **2 Main.cpp**

The main.cpp consists of:

* The following C++ libraries and Class Headers
* srand function to help generate inconsistent random numbers within the ranges mentioned in the latter functions
* Variables that have been declared for the use of this Program
  + string name; - To hold the name of the user
  + int j=1; - used in the for loop to initialize the health to 100 and bool variable, dead to false;
  + int death=0; - this variable sotres the number of players (including the user) that are dead. A player is dead once their health has reached 0 from 100
  + char answer=’a’; - This variable is used for the end of the program when the user has lost and has been asked if he/she would like to play again.
* A Welcome Message
* The declaration to create and array of 5 entries from the class “Zombie”
* A prompt for the user’s name
* A for loop to initialize the values of for each entry for the array Zombie
* The declaration of an object from the class “Combat” to call the function “**PlayerTurn**” from the Combat class

# **3 Classes**

There have been 2 classes that have been used to create this game:

* Zombie Class (Base Class)
* Combat Class (Derived Class)

The Combat Class has a relationship with the Zombie Class via Inheritance, that is, the Combat class has Inherited the variables and functions from the Zombie class, thus making the Zombie class the Base class and the Combat class the Derived class.

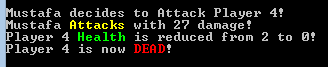
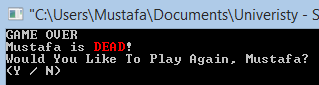
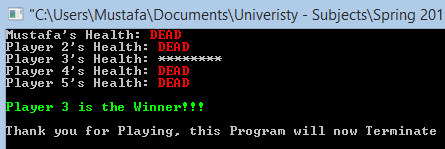
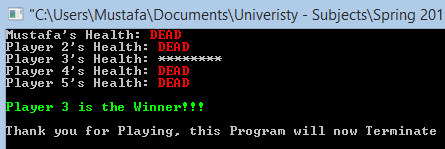
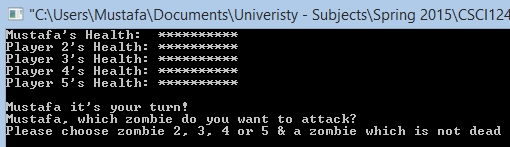
## Zombie Class

The Zombie class was designed to consist of the functions required to perform the various checks and displaying of the data in the form of Health Bars, for example.

Variables  
The Zombie class consists of the following **variables**:

* double health – This variable is used to hold the value of the health which is initialized to 100 at the start of the game.
* int attack; - This variable is used to hold the value of the attack which is a randomly generated number between the range of 25-50
* bool dead; - This bool variable is used to identify if the zombie or user’s character is dead. The character is said to be dead when the health is 0.
* int n; - This is a counter variable which will be used in reference for a few functions.

Functions  
The Zombie Class consists of the following **functions**:

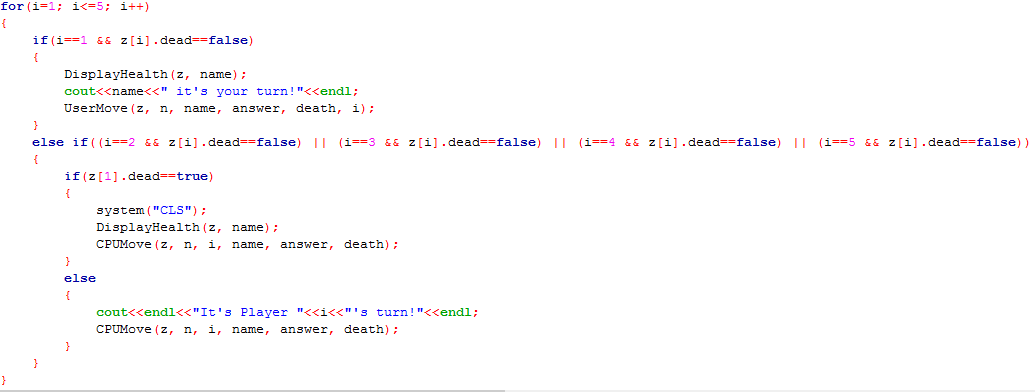
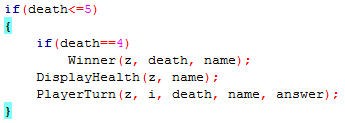
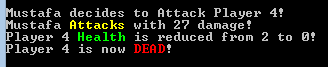
* void IsDead(Zombie z[], int& n, int& death, string& name, char answer, int& i)
  + This function is used to check if the zombie or user is dead. A Player is only dead if that Player’s Health is equal to 0. If a player is dead, it outputs a message saying that the Zombie or User is dead. The death variable is used as a counter, so if a player is dead, it increments the counter.
  + If the User is dead and 3 other Zombies are dead too, the program displays the “Game Over” message and asks the user if he/she would like to play the game again.
  + If the user chooses not to continue playing by entering anything except for “Y” or “y”, then the Winner of the Game is displayed.
  + If the user chooses to play again, the Health and Dead variable are reset to 100 (Health) and false (Dead). death is set to ‘0’, i is set to ‘0’ and answer is set to ‘b’ to satisfy the while condition in the main.
* void Winner(Zombie z[], int& death, string& name)
  + This function is called when 4 out of 5 players are dead. This only shows the final Health of all the Players by calling the function “DisplayHealth”, and it mentions which Player won the game. Then the program is terminated.
* void DisplayHealth(Zombie z[], string& name)
  + A variable called “double life” is declared. The health is divided by 10 and rounded up to the nearest 10th (using the ceil() function) and is stored in the “life” variable.
  + The number of stars printed to represent the life bar is printed according to the value in the “life” variable.
  + For example, if z[i].health is 70, then life is set to 7. The for loop prints out the number of stars based on the value in life. Therefore 7 stars are printed.

## Combat Class

The Combat class was designed to handle the game operations and processes.

Variables  
The Combat class doesn’t consist of any variable as all of the variables are inherited from the base class, the Zombie class.

Functions  
However, the Combat class consists of the main functions that handle the operations and processes of the game mechanics. The Combat class consists of the following **functions**:

* void PlayerTurn(Zombie z[], int& i, int& death, string& name, char answer)
  + The function starts with a for loop that runs through each player (numbers 1-5), based on the player number, another function is called. If i=1, it’s the User’s turn and so the “UserMove” function is called. If i is equal to any number between 2-5, the “CPUMove” function is called.
  + The next part of the “PlayerTurn” function checks if the death counter is less than 5, if yes, then it checks to see if the death counter is equal to 4, if yes, it call the “Winner” function. If the death counter is not equal to 4, it calls the “DisplayHealth” function followed by recursively calling the “PlayerTurn” function.
  + Therefore, this functions shows various function calls based on different if statements and uses recursion to let the game continue until there is only 1 Winner left
* void UserMove(Zombie z[], int& n, string& name, char answer, int& death, int& i)
  + This Function asks the User which Zombie they would like to attack. The user is only allowed to choose between Zombie 2-5 as they are only 5 players in total.
  + For the Attach variable, a random number is generated between the range of 25-50. The Health value is deducted by the Attack value, and so a new Health value is set. For example if Health=100 and Attack=25, the Health variable is now set to 75 (Health-Attack(100-25)).
  + If the new Health value is less than 0, it is set to 0 as the final value.
  + Once the new Health Value is set, the “IsDead” function is called to check if the Zombie is dead based on the new Health of the Zombie
* void CPUMove(Zombie z[], int& n, int& i, string& name, char answer, int& death)
  + This function performs a similar task as the “UserMove” function, except the Player that the Zombie would attack is randomly generated between the values of 1-5 but it can’t be a the value equal to the Player’s number otherwise it would attack itself which is not a valid move. That is, Player 3 cannot choose to attack Player 3.
  + Once the new Health Value is set, the “IsDead” function is called to check if the Zombie is dead based on the new Health of the Zombie