#include <stdio.h>

#include <stdbool.h>

#include <stdlib.h>

#include <string.h>

#include <time.h>

#define MAP\_SIZE\_1 10

#define MAP\_SIZE\_2 20

typedef struct {

int x;

int y;

} Player;

void limparBuffer() {

int c;

while ((c = getchar()) != '\n' && c != EOF);

}

void mostrarTutorial() {

system("cls");

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

printf("# TUTORIAL #\n");

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

printf("Bem-vindo ao tutorial!\n");

printf("Use w, a, s, d para mover o jogador.\n");

printf("O jogador interage com o objeto que esta em cima (i).\n");

printf("&: Simbolo que representa o jogador.\n");

printf("\*: Simbolo que representa uma parede, o jogador ao se movimentar nao pode passar pela parede.\n");

printf("@: Simbolo que representa a chave para abrir a porta para finalizar a fase, a porta abre no momento que o jogador interage com a chave.\n");

printf("=: Simbolo que representa a porta aberta quando o jogador interage com a chave.\n");

printf("O: Simbolo que representa um botao que o usuario pode interagir, o botao fica no chao e o jogador deve ficar em cima dele para poder interagir.\n");

printf("#: Simbolo que representa um espinho. A fase e reiniciada quando o jogador toca no espinho, caso a fase seja reiniciada tres vezes, o jogo volta para o menu principal.\n");

printf(">: Simbolo que representa um teletransporte. O teletransporte sempre deve vir em pares, quando o jogador toca em um ele e transportado para o outro e vice-versa.\n");

printf("X: Simbolo que representa o monstro nivel 1. Esse monstro tem um movimento aleatorio para cima, para esquerda, para baixo e para direita. Caso o monstro toque no jogador, a fase e reiniciada.\n");

printf("V: Simbolo que representa o monstro nivel 2. Esse monstro tem uma inteligencia para seguir o jogador (ha varias maneiras de implementar isso). Caso o monstro toque no jogador, a fase e reiniciada.\n");

printf("Pressione qualquer tecla para voltar ao menu.\n");

getchar();

limparBuffer();

}

void movimentoAleatorio(int \*dx, int \*dy) {

int direcao = rand() % 4;

switch (direcao) {

case 0:

\*dx = -1;

\*dy = 0;

break;

case 1:

\*dx = 1;

\*dy = 0;

break;

case 2:

\*dx = 0;

\*dy = -1;

break;

case 3:

\*dx = 0;

\*dy = 1;

break;

}

}

void movimentarMonstro(int \*x, int \*y, char map[MAP\_SIZE\_2][MAP\_SIZE\_2]) {

int dx, dy;

movimentoAleatorio(&dx, &dy);

// Verificar se a próxima posição é válida

int new\_x = \*x + dx;

int new\_y = \*y + dy;

if (new\_x >= 0 && new\_x < MAP\_SIZE\_2 && new\_y >= 0 && new\_y < MAP\_SIZE\_2 && map[new\_x][new\_y] != '\*') {

\*x = new\_x;

\*y = new\_y;

}

}

bool toquesMonstro(int player\_x, int player\_y, int monstro\_x, int monstro\_y) {

return (player\_x == monstro\_x && player\_y == monstro\_y);

}

void jogarFase2();

void jogarFase1() {

char map[MAP\_SIZE\_1][MAP\_SIZE\_1] = {

{'\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*'},

{'\*', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '\*'},

{'\*', ' ', '\*', '\*', '\*', '\*', '\*', ' ', ' ', '\*'},

{'\*', ' ', ' ', ' ', ' ', ' ', '\*', ' ', ' ', '\*'},

{'\*', ' ', '\*', '\*', ' ', '\*', '\*', ' ', ' ', '\*'},

{'\*', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '\*'},

{'\*', ' ', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*'},

{'\*', '@', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'D'},

{'\*', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '\*'},

{'\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*'}

};

Player player = {1, 1};

char comando[5];

int chave\_x = 7;

int chave\_y = 1;

bool chavePega = false;

bool portaDestrancada = false;

int monstro\_x = 5; // Posição inicial do monstro (linha)

int monstro\_y = 5; // Posição inicial do monstro (coluna)

int toquesMonstroCount = 0; // Contador de toques no monstro

int espinhosCount = 0; // Contador de toques nos espinhos

int i, j;

while (true) {

system("cls");

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

printf("# FASE 1 #\n");

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

for (i = 0; i < MAP\_SIZE\_1; i++) {

for (j = 0; j < MAP\_SIZE\_1; j++) {

if (i == player.x && j == player.y)

printf("& ");

else

printf("%c ", map[i][j]);

}

printf("\n");

}

printf("\nDigite um comando (w, a, s, d para mover; i para interagir; two2 para pular para a Fase 2): ");

scanf("%4s", comando);

limparBuffer();

int dx = 0, dy = 0;

if (strcmp(comando, "w") == 0)

dx = -1;

else if (strcmp(comando, "a") == 0)

dy = -1;

else if (strcmp(comando, "s") == 0)

dx = 1;

else if (strcmp(comando, "d") == 0)

dy = 1;

else if (strcmp(comando, "i") == 0) {

if (player.x == chave\_x && player.y == chave\_y && !chavePega) {

printf("Voce pegou a chave!\n");

chavePega = true;

map[chave\_x][chave\_y] = ' ';

portaDestrancada = true;

// Abrir as portas automaticamente quando a chave é pega

for (i = 0; i < MAP\_SIZE\_1; i++) {

for (j = 0; j < MAP\_SIZE\_1; j++) {

if (map[i][j] == 'D')

map[i][j] = '=';

}

}

} else if (map[player.x][player.y] == 'D' && chavePega) {

printf("Voce abriu a porta!\n");

map[player.x][player.y] = '=';

portaDestrancada = true;

}

} else if (strcmp(comando, "two2") == 0) {

jogarFase2();

return;

} else {

printf("Comando invalido. Tente novamente.\n");

}

int new\_x = player.x + dx;

int new\_y = player.y + dy;

if (new\_x >= 0 && new\_x < MAP\_SIZE\_1 && new\_y >= 0 && new\_y < MAP\_SIZE\_1 && map[new\_x][new\_y] != '\*') {

player.x = new\_x;

player.y = new\_y;

}

if (map[player.x][player.y] == 'D' && !portaDestrancada) {

printf("A porta esta trancada! Encontre a chave para destranca-la.\n");

}

if (player.x == 7 && player.y == MAP\_SIZE\_1 - 2 && portaDestrancada) {

printf("Voce atravessou a porta e passou para a Fase 2!\n");

jogarFase2();

return;

}

if (toquesMonstro(player.x, player.y, monstro\_x, monstro\_y)) {

printf("Voce tocou no monstro! A fase sera reiniciada.\n");

toquesMonstroCount++;

if (toquesMonstroCount >= 3) {

printf("Voce tocou no monstro 3 vezes! Game Over.\n");

printf("Pressione qualquer tecla para voltar ao menu principal.\n");

getchar();

limparBuffer();

return;

}

player.x = 1;

player.y = 1;

chavePega = false;

portaDestrancada = false;

}

}

}

void jogarFase2() {

char map[MAP\_SIZE\_2][MAP\_SIZE\_2] = {

{'\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*'},

{'\*', ' ', ' ', ' ', 'X', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '\*'},

{'\*', ' ', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', ' ', '\*'},

{'\*', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '\*'},

{'\*', ' ', '\*', '\*', ' ', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', ' ', '\*'},

{'\*', ' ', ' ', ' ', '#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '\*'},

{'\*', ' ', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', ' ', '\*'},

{'\*', '@', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '\*'},

{'\*', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '\*'},

{'\*', ' ', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', ' ', '\*'},

{'\*', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '\*'},

{'\*', ' ', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', ' ', '\*'},

{'\*', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '\*'},

{'\*', ' ', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', ' ', '\*'},

{'\*', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '\*'},

{'\*', ' ', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', ' ', '\*'},

{'\*', ' ', ' ', ' ', '#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '\*'},

{'\*', ' ', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', ' ', '\*'},

{'\*', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '\*'},

{'\*', '\*', 'D', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*'}

};

Player player = {1, 1};

char comando[5];

int chave\_x = 7;

int chave\_y = 1;

bool chavePega = false;

bool portaDestrancada = false;

int monstro\_x = 10;

int monstro\_y = 10;

int toquesMonstroCount = 0;

int espinhosCount = 0;

int i, j;

while (true) {

system("cls");

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

printf("# FASE 2 #\n");

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

for (i = 0; i < MAP\_SIZE\_2; i++) {

for (j = 0; j < MAP\_SIZE\_2; j++) {

if (i == player.x && j == player.y)

printf("& ");

else

printf("%c ", map[i][j]);

}

printf("\n");

}

printf("\nDigite um comando (w, a, s, d para mover; i para interagir): ");

scanf("%4s", comando);

limparBuffer();

int dx = 0, dy = 0;

if (strcmp(comando, "w") == 0)

dx = -1;

else if (strcmp(comando, "a") == 0)

dy = -1;

else if (strcmp(comando, "s") == 0)

dx = 1;

else if (strcmp(comando, "d") == 0)

dy = 1;

else if (strcmp(comando, "i") == 0) {

if (player.x == chave\_x && player.y == chave\_y && !chavePega) {

printf("Voce pegou a chave!\n");

chavePega = true;

map[chave\_x][chave\_y] = ' ';

portaDestrancada = true;

// Abrir as portas automaticamente quando a chave é pega

for (i = 0; i < MAP\_SIZE\_2; i++) {

for (j = 0; j < MAP\_SIZE\_2; j++) {

if (map[i][j] == 'D')

map[i][j] = '=';

}

}

} else if (map[player.x][player.y] == 'D' && chavePega) {

printf("Voce abriu a porta!\n");

map[player.x][player.y] = '=';

portaDestrancada = true;

}

} else {

printf("Comando invalido. Tente novamente.\n");

}

int new\_x = player.x + dx;

int new\_y = player.y + dy;

if (new\_x >= 0 && new\_x < MAP\_SIZE\_2 && new\_y >= 0 && new\_y < MAP\_SIZE\_2 && map[new\_x][new\_y] != '\*') {

player.x = new\_x;

player.y = new\_y;

}

if (map[player.x][player.y] == 'D' && !portaDestrancada) {

printf("A porta esta trancada! Encontre a chave para destranca-la.\n");

}

if (player.x == 7 && player.y == MAP\_SIZE\_2 - 2 && portaDestrancada) {

printf("Voce atravessou a porta e concluiu o jogo!\n");

printf("Pressione qualquer tecla para voltar ao menu principal.\n");

getchar();

limparBuffer();

return;

}

if (map[player.x][player.y] == '#') {

printf("");

espinhosCount++;

if (espinhosCount >= 3) {

printf("-----------\n|GAME OVER|\n-----------");

printf("Pressione qualquer tecla para voltar ao menu principal.\n");

getchar();

limparBuffer();

return;

}

player.x = 1;

player.y = 1;

chavePega = false;

portaDestrancada = false;

}

if (toquesMonstro(player.x, player.y, monstro\_x, monstro\_y)) {

printf("");

toquesMonstroCount++;

if (toquesMonstroCount >= 3) {

printf("-----------\n|GAME OVER|\n-----------");

printf("Pressione qualquer tecla para voltar ao menu principal.\n");

getchar();

limparBuffer();

return;

}

player.x = 1;

player.y = 1;

chavePega = false;

portaDestrancada = false;

}

movimentarMonstro(&monstro\_x, &monstro\_y, map);

}

}

void exibirMenu() {

int opcao;

do {

system("cls");

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

printf("# MENU #\n");

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

printf("1. Jogar\n");

printf("2. Tutorial\n");

printf("3. Sair\n\n");

printf("Escolha uma opcao: ");

scanf("%d", &opcao);

limparBuffer();

switch (opcao) {

case 1:

jogarFase1();

break;

case 2:

mostrarTutorial();

break;

case 3:

printf("Saindo do jogo...\n");

break;

default:

printf("Opcao invalida. Tente novamente.\n");

}

} while (opcao != 3);

}

int main() {

srand(time(NULL));

exibirMenu();

return 0;

}

—----------------------------------------------------------------------------------------------------------------------

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h> // Necessário para utilizar o tipo bool

#define WIDTH 40

#define HEIGHT 40

char map[HEIGHT][WIDTH + 1] = {

"########################################",

"# # # \* #",

"# # ####### #",

"# ####### # # #",

"# # # #",

"# \* ########### # # #####",

"# # # ##### ## #",

"# # # # #",

"# ####### # ####### #######",

"# # # # # # #",

"# # O # ### # # # #",

"# \*##### # # # # # ####### #",

"# # # ##### # # # #",

"# # # # # ####### #",

"# # ########## # # # #####",

"# < # # # #",

"######################## # @ ########",

"# # # # # ",

"# ######################### # # #",

"# #### # ",

"# > D ",

"########################################",

};

int playerX = 2, playerY = 2; // Posição inicial do jogador

int startX = 2, startY = 2; // Posição inicial do jogador (para reiniciar)

bool collectedKey = false; // Indica se a chave foi coletada

int collisionCount = 0; // Contador de colisões com '\*'

bool vActivated = false; // Indica se o V está ativo

int vX = 13, vY = 13; // Posição inicial do V

char lastPlayerDirection = 'd'; // Última direção do jogador

void clearScreen() {

system("cls");

}

void drawMap() {

clearScreen();

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

printf("%s\n", map[i]);

}

printf("Collisions with '\*': %d\n", collisionCount);

}

// Função para lidar com a interação do jogador com elementos do mapa

void interact() {

if (map[playerY][playerX] == '@' && !collectedKey) {

collectedKey = true;

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

for (int j = 0; j < WIDTH; j++) {

if (map[i][j] == 'D') {

map[i][j] = '=';

}

}

}

} else if (map[playerY][playerX] == 'O') {

vActivated = true;

lastPlayerDirection = 'd'; // Define a última direção do jogador como 'd' (direita) ao interagir com 'O'

}

}

// Função para mover o jogador

void movePlayer(char direction) {

int newX = playerX;

int newY = playerY;

switch(direction) {

case 'w': newY--; break;

case 'a': newX--; break;

case 's': newY++; break;

case 'd': newX++; break;

case 'i': interact(); return;

default: return;

}

if (map[newY][newX] == '<') {

// Teleporta o jogador para o outro teleportador correspondente

for (int y = 0; y < HEIGHT; y++) {

for (int x = 0; x < WIDTH; x++) {

if (map[y][x] == '>') {

playerX = x;

playerY = y;

return; // Sai da função após o teleporte

}

}

}

} else if (map[newY][newX] == '>') {

// Teleporta o jogador para o outro teleportador correspondente

for (int y = 0; y < HEIGHT; y++) {

for (int x = 0; x < WIDTH; x++) {

if (map[y][x] == '<') {

playerX = x;

playerY = y;

return; // Sai da função após o teleporte

}

}

}

}

if (newX >= 0 && newX < WIDTH && newY >= 0 && newY < HEIGHT) {

if (map[newY][newX] == ' ' || map[newY][newX] == '@' || map[newY][newX] == '>' || map[newY][newX] == 'O') {

if (map[newY][newX] == '@' && !collectedKey) {

collectedKey = true;

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

for (int j = 0; j < WIDTH; j++) {

if (map[i][j] == 'D') {

map[i][j] = '=';

}

}

}

}

if (map[newY][newX] == '>') {

newX = 1;

newY = 1;

}

map[playerY][playerX] = ' ';

playerX = newX;

playerY = newY;

map[playerY][playerX] = '&';

lastPlayerDirection = direction; // Atualiza a última direção do jogador

} else if (map[newY][newX] == '\*') {

collisionCount++;

if (collisionCount >= 3) {

clearScreen();

printf("GAME OVER:\n");

playerX = startX;

playerY = startY;

vX = 13;

vY = 13;

vActivated = false;

collectedKey = false;

collisionCount = 0;

drawMap();

return;

} else {

map[playerY][playerX] = ' ';

playerX = startX;

playerY = startY;

map[playerY][playerX] = '&';

}

}

}

}

// Função para mover o V

void moveV() {

if (!vActivated) return;

int dx = 0, dy = 0;

// Verifica a última direção do jogador e atualiza as coordenadas do V de acordo

switch (lastPlayerDirection) {

case 'w': dy = -1; break;

case 'a': dx = -1; break;

case 's': dy = 1; break;

case 'd': dx = 1; break;

}

int newVX = vX + dx;

int newVY = vY + dy;

// Verifica se a próxima posição é válida e se não é a posição anterior do jogador

if ((map[newVY][newVX] == ' ' || map[newVY][newVX] == '&') &&

!(newVX == playerX && newVY == playerY)) {

map[vY][vX] = ' ';

vX = newVX;

vY = newVY;

map[vY][vX] = 'V';

}

if (vX == playerX && vY == playerY) {

printf("GAME OVER: The V caught you!\n");

exit(0);

}

}

int main() {

char input;

map[vY][vX] = 'V'; // Define a posição inicial do V

map[playerY][playerX] = '&'; // Define a posição inicial do jogador '&'

drawMap();

while (1) {

input = getchar();

movePlayer(input);

if (vActivated) {

moveV();

}

drawMap();

}

return 0;

}