Documentação de Software - Sistema de Compra de Ingressos

Nome	DRE
Miguel Angelo Gonzaga Marques	122138896
Mateus Augusto do Nascimento Afonso	122149481

Descrição

O Sistema de Compra de Ingressos é uma aplicação de console desenvolvida para o grupo Mariano Pinheiro, um dos maiores grupos de cinemas do Brasil. O sistema permite que os usuários comprem ingressos de filmes, selecionem poltronas desejadas e recebam informações sobre o valor a ser pago. O sistema aplica descontos para ingressos de meia-entrada para estudantes e para clientes do Banco Itasil.

Estruturas de Dados

1. **INGRESSO**
    - `unsigned int ingressos_totais`: quantidade total de ingressos selecionados
    - `unsigned int ingressos_normais`: quantidade de ingressos sem desconto
    - `unsigned int ingressos_meia`: quantidade de ingressos com desconto de meia-entrada
    - `unsigned int ingressos_banco`: quantidade de ingressos com desconto de cliente Itasil

2. **CODIGOS_ITASIL**
    - `unsigned long int codigo_itasil`: código de cliente Itasil
    - `struct _CODIGOS_ITASIL* prox`: ponteiro para o próximo código de cliente Itasil

3. **CODIGOS_ESTUDANTE**
    - `unsigned long int codigo_estudante`: número da carteira de estudante
    - `struct _CODIGOS_ESTUDANTE* prox`: ponteiro para a próxima carteira de estudante

4. **SALAS**
    - `int** sala1`: matriz representando a sala 1 do cinema
    - `int** sala2`: matriz representando a sala 2 do cinema
    - `int** sala3`: matriz representando a sala 3 do cinema
    - `int ingressos_restantes_sala1`: quantidade de ingressos disponíveis na sala 1
    - `int ingressos_restantes_sala2`: quantidade de ingressos disponíveis na sala 2
    - `int ingressos_restantes_sala3`: quantidade de ingressos disponíveis na sala 3

Funções auxiliares

Funções auxiliares são funções definidas para ajudar em tarefas e serem reutilizadas várias vezes em diferentes partes do código, melhorando a organização e a legibilidade do código, e evitando repetição.

1. Função `limpar_buffer()`

    Descrição: Limpa o buffer do teclado, descartando qualquer entrada de dados pendente.

    Parâmetros: Nenhum.

    Retorno: Nenhum.

2. Função `limpar_tela()`

   Descrição: Limpa a tela do console, permitindo uma apresentação mais organizada das informações.

   Parâmetros: Nenhum.

   Retorno: Nenhum.

3. Função `ctoi(c)`

   Descrição: Converte um caractere numérico em um valor inteiro.

   Parâmetros:

   - `c`: Caractere a ser convertido.

   Retorno: Valor inteiro correspondente ao caractere convertido.

4. Função `preencher_matriz(sala)`

   Descrição: Preenche a matriz da sala de cinema com valores iniciais, indicando que todos os lugares estão disponíveis.

   Parâmetros:

   - `sala`: Matriz representando a sala de cinema.

   Retorno: Nenhum.

5. Função `alocar_matriz(linhas, colunas)`

   Descrição: Aloca dinamicamente uma matriz de tamanho específico.

   Parâmetros:

   - `linhas`: Número de linhas da matriz.
   - `colunas`: Número de colunas da matriz.

   Retorno: Ponteiro para a matriz alocada.

6. Função `strcknl(string)`

   Descrição: Verifica se um vetor de caracteres possui o caractere de quebra de linha.

Parâmetros:

- `string`: String a ser verificada.

Retorno: Valor booleano indicando se o vetor tem ou não `\n` nos seus caracteres.

Funções

1. Função `exibir_mapa_sala(sala)`

   Descrição: Exibe o mapa da sala de cinema, indicando quais lugares estão ocupados e quais estão disponíveis.

   Parâmetros:

   - `sala`: Matriz representando a sala de cinema.

   Retorno: Nenhum.

2. Função `selecionar_filme()`

   Descrição: Solicita ao usuário que selecione um dos filmes em cartaz, digitando o número da sala correspondente ao filme desejado.

   Parâmetros: Nenhum.

   Retorno: Valor inteiro referente ao número da sala selecionada (1, 2, 3) ou 0 para encerrar o programa.

3. Função `solicitar_quantidade_ingressos(ingressos_disponiveis, codigos_itasil_utilizados, codigos_estudante_utilizados)`

   Descrição: Solicita ao usuário a quantidade de ingressos que deseja comprar, considerando a quantidade de ingressos disponíveis na sala.

   Parâmetros:

   - `ingressos_disponiveis`: Ponteiro para a variável inteira contendo a quantidade de ingressos disponíveis na sala.
   - `codigos_itasil_utilizados`: Ponteiro para a lista de códigos de cliente Itasil utilizados na sala.

- `codigos_estudante_utilizados`: Ponteiro para a lista de números de carteira de estudante utilizados na sala.

Retorno: Ponteiro para a estrutura `INGRESSO` contendo as informações sobre a quantidade de ingressos selecionada e os descontos aplicados.

4. Função `aplicar_desconto_meia_entrada(codigos_utilizados)`

   Descrição: Verifica se o número da carteira de estudante é válido e se já foi utilizado anteriormente.

   Parâmetros:

   - `codigos_utilizados`: Ponteiro para a lista de números de carteira de estudante utilizados na sala.

   Retorno: Valor booleano indicando se o desconto de meia-entrada foi aplicado com sucesso ou não.

5. Função `verificar_carteira_estudante(numero_carteira, codigos_utilizados)`

   Descrição: Verifica se o número da carteira de estudante é válido e se já foi utilizado anteriormente.

   Parâmetros:

   - `numero_carteira`: Número da carteira de estudante a ser verificado.
   - `codigos_utilizados`: Ponteiro para a lista de números de carteira de estudante utilizados na sala.

   Retorno: Valor inteiro indicando a validade da carteira de estudante (-1 para já utilizada, 0 para inválida, 1 para válida).

6. Função `adicionar_codigo_estudante(codigo, codigos_utilizados)`

   Descrição: Adiciona o número da carteira de estudante utilizado à lista de códigos utilizados.

   Parâmetros:

   - `codigo`: Número da carteira de estudante utilizado.
   - `codigos_utilizados`: Ponteiro para a lista de números de carteira de estudante utilizados na sala.

Retorno: Nenhum.

7. Função `aplicar_desconto_cliente_itasil(codigos_utilizados)`

   Descrição: Verifica se o código de cliente Itasil é válido e se já foi utilizado anteriormente.

   Parâmetros:

   - `codigos_utilizados`: Ponteiro para a lista de códigos de cliente Itasil utilizados na sala.

   Retorno: Valor booleano indicando se o desconto de cliente Itasil foi aplicado com sucesso ou não.

8. Função `verificar_codigo_itasil(codigo, codigos_utilizados)`

   Descrição: Verifica se o código de cliente Itasil é válido e se já foi utilizado anteriormente.

   Parâmetros:

   - `codigo`: Código de cliente Itasil a ser verificado.
   - `codigos_utilizados`: Ponteiro para a lista de códigos de cliente Itasil utilizados na sala.

   Retorno: Valor inteiro indicando a validade do código de cliente Itasil (-1 para já utilizado, 0 para inválido, 1 para válido).

9. Função `adicionar_codigo_itasil(codigo, codigos_utilizados)`

   Descrição: Adiciona o código de cliente Itasil utilizado à lista de códigos utilizados.

   Parâmetros:

   - `codigo`: Código de cliente Itasil utilizado.
   - `codigos_utilizados`: Ponteiro para a lista de códigos de cliente Itasil utilizados na sala.

   Retorno: Nenhum.

10. Função `calcular_valor_total(valores_ingressos)`

    Descrição: Calcula o valor total a ser pago pelos ingressos, considerando a quantidade de ingressos e os descontos aplicados.

    Parâmetros:

    - `valores_ingressos`: Ponteiro para a estrutura `INGRESSO` contendo as informações sobre a quantidade de ingressos e os descontos aplicados.

    Retorno: Valor inteiro correspondente ao valor total a ser pago.

11. Função `agradecer_pedido(valor_total)`

    Descrição: Exibe uma mensagem de agradecimento ao usuário pelo pedido e informa o valor total a ser pago.

    Parâmetros:

    - `valor_total`: Valor total a ser pago pelos ingressos.

    Retorno: Nenhum.

12. Função `marcar_poltronas(filme, sala, structIngresso)`

    Descrição: Permite ao usuário selecionar as poltronas desejadas para os ingressos e marca-as como ocupadas na matriz da sala.

    Parâmetros:

    - `filme`: Número da sala correspondente ao filme selecionado.
    - `sala`: Matriz representando a sala de cinema, mesmo que a função esteja sem código, deve apenas colocar na documentação a descrição da função e seus parâmetros, retorno e o que ela faz no sistema.

    Retorno: Nenhum.

Fluxo de Execução

1. A função `main()` é chamada para iniciar o programa.
2. São declaradas as variáveis necessárias, como `filme`, `valor_total`, `reiniciar`, e as listas encadeadas `codigos_itasil_utilizados_sala_X` e

`codigos_estudante_utilizados_sala_X` para armazenar os códigos de cliente Itasil e carteiras de estudante utilizados em cada sala, respectivamente.

3. A estrutura `SALAS` é alocada dinamicamente na memória para armazenar as informações das salas de cinema.
4. As variáveis `ingressos` e `salas->ingressos_restantes_salaX` são inicializadas com valores padrão.
5. As matrizes `salas->salaX` são alocadas dinamicamente e preenchidas com valores iniciais.
6. Inicia-se um loop infinito para que o sistema rode até que o usuário queira que ele seja encerrado.
7. A tela é limpa para exibir um menu onde o usuário pode selecionar o filme em cartaz utilizando a função `selecionar_filme()`.
8. Se o valor de `filme` for igual a 0, o loop é interrompido e o programa é encerrado.
9. Dependendo do valor de `filme`, o usuário é solicitado a informar a quantidade de ingressos desejada utilizando a função `solicitar_quantidade_ingressos()`.
10. Para cada ingresso com desconto de meia-entrada, o usuário é solicitado a informar o número da carteira de estudante utilizando a função `aplicar_desconto_meia_entrada()`.
11. Para cada ingresso com desconto de cliente Itasil, o usuário é solicitado a informar o código de cliente Itasil utilizando a função `aplicar_desconto_cliente_itasil()`.
12. A função `marcar_poltronas()` é chamada para que o usuário selecione as poltronas desejadas para os ingressos.
13. O valor total a ser pago é calculado utilizando a função `calcular_valor_total()`.
14. Uma mensagem de agradecimento é exibida, informando o valor total a ser pago, utilizando a função `agradecer_pedido()`.
15. O usuário é solicitado a informar se deseja realizar um novo pedido, digitando 'S' ou 'N'.
16. O buffer do teclado é limpo utilizando a função `limpar_buffer()`.
17. Se o valor digitado for diferente de 'S' ou 's', o loop é interrompido e o programa é encerrado.

Considerações adicionais

- Durante a implementação, foram realizado validações e tratamento de erros para garantir que o usuário insira valores válidos em cada etapa do processo.
- Foram utilizadas boas práticas de programação, como modularização, comentários e nomenclatura adequada de variáveis e funções, para tornar o código mais legível e fácil de manter.