****

**SUBMITTED BY:**

UMAIR AFZAL 21F-9612

NAYYAR ABBASI 21F-9173

**OPERATING SYSTEMS**

**SEMESTER PROJECT**

**CODE:**

#include <iostream>

#include <fstream>

#include <pthread.h>

#include <string>

#include <cstdlib>

#include <limits>

#include <semaphore.h>

#include <unistd.h>

#include <cstring>

using namespace std;

sem\_t sema;

enum MarbleColor { RED, YELLOW, BLUE, BLACK };

enum PotionTile { HEALTH, STRENGTH, INVISIBILITY, SPEED };

enum DeskBoard { DESK, BOARD };

class Player {

private:

string name;

int score;

int redMarbles;

int yellowMarbles;

int blueMarbles;

int blackMarbles;

bool potionTiles[4];

DeskBoard deskBoard;

int pipefd[2];

public:

Player(const string& playerName) {

name = playerName;

score = 0;

redMarbles = 0;

yellowMarbles = 0;

blueMarbles = 0;

blackMarbles = 0;

for (int i = 0; i < 4; ++i) {

potionTiles[i] = false;

}

deskBoard = DESK;

}

string getPlayerName() {

return name;

}

int getPlayerScore() {

return score;

}

void setYellowMarbles(int yellowMs) {

yellowMarbles = yellowMs;

}

void setBlueMarbles(int blueMs) {

blueMarbles = blueMs;

}

void setRedMarbles(int redMs) {

redMarbles = redMs;

}

void setBlackMarbles(int blackMs) {

blackMarbles = blackMs;

}

void drawMarble(unsigned int seed) {

int colorIndex = rand() % 4;

switch (colorIndex) {

case 0: redMarbles++; break;

case 1: yellowMarbles++; break;

case 2: blueMarbles++; break;

case 3: blackMarbles++; break;

default: break;

}

}

void displayInventory() {

cout << "Player, " << name << "'s Inventory:\n";

cout << "Score: " << score << "\n";

cout << "+-----------------+\n";

cout << "| Red Marbles: | " << redMarbles << "\n";

cout << "| Yellow Marbles: | " << yellowMarbles << "\n";

cout << "| Blue Marbles: | " << blueMarbles << "\n";

cout << "| Black Marbles: | " << blackMarbles << "\n";

cout << "+-----------------+\n";

cout << "Potion Tiles:\n";

for (int i = 0; i < 4; ++i) {

cout << "Potion Tile " << i + 1 << ": " << (potionTiles[i] ? "Collected" : "Not collected") << "\n";

}

cout << "Desk board: " << (deskBoard == DESK ? "Desk" : "Board") << "\n\n";

}

void craftPotion() {

sem\_wait(&sema);

if (redMarbles >= 1 && yellowMarbles >= 1 && blueMarbles >= 1 && blackMarbles >= 1) {

score++;

redMarbles--;

yellowMarbles--;

blueMarbles--;

blackMarbles--;

int potionIndex = rand() % 4;

potionTiles[potionIndex] = true;

cout << "Player " << name << " crafted a potion!\n";

} else {

cout << "Not enough marbles to craft a potion.\n";

}

sem\_post(&sema);

}

void savePlayerDataToFile() {

sem\_wait(&sema);

ofstream file("player\_data.txt");

if (file.is\_open()) {

file << "Player Name: " << name << "\n";

file << "Score: " << score << "\n";

write(pipefd[1], name.c\_str(), strlen(name.c\_str()) + 1);

file.close();

close(pipefd[1]);

} else {

cout << "Unable to save player data.\n";

}

sem\_post(&sema);

}

void loadPlayerDataFromFile() {

char buffer[50];

sem\_wait(&sema);

ifstream file("player\_data.txt");

if (file.is\_open()) {

string playerName;

int loadedScore;

file >> playerName;

file >> loadedScore;

name = playerName;

score = loadedScore;

file.close();

} else {

cout << "Unable to load player data.\n";

}

sem\_post(&sema);

}

};

class UI {

public:

static void clearScreen() {

// Use tput to set the background color to black

system("tput setab 0");

cout << "\033[2J\033[1;1H";

}

static void waitForEnter() {

cout << "Press Enter to continue...";

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cin.get();

}

static int getMenuChoice() {

int choice;

cout << "Choose an action:\n";

cout << "1. Draw Marble\n";

cout << "2. Craft Potion\n";

cout << "3. End Turn\n";

while (!(cin >> choice)) {

cout << "Invalid input. Please enter a number.\n";

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

}

return choice;

}

};

void\* playGame(void\* args) {

Player\* player = static\_cast<Player\*>(args);

unsigned int seed = 42;

const int maxTurns = 10;

Player\* currentPlayer = player;

UI::waitForEnter(); // Wait for the user to press Enter to start the game

for (int turn = 1; turn <= maxTurns; ++turn) {

currentPlayer->drawMarble(seed);

UI::clearScreen();

currentPlayer->displayInventory();

int choice = UI::getMenuChoice();

switch (choice) {

case 1:

currentPlayer->drawMarble(seed);

break;

case 2:

currentPlayer->craftPotion();

break;

case 3:

currentPlayer = (currentPlayer == &player[0]) ? &player[1] : &player[0];

break;

default:

cout << "Invalid choice. Try again.\n";

--turn;

}

}

}

void\* displayInventory(void\* args) {

Player\* player = static\_cast<Player\*>(args);

UI::clearScreen();

player->displayInventory();

UI::waitForEnter();

}

void\* savePlayerDataToFile(void\* args) {

Player\* player = static\_cast<Player\*>(args);

player->savePlayerDataToFile();

}

void\* loadPlayerDataFromFile(void\* args) {

Player\* player = static\_cast<Player\*>(args);

player->loadPlayerDataFromFile();

}

int main() {

sem\_init(&sema, 0, 1);

unsigned int seed = 42;

string p1Name, p2Name;

// Clear the screen

UI::clearScreen();

cout << "Welcome to the Potion Tiles Game!\n";

cout << "Made by Umair Afzal and Nayyer Maqsood\n";

UI::waitForEnter(); // Wait for the user to press Enter to continue

cout << "Enter the Player 1 Name : ";

getline(cin, p1Name);

cout << "Enter the Player 2 Name : ";

getline(cin, p2Name);

Player playerNum[2] = { Player(p1Name), Player(p2Name) };

playerNum[0].setYellowMarbles(0);

playerNum[1].setYellowMarbles(0);

pthread\_t displayInventoryP1, displayInventoryP2;

// Clear the screen again

UI::clearScreen();

cout << "Player " << playerNum[0].getPlayerName() << "'s Initial Inventory:\n";

pthread\_create(&displayInventoryP1, NULL, displayInventory, (void\*)&playerNum[0]);

pthread\_join(displayInventoryP1, NULL);

// Clear the screen

UI::clearScreen();

cout << "Player " << playerNum[1].getPlayerName() << "'s Initial Inventory:\n";

pthread\_create(&displayInventoryP2, NULL, displayInventory, (void\*)&playerNum[1]);

pthread\_join(displayInventoryP2, NULL);

pthread\_t saveDataP1, loadDataP2;

pthread\_create(&saveDataP1, NULL, savePlayerDataToFile, (void\*)&playerNum[0]);

pthread\_create(&loadDataP2, NULL, loadPlayerDataFromFile, (void\*)&playerNum[1]);

pthread\_join(saveDataP1, NULL);

pthread\_join(loadDataP2, NULL);

pthread\_t gameThread;

pthread\_create(&gameThread, NULL, playGame, (void\*)&playerNum);

pthread\_join(gameThread, NULL);

// Clear the screen

UI::clearScreen();

cout << "Game Over!\n";

cout << "Scores:\n";

cout << playerNum[0].getPlayerName() << ": " << playerNum[0].getPlayerScore() << "\n";

cout << playerNum[1].getPlayerName() << ": " << playerNum[1].getPlayerScore() << "\n";

if (playerNum[0].getPlayerScore() > playerNum[1].getPlayerScore()) {

cout << "The winner is: " << playerNum[0].getPlayerName() << "\n";

} else if (playerNum[0].getPlayerScore() < playerNum[1].getPlayerScore()) {

cout << "The winner is: " << playerNum[1].getPlayerName() << "\n";

} else {

cout << "It's a tie!\n";

}

sem\_destroy(&sema);

return 0;

}

**OUTPUT:**

A screenshot of a computer

Description automatically generated

1. When I press continue it demands the player’s names.

A screenshot of a computer

Description automatically generated

1. After entering the name press enter it will display both player’s initial inventory with empty space and then press again continue to draw the tables of both player's inventory after the press continue again then the game will start enjoy!!

A screenshot of a computer

Description automatically generated

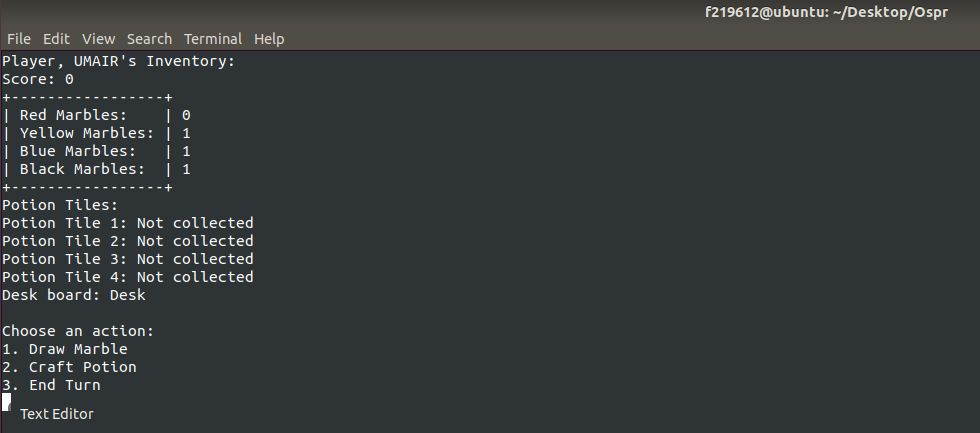
A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. When I press 1 the game draw the marbles and drawing it randomly as seen below.



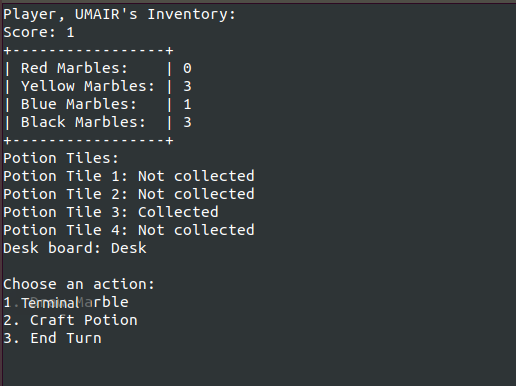
A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. When I press 2 it craft a portion and randomly collected and score will be incremented.



1. When I press 3 the player changed the turn and another play start it.

