**Documentação do Projeto "Snake Game"**

**Visão Geral**

Este projeto implementa um jogo da cobrinha (Snake Game) em C para o console do Windows. O objetivo do jogo é controlar uma cobra que se move pelo campo, comer a comida para crescer, e evitar colidir com as bordas do campo ou com o próprio corpo.

**Estruturas e Definições**

**Cores**

O enum Cores define as cores que podem ser usadas no console do Windows:

enum Cores {

PRETO,

AZUL,

VERDE,

CIANO,

VERMELHO,

MAGENTA,

MARROM,

CINZACLARO,

CINZAESCURO,

AZULCLARO,

VERDECLARO,

CIANOCLARO,

VERMELHOCLARO,

MAGENTACLARO,

AMARELO,

BRANCO

};

**Constantes**

As constantes definem as dimensões do campo de jogo, caracteres para representar comida e borda, e o número máximo de pontuações armazenadas:

#define LARGURA\_MAX 60

#define ALTURA\_MAX 30

#define COMPRIMENTO\_INICIAL 5

#define COMIDA '$'

#define BORDA\_CHAR '#'

#define MAX\_PONTUACOES 10

**Estruturas**

**Estrutura EntradaPontuacao**

Armazena uma entrada de pontuação:

typedef struct {

int pontuacao;

} EntradaPontuacao;

**Estrutura Ponto**

Representa um ponto no campo de jogo:

typedef struct {

int x, y;

} Ponto;

**Estrutura Cobra**

Armazena as informações sobre a cobra:

typedef struct {

Ponto corpo[LARGURA\_MAX \* ALTURA\_MAX];

int comprimento;

int direcao; // 0: Cima, 1: Direita, 2: Baixo, 3: Esquerda

} Cobra;

**Variáveis Globais**

* historicoPontuacoes: Armazena o histórico de pontuações.
* numPontuacoes: Número atual de pontuações no histórico.
* cobra: Ponteiro para a estrutura Cobra.
* comida: Ponteiro para a estrutura Ponto representando a comida.
* pontuacao: Pontuação atual do jogador.
* corTexto: Cor inicial do texto (inicialmente VERDECLARO).

**Funções**

**Funções de Utilidade**

**gotoxy(int x, int y)**

Posiciona o cursor na posição (x, y) do console:

void gotoxy(int x, int y) {

COORD coord;

coord.X = x;

coord.Y = y;

SetConsoleCursorPosition(GetStdHandle(STD\_OUTPUT\_HANDLE), coord);

}

**setCor(int cor)**

Define a cor do texto no console:

void setCor(int cor) {

SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), cor);

}

**Funções de Inicialização e Desenho**

**inicializar()**

Inicializa as variáveis e configura a cobra e a comida:

void inicializar() {

cobra = (Cobra \*)malloc(sizeof(Cobra));

comida = (Ponto \*)malloc(sizeof(Ponto));

cobra->comprimento = COMPRIMENTO\_INICIAL;

cobra->direcao = 1;

pontuacao = 0;

// Inicializar posição da cobra

for (int i = 0; i < COMPRIMENTO\_INICIAL; ++i) {

cobra->corpo[i].x = LARGURA\_MAX / 2 - i;

cobra->corpo[i].y = ALTURA\_MAX / 2;

}

// Colocar comida inicial

comida->x = rand() % (LARGURA\_MAX - 2) + 1;

comida->y = rand() % (ALTURA\_MAX - 2) + 1;

}

**desenharBorda()**

Desenha a borda do campo de jogo:

void desenharBorda() {

// Desenhar borda superior

for (int i = 0; i < LARGURA\_MAX; ++i) {

gotoxy(i, 0);

printf("%c", BORDA\_CHAR);

}

// Desenhar borda inferior

for (int i = 0; i < LARGURA\_MAX; ++i) {

gotoxy(i, ALTURA\_MAX - 1);

printf("%c", BORDA\_CHAR);

}

// Desenhar borda esquerda

for (int i = 1; i < ALTURA\_MAX - 1; ++i) {

gotoxy(0, i);

printf("%c", BORDA\_CHAR);

}

// Desenhar borda direita

for (int i = 1; i < ALTURA\_MAX - 1; ++i) {

gotoxy(LARGURA\_MAX - 1, i);

printf("%c", BORDA\_CHAR);

}

}

**desenhar()**

Desenha a cobra e a comida no campo:

void desenhar() {

setCor(corTexto); // Definindo a cor do texto

// Desenhar cobra

for (int i = 0; i < cobra->comprimento; ++i) {

gotoxy(cobra->corpo[i].x, cobra->corpo[i].y);

printf("o");

}

// Desenhar comida

gotoxy(comida->x, comida->y);

printf("%c", COMIDA);

}

**Funções de Controle do Jogo**

**limparCaudaCobra()**

Limpa a cauda da cobra no campo:

void limparCaudaCobra() {

gotoxy(cobra->corpo[cobra->comprimento - 1].x, cobra->corpo[cobra->comprimento - 1].y);

printf(" ");

}

**verificarColisao()**

Verifica se a cobra colidiu com as bordas ou com ela mesma:

bool verificarColisao() {

// Verificar colisão com as paredes

if (cobra->corpo[0].x <= 0 || cobra->corpo[0].x >= LARGURA\_MAX - 1 ||

cobra->corpo[0].y <= 0 || cobra->corpo[0].y >= ALTURA\_MAX - 1)

return true;

// Verificar colisão com o próprio corpo

for (int i = 1; i < cobra->comprimento; ++i) {

if (cobra->corpo[0].x == cobra->corpo[i].x && cobra->corpo[0].y == cobra->corpo[i].y)

return true;

}

return false;

}

**moverCobra()**

Move a cobra na direção atual:

void moverCobra() {

limparCaudaCobra();

// Mover o corpo

for (int i = cobra->comprimento - 1; i > 0; --i) {

cobra->corpo[i] = cobra->corpo[i - 1];

}

// Mover a cabeça

switch (cobra->direcao) {

case 0: cobra->corpo[0].y--; break; // Cima

case 1: cobra->corpo[0].x++; break; // Direita

case 2: cobra->corpo[0].y++; break; // Baixo

case 3: cobra->corpo[0].x--; break; // Esquerda

}

}

**esperarEntrada()**

Espera por uma tecla pressionada para continuar:

void esperarEntrada() {

// Esperar por tecla pressionada

gotoxy(LARGURA\_MAX / 2 - 15, ALTURA\_MAX / 2 + 10);

printf("Pressione qualquer tecla para continuar...");

getch();

}

**pausarJogo(bool \*jogando)**

Pausa o jogo e espera por uma tecla para continuar ou reiniciar:

void pausarJogo(bool \*jogando) {

printf("\nJogo pausado. Pressione 'P' para continuar ou 'R' para reiniciar.");

while (true) {

char teclaRetomar = getch();

if (teclaRetomar == 'p' || teclaRetomar == 'P') {

printf("\nRetomando o jogo...");

return;

}

if (teclaRetomar == 'r' || teclaRetomar == 'R') {

\*jogando = false;

return;

}

}

}

**atualizar(bool \*jogando)**

Atualiza o estado do jogo, move a cobra, verifica colisões e se a cobra comeu a comida:

void atualizar(bool \*jogando) {

// Mover cobra

moverCobra();

// Verificar colisão

if (verificarColisao()) {

// Mostrar mensagem de "Game Over"

system("cls");

gotoxy(LARGURA\_MAX / 2 - 7, ALTURA\_MAX / 2 - 1);

printf("####################");

gotoxy(LARGURA\_MAX / 2 - 7, ALTURA\_MAX / 2);

printf("# #");

gotoxy(LARGURA\_MAX / 2 - 7, ALTURA\_MAX / 2 + 1);

printf("# Game Over #");

gotoxy(LARGURA\_MAX / 2 - 7, ALTURA\_MAX / 2 + 2);

printf("# #");

gotoxy(LARGURA\_MAX / 2 - 7, ALTURA\_MAX / 2 + 3);

printf("####################");

gotoxy(LARGURA\_MAX / 2 - 7, ALTURA\_MAX / 2 + 5);

printf("Pontuacao: %d", pontuacao);

// Adicionar a pontuação atual ao histórico de pontuações

adicionarAoHistoricoPontuacoes(pontuacao);

// Esperar por tecla pressionada

esperarEntrada();

\*jogando = false;

return;

}

// Verificar se a cobra comeu a comida

if (cobra->corpo[0].x == comida->x && cobra->corpo[0].y == comida->y) {

cobra->comprimento++;

pontuacao++;

// Colocar nova comida

comida->x = rand() % (LARGURA\_MAX - 2) + 1;

comida->y = rand() % (ALTURA\_MAX - 2) + 1;

}

desenhar();

}

**Funções de Menu e Histórico**

**mostrarMenu()**

Exibe o menu principal do jogo:

void mostrarMenu() {

system("cls");

int larguraMenu = 30;

int alturaMenu = 13;

int startX = (LARGURA\_MAX - larguraMenu) / 2;

int startY = (ALTURA\_MAX - alturaMenu) / 2;

gotoxy(startX, startY);

printf("##############################");

gotoxy(startX, startY + 1);

printf("# #");

gotoxy(startX, startY + 2);

printf("# Snake Game #");

gotoxy(startX, startY + 3);

printf("# #");

gotoxy(startX, startY + 4);

printf("# 1. Jogar #");

gotoxy(startX, startY + 5);

printf("# 2. Sair #");

gotoxy(startX, startY + 6);

printf("# 3. Historico #");

gotoxy(startX, startY + 7);

printf("# 4. Mudar Cor #");

gotoxy(startX, startY + 8);

printf("# #");

gotoxy(startX, startY + 9);

printf("##############################");

gotoxy(startX, startY + 10);

printf("# Escolha uma opcao: #");

gotoxy(startX, startY + 11);

printf("##############################");

}

**adicionarAoHistoricoPontuacoes(int pontuacaoAtual)**

Adiciona uma nova pontuação ao histórico de pontuações:

void adicionarAoHistoricoPontuacoes(int pontuacaoAtual) {

if (numPontuacoes < MAX\_PONTUACOES) {

historicoPontuacoes[numPontuacoes].pontuacao = pontuacaoAtual;

numPontuacoes++;

} else {

// Deslocar as pontuações existentes para abrir espaço para a nova pontuação

for (int i = 0; i < MAX\_PONTUACOES - 1; i++) {

historicoPontuacoes[i].pontuacao = historicoPontuacoes[i + 1].pontuacao;

}

// Adicionar a nova pontuação no final do histórico

historicoPontuacoes[MAX\_PONTUACOES - 1].pontuacao = pontuacaoAtual;

}

}

**mudarCorTexto()**

Permite ao usuário mudar a cor do texto:

void mudarCorTexto() {

system("cls");

printf("Escolha a cor:\n");

printf("0 - Preto\n");

printf("1 - Azul\n");

printf("2 - Verde\n");

printf("3 - Ciano\n");

printf("4 - Vermelho\n");

printf("5 - Magenta\n");

printf("6 - Marrom\n");

printf("7 - Cinza Claro\n");

printf("8 - Cinza Escuro\n");

printf("9 - Azul Claro\n");

printf("10 - Verde Claro\n");

printf("11 - Ciano Claro\n");

printf("12 - Vermelho Claro\n");

printf("13 - Magenta Claro\n");

printf("14 - Amarelo\n");

printf("15 - Branco\n");

int escolha;

scanf("%d", &escolha);

if (escolha >= 0 && escolha <= 15) {

corTexto = escolha;

} else {

printf("Opcao invalida!\n");

}

printf("\nPressione qualquer tecla para voltar ao menu inicial...");

getch();

}

**Função Principal**

A função principal (main) controla o fluxo do jogo, exibindo o menu, iniciando o jogo, pausando, e atualizando o estado do jogo conforme a interação do usuário:

int main() {

bool jogando = false;

bool pausado = false;

char escolha;

while (true) {

mostrarMenu();

escolha = getch();

switch (escolha) {

case '1':

inicializar();

system("cls"); // Limpa a tela para remover o menu

desenharBorda();

jogando = true;

pausado = false;

break;

case '2':

free(cobra);

free(comida);

exit(0);

case '3':

mostrarHistoricoPontuacoes();

break;

case '4':

mudarCorTexto();

break;

case 'p':

case 'P':

if (jogando && !pausado) {

pausarJogo(&jogando);

system("cls"); // Limpa a tela para remover a mensagem de pausa

desenharBorda(); // Redesenha a borda após retomar o jogo

pausado = false;

}

break;

case 'r':

case 'R':

printf("Reiniciando o jogo...\n");

inicializar();

system("cls"); // Limpa a tela para remover o menu

desenharBorda();

jogando = true;

pausado = false;

break;

}

while (jogando) {

Sleep(100);

if (!pausado) {

atualizar(&jogando);

}

// Entrada de controle

if (kbhit()) {

char ch = getch();

switch (ch) {

case 'w':

if (cobra->direcao != 2) cobra->direcao = 0;

break;

case 'd':

if (cobra->direcao != 3) cobra->direcao = 1;

break;

case 's':

if (cobra->direcao != 0) cobra->direcao = 2;

break;

case 'a':

if (cobra->direcao != 1) cobra->direcao = 3;

break;

case 'p':

case 'P':

pausarJogo(&jogando);

system("cls"); // Limpa a tela para remover a mensagem de pausa

desenharBorda(); // Redesenha a borda após retomar o jogo

pausado = false;

break;

}

}

}

}

return 0;

}

**Explicação do Funcionamento**

**Inicialização da Cobra e da Comida**

Quando o jogo começa (inicializar()), a cobra é posicionada no centro do campo com um comprimento inicial definido por COMPRIMENTO\_INICIAL. A comida é colocada em uma posição aleatória dentro do campo.

**Movimento e Crescimento da Cobra**

A cobra se move na direção especificada (cima, direita, baixo, esquerda). Quando a cobra come a comida (a posição da cabeça da cobra coincide com a posição da comida), seu comprimento aumenta e a comida é reposicionada aleatoriamente.

**Verificação de Colisões**

A função verificarColisao() verifica se a cobra colidiu com as bordas do campo ou com ela mesma. Se ocorrer uma colisão, o jogo termina e o jogador vê a mensagem "Game Over".

**Pontuações e Histórico**

A pontuação do jogador é incrementada cada vez que a cobra come a comida. As pontuações são armazenadas em um histórico que pode ser visualizado no menu principal.

**Menu Principal**

O menu principal permite ao jogador iniciar um novo jogo, ver o histórico de pontuações, mudar a cor do texto ou sair do jogo. Durante o jogo, o jogador pode pausar ou reiniciar o jogo.

**Controle de Direção**

O jogador controla a direção da cobra usando as teclas w, a, s, d para cima, esquerda, baixo e direita, respectivamente.

**Pausa e Reinício**

O jogador pode pausar o jogo pressionando 'p' e reiniciá-lo pressionando 'r'. A pausa exibe uma mensagem e espera uma tecla de retomada ou reinício.

**Considerações Finais**

Este projeto ilustra o uso de ponteiros para gerenciar a memória dinâmica, a manipulação de entrada de teclado e a atualização gráfica do console. A separação de funções facilita a compreensão e a manutenção do código, permitindo a adição de novas funcionalidades, como a mudança de cor do texto, sem grandes modificações na estrutura existente.