

INSTITUTO POLITECNICO DE BEJA

Escola superior de Tecnologia e Gestão

Licenciatura de Engenharia Informática

Sistemas Operativos

Trabalho Prático

Relatório Técnico do

Jogo “***Hunter Hallow***”

Rute Figueiredo, nº 15408

Tiago Campos, nº 13953

25 de Novembro de 2017

***Índice***

[Introdução 3](#_Toc499570081)

[Mecânica do Jogo 4](#_Toc499570082)

[Desenvolvimento do Jogo 5](#_Toc499570083)

[Conclusão 10](#_Toc499570084)

[Referencias 11](#_Toc499570085)

[Anexos 12](#_Toc499570086)

# ***Introdução***

O jogo “Hunter Hallow” é desenvolvido no âmbito da disciplina de Sistemas Operativos, o professor da disciplina propôs desenvolver um jogo em C em que o tema do jogo fica a cargo dos grupos.

O jogo é inspirado na série Monster Hunters, este trabalho tem vários monstros pré-definidos, vários itens sendo estes carregados de um ficheiro binário e só podendo ser apanhados nas salas depois de derrotar o monstro que esta nessas salas assim como os tesouros também só podem ser apanhados depois de derrotar os monstros.

Para derrotar o monstro final o jogador terá de derrotar todos os monstros antes de chegar à arena final pois sem apanhar os itens não vai ter a força necessária para derrotar o rei dos monstros.

# ***Mecânica do Jogo***

* 1. ***História do Jogo***

O jogo trata-se de um jogo de aventura e ação, o jogo é inspirado em alguns jogos do género, o jogo trata-se de um Soldado que anda em busca de glória, reputação e ser o melhor.

Este Soldado pertence a uma organização chamada Hunter Hallow esta organização tem como objetivo salvar a humanidade dos monstros e caçar esses monstros para impedir que estes se espalhem pelo planeta.

* 1. ***Mapa do jogo***

O mapa do jogo representa uma pequena vila que tem uma armadura e uma espada lendárias, esta vila foi tomada por monstros selvagens sendo um deles o rei dos monstros.

* 1. ***Objetivos do jogo***

Os objetivos que o jogador vai ter de alcançar são eliminar os quatro monstros de elite, derrotar o rei dos monstros para poder recuperar os itens lendários e salvar a vila dos monstros.

* 1. ***Super User***

O super user é um utilizador especial que tem todos os status ao máximo e não perde com nenhum dos monstros, este só pode ser ativado por um dos seguintes comandos “SU”, “su”, “Super User” ou “super user”.

***Desenvolvimento do Jogo***

* + 1. *Definição das Estruturas*

**Player**

struct Player {

char namePlayer[MAX\_NAME];

int energyPlayer;

int damage;

int critic;

int cellPlayer;

int itemPlayer;

int treasurePlayer;

};

**Monster**

struct Monster {

char nameMosnter[MAX\_NAME];

int lifeMonster;

int damageMonster;

int criticMonster;

int cellMonster;

int itemMonster;

int treasureMonster;

int nMonsters;

};

**Item**

struct Item {

int CodItem;

char NameItem[MAX\_NAME];

int DamageItem;

int CriticItem;

int PositionItem;

int LifeItem;

};

**Treasure**

struct Tresure {

int CodTresure;

char NameTreasure[MAX\_NAME];

int Gold;

int PositionTresure;

};

**Cell**

struct Cell {

int north;

int south;

int east;

int west;

int up;

int down;

int treasureCell;

int itemCell;

char descriptionCell[MAX\_DESCRIPTION\_CELL];

};

**Map**

struct Map

{

struct Cell cell[MAX\_CELLS];

struct Item item[MAX\_ITEMS];

struct Tresure treasure[MAX\_TREASURE];

int nCells;

};

**SaveGame**

struct SaveGame {

struct Monster saveMonster;

};

* + 1. *Inicialização do Jogador e do Monstro*
* A inicialização do Jogador é feita através do método *InsertPlayer*, que pede o nome do jogador e inicializa-o com esse nome e as características que estão pré-definidas para o início do jogo (Energy=100, Damage=50, Critic=10). No caso do jogador inserir um dos comandos de SuperUser em vez do seu nome, o jogador recebe valores muito mais generosos que vão facilitar bastante o jogo (Energy=10000, Damage=1000, Critic=200, Item=Gold Armor + Sword, Treasure=Cofre Grande).
* A inicialização do Monstro é feita através da função InicializeMonster. São inicializados 5 monstros que permanecem sempre nas mesmas salas e outros 5 que se movimentam de forma aleatória pelo mapa. Um dos monstros é o que vai definir se o jogo é ganho ou não, porque é apenas depois deste ser derrotado que o jogador ganha.
  + 1. *Inicialização do Mapa no código*

Para a inicialização do Mapa inicialmente foi criada uma função (*InitMap*) que inicializasse o mapa estando o mesmo definido nessa mesma função.

* + 1. *Inicialização dos Itens e dos Tesouros no código*
* Inicialmente foi criada uma função que inicializa os itens e os tesouros a partir do código – *InitItemPlusTreasure*
  1. ***Segunda Fase do Desenvolvimento***
     1. *Inicialização do Mapa através da leitura do Ficheiro*

Foi criada uma função (*LoadMapFromFile*) que vai descarregar as características do mapa a um ficheiro com o nome “map.txt” com a seguinte estrutura:

*north south west east up down item treasure*

*descriptionCell*

* + 1. *Inicialização dos Itens e dos Tesouros a partir de um Ficheiro*

Foram criadas duas funções para carregarem os itens e os tesouros a partir de ficheiros “.txt” – *InitObejectItem* e *InitObejectTreasure*

Após isso ainda foi criada outra função para carregar os itens a partir de um ficheiro binário com o nome “objectos.dat” - *InitObejectItemBin*

* + 1. *Desenvolvimento da Função para fazer o Jogador andar no Mapa*

A função criada para definir o movimento do jogador no mapa é a função *PlayerWalk.* Esta função apresenta ao jogador uma lista de opções:

1 – Norte

2 – Sul

3 – Oeste

4 – Este

5 – Subir

6 – Descer

7 – Save

8 – Menu Principal

Consoante o que o jogador escolher pode-se movimentar pelo mapa (caso exista uma forma de passagem na direção escolhida), gravar o jogo ou ir para o menu principal.

* + 1. *Desenvolvimento da Função para fazer os Monstros andarem no Mapa*

Para este objetivo foi criada a função *MonstersWalk,* que escolhe um dos monstros, que se movimentam pelas salas, aleatoriamente e escolhe uma direção aleatória também para este se movimentar. Após isso, é verificado se o monstro se pode movimentar nessa direção, e caso o player seja super user ainda escreve na consola onde se encontra o monstro.

* + 1. *Desenvolvimento da Função para a Batalha*

Para a batalha entre o player e qualquer um dos monstros foi desenvolvida a função *Battle* que vai sortear valores para o ataque crítico do Player e do Monstro que vai depois modificar o valor de ataque de cada um deles. É ainda sorteado se o Player e o Monster atacam ou falham os seus ataques. O Player vai sendo informado sobre todos os momentos de ataque, e quando a energia do Monstro for menor ou igual a zero a luta acaba e o jogador recolhe o item e o tesouro que estejam nessa sala.

* 1. ***Terceira Fase do Desenvolvimento***
     1. *Permitir o Jogador apanhar os Itens e os Tesouros*

Para permitir que o jogador consiga apanhar os itens que estão na sala depois de derrotar um dos monstros, para isso é usado um if que vê se o monstro tem vida se não o jogador apanha o item que esta na sala assim como o tesouro.

if (monster[i].lifeMonster > 0) printf("Para tua sorte Soldado o Monstro %s", monster[i].nameMosnter);

printf(" acabou de falhar o ataque\n");

}

else {

int itemSelect = pMap->cell[pPlayer->cellPlayer].itemCell;

int treasureSelected = pMap->cell[pPlayer->cellPlayer].treasureCell;

if (itemSelect != -1) {

pPlayer->damage = pPlayer->damage + pMap->item[itemSelect].DamageItem;

pPlayer->critic = pPlayer->critic + pMap->item[itemSelect].CriticItem;

pPlayer->energyPlayer = pPlayer->energyPlayer + pMap->item[itemSelect].LifeItem;

printf("Item Adicionado: %s", pMap->item[itemSelect].NameItem);

printf("Dano Adicionado: %d", pMap->item[itemSelect].DamageItem);

printf("Dano critico Adicionado: %d", pMap->item[itemSelect].CriticItem);

printf("HP Adicionado: %d", pMap->item[itemSelect].LifeItem);

}

if (treasureSelected != -1) {

pPlayer->treasurePlayer = pPlayer->treasurePlayer + pMap->treasure[treasureSelected].Gold;

printf("\n");

printf("Tesouro encontrado: %s\n", pMap->treasure[treasureSelected].NameTreasure);

printf("Gold Adicionado: %d\n", pMap->treasure[treasureSelected].Gold);

}

}

* + 1. *Implementar o Save e o Load*

O save e o load são duas funções que usam o mesmo ficheiro binário, o nome deste ficheiro é atribuído pelo utilizador.

No save os dados são carregados para uma estrutura SaveGame que guarda os dados do jogador e dos monstros para depois os escrever no ficheiro binária através da instrução:

fwrite(pPlayer, sizeof(struct Player), 1, f);

fwrite(&save, sizeof(struct SaveGame), 1, f);

Para fazer o load do jogo o utilizador tem de inserir o nome do save que pretende carregar, depois do utilizador inserir o nome do ficheiro o ficheiro é lido e carrega os dados para a as estruturas do jogador e do monstro através dos seguintes comandos:

fread(pPlayer, sizeof(struct Player), 1, f);

fread(&save, sizeof(struct SaveGame), 1, f);

* + 1. *Revisão do Código e Correção de Bugs*

Depois de colocar todas as funções base a funcionar, o jogo foi testado e identificado os bugs para poderem ser corrigidos, tais como problemas com algumas condições na função da batalha e na função de carregar os itens do ficheiro, assim como correções ortográficas dos prints e ainda havia algumas instruções que estavam mal colocadas.

* + 1. *Inicialização dos Itens através de um ficheiro binário*

Tal como pedido no enunciado do trabalho é necessário carregar os itens de um ficheiro binário para isso foi criada uma função que lê um ficheiro txt e o converte para binário, que por sua vez através de uma função que lê o ficheiro binário e carrega para a estrutura os dados dos itens.

void InitObejectItem(struct Map \*pMap);

void InitObejectItemBin(struct Map \*pMap);

* + 1. *Melhorias da API*

Para a API apenas foi utilizado a função de limpar o ecrã que foi dada na aula pelo professor e ainda foram inseridos alguns prints que ajudam o utilizador a ter melhor jogabilidade.

# ***Conclusão***

Ao realizar este jogo foi necessário dividir a sua realização por fazes para facilitar a criação de métodos assim com a gestão de tempo e para facilitar a correção de erros que foram aparecendo ao longo do desenvolvimento do jogo.

Um dos objetivos que se pretendia era poder inserir diferentes mapas e diferentes ficheiros com diferentes tesouros e itens ou apenas adicionar mas salas ou itens e tesouros nos ficheiros. Para isso o código foi construído para ser o mais flexível possível suportando assim diferentes mapas ou itens e tesouros.

Alguns dos problemas e ou bugs surgiram durante a criação da função batalha, da inicialização dos monstros, a função também teve alguns destes problemas pois devido ao grande numero de monstros e ainda o facto de ter de pegar um ficheiro txt e ter de o converter para binário e depois ter de carregar os itens a partir desse ficheiro binário dai ter surgido alguns bugs.

# ***Referencias***

# ***Anexos***

#include "stdafx.h"

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

#include <string.h>

#include "windows.h"

#define MAX\_NAME 20

#define MAX\_MOSNTERS 100

#define MAX\_ITEMS 1000

#define MAX\_TREASURE 1000

#define MAX\_CELLS 200

#define MAX\_DESCRIPTION\_CELL 1000

#define NCELLS 200

#define MAX\_LINE 1000

#define MAX\_FILENAME 30

#define EMPTY \_T( ' ' )

/\*

Estruturas begin

\*/

struct Player {

char namePlayer[MAX\_NAME];

int energyPlayer;

int damage;

int critic;

int cellPlayer;

int itemPlayer;

int treasurePlayer;

};

struct Monster {

char nameMosnter[MAX\_NAME];

int lifeMonster;

int damageMonster;

int criticMonster;

int cellMonster;

int itemMonster;

int treasureMonster;

int nMonsters;

};

struct Item {

int CodItem;

char NameItem[MAX\_NAME];

int DamageItem;

int CriticItem;

int PositionItem;

int LifeItem;

};

struct Tresure {

int CodTresure;

char NameTreasure[MAX\_NAME];

int Gold;

int PositionTresure;

};

struct Cell {

int north;

int south;

int east;

int west;

int up;

int down;

int treasureCell;

int itemCell;

char descriptionCell[MAX\_DESCRIPTION\_CELL];

};

struct Map

{

struct Cell cell[MAX\_CELLS];

struct Item item[MAX\_ITEMS];

struct Tresure treasure[MAX\_TREASURE];

int nCells;

};

struct SaveGame {

struct Monster saveMonster;

};

/\*

Struct END

\*/

void FunctionClear(); // esta função limpa o ecrã

void InsertPlayer(Player \*pPlayer);/\*Funça que permite o utilizador inserir o seu avatar\*/

void PrintPlayer(Player \*pPlayer);/\*mostra os status do avatar no ecrâ\*/

void InicializeMonster(Monster monster[]);// inicializa varios monstros no jogo

void PrintMonster(Monster monster[]);/\*mostra os status do monstro no ecrâ\*/

int InitMap(Cell cells[]);// inicia o mapa do jogo atraves de variáveis fixas

void LoadMapFromFile(Map \*pMap); // carrega o mapa do jogo de um ficheiro txt

void PrintMapFromFile(Map \*pMap); // faz o print de todas as salas do jogo

void InitItemPlusTreasure(Map \*pMap);//inicia os objectos de forma fixa

void InitObejectItem(Map \*pMap); // carrega os objectos de um ficheiro txt e transforma os em ficheiro bin

void InitObejectItemBin(Map \*pMap); // carrega os objectos de um ficheiro bin

void InitObejectTreasure( Map \*pMap); //carrega os tesouros de ficheiro em txt

void PlayerWalk(Player \*pPlayer, Map \*pMap, Monster monster[]);/\*~Função que cria o mapa do jogo\*/

void MonstersWalk(Player \*pPlayer,Map \*pMap, Monster monster[]);/\*Função que permite o monstro se mover pelo mapa sozinho\*/

void Battle(Player \*pPlayer, Map \*pMap, Monster monster[]);// Batalha entre os monstros e o jogador

void EndGame( Player \*pPlayer, Monster monster[], Map \*pMap);/\*Função que determina quando o jogo acaba\*/

void SaveGame(Player \*pPlayer, Monster monster[]); // garda o jogo num ficheiro em binário

void LoadGame(Player \*pPlayer, Monster monster[]); // carrega o jogo de um ficheiro em binário

/\*

---------- Hunter Hallow --------

O main é onde as funções principais são chamados

\*/

int main()

{

struct Player player;

struct Monster monster[MAX\_MOSNTERS];

struct Cell cells[MAX\_CELLS];

struct Map map;

int nCells;

printf("--------------------------------\n");

printf(" HUNTER HALLOW \n");

printf(" First Episode \n");

printf("--------------------------------\n");

printf("\n");

int option = 0;

printf("Pretende iniciar um novo desafio soldado? \n");

printf(" 1 - Novo Jogo 2 - Continuar Jogo 3 - Sair do Jogo\n");

scanf("%d", &option);

switch (option) {

case 1:

InsertPlayer(&player);

PrintPlayer(&player);

InicializeMonster(monster);

//PrintMonster(monster);

break;

case 2:

LoadGame(&player, monster);

PrintPlayer(&player);

break;

case 3:

exit(0);

break;

default:

printf("Soldado estas a dormir e não percebeste as instruções vou explicar de uma maneira mais facil se for possivel\n apenas pode iserir numeros de 1 a 3\n");

break;

}

LoadMapFromFile(&map);

//PrintMapFromFile(&map);

//map.nCells = InitMap(cells);

//InitItemPlusTreasure(&map);

//InitObejectItem(&map);

InitObejectItemBin(&map);

InitObejectTreasure(&map);

while (player.cellPlayer != (map.nCells + 1)) {

PlayerWalk(&player, &map, monster);

MonstersWalk(&player, &map, monster);

Battle(&player, &map, monster);

EndGame(&player, monster, &map);

}

return 0;

}

/\*

Função que tem como objetivo limpar a consola com se fosse um "System("cls")"

esta função foi retirado de um dos documentos de apoio do Professor Luis Garcia

\*/

void FunctionClear() {

\_tsetlocale(LC\_ALL, \_T("Portuguese"));

HANDLE hStdout = GetStdHandle(STD\_OUTPUT\_HANDLE);

CONSOLE\_SCREEN\_BUFFER\_INFO strConsoleInfo;

GetConsoleScreenBufferInfo(hStdout, &strConsoleInfo);

COORD Home = { 0, 0 };

DWORD hWrittenChars;

/\*limpa os caracteres\*/

FillConsoleOutputCharacter(hStdout, EMPTY, strConsoleInfo.dwSize.X \* strConsoleInfo.dwSize.Y,

Home, &hWrittenChars);

/\*limpa a formatação\*/

FillConsoleOutputAttribute(hStdout, strConsoleInfo.wAttributes,

strConsoleInfo.dwSize.X \* strConsoleInfo.dwSize.Y, Home, &hWrittenChars);

SetConsoleCursorPosition(hStdout, Home);

}

/\*

Esta função inicializa o avatar do jogador no jogo,

pondendo tambem definir o modo de jogo e a dificuldade do jogo

\*/

void InsertPlayer(Player \*pPlayer) {

printf("\n");

printf("Soldado insere o teu nome! \n");

scanf("%s", pPlayer->namePlayer);

if ((strcmp(pPlayer->namePlayer, "SU") == 0) || (strcmp(pPlayer->namePlayer, "su") == 0) ||

(strcmp(pPlayer->namePlayer, "super user") == 0) || (strcmp(pPlayer->namePlayer, "Super User") == 0)) {

// selecionar modo de jogo / dificuldade

pPlayer->energyPlayer = 10000;

pPlayer->damage = 1000;

pPlayer->critic = 200;

pPlayer->itemPlayer = 5;

pPlayer->cellPlayer = 0;

pPlayer->treasurePlayer = 5;

}

else {

// selecionar modo de jogo / dificuldade

pPlayer->energyPlayer = 100;

pPlayer->damage = 50;

pPlayer->critic = 10;

pPlayer->itemPlayer = 0;

pPlayer->cellPlayer = 0;

pPlayer->treasurePlayer = 0;

}

}

/\* Mostrar Jogador

Esta funçao tem como função principal mostrar os status do jogador no ecrã

\*/

void PrintPlayer(Player \*pPlayer) {

printf("\n");

printf("Bem vindo Hunter: %s \n", pPlayer->namePlayer);

printf("A tua Vida: %d \n", pPlayer->energyPlayer);

printf("Os teus Items: %d \n", pPlayer->itemPlayer);

printf("A tua localisação: %d \n", pPlayer->cellPlayer);

printf("Dinheiro: %d \n", pPlayer->treasurePlayer);

printf("Estas pronto para o desafio Soldado !!!\n");

}

/\*

Nesta função são inicializados os monstros do jogo

\*/

void InicializeMonster(Monster monster[]) {

int count = 0;

//Monstro Principal - só depois de ser derrotado é que o jogador ganha

monster[0].cellMonster = 13;

strcpy(monster[0].nameMosnter, "LUCIFER");

monster[0].damageMonster = 60;

monster[0].lifeMonster = 1000;

monster[0].criticMonster = 70;

monster[0].itemMonster = 5;

monster[0].treasureMonster = 5;

count++;

//Monstro da cell 2

monster[1].cellMonster = 2;

strcpy(monster[1].nameMosnter, "AZREL");

monster[1].damageMonster = 10;

monster[1].lifeMonster = 100;

monster[1].criticMonster = 10;

monster[1].itemMonster = 1;

monster[1].treasureMonster = 1;

count++;

//Monstro da cell 3

monster[2].cellMonster = 3;

strcpy(monster[2].nameMosnter, "HERCULES");

monster[2].damageMonster = 20;

monster[2].lifeMonster = 200;

monster[2].criticMonster = 20;

monster[2].itemMonster = 2;

monster[2].treasureMonster = 2;

count++;

//Monstro da cell 8

monster[3].cellMonster = 8;

strcpy(monster[3].nameMosnter, "JACLINE");

monster[3].damageMonster = 30;

monster[3].lifeMonster = 300;

monster[3].criticMonster = 30;

monster[3].itemMonster = 3;

monster[3].treasureMonster = 3;

count++;

//Monstro da cell 9

monster[4].cellMonster = 9;

strcpy(monster[4].nameMosnter, "HADES");

monster[4].damageMonster = 40;

monster[4].lifeMonster = 500;

monster[4].criticMonster = 4;

monster[4].itemMonster = 4;

monster[4].treasureMonster = 4;

count++;

//Monstros lv1

monster[5].cellMonster = 10;

strcpy(monster[5].nameMosnter, "ESCLETO 1");

monster[5].damageMonster = 10;

monster[5].lifeMonster = 100;

monster[5].criticMonster = 10;

monster[5].itemMonster = 0;

monster[5].treasureMonster = 1;

count++;

//Monstros lv1

monster[6].cellMonster = 10;

strcpy(monster[6].nameMosnter, "ESCLETO 2");

monster[6].damageMonster = 10;

monster[6].lifeMonster = 100;

monster[6].criticMonster = 10;

monster[6].itemMonster = 0;

monster[6].treasureMonster = 1;

count++;

//Monstros lv1

monster[7].cellMonster = 10;

strcpy(monster[7].nameMosnter, "ESCLETO 3");

monster[7].damageMonster = 10;

monster[7].lifeMonster = 100;

monster[7].criticMonster = 10;

monster[7].itemMonster = 0;

monster[7].treasureMonster = 1;

count++;

//Monstros lv1

monster[8].cellMonster = 10;

strcpy(monster[8].nameMosnter, "ESCLETO 4");

monster[8].damageMonster = 10;

monster[8].lifeMonster = 100;

monster[8].criticMonster = 10;

monster[8].itemMonster = 0;

monster[8].treasureMonster = 1;

count++;

monster[0].nMonsters = count;

}

/\*

Este função vai fazer o print de todos os monstros no jogo

\*/

void PrintMonster(Monster monster[]) {

for (int i = 0; i < monster[0].nMonsters; i++) {

printf("\n");

printf("Monstro %s\n", monster[i].nameMosnter);

printf("Monstro HP: %d\n", monster[i].lifeMonster);

printf("Monstro ATK: %d\n", monster[i].damageMonster);

printf("Monstro CTRITIC: %d\n", monster[i].criticMonster);

printf("Monstro ROOM: %d\n", monster[i].cellMonster);

printf("Monstro ITEM: %d\n", monster[i].itemMonster);

printf("Monstro TREASURE: %d\n", monster[i].treasureMonster);

}

}

/\*

Função que inicializa os items e os tesouros no jogo

\*/

void InitItemPlusTreasure(struct Map \*pMap) {

//item 1

pMap->item[0].CodItem = 1;

pMap->item[0].PositionItem = 2;

strcpy(pMap->item[0].NameItem, "Wood Armor");

pMap->item[0].CriticItem = 0;

pMap->item[0].DamageItem = 0;

pMap->item[0].LifeItem = 0;

//item 2

pMap->item[1].CodItem = 2;

pMap->item[1].PositionItem = 3;

strcpy(pMap->item[1].NameItem, "Wood Sword");

pMap->item[1].CriticItem = 0;

pMap->item[1].DamageItem = 0;

pMap->item[1].LifeItem = 0;

//item 3

pMap->item[2].CodItem = 3;

pMap->item[2].PositionItem = 8;

strcpy(pMap->item[2].NameItem, "Steel Armor");

pMap->item[2].CriticItem = 0;

pMap->item[2].DamageItem = 0;

pMap->item[2].LifeItem = 100;

//item 4

pMap->item[3].CodItem = 4;

pMap->item[3].PositionItem = 9;

strcpy(pMap->item[3].NameItem, "Steel Sword");

pMap->item[3].CriticItem = 0;

pMap->item[3].DamageItem = 0;

pMap->item[3].LifeItem = 0;

//item 5

pMap->item[4].CodItem = 5;

pMap->item[4].PositionItem = 13;

strcpy(pMap->item[4].NameItem, "Gold Armor + Sword");

pMap->item[4].CriticItem = 0;

pMap->item[4].DamageItem = 0;

pMap->item[4].LifeItem = 0;

//Tesouro 1

pMap->treasure[0].CodTresure = 1;

pMap->treasure[0].PositionTresure = 2;

strcpy(pMap->treasure[0].NameTreasure, "Bolsa de Moedas");

pMap->treasure[0].Gold = 500;

//Tesouro 2

pMap->treasure[1].CodTresure = 2;

pMap->treasure[1].PositionTresure = 3;

strcpy(pMap->treasure[1].NameTreasure, "Bau Pequeno");

pMap->treasure[1].Gold = 500;

//Tesouro 3

pMap->treasure[2].CodTresure = 3;

pMap->treasure[2].PositionTresure = 8;

strcpy(pMap->treasure[2].NameTreasure, "Bau Médio");

pMap->treasure[2].Gold = 500;

//Tesouro 4

pMap->treasure[3].CodTresure = 4;

pMap->treasure[3].PositionTresure = 9;

strcpy(pMap->treasure[3].NameTreasure, "Bau Grande");

pMap->treasure[3].Gold = 500;

//Tesouro 5

pMap->treasure[4].CodTresure = 5;

pMap->treasure[4].PositionTresure = 13;

strcpy(pMap->treasure[4].NameTreasure, "Cofre Grande");

pMap->treasure[4].Gold = 500;

}

/\*

Nesta função inicializa os objectos do ficheiro txt e transforma os num ficheiro bin

\*/

void InitObejectItem(struct Map \*pMap) {

FILE \*f, \*f2;

char l[MAX\_LINE];

int line = 1;

int i = 0;

int count = 0;

int id, position, damage, critic, hp;

f = fopen("items.txt", "r");

f2 = fopen("objectos.dat", "w");

while (fgets(l, MAX\_LINE, f) != NULL) {

if (line == 1 && strcmp(l, "\n") != 0) {

sscanf(l, "%d %d %d %d %d", &id, &position, &damage, &critic, &hp);

pMap->item[i].CodItem = id;

pMap->item[i].PositionItem = position;

pMap->item[i].DamageItem = damage;

pMap->item[i].CriticItem = critic;

pMap->item[i].LifeItem = hp;

line++;

}

else if (strcmp(l, "\n") != 0 && l != " ") {

strcpy(pMap->item[i].NameItem, l);

line = 1;

i++;

count++;

}

else {

//do Nothing

}

}

for (int i = 0; i < count; i++) { // escrever fich bin

fwrite(&pMap->item[i], sizeof(struct Item), 1, f2);

}

fclose(f);

fclose(f2);

}

/\*

Nesta função incializa-se os itens de um ficheiro binario

\*/

void InitObejectItemBin(struct Map \*pMap) {

struct Item item;

FILE \*f;

char l[MAX\_LINE];

int i = 0;

f = fopen("objectos.dat", "r");

while (((fread(&item, sizeof(struct Item), 1, f)) > 0) != NULL) {

pMap->item[i].CodItem = item.CodItem;

pMap->item[i].PositionItem = item.PositionItem;

pMap->item[i].DamageItem = item.DamageItem;

pMap->item[i].CriticItem = item.CriticItem;

pMap->item[i].LifeItem = item.LifeItem;

strcpy(pMap->item[i].NameItem, item.NameItem);

i++;

}

fclose(f);

}

/\*

Esta função lê os tesouros existentes no mapa do ficheiro txt

\*/

void InitObejectTreasure(Map \*pMap){

FILE \*f;

char l[MAX\_LINE];

int line = 1;

int i = 0;

int count = 0;

int id, position, gold;

f = fopen("treasures.txt", "r");

while (fgets(l, MAX\_LINE, f) != NULL) {

if (line == 1 && strcmp(l, "\n") != 0) {

sscanf(l, "%d %d %d", &id, &position, &gold);

pMap->treasure[i].CodTresure = id;

pMap->treasure[i].PositionTresure = position;

pMap->treasure[i].Gold = gold;

line++;

}

else if (strcmp(l, "\n") != 0 && l != " ") {

strcpy(pMap->treasure[i].NameTreasure, l);

line = 1;

i++;

count++;

}

else {

//do Nothing

}

}

fclose(f);

}

/\*

Esta função serve inicia o mapa numa primeira versão antes de implementar a leitura do mapa apartir de um ficheiro

\*/

int InitMap(Cell cells[]) {

//cell 0

cells[0].north = 1;

cells[0].south = -1;

cells[0].west = 20;

cells[0].east = 19;

cells[0].up = -1;

cells[0].down = -1;

strcpy(cells[0].descriptionCell, "Bem vido à Vila Sunshine, desejamos lhe a maior sorte para a sua missão!");

cells[0].treasureCell = -1;

cells[0].itemCell = -1;

//cell 1

cells[1].north = 4;

cells[1].south = 0;

cells[1].west = 3;

cells[1].east = -1;

cells[1].up = 2;

cells[1].down = -1;

strcpy(cells[1].descriptionCell, "Neste mumento soldado encontraste entre duas arenas, entra lá e descobre o que se passa!");

cells[1].treasureCell = -1;

cells[1].itemCell = -1;

//cell 2

cells[2].north = -1;

cells[2].south = -1;

cells[2].west = -1;

cells[2].east = 1;

cells[2].up = -1;

cells[2].down = 1;

strcpy(cells[2].descriptionCell, "Nesta arena podes encontrar uma armadura que te vai dar mais HP");

cells[2].treasureCell = 1;

cells[2].itemCell = 1;

//cell 3

cells[3].north = -1;

cells[3].south = -1;

cells[3].west = -1;

cells[3].east = 1;

cells[3].up = -1;

cells[3].down = -1;

strcpy(cells[3].descriptionCell, "Nesta arena podes encontrar uma armadura que te vai dar mais DANO");

cells[3].treasureCell = 2;

cells[3].itemCell = 2;

//cell 4

cells[4].north = 7;

cells[4].south = 1;

cells[4].west = 5;

cells[4].east = 6;

cells[4].up = -1;

cells[4].down = -1;

strcpy(cells[4].descriptionCell, "Chegaste a um crusamento podes continar em frente para chegar a arena PRINCIPAL ou ir por um caminho maior");

cells[4].treasureCell = -1;

cells[4].itemCell = -1;

//cell 5

cells[5].north = -1;

cells[5].south = -1;

cells[5].west = 22;

cells[5].east = 5;

cells[5].up = -1;

cells[5].down = -1;

strcpy(cells[5].descriptionCell, "Por este caminho vais pela direita");

cells[5].treasureCell = -1;

cells[5].itemCell = -1;

//cell 6

cells[6].north = -1;

cells[6].south = -1;

cells[6].west = 4;

cells[6].east = 17;

cells[6].up = -1;

cells[6].down = -1;

strcpy(cells[6].descriptionCell, "Por este caminho vais pela esquerda");

cells[6].treasureCell = -1;

cells[6].itemCell = -1;

//cell 7

cells[7].north = 10;

cells[7].south = 4;

cells[7].west = 9;

cells[7].east = 8;

cells[7].up = -1;

cells[7].down = -1;

strcpy(cells[7].descriptionCell, "Neste mumento soldado encontraste entre duas arenas, entra lá e descobre o que se passa!");

cells[7].treasureCell = -1;

cells[7].itemCell = -1;

//cell 8

cells[8].north = -1;

cells[8].south = -1;

cells[8].west = 7;

cells[8].east = 1;

cells[8].up = -1;

cells[8].down = -1;

strcpy(cells[8].descriptionCell, "Nesta arena exite uma besta feroz que tens de eliminar!!!");

cells[8].treasureCell = 3;

cells[8].itemCell = 3;

//cell 9

cells[9].north = -1;

cells[9].south = -1;

cells[9].west = -1;

cells[9].east = 7;

cells[9].up = -1;

cells[9].down = -1;

strcpy(cells[9].descriptionCell, "Nesta arena vais poder sentir o poder de uma criatura quase tão poderoza como o rei dos monstros");

cells[9].treasureCell = 4;

cells[9].itemCell = 4;

//cell 10

cells[10].north = -1;

cells[10].south = 7;

cells[10].west = 11;

cells[10].east = 12;

cells[10].up = 13;

cells[10].down = -1;

strcpy(cells[10].descriptionCell, "Neste mumento soldado ou sobes as escadas e entras na arena principal ou desistes, a escolha é tua");

cells[10].treasureCell = -1;

cells[10].itemCell = -1;

//cell 11

cells[11].north = -1;

cells[11].south = -1;

cells[11].west = 14;

cells[11].east = 10;

cells[11].up = -1;

cells[11].down = -1;

strcpy(cells[11].descriptionCell, "Por este caminho vais pela direita");

cells[11].treasureCell = -1;

cells[11].itemCell = -1;

//cell 12

cells[12].north = -1;

cells[12].south = -1;

cells[12].west = 10;

cells[12].east = 15;

cells[12].up = -1;

cells[12].down = -1;

strcpy(cells[12].descriptionCell, "Por este caminho vais pela esquerda");

cells[12].treasureCell = -1;

cells[12].itemCell = -1;

//cell 13

cells[13].north = -1;

cells[13].south = -1;

cells[13].west = -1;

cells[13].east = -1;

cells[13].up = -1;

cells[13].down = 10;

strcpy(cells[13].descriptionCell, "Muitos parabens pela tua CORAGEM !!! E boa sorte para o teu desafio soldado");

cells[13].treasureCell = 5;

cells[13].itemCell = 5;

//cell 14

cells[14].north = -1;

cells[14].south = 23;

cells[14].west = -1;

cells[14].east = 11;

cells[14].up = -1;

cells[14].down = -1;

strcpy(cells[14].descriptionCell, "Estas num beco ou vias em direção há saida da vila ou voltas para tras");

cells[14].treasureCell = -1;

cells[14].itemCell = -1;

//cell 15

cells[15].north = -1;

cells[15].south = 16;

cells[15].west = 12;

cells[15].east = -1;

cells[15].up = -1;

cells[15].down = -1;

strcpy(cells[15].descriptionCell, "Estas num beco ou vias em direção há saida da vila ou voltas para tras");

cells[15].treasureCell = -1;

cells[15].itemCell = -1;

//cell 16

cells[16].north = 15;

cells[16].south = 17;

cells[16].west = -1;

cells[16].east = -1;

cells[16].up = -1;

cells[16].down = -1;

strcpy(cells[16].descriptionCell, "Nesta rua ou voltas para tras ou continuas em direção saida da vila");

cells[16].treasureCell = -1;

cells[16].itemCell = -1;

//cell 17

cells[17].north = 16;

cells[17].south = 18;

cells[17].west = 6;

cells[17].east = -1;

cells[17].up = -1;

cells[17].down = -1;

strcpy(cells[17].descriptionCell, "Chegaste a meio da cidade do lado este");

cells[17].treasureCell = -1;

cells[17].itemCell = -1;

//cell 18

cells[18].north = 17;

cells[18].south = 19;

cells[18].west = -1;

cells[18].east = -1;

cells[18].up = -1;

cells[18].down = -1;

strcpy(cells[18].descriptionCell, "Ou voltas para a saida da vila ou vias em direção ao rei do montro");

cells[18].treasureCell = -1;

cells[18].itemCell = -1;

//cell 19

cells[19].north = 18;

cells[19].south = -1;

cells[19].west = 0;

cells[19].east = -1;

cells[19].up = -1;

cells[19].down = -1;

strcpy(cells[19].descriptionCell, "Tem coraje e vai em frente teras grandes reconpensas");

cells[19].treasureCell = -1;

cells[19].itemCell = -1;

//cell 20

cells[20].north = 21;

cells[20].south = -1;

cells[20].west = -1;

cells[20].east = 0;

cells[20].up = -1;

cells[20].down = -1;

strcpy(cells[20].descriptionCell, "Tem coraje e vai em frente teras grandes reconpensas");

cells[20].treasureCell = -1;

cells[20].itemCell = -1;

//cell 21

cells[21].north = 22;

cells[21].south = 20;

cells[21].west = -1;

cells[21].east = -1;

cells[21].up = -1;

cells[21].down = -1;

strcpy(cells[21].descriptionCell, "Ou voltas para a saida da vila ou vias em direção ao rei do montro");

cells[21].treasureCell = -1;

cells[21].itemCell = -1;

//cell 22

cells[22].north = 23;

cells[22].south = 22;

cells[22].west = -1;

cells[22].east = 5;

cells[22].up = -1;

cells[22].down = -1;

strcpy(cells[22].descriptionCell, "Chegaste a meio da cidade do lado oeste");

cells[22].treasureCell = -1;

cells[22].itemCell = -1;

//cell 23

cells[23].north = 14;

cells[23].south = 22;

cells[23].west = -1;

cells[23].east = -1;

cells[23].up = -1;

cells[23].down = -1;

strcpy(cells[23].descriptionCell, "Estas num beco ou vias em direção há saida da vila ou voltas para tras");

cells[23].treasureCell = -1;

cells[23].itemCell = -1;

return 24;

}

/\*

Esta função carrega o mapa do jogo de um ficheiro com o nome "map.txt"

\*/

void LoadMapFromFile(struct Map \*pMap) {

FILE \*f;

char l[MAX\_LINE];

int line = 1;

int i = 0;

int count = 0;

int north;

int south;

int west;

int east;

int up;

int down;

int item;

int treasure;

f = fopen("map.txt", "r");

while (fgets(l, MAX\_LINE, f) != NULL) {

if (line == 1 && strcmp(l, "\n") != 0) {

sscanf(l, "%d %d %d %d %d %d %d %d", &north, &south, &west,

&east, &up, &down, &item, &treasure);

pMap->cell[i].north = north;

pMap->cell[i].south = south;

pMap->cell[i].west = west;

pMap->cell[i].east = east;

pMap->cell[i].up = up;

pMap->cell[i].down = down;

pMap->cell[i].itemCell = item;

pMap->cell[i].treasureCell = treasure;

line++;

}

else if (strcmp(l, "\n") != 0 && l != " ") {

strcpy(pMap->cell[i].descriptionCell, l);

line = 1;

i++;

count++;

}

else {

//do Nothing

}

}

pMap->nCells = count;

fclose(f);

}

/\*

Esta função mostra todas as salas do jogo no ecrã

\*/

void PrintMapFromFile(Map \*pMap) {

for (int i = 0; i < pMap->nCells; i++) {

printf("\n");

printf("Norte: %d \n", pMap->cell[i].north);

printf("Sul: %d \n", pMap->cell[i].south);

printf("Oeste: %d \n", pMap->cell[i].west);

printf("Este: %d \n", pMap->cell[i].east);

printf("Cima: %d \n", pMap->cell[i].up);

printf("Baixo: %d \n", pMap->cell[i].down);

printf("Item: %d \n", pMap->cell[i].itemCell);

printf("Tesouro: %d \n", pMap->cell[i].treasureCell);

printf("Descrição: %s \n", pMap->cell[i].descriptionCell);

}

}

/\*

Esta função é o que permite o jogador navegar no mapa e usar algumas das funçoes implementadas

\*/

void PlayerWalk(struct Player \*pPlayer, struct Map \*pMap, struct Monster monster[]) {

int option;

//FunctionClear();

printf("\n");

printf(" --------------------------------------------------------------------\n");

printf(" Para andar na aldeia seleciona uma das opções abaixo Soldado\n");

printf("1 - Norte 2 - Sul 3 - Oeste 4 - Este 5 - Subir 6 - Descer 7 - Save 8 - Menu Principal\n");

printf(" --------------------------------------------------------------------\n");

printf("\n");

printf("Posiçao atual: %d\n", pPlayer->cellPlayer);

printf("\n");

printf("%s", pMap->cell[pPlayer->cellPlayer].descriptionCell);

printf("\n");

printf("Escolhe a direção que queres soldado: \n");

scanf("%d", &option);

FunctionClear();

switch (option)

{

case 1:

if (pMap->cell[pPlayer->cellPlayer].north == -1) {

printf("Lamento mas nao podes atravesar paredes !!!");

}

else {

pPlayer->cellPlayer = pMap->cell[pPlayer->cellPlayer].north;

}

break;

case 2:

if (pMap->cell[pPlayer->cellPlayer].south == -1) {

printf("Lamento mas nao podes atravesar paredes !!!");

}

else { pPlayer->cellPlayer = pMap->cell[pPlayer->cellPlayer].south; }

break;

case 3:

if (pMap->cell[pPlayer->cellPlayer].west == -1) {

printf("Lamento mas nao podes atravesar paredes !!!");

}

else { pPlayer->cellPlayer = pMap->cell[pPlayer->cellPlayer].west; }

break;

case 4:

if (pMap->cell[pPlayer->cellPlayer].east == -1) {

printf("Lamento mas nao podes atravesar paredes !!!");

}

else { pPlayer->cellPlayer = pMap->cell[pPlayer->cellPlayer].east; }

break;

case 5:

if (pMap->cell[pPlayer->cellPlayer].up == -1) {

printf("Lamento mas nao podes atravesar paredes !!!");

}

else { pPlayer->cellPlayer = pMap->cell[pPlayer->cellPlayer].up; }

break;

case 6:

if (pMap->cell[pPlayer->cellPlayer].down == -1) {

printf("Lamento mas nao podes atravesar paredes !!!");

}

else { pPlayer->cellPlayer = pMap->cell[pPlayer->cellPlayer].down; }

break;

case 7:

SaveGame( pPlayer, monster);

printf("-----------------------------------\n");

printf("O Jogo foi guardado com sucesso\n");

break;

case 8:

main();

break;

default:

printf("O valor introduzido é invalido!!! \n Insira novamente um numero de 1 a 8 \n");

break;

}

}

/\*

Esta Função faz com que os monstros do indice 5 a 7 andem pelo mapa de forma aleatória e sendo estes escolhidos de forma aleatória

\*/

void MonstersWalk(struct Player \*pPlayer, struct Map \*pMap, Monster monster[]) {

int randMoveMonster = rand() % 6 + 1;

int randMonster = 0;

while (randMonster < 5 && randMonster > 8) {

randMonster = rand() % 8 + 5;

}

if ((strcmp(pPlayer->namePlayer, "SU") == 0) || (strcmp(pPlayer->namePlayer, "su") == 0) ||

(strcmp(pPlayer->namePlayer, "super user") == 0) || (strcmp(pPlayer->namePlayer, "Super User") == 0)) {

switch (randMoveMonster)

{

case 1:

if (pMap->cell[monster[randMonster].cellMonster].north == -1) {

printf("Monstro não se moveu!\n");

}

else {

monster[randMonster].cellMonster = pMap->cell[monster[randMonster].cellMonster].north;

printf("\n");

printf("Monstro: %s\n", monster[randMonster].nameMosnter);

printf("Monstro foi para a sala: %d\n", monster[randMonster].cellMonster);

}

break;

case 2:

if (pMap->cell[monster[randMonster].cellMonster].south == -1) {

printf("Monstro não se moveu!\n");

}

else {

monster[randMonster].cellMonster = pMap->cell[monster[randMonster].cellMonster].south;

printf("\n");

printf("Monstro: %s\n", monster[randMonster].nameMosnter);

printf("Monstro foi para a sala: %d\n", monster[randMonster].cellMonster);

}

break;

case 3:

if (pMap->cell[monster[randMonster].cellMonster].west == -1) {

printf("Monstro não se moveu!\n");

}

else {

monster[randMonster].cellMonster = pMap->cell[monster[randMonster].cellMonster].west;

printf("\n");

printf("Monstro: %s\n", monster[randMonster].nameMosnter);

printf("Monstro foi para a sala: %d\n", monster[randMonster].cellMonster);

}

break;

case 4:

if (pMap->cell[monster[randMonster].cellMonster].east == -1) {

printf("Monstro não se moveu!\n");

}

else {

monster[randMonster].cellMonster = pMap->cell[monster[randMonster].cellMonster].east;

printf("\n");

printf("Monstro: %s\n", monster[randMonster].nameMosnter);

printf("Monstro foi para a sala: %d\n", monster[randMonster].cellMonster);

}

break;

case 5:

if (pMap->cell[monster[randMonster].cellMonster].up == -1) {

printf("Monstro não se moveu!\n");

}

else {

printf("Monstro não se moveu!\n");

}

break;

case 6:

if (pMap->cell[monster[randMonster].cellMonster].down == -1) {

printf("Monstro não se moveu!\n");

}

else

{

printf("Monstro não se moveu!\n");

}

break;

default:

break;

}

}

else {

switch (randMoveMonster)

{

case 1:

if (pMap->cell[monster[randMonster].cellMonster].north == -1) {

}

else {

monster[randMonster].cellMonster = pMap->cell[monster[randMonster].cellMonster].north;

}

break;

case 2:

if (pMap->cell[monster[randMonster].cellMonster].south == -1) {

}

else {

monster[randMonster].cellMonster = pMap->cell[monster[randMonster].cellMonster].south;

}

break;

case 3:

if (pMap->cell[monster[randMonster].cellMonster].west == -1) {

}

else {

monster[randMonster].cellMonster = pMap->cell[monster[randMonster].cellMonster].west;

}

break;

case 4:

if (pMap->cell[monster[randMonster].cellMonster].east == -1) {

}

else {

monster[randMonster].cellMonster = pMap->cell[monster[randMonster].cellMonster].east;

}

break;

case 5:

if (pMap->cell[monster[randMonster].cellMonster].up == -1) {

}

else {

}

break;

case 6:

if (pMap->cell[monster[randMonster].cellMonster].down == -1) {

}

else

{

}

break;

default:

break;

}

}

}

/\*

Nesta Função o jogador batalha com um monstro ou varios caso exista algum monstro na sua cela

\*/

void Battle(struct Player \*pPlayer, struct Map \*pMap, Monster monster[]) {

int randCriticPlayer = 0; // é sorteado um valor de ataque critico

int randAtkPlayer = 0; //O Player acerta o ataque ou falha

int newatkPlayer = 0; //Novo valor de ataque do jogador

int randCriticMonster = 0; // é sorteado um valor de ataque critico

int randAtkMonster = 0; //O Player acerta o ataque ou falha

int newatkMonster = 0; //Novo valor de ataque do monstro

for (int i = 0; i < monster[0].nMonsters; i++) {// enquanto houver monstros na sala o jodador vai lutando contra eles

while (pPlayer->cellPlayer == monster[i].cellMonster && pPlayer->energyPlayer > 0 && monster[i].lifeMonster > 0) {

randAtkPlayer = rand() % 6;

randCriticPlayer = rand() % pPlayer->critic;

newatkPlayer = ((pPlayer->damage \* randCriticPlayer) / 100);

newatkPlayer = newatkPlayer + pPlayer->damage;

if (randAtkPlayer > 3 && pPlayer->energyPlayer > 0) {// o jogador ataca

monster[i].lifeMonster = monster[i].lifeMonster - newatkPlayer;

printf("\n");

printf("Monstro: %s\n", monster[i].nameMosnter);//o jogador falha o ataque

printf("Dano tirado ao Monstro: %d\n", newatkPlayer);

printf("HP atual do Monstro: %d\n", monster[i].lifeMonster);

}

else {

if (pPlayer->energyPlayer > 0) {

printf("Parabens acabaste de falar completamente o ATAQUE!!!\n");

}

else {

//do nothing

}

}

randAtkMonster = rand() % 6;

randCriticMonster = rand() % monster[i].criticMonster;

newatkMonster = ((monster[i].damageMonster \* randCriticMonster) / 100);

newatkMonster += monster[i].damageMonster;

if (randAtkMonster > 3 && monster[i].lifeMonster > 0) { // o mosntro ataca

pPlayer->energyPlayer -= newatkMonster;

printf("\n");

printf("Soldado: %s\n", pPlayer->namePlayer);

printf("Dano tirado pelo Monstro: %d\n", newatkMonster);

printf("HP: %d\n", pPlayer->energyPlayer);

}

else {

if (monster[i].lifeMonster > 0) {// o monstro falhou o ataque

printf("Para tua sorte Soldado o Monstro %s", monster[i].nameMosnter);

printf(" acabou de falhar o ataque\n");

}

else { //Quando a vida do monstro for menor que 0 o jogador podera receber os items que estão na sala

int itemSelect = pMap->cell[pPlayer->cellPlayer].itemCell; // esta variavel guarda o valor do item que se encontra na mesma sala do jogador

int treasureSelected = pMap->cell[pPlayer->cellPlayer].treasureCell; //esta variaver guarda o valor do tesouro que se encontra na mesma sala do jogador

if (itemSelect != -1) { // jogador apanha item

pPlayer->damage = pPlayer->damage + pMap->item[itemSelect].DamageItem;

pPlayer->critic = pPlayer->critic + pMap->item[itemSelect].CriticItem;

pPlayer->energyPlayer = pPlayer->energyPlayer + pMap->item[itemSelect].LifeItem;

printf("Item Adicionado: %s", pMap->item[itemSelect].NameItem);

printf("Dano Adicionado: %d", pMap->item[itemSelect].DamageItem);

printf("Dano critico Adicionado: %d", pMap->item[itemSelect].CriticItem);

printf("HP Adicionado: %d", pMap->item[itemSelect].LifeItem);

}

if (treasureSelected != -1) {// jogador apanha tesouro

pPlayer->treasurePlayer = pPlayer->treasurePlayer + pMap->treasure[treasureSelected].Gold;

printf("\n");

printf("Tesouro encontrado: %s\n", pMap->treasure[treasureSelected].NameTreasure);

printf("Gold Adicionado: %d\n", pMap->treasure[treasureSelected].Gold);

}

}

}

} // end while

} // end for

}

/\*

Esta função guarda o jogo num ficheiro binário em que o nume é atribuido pelo utilizador

\*/

void SaveGame(struct Player \*pPlayer, struct Monster monster[]) {

struct SaveGame save;

FILE \*f;

char fileName[MAX\_FILENAME];

printf("Insere um nome para o ficheiro: \n");

scanf("%s", fileName);

//fgets(fileName, MAX\_NAME, stdin);

fileName[strlen(fileName) - 1] = '\0';

strcat(fileName, ".bin");

f = fopen(fileName, "wb");

fwrite(pPlayer, sizeof(struct Player), 1, f);

for (int i = 0; i < monster[0].nMonsters; i++) {

save.saveMonster.cellMonster = monster[i].cellMonster;

save.saveMonster.criticMonster = monster[i].criticMonster;

save.saveMonster.damageMonster = monster[i].damageMonster;

save.saveMonster.itemMonster = monster[i].itemMonster;

save.saveMonster.lifeMonster = monster[i].lifeMonster;

strcpy(save.saveMonster.nameMosnter, monster[i].nameMosnter);

save.saveMonster.treasureMonster = monster[i].treasureMonster;

fwrite(&save, sizeof(struct SaveGame), 1, f);

}

fclose(f);

}

/\*

Esta função carrega o jogo de um ficheiro binário, o utilizador pode escolher qual é o save que quer carregar

\*/

void LoadGame(struct Player \*pPlayer, struct Monster monster[]) {

struct SaveGame save;

FILE \*f;

char fileName[MAX\_FILENAME];

printf("Insere o nome do savefile que pretendes continuar: \n");

//fgets(fileName, MAX\_NAME, stdin);

scanf("%s", fileName);

strcat(fileName, ".bin");

f = fopen(fileName, "rb");

if (f == NULL) {

printf("Não existe nenhum jogo com esse nome!");

LoadGame(pPlayer, monster);

}

else {

fread(pPlayer, sizeof(struct Player), 1, f);

for (int i = 0; i < monster[0].nMonsters; i++) {

fread(&save, sizeof(struct SaveGame), 1, f);

monster[i].cellMonster = save.saveMonster.cellMonster;

monster[i].criticMonster = save.saveMonster.criticMonster;

monster[i].damageMonster = save.saveMonster.damageMonster;

monster[i].itemMonster = save.saveMonster.itemMonster;

monster[i].lifeMonster = save.saveMonster.lifeMonster;

strcpy(monster[i].nameMosnter, save.saveMonster.nameMosnter);

monster[i].treasureMonster = save.saveMonster.treasureMonster;

}

fclose(f);

}

}

/\*

Esta função verifica se o jogador ganha o jogo o se o jodador morre e mostra as mensagens correspondentes

\*/

void EndGame(struct Player \*pPlayer, struct Monster monster[], struct Map \*pMap) {

for (int i = 0; i < monster[0].nMonsters; i++) {

if(strcmp(monster[i].nameMosnter, "LUCIFER") == 0 && pPlayer->energyPlayer > 0 && monster[i].lifeMonster <= 0) {

FunctionClear();

printf("Muitos Parabens soldado acabaste de derrotar rei dos monstros e salvaste a vila da destruição\n");

printf("Agora que recuperaste a lendaria armadura mais a lendaria espada das maos do %s\n", monster[i].nameMosnter);

printf("Por este feito decidimos-te prover a Class B soldado %s\n",pPlayer->namePlayer);

printf("Aconselho-te a não tirar muito tempo de ferias, estou com sentimento que em breve vamos precisar de ti ... \n");

printf("\n");

printf("---------------------------------------------------------------------------------------------------------------------\n");

printf("\n");

printf("Jogo desenvolvido por: \n");

printf("Rute Figeiredo\n");

printf("Tiago Campos\n");

printf("\n");

}

if(pPlayer->energyPlayer < 0){

FunctionClear();

printf("O Soldado %s", pPlayer->namePlayer);

printf(" foi um soldado exemplar, mas ficou demasiado convencido e acabou por se descuidar\n");

printf("e isso foi o seu fim -\_-\n");

printf("R.I.P : &s", pPlayer->namePlayer);

main();

}

}

}