

LERY Tetris

Eduardo de Azevêdo dos Santos

14/0136967

Luiz Felipe Carvalho Duarte

14/0151958

Rafael Cascardo Campos

14/0159401

Yero Távora

14/0166378

Brasília - DF, junho de 2016

Universidade de Brasília, Departamento de Ciência da Computação

1 Introdução

LERY Tetris é um jogo programado na linguagem C. Feito como projeto da disciplina de Programação Sistemática da Universidade de Brasília, o jogo ganha seu nome como uma sequência das iniciais de cada um dos autores de forma a ficar sonora. Segue o link do repositório do GitHub utilizado para a confecção do programa:
<https://github.com/gerudoking/lery-tetris.git>

2 Estrutura do Programa

O programa é composto de nove módulos, sendo quatro destes lógicos, contendo a interface das funções e a declaração dos registros, quatro sendo módulos físicos, com a implementação das funções, e um módulo contendo a função main do programa. O jogo utiliza da biblioteca ncurses para montar sua interface.

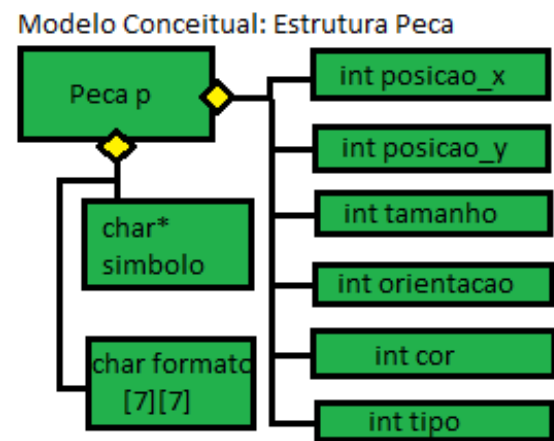
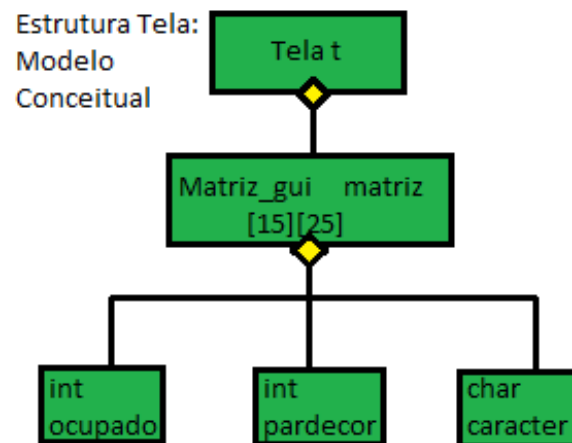
São módulos do programa os listados a seguir:

- Tela.h
- Tela.c
- Peca.h
- Peca.c
- Placar.h
- Placar.c
- Engine.h
- Engine.c
- main.c

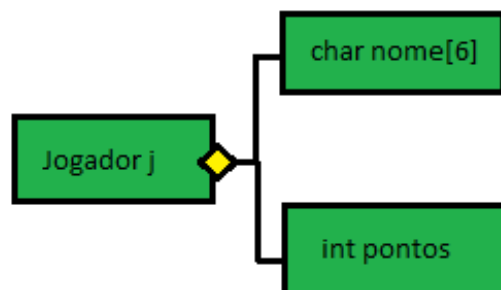
Cada módulo será devidamente explicado em sua respectiva seção.

3 Modelos das Estruturas

Modelos das estruturas utilizadas no projeto:



Modelo Conceitual:
Estrutura Jogador



4 Tela

4.1 Tela.h

Arquivo do tipo header(.h) contendo a declaração da struct Tela e as interfaces das funções utilizadas no módulo Tela.c.

A struct Tela é composta por outra struct, chamada Tipomatriz. Esta, por sua vez, é composta de tres membros. Estes são:

- ocupado, do tipo inteiro;
- pardecor, do tipo inteiro;
- caracter, do tipo caractere.

As structs estão declaradas da seguinte maneira:

```
struct tipomatriz{  
int ocupado;  
int pardecor;  
char caracter;  
};  
typedef struct tipomatriz Tipomatriz;  
  
struct tela{  
Tipomatriz matriz_gui[15][25];  
};  
typedef struct tela Tela
```

O módulo ainda possui seis interfaces:

```
WINDOW* cria_nova_janela(int alt, int larg, int starty, int startx);
```

Função que recebe quatro valores. Um correspondente a altura de uma janela padrão da biblioteca ncurses, outro a largura da mesma, e por fim os valores das posições iniciais y e x, respectivamente, da janela no terminal.

```
void destroi_janela(WINDOW* janela_local);
```

Em suma, recebe uma janela padrão da biblioteca ncurses e não possui retorno. Foi implementada porém não utilizada no programa.

```
Tela* cria_tela();
```

Função que não recebe parâmetros, retorna uma struct do tipo Tela.

```
int mostra_tela(Tela* t)
```

Esta função recebe uma struct do tipo Tela e retorna um valor do tipo inteiro(utilizado para testes).

```
int mostra_tela_inicial();
```

Função que não recebe parâmetros e retorna um valor do tipo inteiro(utilizado para testes).

```
int mostra_tela_final(int pont);
```

Função que recebe um inteiro representando a pontuação do jogador e retorna um inteiro(utilizado para testes).

```
int mostra_tela_placar(int pont);
```

Função que recebe um inteiro representando a pontuação do jogador e retorna um inteiro(utilizado para testes).

4.2 Tela.c

Arquivo do tipo .c, contém a implementação das funções declaradas em Tela.h.

`cria_nova_janela`: A função irá utilizar os parâmetros dados para criar uma janela padrão da biblioteca ncurses, para posteriormente desenhar uma caixa para servir como bordas da janela. Irá então dar "refresh" na janela para aplicar as mudanças. Por fim, retorna a janela criada.

`cria_tela`: Esta função irá alocar espaço para a struct `t` do tipo `Tela`, testar seu sucesso, em caso de falha retornando `NULL`, e então irá definir cada campo de cada posição do membro `matriz_gui` de forma que o campo `caracter` seja um caractere espaço, o campo `ocupado` seja definido como zero, e o campo `pardecor` como um. Por fim retornará a tela criada.

`mostra_tela`: Função que recebe uma `Tela t` previamente criada. Começa limpando a tela com o comando `refresh()`, para então gerar uma janela padrão do ncurses para servir como bordas para a interface do jogo. Em um laço duplo, irá então reproduzir na tela os caracteres de cada posição da `matriz_gui` com suas respectivas cores, também definidas em `matriz_gui`.

`mostra_tela_inicial`: A função `mostra_tela_inicial()` não recebe parâmetros. Esta irá criar uma tela `t`, mostrá-la, para então escrever por cima as mensagens de boas-vindas ao jogador em suas respectivas cores. Por fim, libera a estrutura `t`.

`mostra_tela_final`: Função idêntica a função `mostra_tela_final`, exceto que esta recebe um inteiro correspondente a pontuação do jogador e o imprime junto de sua respectiva mensagem.

5 Pecas

5.1 Peca.h

Módulo contendo a declaração da struct peca, a qual é composta pelos seguintes membros:

- posicao_x, do tipo inteiro;
- posicao_y, do tipo inteiro;
- tamanho, do tipo inteiro;
- orientacao, do tipo inteiro;
- cor, do tipo inteiro;
- simbolo, do tipo ponteiro para caractere.

Segue a declaração da struct:

```
typedef struct peca{  
int posicao_x , posicao_y , tamanho , orientacao , cor , tipo ;  
char* simbolo ;  
char formato [ 7 ] [ 7 ] ;  
}peca ;
```

```
peca* nova_peca ( Tela* tela ) ;
```

Função que recebe a tela de jogo onde a peca deve ser impressa e retorna a peca que foi criada

```
int move_peca_x ( Tela* tela , peca* a , int direcao ) ;
```

Função que recebe a tela de jogo onde a peca deve ser impressa, a peca que deve ser impressa e a direcao para onde ela deve se mover e retorna um inteiro usado para testes

```
int move_peca_y ( Tela* tela , peca* a ) ;
```

Função que recebe a tela de jogo onde a peça deve ser impressa e a peça que deve ser impressa e retorna um inteiro usado para testes

```
void rotaciona_peca ( Tela* tela , peca* a ) ;
```

Função que recebe a tela de jogo onde a peça deve ser impressa e a peça que deve ser impressa

5.2 Peca.c

Arquivo do tipo .c, tem as seguintes funções implementadas:

`nova_pec`: Função que recebe uma tela, então aloca espaço para uma estrutura do tipo peça, e faz as suas devidas inicializações, atribuindo a seu campo tamanho um valor aleatório entre três e cinco, escolhendo aleatoriamente uma orientação(peça na horizontal ou vertical), e definindo uma cor aleatória para a peça(vermelho, azul ou verde). Por fim, a função faz mudanças na `matriz_gui` da tela para que a peça possa ser mostrada. Retorna a estrutura da peça ao final.

`move_pec_x`: Recebe uma tela, uma peça, e um inteiro direção, que pode ser menos um ou um, indicando movimento para a esquerda ou direita, respectivamente. Faz-se então uma checagem quanto a orientação da peça, e então uma checagem para ver se é possível o movimento da peça naquela direção, ou seja, se não há colisões. A função tem diferentes retornos do tipo inteiro, para que cada situação seja tratada de maneira diferente pela função que a chamar.

`move_pec_y`: Recebe uma tela, uma peça. Move a peça para baixo, checando se há colisões. Caso haja, retorna um valor. Caso contrário, move a peça para baixo, atualizando sua posição na `matriz_gui`.

6 Engine

6.1 Engine.h

O módulo Engine.h possui a interface de diversas funções que serão implementadas no Engine.c. São as seguinte:

```
void inicia_ncurses ();
```

Uma função sem parâmetros e sem retorno.

```
void finaliza_ncurses ();
```

Uma função sem parâmetros e sem retorno.

```
int deleta_linha (Tela* tela , int linha );
```

Esta função recebe um ponteiro para Tela, e um inteiro correspondente a uma linha.

```
int movimento (Tela* tela );
```

Função que recebe ponteiro de tela e retorna um inteiro.

6.2 Engine.c

Arquivo do tipo .c, tem as seguintes funções implementadas:

`inicia_ncurses`: Função que chama várias outras funções da ncurses, assim inicializando as funcionalidades da biblioteca.

`finaliza_ncurses`: Função que finaliza as funcionalidades da biblioteca ncurses, chamando a função `endwin()`.

`deleta_linha`: Função que pega uma linha preenchida na matriz_gui, e a desocupa, posteriormente pegando todas as posições ocupadas acima e as reatribuindo uma posição abaixo.

`movimento`: A função que ao começar cria uma nova peça e a mostra na tela, então entra em um laço de repetição responsável por receber as entradas de movimento da peça até ela ser fixada na matriz_gui ao encerrar seu movimento. Dentro do laço é contado o tempo de espera por entrada do jogador e são tratado os casos de saída para as

funções de movimento, e é chamada a função `mostra_tela` novamente. Ao final, é feita uma checagem se alguma linha está completamente ocupada. Caso esteja, é chamada a função `deleta_linha`.

7 Placar

7.1 Placar.h

Módulo contendo a declaração da struct `Jogador`, o qual é composto pelos seguintes membros:

- vetor de caracteres do tipo `char`, `nome`
- valor do tipo inteiro, `pontos`

Segue declaração da struct:

```
struct tipojogador{  
    char nome[6];  
    int pontos;  
};  
typedef struct tipojogador Jogador;
```

```
void cria_placar(void);
```

Função que não recebe argumentos e não retorna nada.

```
void atualiza_placar(int pontuacao);
```

Função que recebe a `pontuacao` a ser escrita no arquivo de placares e não retorna nada.

```
void mostra_placar(void);
```

Função que não recebe argumentos e não retorna nada.

7.2 Placar.c

Arquivo do tipo .c, implementa as seguintes funções:

`cria_placar`: Esta função abre um arquivo no modo leitura, checando se ele existe. Caso não exista, abrirá o arquivo no modo escrita+, inicializando o .txt que contém as informações do placar do jogo.

`atualiza_placar`: A função pede entrada ao usuário de um nome de até 5 letras, e logo após checa se a pontuação obtida no jogo supera alguma das gravadas em disco. No caso de superar, o novo registro Jogador é escrito na posição correspondente, mantendo um arquivo de registros ordenados pela pontuação. Caso contrário, não será escrito nada no arquivo .txt.

`mostra_placar`: A função `mostra_placar` abre o arquivo `placar.txt` em modo leitura, lê cinco linhas escritas no arquivo e as imprime na tela, sendo estas linhas o nome e a pontuação de cada jogador, ordenado por pontuação.

8 A função Main

```
int main(){

    srand(time(NULL));

    int end = 0;
    int pontuacao = 0;

    inicia_ncurses();

    Tela* t = cria_tela();

    mostra_tela_inicial();
    getch();

    while(end == 0){

        mostra_tela(t);
        end = movimento(t, &pontuacao);
    }

    mostra_tela_final(pontuacao*100);
    getch();

    mostra_tela_placar(pontuacao*100);
    getch();

    free(t);
    finaliza_ncurses();

    return 0;
}
```

Nesta função, é inicializada a seed para as funções que usam aleatoriedade, é criada uma estrutura Tela, e então se mostra a tela de boas vindas, para posteriormente utilizar a estrutura criada dentro de um laço que termina quando o jogador perder o jogo. O laço irá constantemente se chama mostra_tela e movimento, sendo que esta retorna se o jogo está perdido ou não. Por fim, mostra-se a tela final, libera-se a estrutura Tela, e finaliza-se o ncurses.

9 Divisão do Trabalho

- Tela.h, Tela.c, Placar.h e Placar.c: Eduardo de Azevêdo dos Santos
- Pecas.h e Pecas.c: Luiz Felipe Carvalho Duarte
- Engine.h e Engine.c: Yero Távora; Rafael Cascardo Campos

No geral, todos ajudaram aos outros com dicas e ajustes. O makefile foi feito e atualizado por todos, e a documentação feita por Eduardo de Azevêdo dos Santos e Rafael Cascardo Campos em LaTeX.