|  |  |
| --- | --- |
|  | **2011** |
|  | **Escola Superior de Tecnologia e Gestão**  **Docente:**  Luís Garcia  **Discentes:**  6170 – Bruno Moreira  6032 – Luís Costa |

****

|  |
| --- |
| **[Sistemas Operativos]**  **Trabalho de Grupo 2** |
| Programação de um Jogo de Aventuras - Parte II |

Índice

[Introdução 3](#_Toc297595572)

[1 Descrição do programa 4](#_Toc297595573)

[1.1 Cliente-Servidor 4](#_Toc297595574)

[1.2 Versão Servidor 5](#_Toc297595575)

[1.3 Threads do Servidor 5](#_Toc297595576)

[1.3.1 Thread do Jogador 5](#_Toc297595577)

[1.3.2 Thread do Monstro 5](#_Toc297595578)

[1.4 Versão Cliente 6](#_Toc297595579)

[1.4.1 Thread Mensagens 6](#_Toc297595580)

[1.4.2 Desenho das janelas 6](#_Toc297595581)

[2 Conclusão 7](#_Toc297595582)

[3 Bibliografia 8](#_Toc297595583)

[ANEXOS 9](#_Toc297595584)

# Introdução

Segundo trabalho de grupo desenvolvido no âmbito da disciplina de Sistemas Operativos, da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Beja, curso de Engenharia Informática. Este tem como objectivo dar continuidade ao desenvolvimento do jogo de aventuras gráficas do trabalho anterior, adicionando-lhe algumas melhorias recorrendo à programação concorrente e ao ambiente gráfico do Ms Windows.

# Descrição do programa

## Cliente-Servidor

Esta versão do jogo consiste no desenvolvimento de um programa cliente que se liga a um programa servidor pela rede.

Para tal foram utilizados named pipes e threads para controlar a comunicação.

A comunicação entre as duas versões teve que ser protocolada, por forma que ambas as versões do programa pudessem compreender-se.

Assim foram protocoladas as seguintes mensagens:

»L|inicia\_luta|resultado\_luta  
Esta mensagem indica ao destinatário que se trata de uma luta, pois começa com a palavra “»L”. O segundo parâmetro da mensagem indica se deve ou não iniciar uma luta, e o terceiro se o jogador venceu ou perdeu a luta.

»M|nome|energia|localização

Esta mensagem indica ao destinatário que lhe vai passar a informação do monstro, pois começa com a palavra “»M”. O segundo parâmetro da mensagem indica o nome do monstro, o segundo a energia e o terceiro a sua localização.

»J|nome|energia|localização|tem\_tesouro

Esta mensagem indica ao destinatário que lhe vai passar a informação do jogador, pois começa com a palavra “»J”. O segundo parâmetro da mensagem indica o nome do monstro, o segundo a energia, o terceiro a localização actual do jogador e o terceiro indica se o jogador tem ou não o tesouro em sua posse.

»C|descrição|norte|sul|este|oeste

Esta mensagem indica ao destinatário que lhe vai passar a informação de uma célula, pois começa com a palavra “»C”. O segundo parâmetro da mensagem indica a descrição da célula, o segundo a saída a norte, o terceiro a saída a sul, o quarto a saída a este e o quinto a saída a oeste.

»S|ecrã"

Esta mensagem indica ao destinatário que deverá desenhar um determinado ecrã, pois começa com a palavra “»S”. O segundo parâmetro da mensagem indica qual o ecrã que deverá desenhar.

»I|texto"

Esta mensagem indica ao destinatário que deverá solicitar um comando, pois começa com a palavra “»I”. O segundo parâmetro da mensagem indica qual a mensagem que deve ser apresentada para solicitar o referido comando.

## Versão Servidor

A versão servidor contém toda a gestão do jogo, ou seja gere o mapa, jogador e monstro, ordenando à versão cliente que apresente dados no ecrã ou que solicite ao utilizador a introdução de dados. Para controlar a gestão do jogo foram utilizadas threads.

## Threads do Servidor

Ao iniciar um novo jogo o sistema cria duas threads diferentes, uma para tratar a interacção com o cliente e outra para tratar o movimento do monstro. Estas threads correm sincronizadas de forma a garantir que o jogador e o monstro não se movimentam enquanto estão a lutar. A sincronização das threads é garantida com a utilização de mutex’s.

### Thread do Jogador

Esta thread é responsável pela interacção com a versão cliente, recebendo os comandos introduzidos pelo utilizador e enviado para o cliente o que deve ser apresentado no ecrã.

É também esta thread que realiza a actualização da localização do jogador no mapa.

### Thread do Monstro

Esta thread é responsável pela movimentação do monstro pelo mapa, decidindo quando e para onde este se deverá movimentar, actualizando posteriormente esta informação.

## Versão Cliente

A versão cliente é um terminal “estúpido” que se limita a cumprir ordens que a versão servidor lhe envia, nomeadamente solicitar comandos ao utilizador e apresentar informação no ecrã. Para controlar a interacção com a versão servidor foi necessário recorrer a threads.

Esta versão utiliza a API do MS Windows para o desenho das janelas.

### Thread Mensagens

Esta thread é responsável pelo tratamento das mensagens enviadas pela versão servidor, decifrando o protocolo estabelecido e chamando as funções que executam as acções solicitadas.

Esta thread controla a leitura de mensagens utilizando uma variável do tipo boleano que define quando se pode ou não ler do pipe.

A actualização da variável de controlo é efectuada com a utilização de mutex’s, garantido assim a sua integridade.

### Desenho das janelas

A utilização da API do MS Windows para desenhar as janelas do jogo, serve para apresentar as descrições do jogador e monstro, assim como a imagem correspondente à célula em que o jogador se encontra.

# Conclusão

Conseguimos cumprir o objectivo do trabalho que consistia em desenvolver uma versão melhorada do jogo de aventuras. O grupo achou este trabalho interessante na medida em que nos permitiu aumentar o nosso conhecimento na utilização de threads, que até à data não tinha sido utilizado por qualquer membro do nosso grupo. Quanto às dificuldades na realização do trabalho, o grupo teve algumas relacionadas com o debug da assincronia entre threads, que nem sempre é fácil de executar. O grupo também achou que o manuseamento das janelas utilizando a API do Ms Windows é extremamente trabalhoso e não justifica a sua utilização nos dias que correm. Compreendemos no entanto a utilização desta tecnologia tendo em conta que nos permitiu compreender melhor o comportamento da interface gráfica no Ms Windows.

# Bibliografia

1. Win32 Programming, FunctionX, 2011

<http://www.functionx.com/win32>

1. Documentação da linguagem C e exemplos

<http://estig.e-learning.ipbeja.pt/course/view.php?id=1015>

# ANEXOS

Anexo 1: Código do Programa

//Descritivo:Primeiro trabalho de grupo da disciplina de SO

//Titulo:DLL para o jogo de aventura

//Autores: Luís Costa Nº6032, Bruno Moreira Nº6170

#include "stdafx.h"

#include "string.h"

#include <ctime>

#include <malloc.h>

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

#include <ctype.h>

#include "windows.h"

#include "stdafx.h"

#include "JogoSO.h"

//include da DLL

#include "JogoSODLL.h"

#define MAX\_CELULAS 9 // número máximo de células do mapa

#define MAX\_COL 400 // tamanho máximo da linha

//-------------------------------------------------------------------------------------------------

// ESTRUTURAS

//-------------------------------------------------------------------------------------------------

//Jogador

//---------------------

struct Jogador

{

char\* nome[50]; //Nome do jogador

int energia; //Energia do jogador

int localizacao; //Celula onde o jogador se encontra

int flg\_tem\_tesouro; //Se o jogador tem ou não tesouro

};

struct Celula

{

int norte; //Qual a celula a norte

int sul; //Qual a celula a sul

int este; //Qual a celula a este

int oeste; //Qual a celula a oeste

char\* descricao[400]; //Descrição da celula

int item; //Item que existe na celula

};

//Monstro

//---------------------

struct Monstro

{

char\* nome[50];

int energia; //Energia do jogador

int localizacao; //Celula onde o jogador se encontra

};

#ifdef \_DEBUG

#define new DEBUG\_NEW

#endif

//-------------------------------------------------------------------------------------------------

// FUNÇÕES

//-------------------------------------------------------------------------------------------------

//Inicializa o jogador

void inicializa\_jogador(struct Jogador \*pJogador, char\* nome, bool blnSuperUser)

{

// associa os dados por defeito ao jogador

strcpy((char\*) pJogador->nome, nome);

pJogador->localizacao = 0;

pJogador->flg\_tem\_tesouro = -1;

if (blnSuperUser == true) {

pJogador->energia = 9999; // energia do superuser

}

else {

pJogador->energia = 100; // energia do utilizador normal

}

}

// inicilializa mapa com base num ficheiro

void inicializa\_mapa\_ficheiro(struct Celula pMapa[], char\* pFicheiroMapa)

{

printf("A inicializar o mapa do ficheiro...\n");

#define MAX\_LIN 80

#define CAMPOS 7

FILE \*f;

char l[ MAX\_LIN ];

f = fopen( pFicheiroMapa, "r" );

// inicia a linha

int iLinha = 0;

// inicia o contador

int iIndice = -1;

// percorre o ficheiro até encontrar o fim

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

// se o resto da divisão do número da linha pelo número de linhas por célula

// for igual a 0, quer dizer que é um novo registo

if (iLinha % CAMPOS == 0)

{

// incrementa o indice

iIndice = iIndice + 1;

}

// imprime a linha

int iResto = iLinha % CAMPOS;

switch ( iResto )

{

case 0: // Descrição

strcpy((char\*)pMapa[iIndice].descricao, rtrim(l));

break;

case 1: // Norte

pMapa[iIndice].norte = atoi (l);

break;

case 2: // Sul

pMapa[iIndice].sul = atoi (l);

break;

case 3: // Este

pMapa[iIndice].este = atoi (l);

break;

case 4: // Oeste

pMapa[iIndice].oeste = atoi (l);

break;

case 5: // Item

pMapa[iIndice].item = atoi (l);

break;

}

// incrementa o número da linha

iLinha = iLinha + 1;

}

fclose( f );

printf("Concluiu a inicialização do mapa com base num ficheiro\n");

}

// Inicilializa mapa com base num ficheiro binário

void inicializa\_mapa\_ficheiro\_bin(struct Celula pMapa[], char\* pFicheiroMapa)

{

printf("A inicializar o mapa de um ficheiro binário\n");

#define MAX\_LIN 400

#define CAMPOS 7

FILE \*f;

char l[ MAX\_LIN ];

f = fopen( pFicheiroMapa, "rb" );

// inicia a linha

int iLinha = 0;

// inicia o contador

int iIndice = -1;

if (f != NULL)

{

// percorre o ficheiro até encontrar o fim

while( fread(l,sizeof(char), MAX\_LIN, f) != NULL ){

// se o resto da divisão do número da linha pelo número de linhas por célula

// for igual a 0, quer dizer que é um novo registo

if (iLinha % CAMPOS == 0)

{

// incrementa o indice

iIndice = iIndice + 1;

}

// imprime a linha

int iResto = iLinha % CAMPOS;

if (iResto > 0) {int a = atoi(l);}

switch ( iResto )

{

case 0: // Descrição

strcpy((char\*)pMapa[iIndice].descricao, l);

break;

case 1: // Norte

pMapa[iIndice].norte = atoi (l);

break;

case 2: // Sul

pMapa[iIndice].sul = atoi (l);

break;

case 3: // Este

pMapa[iIndice].este = atoi (l);

break;

case 4: // Oeste

pMapa[iIndice].oeste = atoi (l);

break;

case 5: // Item

pMapa[iIndice].item = atoi (l);

break;

}

// incrementa o número da linha

iLinha = iLinha + 1;

}

fclose( f );

printf("Concluiu a inicialização do mapa com base num ficheiro binário\n");

}

else

{

printf("Erro a abrir o ficheiro %s",pFicheiroMapa);

system("pause");

}

}

//Inicializa o mapa da aventura

//--------------------------------

// +------+------+------+

// | | |

// |0 |1 |2 |

// +------+-- --+-- --+

// | |

// |3 |4 |5 T|

// +-- --+------+------+

// | |

// |6 |7 |8 |

// +------+------+------+

void inicializa\_mapa(struct Celula pMapa[], char\* pFicheiroMapa)

{

printf("A inicializar o mapa...\n");

// se o tamanho da string pFicheiroMapa for superior a 0 então é

// para carregar o mapa do ficheiro

if (strlen(pFicheiroMapa) > 1)

{

// carrega mapa do ficheiro

inicializa\_mapa\_ficheiro\_bin(pMapa, pFicheiroMapa);

}

else

{

//Construção da sala 0

//----------------------

strcpy((char\*) pMapa[0].descricao, "Encontraste na entrada do restaurante…O ambiente é muito sombrio avistando-se apenas uma pequena luz a ESTE.\n");

pMapa[0].norte = -1;

pMapa[0].sul = -1;

pMapa[0].este = 1;

pMapa[0].oeste = -1;

pMapa[0].item = -1;

//Construção da sala 1

//----------------------

strcpy((char\*) pMapa[1].descricao, "Entraste numa sala que parece ser a sala de espera dos clientes. Para além de algumas cadeiras e uma televisão existem apenas alguns quadros na parede.\n");

pMapa[1].norte = -1;

pMapa[1].sul = 4;

pMapa[1].este = -1;

pMapa[1].oeste = 0;

pMapa[1].item = -1;

//Construção da sala 2

//----------------------

strcpy((char\*) pMapa[2].descricao, "Abriste a porta com dificuldade e entraste no que parece ser o armazém do restaurante. Nas prateleiras encontram-se inúmeros frascos com lagartos, insectos e uma arca frigorífica onde consegues ver alguns gatos…\n");

pMapa[2].norte = -1;

pMapa[2].sul = 5;

pMapa[2].este = -1;

pMapa[2].oeste = -1;

pMapa[2].item = -1;

//Construção da sala 3

//----------------------

strcpy((char\*) pMapa[3].descricao, "Encontras-te numa das salas de jantar do restaurante. As mesas estão todas postas e prontas para a hora de jantar. Olhas ao fundo e vez o que parecem ser umas escadas para o 2º andar.\n");

pMapa[3].norte = -1;

pMapa[3].sul = 6;

pMapa[3].este = 4;

pMapa[3].oeste = -1;

pMapa[3].item = -1;

//Construção da sala 4

//----------------------

strcpy((char\*) pMapa[4].descricao, "Entras no corredor do restaurante…Sentes o ambiente algo pesado muito por culpa de uma música tradicional chinesa que ouves vinda da sala a ESTE.\n");

pMapa[4].norte = 1;

pMapa[4].sul = -1;

pMapa[4].este = 5;

pMapa[4].oeste = 3;

pMapa[4].item = -1;

//Construção da sala 5

//----------------------

strcpy((char\*) pMapa[5].descricao, "Ao entrares da sala deparaste com uma cena macabra e arrepiante…Encontras o grande jogador chinês a ser torturado de uma forma desumana, sendo obrigado a ouvir a entrevista do Paulo Futre. De imediato o desamarras agradecendo-te este por lhe teres salvo a vida.\n");

pMapa[5].norte = 2;

pMapa[5].sul = -1;

pMapa[5].este = -1;

pMapa[5].oeste = 4;

pMapa[5].item = 0;

//Construção da sala 6

//----------------------

strcpy((char\*) pMapa[6].descricao, "Estás no segundo andar do restaurante. Á tua frente tens umas escadas para o 1º andar. Olhas em teu redor e vez apenas alguns vasos chineses.\n");

pMapa[6].norte = 3;

pMapa[6].sul = -1;

pMapa[6].este = 7;

pMapa[6].oeste = -1;

pMapa[6].item = -1;

//Construção da sala 7

//----------------------

strcpy((char\*) pMapa[7].descricao, "Abres uma porta de correr e entras num pequeno quarto. Á tua direita vez algumas camas por fazer e muita roupa espalhada no chão.\n");

pMapa[7].norte = -1;

pMapa[7].sul = -1;

pMapa[7].este = 8;

pMapa[7].oeste = 6;

pMapa[7].item = -1;

//Construção da sala 8

//----------------------

strcpy((char\*) pMapa[8].descricao, "Aproximaste da porta e sentes um cheiro muito intenso a plástico estilo loja dos chineses! Abres a porta e encontras a fonte do forte cheiro…uma enorme quantidade de caixas com letras chinesas no exterior. Algumas delas então abertas evidenciando o seu conteúdo, camisolas da ''adidaas''\n");

pMapa[8].norte = -1;

pMapa[8].sul = -1;

pMapa[8].este = -1;

pMapa[8].oeste = 7;

pMapa[8].item = -1;

}

}

// testes da inicialização do mapa

void inicializa\_mapa\_teste (struct Celula pMapa[])

{

printf ("inicializa\_mapa\_teste\n");

for (int i = 0; i < MAX\_CELULAS; i++)

{

printf("[%s]\n", pMapa[i].descricao);

printf("Norte: %d\n", pMapa[i].norte);

printf("Sul: %d\n", pMapa[i].sul);

printf("Este: %d\n", pMapa[i].este);

printf("Oeste: %d\n", pMapa[i].oeste);

printf("Item: %d\n\n", pMapa[i].item);

}

system("pause");

}

// testes da inicialização do jogador

void inicializa\_jogador\_teste ( struct Jogador \*pJogador )

{

printf ("inicializa\_jogador\_teste\n");

printf("Nome: %s\n", pJogador->nome);

printf("Energia: %d\n", pJogador->energia);

printf("Localização: %d\n", pJogador->localizacao);

printf("Tesouro: %d\n", pJogador->flg\_tem\_tesouro);

system("pause");

}

// testes da inicialização do monstro

void inicializa\_monstro\_teste ( struct Monstro \*pMonstro )

{

printf ("inicializa\_monstro\_teste\n");

printf("Nome: %s\n", pMonstro->nome);

printf("Energia: %d\n", pMonstro->energia);

printf("Localização: %d\n", pMonstro->localizacao);

system("pause");

}

// inicializa o monstro

void inicializa\_monstro(struct Monstro \*pMonstro)

{

// associa os dados por defeito ao monstro

strcpy((char\*) pMonstro->nome, "Paulo Futre");

pMonstro->energia = 50;

pMonstro->localizacao = 4;

}

// movimenta o monstro

void movimenta\_monstro(struct Monstro \*pMonstro, struct Celula pMapa[])

{

// faz random para decidir se movimenta ou não

int iMovimenta = GetRandomNumber(0, 1);

// é para movimentar

if ( iMovimenta == 1 && pMonstro->localizacao != -1)

{

// verifica quais as saídas existentes na localização actual do monstro

int iLocalizacaoActual = pMonstro->localizacao;

int iEste = pMapa[iLocalizacaoActual].este;

int iNorte = pMapa[iLocalizacaoActual].norte;

int iSul = pMapa[iLocalizacaoActual].sul;

int iOeste = pMapa[iLocalizacaoActual].oeste;

// adiciona as entradas com valor maior ou igual a 0 a um array

unsigned int iSaidas[4];

int iPos = -1;

if (iEste >= 0) { iPos++; iSaidas[iPos]=iEste; }

if (iNorte >= 0) { iPos++; iSaidas[iPos]=iNorte; }

if (iSul >= 0) { iPos++; iSaidas[iPos]=iSul; }

if (iOeste >= 0) { iPos++; iSaidas[iPos]=iOeste; }

// faz random das saídas existentes

int iNovaLocalizacao = iSaidas[GetRandomNumber(0, iPos)];

// altera a localização do monstro

pMonstro->localizacao = iNovaLocalizacao;

}

}

// Aceita comando do jogador

int aceita\_comando\_jogador(char \*sComando, struct Jogador \*pJogador, struct Celula pMapa[])

{

// valida a localização do jogador

int iLocalizacaoJogador = pJogador->localizacao;

int iAccao = -1;

// validar comandos disponíveis

sComando = strupr(sComando);

if ( strcmp(sComando, "N") == 0 ) {

iAccao = pMapa[iLocalizacaoJogador].norte;

}

if ( strcmp(sComando, "S") == 0 ) {

iAccao = pMapa[iLocalizacaoJogador].sul;

}

if ( strcmp(sComando, "E") == 0 ) {

iAccao = pMapa[iLocalizacaoJogador].este;

}

if ( strcmp(sComando, "O") == 0 ) {

iAccao = pMapa[iLocalizacaoJogador].oeste;

}

if ( strcmp(sComando, "G") == 0 ) {

iAccao = 100;

}

if ( strcmp(sComando, "0") == 0 ) {

iAccao = 101;

}

// retorna o movimento que foi feito

return iAccao;

}

// Movimenta ao jogador para a localização pretendida

void movimenta\_jogador(int iLocalizacao, struct Jogador \*pJogador)

{

// validar se o comando recebido está nos comandos disponíveis

if (iLocalizacao >= 0) {

// movimenta jogador

pJogador->localizacao = iLocalizacao;

}

}

// Descreve jogador

void descreve\_jogador(struct Jogador \*pJogador)

{

int nLinhaInicio = 0; //linha da consola

int nColunaInicio = 7; //coluna da consola

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

/\*Guarda os atributos currentes do texto para reposiçãoo no final\*/

CONSOLE\_SCREEN\_BUFFER\_INFO strConsoleInfo;

GetConsoleScreenBufferInfo( hStdout, &strConsoleInfo );

//define a posição do cursor na consula e imprime naquela posição

COORD pos1 = {nColunaInicio, nLinhaInicio};

SetConsoleCursorPosition( hStdout, pos1 );

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

//-----------------------------

COORD pos2 = {nColunaInicio + 8, nLinhaInicio + 1};

SetConsoleCursorPosition( hStdout, pos2 );

/\*Texto e fundo com cores\*/

SetConsoleTextAttribute( hStdout, FOREGROUND\_RED | FOREGROUND\_INTENSITY | BACKGROUND\_INTENSITY | BACKGROUND\_INTENSITY );

printf("DADOS DO JOGADOR");

//-----------------------------

COORD pos3 = {nColunaInicio, nLinhaInicio + 2};

SetConsoleCursorPosition( hStdout, pos3 );

SetConsoleTextAttribute( hStdout, strConsoleInfo.wAttributes );

printf("\*------------------------------\*");

//-----------------------------

COORD pos4 = {nColunaInicio, nLinhaInicio + 3};

SetConsoleCursorPosition( hStdout, pos4 );

printf("\* ");

SetConsoleTextAttribute( hStdout, FOREGROUND\_RED | FOREGROUND\_INTENSITY);

printf("Nome : %s", pJogador->nome);

//----------------------------

COORD pos5 = {nColunaInicio, nLinhaInicio + 4};

SetConsoleCursorPosition( hStdout, pos5 );

SetConsoleTextAttribute( hStdout, strConsoleInfo.wAttributes );

printf("\* ");

SetConsoleTextAttribute( hStdout, FOREGROUND\_RED | FOREGROUND\_INTENSITY);

printf("Energia : %d", pJogador->energia);

//----------------------------

COORD pos6 = {nColunaInicio, nLinhaInicio + 5};

SetConsoleCursorPosition( hStdout, pos6 );

SetConsoleTextAttribute( hStdout, strConsoleInfo.wAttributes );

printf("\*");

//----------------------------

COORD pos7 = {nColunaInicio, nLinhaInicio + 6};

SetConsoleCursorPosition( hStdout, pos7 );

printf("\*------------------------------\*");

COORD pos8 = {nColunaInicio, nLinhaInicio + 7};

SetConsoleCursorPosition( hStdout, pos8 );

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

/\*Repoe os atributos iniciais\*/

SetConsoleTextAttribute( hStdout, strConsoleInfo.wAttributes );

}

// Descreve monstro

void descreve\_monstro(struct Monstro \*pMonstro, struct Celula pMapa[], bool blnSuperUser)

{

int nLinhaInicio = 0; //linha da consola

int nColunaInicio = 38; //coluna da consola

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

/\*Guarda os atributos currentes do texto para reposiçãoo no final\*/

CONSOLE\_SCREEN\_BUFFER\_INFO strConsoleInfo;

GetConsoleScreenBufferInfo( hStdout, &strConsoleInfo );

//define a posição do cursor na consula e imprime naquela posição

COORD pos1 = {nColunaInicio, nLinhaInicio};

SetConsoleCursorPosition( hStdout, pos1 );

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

//---------------------------------

COORD pos2 = {nColunaInicio + 4, nLinhaInicio + 1};

SetConsoleCursorPosition( hStdout, pos2 );

SetConsoleTextAttribute( hStdout, FOREGROUND\_RED | FOREGROUND\_INTENSITY | BACKGROUND\_INTENSITY | BACKGROUND\_INTENSITY );

printf("DESCRIÇÃO DOS MONSTROS");

//---------------------------------

SetConsoleTextAttribute( hStdout, strConsoleInfo.wAttributes );

COORD pos3 = {nColunaInicio, nLinhaInicio + 2};

SetConsoleCursorPosition( hStdout, pos3 );

printf("\*------------------------------\*");

//---------------------------------

COORD pos4 = {nColunaInicio, nLinhaInicio + 3};

SetConsoleCursorPosition( hStdout, pos4 );

printf("\* ");

SetConsoleTextAttribute( hStdout, FOREGROUND\_RED | FOREGROUND\_INTENSITY);

printf("Nome : %s", pMonstro->nome);

//----------------------------------

COORD pos5 = {nColunaInicio, nLinhaInicio + 4};

SetConsoleCursorPosition( hStdout, pos5 );

SetConsoleTextAttribute( hStdout, strConsoleInfo.wAttributes );

printf("\* ");

SetConsoleTextAttribute( hStdout, FOREGROUND\_RED | FOREGROUND\_INTENSITY);

// se o monstro estiver morto informa que morreu

if (pMonstro->energia <= 0)

{

printf("Energia : Morto");

}

else

{

printf("Energia : %d", pMonstro->energia);

}

// se o modo super user estiver activado mostra a sala do monstro

if ( blnSuperUser == true )

{

// escreve a designação da sala se o monstro estiver vivo

if (pMonstro->energia >= 0)

{

COORD pos6 = {nColunaInicio, nLinhaInicio + 5};

SetConsoleCursorPosition( hStdout, pos6 );

SetConsoleTextAttribute( hStdout, strConsoleInfo.wAttributes );

printf("\* ");

SetConsoleTextAttribute( hStdout, FOREGROUND\_RED | FOREGROUND\_INTENSITY);

printf("Localização: %s", pMapa[pMonstro->localizacao].descricao);

}

}

else

{

SetConsoleTextAttribute( hStdout, strConsoleInfo.wAttributes );

COORD pos6 = {nColunaInicio, nLinhaInicio + 5};

SetConsoleCursorPosition( hStdout, pos6 );

printf("\*", pMapa[pMonstro->localizacao].descricao);

}

SetConsoleTextAttribute( hStdout, strConsoleInfo.wAttributes );

COORD pos7 = {nColunaInicio, nLinhaInicio + 6};

SetConsoleCursorPosition( hStdout, pos7 );

printf("\*------------------------------\*");

//-----------------------------------------

COORD pos8 = {nColunaInicio, nLinhaInicio + 7};

SetConsoleCursorPosition( hStdout, pos8 );

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

}

// Desenha no ecrã o que ocorre no jogo

void descreve\_status(struct Jogador \*pJogador, struct Monstro \*pMonstro, struct Celula pMapa[], bool blnSuperUser)

{

int MAX\_LARGURA = 73;

system("cls"); // limpa ecrã

descreve\_jogador(pJogador);

printf("\n");

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

printf("| DESCRIÇÃO DA LOCALIZAÇÃO |\n");

printf("| |\n");

printf("| ");

// imprime a descrição em conjuntos de 73 caracteres

int i = 0;

char sLocalizacao[ MAX\_COL ];

strcpy((char\*) sLocalizacao, (char\*) pMapa[pJogador->localizacao].descricao);

while( sLocalizacao[i] != '\n')

{

printf("%c", sLocalizacao[i]);

i++;

if ( (i % MAX\_LARGURA) == 0)

{

printf(" |\n| ");

}

}

int x = 0;

//printf("i:%d, largura:%d, conta:%d", i, MAX\_LARGURA, (i % MAX\_LARGURA));

for (x; x <= (MAX\_LARGURA - ((i) % MAX\_LARGURA)); x++)

{

printf(" ");

}

printf("|\n");

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

// se o modo superUser estiver activado mostra o status do monstro

if (blnSuperUser == true)

{

descreve\_monstro(pMonstro, pMapa, blnSuperUser);

}

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

}

// Simula a luta entre jogador e monstro

int lutar(struct Jogador \*pJogador, struct Monstro \*pMonstro, struct Celula pMapa[], bool blnSuperUser)

{

int MAX\_DANO = 30;

int iReturn = 1;

int iHandyCap = 0; // desvantagem do monstro

// valida se o modo é superUser

if (blnSuperUser == true)

{

iHandyCap = 10;

}

// descreve status

system("cls");

descreve\_jogador(pJogador);

descreve\_monstro(pMonstro, pMapa, blnSuperUser);

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

printf("| COMBATE\n");

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

// imprime legenda

printf(" @ - 1\n");

printf("3 - /|\\ - 4\n");

printf(" / \\ - 2\n");

// solicita a posição para atacar (cima, baixo, esquerda, direita, foge)

int iJogadorAtaque;

printf("Qual a posição em que quer atacar?:");

scanf( "%d", &iJogadorAtaque );

// solicita a posição para defender

int iJogadorDefesa;

printf("Qual a posição em que quer defender?:");

scanf( "%d", &iJogadorDefesa );

// random de da posição onde o monstro defende/ataca

// (cima, baixo, esquerda, direita, foge)

int iMonstroAtaque = GetRandomNumber(0, 4);

int iMonstroDefesa = GetRandomNumber(0, 4 + iHandyCap);

// calcula dano

int iDanoJogador = GetRandomNumber(1, MAX\_DANO);

int iDanoMonstro = GetRandomNumber(1, MAX\_DANO - iHandyCap);

// Simula a luta

// ATAQUE

if (iJogadorAtaque == iMonstroDefesa)

{

printf("O monstro defendeu o ataque!\n");

}

else

{

// desconta o dano no monstro

printf("Causou %d pontos de dano no monstro %s\n", iDanoJogador, pMonstro->nome);

pMonstro->energia = pMonstro->energia - iDanoJogador;

}

// DEFESA

if (iMonstroAtaque == iJogadorDefesa)

{

printf("O jogador defendeu o ataque!\n");

}

else

{

// desconta o dano no jogador

printf("O monstro %s causou-lhe %d pontos de dano\n", pMonstro->nome, iDanoMonstro);

pJogador->energia = pJogador->energia - iDanoMonstro;

}

// Valida se a luta terminou

if (pJogador->energia <= 0)

{

// o monstro matou o jogador

printf("Infelizmente o monstro %s matou-o!\n", pMonstro->nome);

iReturn = 0;

}

if (pMonstro->energia <= 0)

{

// o jogador matou o monstro

printf("Parabéns! Matou o monstro %s!\n", pMonstro->nome);

// remove o monstro do mapa

pMonstro->localizacao = -1;

iReturn = 0;

}

// pausa se a luta terminou

system("pause");

return iReturn;

}

// Tenta apanhar o tesouro

bool apanha\_tesouro(struct Jogador \*pJogador, struct Celula pMapa[])

{

if (pMapa[pJogador->localizacao].item == 0)

{

// define o jogador como tendo o tesouro

pJogador->flg\_tem\_tesouro = 0;

// retira o tesouro da célula

pMapa[pJogador->localizacao].item = -1;

// o tesouro foi apanhado

return true;

}

// o tesouro não foi apanhado

return false;

}

// valida os comandos disponíveis para o jogador

char\* valida\_comandos\_disponiveis(struct Celula \*pMapa)

{

int MAX\_CHARS = 200;

char\* sComandosDisponiveis;

sComandosDisponiveis = (char \*) malloc( MAX\_CHARS );

strcpy((char\*) sComandosDisponiveis, "+------------------------------\n");

// valida Norte

if (pMapa->norte >= 0){ strcat((char\*) sComandosDisponiveis, "| N - Norte\n");}

// valida Sul

if (pMapa->sul >= 0) { strcat((char\*) sComandosDisponiveis, "| S - Sul\n"); }

// valida Este

if (pMapa->este >= 0) { strcat((char\*) sComandosDisponiveis, "| E - Este\n"); }

// valida Oeste

if (pMapa->oeste >= 0){ strcat((char\*) sComandosDisponiveis, "| O - Oeste\n");}

// Opções funcionais

// adiciona return

strcat((char\*) sComandosDisponiveis, "|\n");

// Gravar Jogo

strcat((char\*) sComandosDisponiveis, "| G - Gravar Jogo\n");

// sai do Jogo

strcat((char\*) sComandosDisponiveis, "| 0 - Sair do Jogo\n");

return (char\*) sComandosDisponiveis;

}

// valida switches

void validaSwitches(int argc, char\* args[], bool \*pSuperUser, char \*pFicheiroMapa)

{

bool godMode = false;

for (int i = 1; i < argc; i++)

{

if ( strcmp(strupr(args[i]), "SU") == 0 ) { \*pSuperUser = true; } // modo SuperUser

if ( strcmp(strupr(args[i]), "M") == 0 ) { strcpy(pFicheiroMapa, args[i+1]); } // faz load do mapa

}

}

// Se a Localização do Jogador for igual à do Monstro

void valida\_condicoes\_luta(struct Jogador \*pJogador, struct Monstro \*pMonstro, struct Celula pMapa[], bool blnSuperUser)

{

if (pJogador->localizacao == pMonstro->localizacao)

{

// avisa o jogador que se encontra na mesma localização que o monstro e que tem de lutar

printf("Encontraste o agente do jogador ao entrares na sala! Inicia-se uma violenta batalha...\n");

system("pause");

int iResultado = 1;

// Lutar

while (iResultado != 0)

{

// descreve status

iResultado = lutar(pJogador, pMonstro, pMapa, blnSuperUser);

}

}

}

// grava o estado actual do jogo

void grava\_jogo(struct Jogador \*pJogador, struct Monstro \*pMonstro, struct Celula pMapa[])

{

printf("A gravar o jogo...\n");

FILE \*f; // ficheiro

char l[ MAX\_COL ]; // linha

char\* sResposta[2];

strcpy((char\*) sResposta, "");

// valida se já existe uma gravação

char\* sNomeFicheiro[50];

strcpy((char\*) sNomeFicheiro, (char\*) pJogador->nome);

// tenta abrir um ficheiro de gravação já existente

f = fopen( strcat((char\*) sNomeFicheiro, ".txt"), "rb");

if (f != NULL )

{

// se existir pergunta se deseja gravar por cima

while (strcmp(strupr((char\*) sResposta), "S") != 0 && strcmp(strupr((char\*) sResposta), "N") != 0)

{

printf("Já existe um ficheiro de gravação para este utilizador, deseja gravar por cima?(S/N):");

scanf( "%s", sResposta );

}

}

if (strcmp(strupr((char\*) sResposta), "S") == 0 || f == NULL )

{

// abre o ficheiro para escrita

f = fopen((char\*) sNomeFicheiro, "wb");

// grava estrutura do jogador

//---------------------------

fwrite(":JOGADOR", sizeof(char), MAX\_COL, f); // cabeçalho :JOGADOR

strcpy(l, trim((char\*) pJogador->nome)); // nome

fwrite(l, sizeof(char), MAX\_COL, f);

itoa(pJogador->energia, l, 10); // energia

fwrite(trim(l), sizeof(char), MAX\_COL, f);

itoa(pJogador->localizacao, l, 10); // localização

fwrite(trim(l), sizeof(char), MAX\_COL, f);

itoa(pJogador->flg\_tem\_tesouro, l, 10); // tesouro

fwrite(trim(l), sizeof(char), MAX\_COL, f);

// grava estrutura do monstro

//---------------------------

fwrite(":MONSTRO", sizeof(char), MAX\_COL, f); // cabeçalho :MONSTRO

strcpy(l, trim((char\*) pMonstro->nome)); // nome

fwrite(l, sizeof(char), MAX\_COL, f);

itoa(pMonstro->energia, l, 10); // energia

fwrite(trim(l), sizeof(char), MAX\_COL, f);

itoa(pMonstro->localizacao, l, 10); // localização

fwrite(trim(l), sizeof(char), MAX\_COL, f);

// grava estrutura do mapa

//------------------------

fwrite(":MAPA", sizeof(char), MAX\_COL, f); // cabeçalho :MAPA

// grava cada uma das células do mapa

for (int i = 0; i < MAX\_CELULAS; i++)

{

strcpy(l, (char\*) pMapa[i].descricao); // nome da célula

fwrite(l, sizeof(char), MAX\_COL, f);

itoa(pMapa[i].norte, l, 10); // norte

fwrite(l, sizeof(char), MAX\_COL, f);

itoa(pMapa[i].sul, l, 10); // sul

fwrite(l, sizeof(char), MAX\_COL, f);

itoa(pMapa[i].este, l, 10); // este

fwrite(trim(l), sizeof(char), MAX\_COL, f);

itoa(pMapa[i].oeste, l, 10); // oeste

fwrite(l, sizeof(char), MAX\_COL, f);

itoa(pMapa[i].item, l, 10); // item

fwrite(l, sizeof(char), MAX\_COL, f);

}

// fecha o ficheiro

fclose(f);

printf("Jogo gravado com sucesso!");

system("pause");

}

}

// testa o final do jogo

bool testa\_fim\_jogo(struct Jogador \*pJogador)

{

return !(pJogador->energia > 0 && !(pJogador->flg\_tem\_tesouro == 0 && pJogador->localizacao == 0));

}

// novo jogo

void novo\_jogo(struct Jogador \*pJogador, struct Monstro \*pMonstro, struct Celula pMapa[], bool blnSuperUser, char\* pFicheiroMapa)

{

//--------------------------------

//Chama a função para inicializar o jogador

//--------------------------------

//jogador = ( struct Jogador )malloc( sizeof ( struct Jogador ) );

//struct jsw\_node \*rn = ( struct jsw\_node\*)malloc(sizeof(struct jsw\_node));

char \* nomeJogador[50];

// solicita o nome do jogador

printf( "Indique o nome do jogador:" );

scanf( "%s", &nomeJogador );

inicializa\_jogador(pJogador, (char \*) nomeJogador, blnSuperUser );

//--------------------------------

//Chama a função para inicializar o mapa

//--------------------------------

inicializa\_mapa( pMapa, (char \*) pFicheiroMapa );

//inicializa\_mapa\_teste (pMapa);

//system("pause");

//--------------------------------

//Chama a função para inicializar o monstro

//--------------------------------

inicializa\_monstro( pMonstro );

}

// carrega um jogo previamente gravado

void carrega\_jogo(struct Jogador \*pJogador, struct Monstro \*pMonstro, struct Celula pMapa[])

{

printf("A carregar o jogo...\n");

FILE \*f;

char l[ MAX\_COL ];

// carrega o jogo com o nome do jogador

char\* sNomeFicheiro[50];

strcpy((char\*) sNomeFicheiro, (char\*) pJogador->nome);

f = fopen( strcat((char\*) sNomeFicheiro, ".txt"), "rb" );

if ( f != NULL )

{

// lê o conteúdo do ficheiro

while( fread(l, sizeof(char), MAX\_COL, f) != NULL ){

// se a linha for igual a mapa

if (strcmp(strupr(l),":MAPA") == 0)

{

// carrega cada uma das células do mapa

for (int i = 0; i < MAX\_CELULAS; i++)

{

fread(l, sizeof(char), MAX\_COL, f);

strcpy((char\*) pMapa[i].descricao, l); // nome da célula

fread(l, sizeof(char), MAX\_COL, f);

pMapa[i].norte = atoi(l); // norte

fread(l, sizeof(char), MAX\_COL, f);

pMapa[i].sul = atoi(l); // sul

fread(l, sizeof(char), MAX\_COL, f);

pMapa[i].este = atoi(l); // este

fread(l, sizeof(char), MAX\_COL, f);

pMapa[i].oeste = atoi(l); // oeste

fread(l, sizeof(char), MAX\_COL, f);

pMapa[i].item = atoi(l); // item

}

}

if (strcmp(strupr(l),":JOGADOR") == 0)

{

fread(l, sizeof(char), MAX\_COL, f);

strcpy((char\*) pJogador->nome, l); // nome do jogador

fread(l, sizeof(char), MAX\_COL, f);

pJogador->energia = atoi (l); // energia

fread(l, sizeof(char), MAX\_COL, f);

pJogador->localizacao = atoi (l); // localizacao

fread(l, sizeof(char), MAX\_COL, f);

pJogador->flg\_tem\_tesouro = atoi (l); // tesouro

}

if (strcmp(strupr(l),":MONSTRO") == 0)

{

fread(l, sizeof(char), MAX\_COL, f);

strcpy((char\*) pMonstro->nome, l); // nome do monstro

fread(l, sizeof(char), MAX\_COL, f);

pMonstro->energia = atoi (l); // energia

fread(l, sizeof(char), MAX\_COL, f);

pMonstro->localizacao = atoi (l); // localizacao

}

}

fclose( f );

printf("Jogo carregado com sucesso!\n");

system("pause");

}

else

{

printf("Ficheiro de jogo inválido!\n");

system("pause");

}

}

// executa os comandos funcionais

void comandos\_funcionais(int iAccao, struct Jogador \*pJogador, struct Monstro \*pMonstro, struct Celula pMapa[])

{

// grava jogo

if (iAccao == 100) {

grava\_jogo(pJogador, pMonstro, pMapa);

}

// sai do jogo

if (iAccao == 101) {

printf("sai do jogo\n");

system("pause");

exit(EXIT\_SUCCESS);

}

}

// inicia jogo

void inicia\_jogo(struct Jogador \*pJogador, struct Monstro \*pMonstro, struct Celula pMapa[], bool blnSuperUser)

{

//--------------------------------

// Inicia o jogo

//--------------------------------

//Enquanto não for Fim de Jogo

while (testa\_fim\_jogo(pJogador) == false)

{

int iAccao = -1;

while (iAccao < 0)

{

// Descrever a Localização do Jogador

descreve\_status(pJogador, pMonstro, pMapa, blnSuperUser);

// valida se o monstro encontrou o jogador, se encontrou inicia a luta

valida\_condicoes\_luta(pJogador, pMonstro, pMapa, blnSuperUser);

// Aceitar Comando do Jogador

char\* sComando[2];

strcpy((char\*) sComando, "");

// validar/imprimir comandos disponíveis

char\* sComandosDisponiveis = valida\_comandos\_disponiveis(&pMapa[pJogador->localizacao]);

printf("%s\n", sComandosDisponiveis);

// solicita comando ao jogador

printf("Insira um Comando: ");

scanf( "%s", sComando );

iAccao = aceita\_comando\_jogador((char\*) sComando, pJogador, pMapa);

// se o comando for inválido dá mensagem de erro

if (iAccao < 0)

{

printf("Comando inválido!\n");

system("pause");

}

}

// se a acção for igual ou superior a 100 é uma acção funcional

if (iAccao >= 100)

{

comandos\_funcionais(iAccao, pJogador, pMonstro, pMapa);

}

else

{

// Movimentar Monstro

movimenta\_monstro(pMonstro, pMapa);

// Movimentar Jogador

movimenta\_jogador(iAccao, pJogador);

// apanha o tesouro

bool tesouro = apanha\_tesouro(pJogador, pMapa);

if (tesouro == true)

{

//printf("Apanhou o tesouro! %d\n", jogador.flg\_tem\_tesouro);

//system("pause");

}

// valida localizações para luta

valida\_condicoes\_luta(pJogador, pMonstro, pMapa, blnSuperUser);

}

}

// terminou o jogo

system("cls");

if (pJogador->energia > 0)

{

// ganhou

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

printf("| Sais rapidamente do restaurante levando contigo o grandioso jogador. A |\n");

printf("| notícia espalhou-se rapidamente por tudo o mundo e o seu já se encontra |\n");

printf("| inundado de charters…Cumpriste o teu objectivo e salvastes a época do |\n");

printf("| clube! |\n");

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

}

else

{

// perdeu

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

printf("| Infelizmente não foste capaz de salvar o grande jogador chinês! |\n");

printf("| Os charters de chineses nunca virão a ser o que se esperava! |\n");

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

}

printf("\n\n");

printf("\_\_\_\_ \_ \_ \_ \_\_\_ \_\_\_\_ \_ \_\_\_\_ \_\_\_\_ \_\_\_\_ /\n");

printf("|\_\_\_ | |\\/| | \\ | | | | | | \_\_ | | / \n");

printf("| | | | |\_\_/ |\_\_| \_| |\_\_| |\_\_] |\_\_| . \n");

system( "pause" );

}

// menu principal

void menu\_principal()

{

system("cls");

// escreve a história do jogo

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

printf("| Estamos no ano de 2010 e o Sport Lisboa e Benfica está a sofrer uma das |\n");

printf("| piores épocas de sempre. A sua única esperança é conseguir contratar o |\n");

printf("| maior jogador chinês da actualidade de modo a ganhar vantagem na luta |\n");

printf("| contra os seus mais directos adversários no campeonato. Para isso é |\n");

printf("| necessário resgatar o jogador que se encontra preso pelo seu agente num |\n");

printf("| restaurante chinês. A tarefa não vai ser fácil…mas não é impossível! |\n");

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

// adiciona uns espaços antes do menú

printf("\n\n");

// escreve o menu

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

printf("| 1 - Novo Jogo |\n");

printf("| 2 - Carregar Jogo |\n");

printf("| 3 - Converter mapa para binário |\n");

printf("| |\n");

printf("| 0 - Sair |\n");

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

}

// The one and only application object

CWinApp theApp;

using namespace std;

int main(int argc, char \* argv[], char \* envp[])

{

int nRetCode = 0;

HMODULE hModule = ::GetModuleHandle(NULL);

if (hModule != NULL)

{

// initialize MFC and print and error on failure

if (!AfxWinInit(hModule, NULL, ::GetCommandLine(), 0))

{

// TODO: change error code to suit your needs

\_tprintf(\_T("Fatal Error: MFC initialization failed\n"));

nRetCode = 1;

}

else

{

int iResultado;

setlocale( LC\_ALL, "Portuguese" );

srand(time(0));

//--------------------------------

//Chama a função para validar os switches

//--------------------------------

bool blnSuperUser = false; // define se está ou não no modo superuser

char\* ficheiroMapa[100]; // nome do ficheiro do mapa

strcpy((char\*) ficheiroMapa, "");

validaSwitches(argc, argv, &blnSuperUser, (char\*) ficheiroMapa);

//--------------------------------

// Define as estruturas a serem utilizadas no jogo

//--------------------------------

struct Celula mapa[MAX\_CELULAS]; //Cria um array de células para o mapa do jogo

struct Jogador jogador;

struct Monstro monstro;

//--------------------------------

// Chama a função que imprime o menu principal

//--------------------------------

int iOpcao = -1;

bool blnOpcaoMenu = false;

while (iOpcao != 0)

{

// imprime menu

menu\_principal();

// solicita a opção ao jogador

printf("Escolha um opção:");

scanf( "%d", &iOpcao );

// Valida a opção escolhida

switch ( iOpcao )

{

case 0: // Sai do jogo

printf("Saiu do jogo");

break;

case 1: // Novo jogo

novo\_jogo(&jogador, &monstro, mapa, blnSuperUser, (char\*) ficheiroMapa);

inicia\_jogo(&jogador, &monstro, mapa, blnSuperUser);

break;

case 2: // Carregar jogo

// solicita o nome do jogo

printf("Qual o nome do jogo?:");

scanf( "%s", &jogador.nome );

carrega\_jogo(&jogador, &monstro, mapa);

inicia\_jogo(&jogador, &monstro, mapa, blnSuperUser);

break;

case 3: // Converter mapa para binário

converte\_ficheiro();

break;

}

}

return 0;

}

}

else

{

// TODO: change error code to suit your needs

\_tprintf(\_T("Fatal Error: GetModuleHandle failed\n"));

nRetCode = 1;

}

return nRetCode;

}

Anexo 2: Código da DLL

//Descritivo:Primeiro trabalho de grupo da disciplina de SO

//Titulo:DLL para o jogo de aventura

//Autores: Luís Costa Nº6032, Bruno Moreira Nº6170

#include "stdafx.h"

#include "JogoSODLL.h"

#ifdef \_DEBUG

#define new DEBUG\_NEW

#endif

BEGIN\_MESSAGE\_MAP(CJogoSODLLApp, CWinApp)

END\_MESSAGE\_MAP()

// CJogoSODLLApp construction

CJogoSODLLApp::CJogoSODLLApp()

{

// TODO: add construction code here,

// Place all significant initialization in InitInstance

}

// The one and only CJogoSODLLApp object

CJogoSODLLApp theApp;

// CJogoSODLLApp initialization

BOOL CJogoSODLLApp::InitInstance()

{

CWinApp::InitInstance();

return TRUE;

}

\_\_declspec( dllexport ) char\* WINAPI rtrim(char \*s)

{

char\* back = s + strlen(s);

while(isspace(\*--back));

\*(back+1) = '\0';

return s;

}

\_\_declspec( dllexport ) char \* WINAPI ltrim(char \*s)

{

while(isspace(\*s)) s++;

return s;

}

\_\_declspec( dllexport ) char \* WINAPI trim(char \*s)

{

return rtrim(ltrim(s));

}

// Generate a random number between nLow and nHigh (inclusive)

\_\_declspec( dllexport ) unsigned int WINAPI GetRandomNumber(int nLow, int nHigh)

{

return (rand() % (nHigh - nLow + 1)) + nLow;

}

\_\_declspec( dllexport ) void WINAPI converte\_ficheiro()

{

#define MAX\_COL 400

printf("A converter o ficheiro para binário...");

system("pause");

FILE \*forigem; // ficheiro de origem

FILE \*fdestino; // ficheiro dedestino

char l[ MAX\_COL ]; // linha

char\* nomeMapaOrigem[100]; // nome do ficheiro do mapa

char\* nomeMapaDestino[100];

strcpy((char\*) nomeMapaOrigem, "");

strcpy((char\*) nomeMapaDestino, "");

//Pede o nome do ficheiro do mapa a converte

printf("Nome do ficheiro a converter (não indicar extensão):");

scanf("%s", nomeMapaOrigem);

strcpy((char\*) nomeMapaDestino, (char\*) nomeMapaOrigem);

//Abre o ficheiro do mapa de origem

forigem = fopen( strcat((char\*) nomeMapaOrigem, ".txt"), "r");

//Abre o ficheiro do mapa de destino em binário

fdestino = fopen( strcat((char\*) nomeMapaDestino, "b.txt"), "wb");

if (forigem != NULL)

{

if (fdestino != NULL)

{

// percorre o ficheiro de origem até encontrar o fim

while( fgets(l, MAX\_COL, forigem) != NULL )

{

//grava a linha no ficheiro binário

fwrite(l, sizeof(char), MAX\_COL, fdestino);

}

fclose(fdestino);

}

else

{

printf("Erro a abrir/criar o ficheiro binário/n");

}

fclose(forigem);

printf("Ficheiro convertido com sucesso!/n");

system("pause");

}

else

{

printf("Erro a ler o ficheiro %s/n", nomeMapaOrigem);

}

}