Bryan

Nguyen

CSC-17C

PROJECT 2

(47341)

**Introduction**

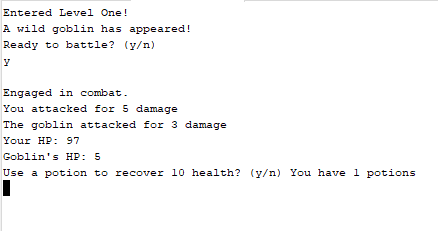
The project is a type of battle game where the player tries to defeat the monster. The player must use his items wisely and have some luck in order to beat the game. The monsters on each level gain increasingly higher health and attack points. Every monster has a chance to drop a potion upon death. At the end of each level, if the player successfully defeats the monster, he will also gain attack, almost like a level up. The user will continue the game by entering ‘y’ or ‘Y’ and a random number generator will decide how much damage is inflicted, based on how much attack the player and monster has. For example, if the player has 15 attack, he will do damage anywhere between 1-15 inclusive. The player will be given the option to use a potion only if he has taken damage and is not dead. Entering ‘n’ when given the choice to use a potion will keep the potion in the player’s inventory. Entering ‘n’ when given the choice to continue will simply end the game and the player must restart from the beginning when the game is begun. The player’s status progression will output at the end of each battle. Once the player wins the game or is defeated, all of the monsters’ total health will be displayed for them to see the progression.

**Summary**

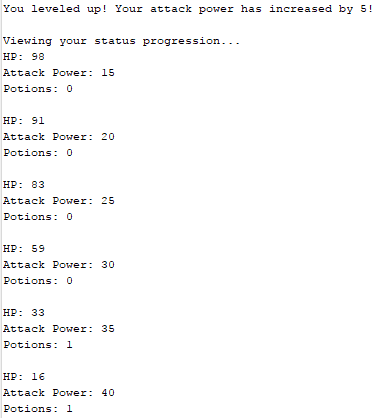
The project is approximately 930 lines of code. It meets the criteria for the project because it consists of maps, sets, lists, stacks, queues, iterators, and algorithms. It took roughly a week to finish the project. There are 7 main variables, about 5 constructs and 2 objects. The idea of the project itself was not difficult to code, but it seems that it is better to learn the STL with a simpler program at first. The STL was difficult to implement because it is not like what I have seen or utilized before. The iterators were probably the most confusing part.

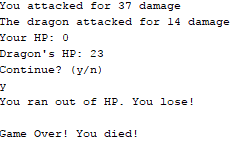
**Description:**

The project was coded focused mainly on objects utilizing the specified C++ STL along with hashes, graphs, trees, and recursive sorts.

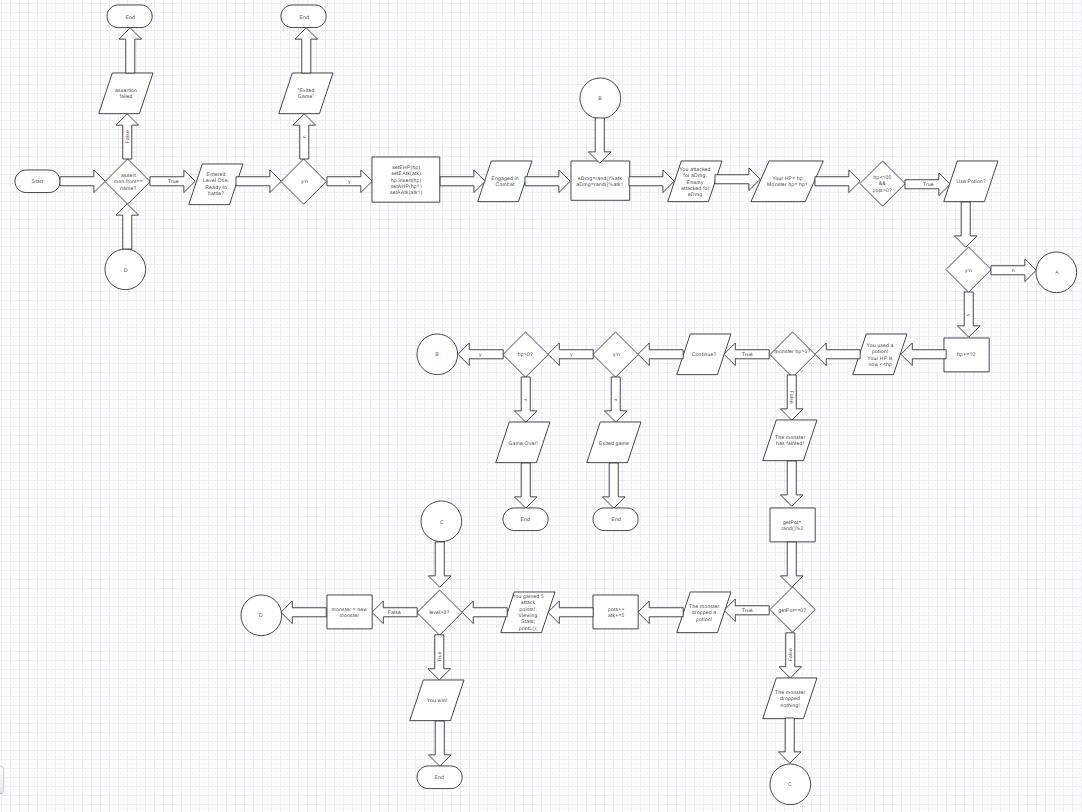








Attempted to zig zag the flow chart to make it more readable but it still does not appear to be.



**Pseudocode**

Display Entered Game

Display Login

Enter login

If(login)

Output “logged in”

Else

Output “Create account”

Username.txt

Display ready to battle?

Get y/n

If n

Display Exited Game

If y

Set monster HP and attack

Set player HP and attack

Do

playerHP – rand()%(monster attack)+1

monsterHP – rand()%(player attack)+1

Display playerHP and monsterHP

Display use potion?

Get y/n

If y

playerHP+=10;

Display Potion Used! Hp: playerHP

If monsterHP==0

Display You defeated the monster!

Win++

If playerHP==0

Display You Lose

monster = new monster

while Win<#monsters

if win==#monsters

Display You Win!

**Classes**

|  |
| --- |
| Status |
| Lv |
| Graph |
| Login |
| MergeSort |

**Important Variables and Constructs**

|  |  |  |
| --- | --- | --- |
| Set<int> | hp | Levels.h (27) |
| Map<const int> bool | Tier | Levels.h (28) |
| List<Lv> | L | Levels.h (29) |
| Srand |  | Main.cpp (23) |
| Int | ally, enemy, pots, aAtk, eAtk, atk1, atk2 | Status.h (20-21) |
| Queue<string> | Mon | 21 |
| Stack<string> | Monster | 22 |

**Important Concepts**

|  |  |
| --- | --- |
| **List** | **Levels.h(52-57) levels.cpp(128,232,335)….** |
| **Iterator** | **Levels.h (53) levels.cpp(131,235,338)….** |
| **Algorithm** | **Levels.h(60) main.cpp(35)** |
| **Queue** | **Levels.cpp(39-40, 143-144, 247-248)…** |
| **Stack** | **Levels.cpp(41-42, 145-146, 249-250)….** |
| **Set** | **Levels.cpp(63,166,270,373,476)….** |
| **Map** | **Levels.cpp(132,141,236,245,339,348)….** |
| **Hash** | **Login.cpp(23,56)** |
| **Recursive Sort** | **mergeSort.cpp, main.cpp** |
| **Graphs** | **Graph.cpp, main.cpp** |
| **Trees** | **Tree.h, main.cpp** |

**Doxygen files included in order to view methods.**

I converted a piece of code from the iterator of line 53 of Levels.h in order to suit my list. I also converted the algorithm from the sgi website. I learned the C++ STL concepts from the SGI website provided, so they all should have similarities.

**Code**

**MAIN.CPP**

#include <cstdlib>

#include <ctime>

#include "levels.h"

#include "login.h"

#include "mergeSort.h"

#include "Tree.h"

#include "Graph.h"

using namespace std;

void itpre(Node\*);

int main(int argc, char\*\* argv) {

srand(static\_cast<unsigned int>(time(0)));

cout << "Sorting numbers..." << endl;

MergeSort a;

a.fill(10);

a.print(5);

a.mergeSort(0,10);

a.print(5);

cout << "Converting levels into nodes..." << endl;

Node \*root = newNode(1);

root->left = newNode(3);

root->right = newNode(5);

root->left->left = newNode(2);

root->left->right = newNode(4);

root->right->left = newNode(6);

itpre(root);

cout << endl;

cout << "Calculating Lucky Rounds" << endl;

Graph g(4);

g.addEdge(0, 1);

g.addEdge(0, 2);

g.addEdge(1, 2);

g.addEdge(2, 0);

g.addEdge(2, 3);

g.addEdge(3, 3);

g.BFS(2);

cout << endl << endl;

Lv game;

game.read();

game.l1();

game.l2();

game.l3();

game.l4();

game.l5();

game.l6();

game.l7();

game.l8();

game.prntHP();

game.outWin();

game.rank();

return 0;

}

void itpre(Node\* root){

if (root == NULL)

return;

stack<Node \*> nodeStack;

nodeStack.push(root);

while (nodeStack.empty() == false)

{

struct Node \*node = nodeStack.top();

printf ("%d ", node->data);

nodeStack.pop();

if (node->right)

nodeStack.push(node->right);

if (node->left)

nodeStack.push(node->left);

}

}

**LEVELS.H**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

/\*

\* File: levels.h

\* Author: Bryan

\*

\* Created on October 27, 2017, 5:33 PM

\*/

#ifndef LEVELS\_H

#define LEVELS\_H

#include <list>

#include <map>

#include <set>

#include <iterator>

#include <queue>

#include <stack>

#include <assert.h>

#include "status.h"

using namespace std;

class Lv:public Status, public Login{

private:

set< int > hp;

map<const int, bool> tier;

list <Lv> l;

public:

Lv();

void l1();

void l2();

void l3();

void l4();

void l5();

void l6();

void l7();

void l8();

void l9();

void l10();

void prntStats(){

cout << "HP: " << ally << endl;

cout << "Attack Power: " << aAtk << endl;

cout << "Potions: " << pots << endl << endl;

}

void prntEStats(){

cout << "HP: " << enemy << endl;

cout << "Attack Power: " << eAtk << endl;

cout << endl;

}

void prntL(){

for(list<Lv>::iterator it = l.begin(); it!=l.end(); it++){

Lv lv = \*it;

lv.prntStats();

}

}

void prntHP(){

cout << "Defeated Monster's Total Health Points:" << endl;

copy(hp.begin(), hp.end(), ostream\_iterator<int>(cout, "\n"));

}

void outWin();

void rank();

};

#endif /\* LEVELS\_H \*/

**LEVELS.CPP**

#include <iostream>

#include <string>

#include <cstdlib>

#include "levels.h"

using namespace std;

Lv::Lv(){

tier[1]=false;

tier[2]=false;

tier[3]=false;

tier[4]=false;

tier[5]=false;

tier[6]=false;

tier[7]=false;

tier[8]=false;

mon.push("Goblin");

mon.push("Zombie");

mon.push("Sand Rat");

mon.push("Wolf");

mon.push("Tiger");

mon.push("Boar");

mon.push("Dragon");

mon.push("Baboon");

monster.push("Goblin");

monster.push("Zombie");

monster.push("Sand Rat");

monster.push("Wolf");

monster.push("Tiger");

monster.push("Boar");

monster.push("Dragon");

monster.push("Baboon");

enemy=0;

ally=0;

pots=0;

}

void Lv::l1(){

assert(mon.front() == "Goblin");

mon.pop();

assert(monster.size()==8);

monster.pop();

cout << "Entered Level One! " << endl;

char reply;

cout << "A wild goblin has appeared! " << endl;

cout << "Ready to battle? (y/n)" << endl;

cin >> reply;

reply = tolower(reply);

while((reply!='y')){

if(tolower(reply=='n')){

cout << "You have exited the game" << endl;

break;

}

cout << "Invalid response. Please enter a valid response: " << endl;

cin >> reply;

reply=tolower(reply);

}

if(reply == 'y'){

cout << endl;

setEHP(10); //Enemy HP

setAHP(100); // Ally HP

hp.insert(10);

char cont=' ';

cout << "Engaged in combat." << endl;

setAAtk(10); //Able to attack up to 10 damage

setEAtk(3); //Able to attack up to 3 damage

pots++;

do{

eDmg(); //Enemy taking damage

cout << "You attacked for " << atk1 << " damage" << endl;

aDmg(); //Ally taking damage

cout << "The goblin attacked for " << atk2 << " damage" << endl;

cout << "Your HP: " << ally << endl;

cout << "Goblin's HP: " << enemy << endl;

if(ally < 100 && pots>0){

char rsp;

cout << "Use a potion to recover 10 health? (y/n) ";

cout << "You have " << getPot() << " potions" << endl;

cin >> rsp;

rsp = tolower(rsp);

while(rsp!='y'){

if(tolower(rsp=='n'))

break;

cout << "Invalid response. Please enter a valid response: ";

cout << endl;

cin >> rsp;

cont = tolower(rsp);

}

if(rsp=='y'){

usePot();

cout << "You used a potion!" << endl;

cout << "Your HP is now " << ally << endl;

}

}

if(enemy>0){

cout << "Continue? (y/n)" << endl;

cin >> cont;

while(cont!='y'){

if(tolower(cont=='n')){

cout << "You have exited the game" << endl;

break;

}

cout << "Invalid response. Please enter a valid response: ";

cout << endl;

cin >> cont;

cont = tolower(cont);

}

}

if(ally==0){

cout << "You ran out of HP. You lose!" << endl;

break;

}

} while(cont=='y' && enemy>0);

cout << endl;

if(enemy == 0){

cout << "The goblin has fainted!" << endl;

gainPot();

cout << "You leveled up! Your attack power has ";

cout << "increased by 5!" << endl;

setAAtk(aAtk+=5);

Lv add;

add.setAHP(ally);

add.setAAtk(aAtk);

add.setPot(pots);

l.push\_back(add);

cout << endl;

cout << "Viewing status progression..." << endl;

prntL();

tier[1]=true;

}

else{

cout << "Game Over! You died!" << endl;

}

}

}

void Lv::l2(){

if(tier[1]!=true)

return;

assert(mon.front() == "Zombie");

mon.pop();

assert(monster.size()==7);

monster.pop();

cout << "Entered Level Two! " << endl;

char reply;

cout << "A zombie has appeared! " << endl;

cout << "Ready to battle? (y/n)" << endl;

cin >> reply;

reply = tolower(reply);

while((reply!='y')){

if(tolower(reply=='n')){

cout << "You have exited the game" << endl;

break;

}

cout << "Invalid response. Please enter a valid response: " << endl;

cin >> reply;

reply=tolower(reply);

}

if(reply == 'y'){

cout << endl;

setEHP(15); //Enemy HP

hp.insert(15);

setAHP(getAHP()); // Ally HP

char cont=' ';

cout << "Engaged in combat." << endl;

setAAtk(getAAtk()); //Able to attack up to 10 damage

setEAtk(5); //Able to attack up to 3 damage

do{

eDmg(); //Enemy taking damage

cout << "You attacked for " << atk1 << " damage" << endl;

aDmg(); //Ally taking damage

cout << "The zombie attacked for " << atk2 << " damage" << endl;

cout << "Your HP: " << ally << endl;

cout << "Zombie's HP: " << enemy << endl;

if(ally < 100 && pots>0){

char rsp;

cout << "Use a potion to recover 10 health? (y/n) ";

cout << "You have " << getPot() << " potions" << endl;

cin >> rsp;

rsp = tolower(rsp);

while(rsp!='y'){

if(tolower(rsp=='n'))

break;

cout << "Invalid response. Please enter a valid response: ";

cout << endl;

cin >> rsp;

cont = tolower(rsp);

}

if(rsp=='y'){

usePot();

cout << "You used a potion!" << endl;

cout << "Your HP is now " << ally << endl;

}

}

if(enemy>0){

cout << "Continue? (y/n)" << endl;

cin >> cont;

while(cont!='y'){

if(tolower(cont=='n')){

cout << "You have exited the game" << endl;

break;

}

cout << "Invalid response. Please enter a valid response: ";

cout << endl;

cin >> cont;

cont = tolower(cont);

}

}

if(ally==0){

cout << "You ran out of HP. You lose!" << endl;

break;

}

} while(cont=='y' && enemy>0);

cout << endl;

if(enemy == 0){

cout << "The zombie has fainted!" << endl;

gainPot();

cout << "You leveled up! Your attack power has ";

cout << "increased by 5!" << endl;

setAAtk(aAtk+=5);

Lv add;

add.setAHP(ally);

add.setAAtk(aAtk);

add.setPot(pots);

l.push\_back(add);

cout << endl;

cout << "Viewing status progression..." << endl;

prntL();

tier[2]=true;

}

else{

cout << "Game Over! You died!" << endl;

}

}

}

void Lv::l3(){

if(tier[2]!=true)

return;

assert(mon.front() == "Sand Rat");

mon.pop();

assert(monster.size()==6);

monster.pop();

cout << "Entered Level Three! " << endl;

char reply;

cout << "A wild sand rat has appeared! " << endl;

cout << "Ready to battle? (y/n)" << endl;

cin >> reply;

reply = tolower(reply);

while((reply!='y')){

if(tolower(reply=='n')){

cout << "You have exited the game" << endl;

break;

}

cout << "Invalid response. Please enter a valid response: " << endl;

cin >> reply;

reply=tolower(reply);

}

if(reply == 'y'){

cout << endl;

setEHP(25); //Enemy HP

hp.insert(25);

setAHP(getAHP()); // Ally HP

char cont=' ';

cout << "Engaged in combat." << endl;

setAAtk(getAAtk()); //Able to attack up to 10 damage

setEAtk(7); //Able to attack up to 3 damage

do{

eDmg(); //Enemy taking damage

cout << "You attacked for " << atk1 << " damage" << endl;

aDmg(); //Ally taking damage

cout << "The sand rat attacked for " << atk2 << " damage" << endl;

cout << "Your HP: " << ally << endl;

cout << "Sand Rat's HP: " << enemy << endl;

if(ally < 100 && pots>0){

char rsp;

cout << "Use a potion to recover 10 health? (y/n) ";

cout << "You have " << getPot() << " potions" << endl;

cin >> rsp;

rsp = tolower(rsp);

while(rsp!='y'){

if(tolower(rsp=='n'))

break;

cout << "Invalid response. Please enter a valid response: ";

cout << endl;

cin >> rsp;

cont = tolower(rsp);

}

if(rsp=='y'){

usePot();

cout << "You used a potion!" << endl;

cout << "Your HP is now " << ally << endl;

}

}

if(enemy>0){

cout << "Continue? (y/n)" << endl;

cin >> cont;

while(cont!='y'){

if(tolower(cont=='n')){

cout << "You have exited the game" << endl;

break;

}

cout << "Invalid response. Please enter a valid response: ";

cout << endl;

cin >> cont;

cont = tolower(cont);

}

}

if(ally==0){

cout << "You ran out of HP. You lose!" << endl;

break;

}

} while(cont=='y' && enemy>0);

cout << endl;

if(enemy == 0){

cout << "The goblin has fainted!" << endl;

gainPot();

cout << "You leveled up! Your attack power has ";

cout << "increased by 5!" << endl;

setAAtk(aAtk+=5);

Lv add;

add.setAHP(ally);

add.setAAtk(aAtk);

add.setPot(pots);

l.push\_back(add);

cout << endl;

cout << "Viewing status progression..." << endl;

prntL();

tier[3]=true;

}

else{

cout << "Game Over! You died!" << endl;

}

}

}

void Lv::l4(){

if(tier[3]!=true)

return;

assert(mon.front() == "Wolf");

mon.pop();

assert(monster.size()==5);

monster.pop();

cout << "Entered Level Four! " << endl;

char reply;

cout << "A Wolf has appeared! " << endl;

cout << "Ready to battle? (y/n)" << endl;

cin >> reply;

reply = tolower(reply);

while((reply!='y')){

if(tolower(reply=='n')){

cout << "You have exited the game" << endl;

break;

}

cout << "Invalid response. Please enter a valid response: " << endl;

cin >> reply;

reply=tolower(reply);

}

if(reply == 'y'){

cout << endl;

setEHP(40); //Enemy HP

hp.insert(40);

setAHP(getAHP()); // Ally HP

char cont=' ';

cout << "Engaged in combat." << endl;

setAAtk(getAAtk()); //Able to attack up to 10 damage

setEAtk(9); //Able to attack up to 3 damage

do{

eDmg(); //Enemy taking damage

cout << "You attacked for " << atk1 << " damage" << endl;

aDmg(); //Ally taking damage

cout << "The Wolf attacked for " << atk2 << " damage" << endl;

cout << "Your HP: " << ally << endl;

cout << "Wolf's HP: " << enemy << endl;

if(ally < 100 && pots>0){

char rsp;

cout << "Use a potion to recover 10 health? (y/n) ";

cout << "You have " << getPot() << " potions" << endl;

cin >> rsp;

rsp = tolower(rsp);

while(rsp!='y'){

if(tolower(rsp=='n'))

break;

cout << "Invalid response. Please enter a valid response: ";

cout << endl;

cin >> rsp;

cont = tolower(rsp);

}

if(rsp=='y'){

usePot();

cout << "You used a potion!" << endl;

cout << "Your HP is now " << ally << endl;

}

}

if(enemy>0){

cout << "Continue? (y/n)" << endl;

cin >> cont;

while(cont!='y'){

if(tolower(cont=='n')){

cout << "You have exited the game" << endl;

break;

}

cout << "Invalid response. Please enter a valid response: ";

cout << endl;

cin >> cont;

cont = tolower(cont);

}

}

if(ally==0){

cout << "You ran out of HP. You lose!" << endl;

break;

}

} while(cont=='y' && enemy>0);

cout << endl;

if(enemy == 0){

cout << "The Wolf has fainted!" << endl;

gainPot();

cout << "You leveled up! Your attack power has ";

cout << "increased by 5!" << endl;

setAAtk(aAtk+=5);

Lv add;

add.setAHP(ally);

add.setAAtk(aAtk);

add.setPot(pots);

l.push\_back(add);

cout << endl;

cout << "Viewing status progression..." << endl;

prntL();

tier[4]=true;

}

else{

cout << "Game Over! You died!" << endl;

}

}

}

void Lv::l5(){

if(tier[4]!=true)

return;

assert(mon.front() == "Tiger");

mon.pop();

assert(monster.size()==4);

monster.pop();

cout << "Entered Level Five! " << endl;

char reply;

cout << "A wild tiger has appeared! " << endl;

cout << "Ready to battle? (y/n)" << endl;

cin >> reply;

reply = tolower(reply);

while((reply!='y')){

if(tolower(reply=='n')){

cout << "You have exited the game" << endl;

break;

}

cout << "Invalid response. Please enter a valid response: " << endl;

cin >> reply;

reply=tolower(reply);

}

if(reply == 'y'){

cout << endl;

setEHP(50); //Enemy HP

hp.insert(50);

setAHP(getAHP()); // Ally HP

char cont=' ';

cout << "Engaged in combat." << endl;

setAAtk(getAAtk()); //Able to attack up to 10 damage

setEAtk(11); //Able to attack up to 3 damage

do{

eDmg(); //Enemy taking damage

cout << "You attacked for " << atk1 << " damage" << endl;

aDmg(); //Ally taking damage

cout << "The tiger attacked for " << atk2 << " damage" << endl;

cout << "Your HP: " << ally << endl;

cout << "Tiger's HP: " << enemy << endl;

if(ally < 100 && pots>0){

char rsp;

cout << "Use a potion to recover 10 health? (y/n) ";

cout << "You have " << getPot() << " potions" << endl;

cin >> rsp;

rsp = tolower(rsp);

while(rsp!='y'){

if(tolower(rsp=='n'))

break;

cout << "Invalid response. Please enter a valid response: ";

cout << endl;

cin >> rsp;

cont = tolower(rsp);

}

if(rsp=='y'){

usePot();

cout << "You used a potion!" << endl;

cout << "Your HP is now " << ally << endl;

}

}

if(enemy>0){

cout << "Continue? (y/n)" << endl;

cin >> cont;

while(cont!='y'){

if(tolower(cont=='n')){

cout << "You have exited the game" << endl;

break;

}

cout << "Invalid response. Please enter a valid response: ";

cout << endl;

cin >> cont;

cont = tolower(cont);

}

}

if(ally==0){

cout << "You ran out of HP. You lose!" << endl;

break;

}

} while(cont=='y' && enemy>0);

cout << endl;

if(enemy == 0){

cout << "The tiger has fainted!" << endl;

gainPot();

cout << "You leveled up! Your attack power has ";

cout << "increased by 5!" << endl;

setAAtk(aAtk+=5);

Lv add;

add.setAHP(ally);

add.setAAtk(aAtk);

add.setPot(pots);

l.push\_back(add);

cout << endl;

cout << "Viewing your status progression..." << endl;

prntL();

tier[5]=true;

}

else{

cout << "Game Over! You died!" << endl;

}

}

}

void Lv::l6(){

if(tier[5]!=true)

return;

assert(mon.front() == "Boar");

mon.pop();

assert(monster.size()==3);

monster.pop();

cout << "Entered Level Six! " << endl;

char reply;

cout << "A wild boar has appeared! " << endl;

cout << "Ready to battle? (y/n)" << endl;

cin >> reply;

reply = tolower(reply);

while((reply!='y')){

if(tolower(reply=='n')){

cout << "You have exited the game" << endl;

break;

}

cout << "Invalid response. Please enter a valid response: " << endl;

cin >> reply;

reply=tolower(reply);

}

if(reply == 'y'){

cout << endl;

setEHP(67); //Enemy HP

hp.insert(67);

setAHP(getAHP()); // Ally HP

char cont=' ';

cout << "Engaged in combat." << endl;

setAAtk(getAAtk()); //Able to attack up to 10 damage

setEAtk(14); //Able to attack up to 3 damage

do{

eDmg(); //Enemy taking damage

cout << "You attacked for " << atk1 << " damage" << endl;

aDmg(); //Ally taking damage

cout << "The boar attacked for " << atk2 << " damage" << endl;

cout << "Your HP: " << ally << endl;

cout << "Boar's HP: " << enemy << endl;

if(ally < 100 && pots>0){

char rsp;

cout << "Use a potion to recover 10 health? (y/n) ";

cout << "You have " << getPot() << " potions" << endl;

cin >> rsp;

rsp = tolower(rsp);

while(rsp!='y'){

if(tolower(rsp=='n'))

break;

cout << "Invalid response. Please enter a valid response: ";

cout << endl;

cin >> rsp;

cont = tolower(rsp);

}

if(rsp=='y'){

usePot();

cout << "You used a potion!" << endl;

cout << "Your HP is now " << ally << endl;

}

}

if(enemy>0){

cout << "Continue? (y/n)" << endl;

cin >> cont;

while(cont!='y'){

if(tolower(cont=='n')){

cout << "You have exited the game" << endl;

break;

}

cout << "Invalid response. Please enter a valid response: ";

cout << endl;

cin >> cont;

cont = tolower(cont);

}

}

if(ally==0){

cout << "You ran out of HP. You lose!" << endl;

break;

}

} while(cont=='y' && enemy>0);

cout << endl;

if(enemy == 0){

cout << "The boar has fainted!" << endl;

gainPot();

cout << "You leveled up! Your attack power has ";

cout << "increased by 5!" << endl;

setAAtk(aAtk+=5);

Lv add;

add.setAHP(ally);

add.setAAtk(aAtk);

add.setPot(pots);

l.push\_back(add);

cout << endl;

cout << "Viewing your status progression..." << endl;

prntL();

tier[6]=true;

}

else{

cout << "Game Over! You died!" << endl;

}

}

}

void Lv::l7(){

if(tier[6]!=true)

return;

assert(mon.front() == "Dragon");

mon.pop();

assert(monster.size()==2);

monster.pop();

cout << "Entered Level Seven! " << endl;

char reply;

cout << "A wild dragon has appeared! " << endl;

cout << "Ready to battle? (y/n)" << endl;

cin >> reply;

reply = tolower(reply);

while((reply!='y')){

if(tolower(reply=='n')){

cout << "You have exited the game" << endl;

break;

}

cout << "Invalid response. Please enter a valid response: " << endl;

cin >> reply;

reply=tolower(reply);

}

if(reply == 'y'){

cout << endl;

setEHP(77); //Enemy HP

hp.insert(77);

setAHP(getAHP()); // Ally HP

char cont=' ';

cout << "Engaged in combat." << endl;

setAAtk(getAAtk()); //Able to attack up to 10 damage

setEAtk(18); //Able to attack up to 3 damage

do{

eDmg(); //Enemy taking damage

cout << "You attacked for " << atk1 << " damage" << endl;

aDmg(); //Ally taking damage

cout << "The dragon attacked for " << atk2 << " damage" << endl;

cout << "Your HP: " << ally << endl;

cout << "Dragon's HP: " << enemy << endl;

if(ally < 100 && pots>0){

char rsp;

cout << "Use a potion to recover 10 health? (y/n) ";

cout << "You have " << getPot() << " potions" << endl;

cin >> rsp;

rsp = tolower(rsp);

while(rsp!='y'){

if(tolower(rsp=='n'))

break;

cout << "Invalid response. Please enter a valid response: ";

cout << endl;

cin >> rsp;

cont = tolower(rsp);

}

if(rsp=='y'){

usePot();

cout << "You used a potion!" << endl;

cout << "Your HP is now " << ally << endl;

}

}

if(enemy>0){

cout << "Continue? (y/n)" << endl;

cin >> cont;

while(cont!='y'){

if(tolower(cont=='n')){

cout << "You have exited the game" << endl;

break;

}

cout << "Invalid response. Please enter a valid response: ";

cout << endl;

cin >> cont;

cont = tolower(cont);

}

}

if(ally==0){

cout << "You ran out of HP. You lose!" << endl;

break;

}

} while(cont=='y' && enemy>0);

cout << endl;

if(enemy == 0){

cout << "The dragon has fainted!" << endl;

gainPot();

cout << "You leveled up! Your attack power has ";

cout << "increased by 5!" << endl;

setAAtk(aAtk+=5);

Lv add;

add.setAHP(ally);

add.setAAtk(aAtk);

add.setPot(pots);

l.push\_back(add);

cout << endl;

cout << "Viewing your status progression..." << endl;

prntL();

tier[7]=true;

}

else{

cout << "Game Over! You died!" << endl;

}

}

}

void Lv::l8(){

if(tier[7]!=true)

return;

assert(mon.front() == "Baboon");

mon.pop();

assert(monster.size()==1);

monster.pop();

cout << "Entered Level Eight! " << endl;

char reply;

cout << "A wild baboon has appeared! " << endl;

cout << "Ready to battle? (y/n)" << endl;

cin >> reply;

reply = tolower(reply);

while((reply!='y')){

if(tolower(reply=='n')){

cout << "You have exited the game" << endl;

break;

}

cout << "Invalid response. Please enter a valid response: " << endl;

cin >> reply;

reply=tolower(reply);

}

if(reply == 'y'){

cout << endl;

setEHP(84); //Enemy HP

hp.insert(84);

setAHP(getAHP()); // Ally HP

char cont=' ';

cout << "Engaged in combat." << endl;

setAAtk(getAAtk()); //Able to attack up to 10 damage

setEAtk(20); //Able to attack up to 3 damage

do{

eDmg(); //Enemy taking damage

cout << "You attacked for " << atk1 << " damage" << endl;

aDmg(); //Ally taking damage

cout << "The baboon attacked for " << atk2 << " damage" << endl;

cout << "Your HP: " << ally << endl;

cout << "Baboon's HP: " << enemy << endl;

if(ally < 100 && pots>0){

char rsp;

cout << "Use a potion to recover 10 health? (y/n) ";

cout << "You have " << getPot() << " potions" << endl;

cin >> rsp;

rsp = tolower(rsp);

while(rsp!='y'){

if(tolower(rsp=='n'))

break;

cout << "Invalid response. Please enter a valid response: ";

cout << endl;

cin >> rsp;

cont = tolower(rsp);

}

if(rsp=='y'){

usePot();

cout << "You used a potion!" << endl;

cout << "Your HP is now " << ally << endl;

}

}

if(enemy>0){

cout << "Continue? (y/n)" << endl;

cin >> cont;

while(cont!='y'){

if(tolower(cont=='n')){

cout << "You have exited the game" << endl;

break;

}

cout << "Invalid response. Please enter a valid response: ";

cout << endl;

cin >> cont;

cont = tolower(cont);

}

}

if(ally==0){

cout << "You ran out of HP. You lose!" << endl;

break;

}

} while(cont=='y' && enemy>0);

cout << endl;

if(enemy == 0){

cout << "The baboon has fainted!" << endl;

gainPot();

cout << "You leveled up! Your attack power has ";

cout << "increased by 5!" << endl;

setAAtk(aAtk+=5);

Lv add;

add.setAHP(ally);

add.setAAtk(aAtk);

add.setPot(pots);

l.push\_back(add);

cout << endl;

cout << "Viewing your status progression..." << endl;

prntL();

tier[8]=true;

}

else{

cout << "Game Over! You died!" << endl;

}

}

}

void Lv::outWin(){

ofstream outfile;

int count = 0;

for(map <const int,bool>::iterator it = tier.begin();it!=tier.end();++it){

if(it->second == true)

count++;

}

outfile.open("players.txt",ios\_base::app);

outfile << name << " " << count << endl;

}

void Lv::rank(){

ifstream infile;

infile.open("players.txt");

string x;

int i=0;

cout << endl;

cout << "Game History and scores" << endl;

while(!infile.eof()){

getline(infile,x);

cout << x << endl;

}

infile.close();

}

**STATUS.H**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

/\*

\* File: status.h

\* Author: Bryan

\*

\* Created on October 27, 2017, 9:00 PM

\*/

#ifndef STATUS\_H

#define STATUS\_H

#include <iostream>

#include "login.h"

#include "mergeSort.h"

using namespace std;

class Status{

protected:

int ally, enemy, pots, aAtk, eAtk, atk1, atk2;

queue<string> mon; //monster

stack<string> monster; //monster's attack

public:

Status(){

ally=0;

enemy=0;

pots=0;

aAtk=0;

eAtk=0;

}

void setEHP(int h){

enemy=h;

}

int getEHP(){

return enemy;

}

void setEAtk(int a){

eAtk=a;

}

void eDmg(){

atk1=rand()%aAtk+1;

enemy-=atk1;

if(enemy<0) enemy=0;

}

void setAHP(int h){

ally=h;

}

int getAHP(){

return ally;

}

void setAAtk(int a){

aAtk=a;

}

int getAAtk(){

return aAtk;

}

void aDmg(){

atk2=rand()%eAtk+1;

ally-=atk2;

if(ally<0) ally=0;

}

void setPot(int p){

pots=p;

}

void gainPot(){

int p = rand()%10;

if(p<5){ cout << "Enemy dropped a potion!" << endl; pots++;}

else cout << "Enemy dropped nothing" << endl;

}

void usePot(){

ally+=10;

pots--;

if(ally>100) ally=100;

}

int getPot(){

return pots;

}

};

#endif /\* STATUS\_H \*/

**LOGIN.H**

#include <string>

#include <fstream>

#include <iostream>

using namespace std;

#ifndef LOGIN\_H

#define LOGIN\_H

class Login{

protected:

string name;

unsigned int password;

unsigned int pw;

public:

Login();

void read();

string getName(){

return name;

}

};

#endif /\* LOGIN\_H \*/

**LOGIN.CPP**

#include <cstdlib>

#include <fstream>

#include <sstream>

#include "login.h"

#include "GeneralHashFunctions.h"

Login::Login(){

}

void Login::read(){

string hash;

cout << "Enter (new) username: ";

cin >> name;

cout << "Enter (new) password: ";

cin >> hash;

pw = RSHash(hash);

string test;

stringstream s;

string x;

string file = name;

ifstream look;

ofstream write, players;

if(look){

look.open(file.c\_str());

getline(look,x);

if(x.empty()){

cout << "Account not found. Creating account..." << endl;

write.open(file.c\_str());

write << pw << endl;

write.close();

players.close();

}

else{

s << x;

s >> password;

if(pw == password){

cout << "You have logged in." << endl << endl;

}

else{

while(pw != password){

cout << "Invalid password. Re-enter your password ";

cout << "or enter -1 to exit." << endl;

cin >> test;

if(test == "-1")

exit(0);

pw = RSHash(test);

cout << endl << endl;

if(pw == password)

cout << "You have logged in" << endl;

}

}

look.close();

}

}

}

**MERGESORT.H**

#ifndef MERGESORT\_H

#define MERGESORT\_H

class MergeSort{

protected:

struct Data{

int size;

int \*sortit;

int \*working;

};

Data\* a;

public:

MergeSort();

void mergeSort(int,int);

void merge(int,int,int);

~MergeSort();

void fill(int);

void print(int);

};

#endif /\* MERGESORT\_H \*/

**MERGESTORT.CPP**

#include <cstdlib>

#include <iostream>

#include "mergeSort.h"

using namespace std;

MergeSort::MergeSort(){

a->size=0;

a->sortit=NULL;

a->working=NULL;

}

void MergeSort::merge(int beg, int nlow, int nhigh){

//Create a merged array

int span=nhigh-beg; //Span the range to merge

int cntl=beg,cnth=nlow;//Independent counters to merge the split

//Fill the array

for(int i=0;i<span;i++){

if(cntl==nlow){//Low done fill with the higher end of array

a->working[i]=a->sortit[cnth++];

}else if(cnth==nhigh){//High done fill with lower end of array

a->working[i]=a->sortit[cntl++];

}else if(a->sortit[cntl]<a->sortit[cnth]){

a->working[i]=a->sortit[cntl++];//Fill with lower end

}else{

a->working[i]=a->sortit[cnth++];//Fill with higher end

}

}

//Copy back from the working array to the sorted array

for(int i=0;i<span;i++){

a->sortit[beg+i]=a->working[i];

}

}

void MergeSort::mergeSort(int beg, int end){

int center=(beg+end)/2;//Split the task down the middle

if((center-beg)>1)mergeSort(beg,center);//Got to be an array to split

if((end-center)>1)mergeSort(center,end);//Got to be an array to split

merge(beg,center,end);//Merge the sorted arrays into 1 larger sorted array

}

void MergeSort::fill(int n){

//Allocate memory

Data \*data=new Data;

data->size=n;

data->sortit=new int[n];

data->working=new int[n];

for(int i=0;i<n;i++){

data->sortit[i]=rand()%100+1;

}

a = data;

}

void MergeSort::print(int perLine){

//First print the unsorted array

cout<<endl;

for(int i=0;i<a->size;i++){

cout<<a->sortit[i]<<" ";

if(i%perLine==(perLine-1))cout<<endl;

}

cout<<endl;

}

MergeSort::~MergeSort(){

delete[] a->sortit;

delete[] a->working;

delete a;

}

**GRAPH.H**

#ifndef GRAPH\_H

#define GRAPH\_H

#include<iostream>

#include <list>

using namespace std;

class Graph{

int V;

list<int> \*adj;

public:

Graph(int);

void addEdge(int, int);

void BFS(int);

};

#endif /\* GRAPH\_H \*/

**GrAPH.CPP**

#include "Graph.h"

Graph::Graph(int V){

this->V = V;

adj = new list<int>[V];

}

void Graph::addEdge(int v, int w)

{

adj[v].push\_back(w);

}

void Graph::BFS(int s)

{

bool \*visited = new bool[V];

for(int i = 0; i < V; i++)

visited[i] = false;

list<int> queue;

visited[s] = true;

queue.push\_back(s);

list<int>::iterator i;

while(!queue.empty())

{

s = queue.front();

cout << s << " ";

queue.pop\_front();

for (i = adj[s].begin(); i != adj[s].end(); ++i)

{

if (!visited[\*i])

{

visited[\*i] = true;

queue.push\_back(\*i);

}

}

}

}

**TREE.H**

#include <cstdlib>

#ifndef TREE\_H

#define TREE\_H

struct Node

{

int data;

struct Node\* left;

struct Node\* right;

};

struct Node\* newNode(int data)

{

struct Node\* node = new struct Node;

node->data = data;

node->left = NULL;

node->right = NULL;

return(node);

}

#endif /\* TREE\_H \*/