Sistema para Salas de Cinema

Programação Orientada a Objetos

Estrutura de Dados I

Bacharelado de Ciência da Computação

UFSCAR – Sorocaba

Augusto Sousa Santos 380245
Bruna Tanios Devienne Pompeu 408506
Hitalo Siqueira 408476
Matheus Casarin Paez 438308
Thales Chagas 408557

Descrição do Programa

Foi proposto desenvolver um sistema que dará suporte a gestão das salas de cinema que fazem parte da rede. O programa permite a vender ingresso, calcular o valor total, emitir o ingresso, remover ingresso da compra, verificar a forma de pagamento (dinheiro ou cartão), verificar o tipo de ingresso (inteiro ou meio), verificar o assento disponível, verificar o nome do filme, o horário e a sala da sessão e verificar a situação de uma sala (em manutenção geral, manutenção de equipamento, em reforma ou disponível).

Forma de Utilização do Programa

Ao iniciar o programa, é apresentado um menu com 4 opções:

Menu de opções :
1 – Gerenciar Salas;
2 – Gerenciar Sessão;
3 – Gerenciar Venda;
0 – Encerrar;

Opção 1 - de Gerenciar Salas, então é apresentado outro menu:

Menu de opções :
1 – Inserir Sala;
2 – Remover Sala;
3 – Interditar Sala;
4 – Ver Salas;
0 – Voltar ao Menu Anterior;

Opção 1 - Inserir Sala: o usuário deve informar a quantidade de fileiras e a quantidade de assentos em cada fileira da sala.

Opção 2 - Remover Sala: o usuário deve informar o número da sala a ser excluída.

Opção 3 - Interditar Sala: o usuário deve informar o número da sala e então aparecerá 4 opções para a situação da sala:

0 – Liberar Sala;
1 – Manutenção de Equipamento;
2 – Reforma;
3 – Manutenção Geral;

O usuário escolhe a situação em que a sala se encontra.

Opção 4 - Ver Salas: é mostrado na tela todas as salas existentes e os assentos disponíveis (1) e indisponíveis (0).

Opção 0 - Voltar ao Menu Anterior: retorna ao menu principal.

Opção 2 - Gerenciar Sessão, então é apresentado outro menu de opções:

1 – Inserir Sessão;
2 – Encerrar Sessão;
3 – Ver Sessões;
4 – Voltar ao Menu Anterior;

Opção 1 - Inserir Sessão: o usuário deve informar o nome do filme, o horário e a sala da sessão.

Opção 2 - Encerrar Sessão: o usuário deve informar o número da sessão que deseja excluir.

Opção 3 - Ver Sessões: são mostradas na tela todas as sessões existentes no cinema.

Opção 0 - Voltar ao Menu Anterior: retorna ao menu principal.

Opção 3 - Gerenciar Venda, então é apresentado outro menu de opções:

1 – Inserir Ingresso;
2 – Remover Ingresso;
3 – Calcular Valor Total;
4 – Emitir Ingresso(s);

Opção 1 - Inserir Ingresso: o usuário deve informar a data da sessão e o tipo do ingresso (inteira ou meia). Aparecerão todas as sessões disponíveis e então o usuário deve selecionar o número da sessão. Depois aparecera a sala com os assentos e suas disponibilidades, o usuário deve escolher o assento de acordo com a letra da fileira e número do assento. Caso o assento esteja ocupado, deverá escolher outro. E então o ingresso é inserido em sua compra.

Opção 2 – Remove Ingresso: o usuário deve informar o número do ingresso que deseja remover. O ingresso é removido da compra e o lugar torna-se disponível.

Opção 3 – Calcular Valor Total: imprime todos os ingressos da venda do usuário e calcula o valor total da venda.

Opção 4 – Emitir Ingresso(s): Imprime o(s) ingresso(s) no arquivo de saída.

Opção 0 – Encerrar: o programa é encerrado.

Descrição dos Algoritmos Utilizados na Resolução do Problema Proposto

Abaixo serão apresentados os métodos das classes existentes no programa com uma explicação sucinta sobre o que cada uma faz.
Obs.: os vetores de fileira e assento foram implementados usando a biblioteca da STL vector.h

1. **Assento**

1.1 bool Assendo::Assento::verificaDisponibilidade()
Método que verifica se o assento esta disponível e retorna true caso esteja e false caso não esteja.

1.2 void Assendo::Assento::setDisponibilidade(bool disponibilidade)
Método que seta o valor da disponibilidade do assento.

1.3 int Assendo::Assento::getIdAssento()
Método que retorna o número de identificação do assento.

1.4 void Assendo::Assento::setIdAssento(int idAssento)
Método que seta o valor do número de identificação do assento.

2. **Fileira**

2.1 void Fileira::setIdFileira(char idFileira);
Método que seta o caracter de identificação da fileira.

2.2 char Fileira::getIdFileira();
Método que retorna o caracter de identificação da fileira.

2.3 void Fileira::iniciaAssentos(int nAssentosFileira);
Método que define o tamanho da fileira de acordo com o número de assentos existentes e inicializa cada assento como disponível (true) e cada idAssento com o número para identificação do assento.

2.4 void Fileira::imprimeAssentos();
Método que imprime os assentos de uma fileira.

2.5 void Fileira::imprimeX();
Método que mostra a numeração dos assentos.

2.6 void Fileira::setDisponibilidade(int assento);
Método que verifica a disponibilidade dos assentos.

2.7 void Fileira::setIdFileira();
Método que atribui uma identificação à fileira.

3. Sala

3.1 Sala::Sala(int numSala, int nFileiras, int nAssentosFileira);
Construtor da classe Sala que define o tamanho de cada fileira e a situação da sala como disponível, inicializa o número da sala, o caracter de identificação de cada fileira e o número de identificação dos assentos em cada fileira.

3.2 int Sala::getNumSala();
Método que retorna o número da sala.

3.3 int Sala::getNFileiras();
Método que retorna o número de fileiras existentes.

3.4 int Sala::getNAssentosFileira();
Método que retorna o número de assentos existentes em cada fileira.

3.5 string Sala::getEstado();
Método que retorna o estado da sala (disponível, em manutenção geral, manutenção de equipamento ou em reforma).

3.6 void Sala::setNumSala(int numSala);
Método que seta o número da sala.

3.7 void Sala::setNFileiras(int nFileiras);
Método que seta a quantidade de fileiras existentes.

3.8 void Sala::setNAssentosFileira(int nAssentosFileira);
Método que seta a quantidade de assentos existentes em cada fileira.

3.9 void Sala::setEstado(string estado);
Método que seta o estado da sala.

3.10 void Sala::imprimeSala();
Método que imprime o número da sala, as fileiras e assentos existentes.

3.11 bool Sala::verificaFileira(char fileira, int assento);
Método que verifica a fileira.

3.12 void Sala::setDisp(char fileira, int assento);
Método que verifica a disponibilidade dos assentos da fileira.

3.13 void Sala::setEstado(string estado);
Método que atribui um estado à sala.

4. Sessão

4.1 Sessao::Sessao(int nSessao, string filme, string horário, int sala, bool encerrada, int numVendido);

Construtor da classe Sessao que inicializa o nome do filme, o horário, a sala e o número da sessão, se a sessão esta encerrada ou não e o número de ingressos vendidos da mesma.

4.2 int Sessao::getSala();

Método que retorna o numera da sala de uma sessão.

4.3 void Sessao::setSala(int sala);

Método que seta o número da sala de uma sessão.

4.4 int Sessao::getNSessao();

Método que retorna a quantidade de sessões existentes no cinema.

4.5 void Sessao::setStatus(bool encerrada);

Método que seta se uma sessao esta encerrada (true) ou não (false).

4.6 bool Sessao::getStatus();

Método que retorna true se uma sessão estiver encerrada e false caso não esteja.

4.7 string Sessao::getHorário();

Método que retorna o horário de uma sessão.

4.8 void Sessao::setHorário(string horário);

Método que seta o horário de uma sessão.

4.9 void Sessao::setNumVendido(int numVendido);

Método que seta o número de ingressos vendidos em uma sessão.

4.10 int Sessao::getNumVendido();

Método que retorna o número de ingressos vendidos em uma sessão.

4.11 string Sessao::getFilme();

Método que retorna o nome do filme de uma sessão.

4.12 void Sessao::setFilme(string nomeFilme);

Método que seta o nome do filme de uma sessão.

4.13 void Sessao::imprimeSessao();

Método que imprime o nome do filme, o horário e o número de uma sessão.

5. Ingresso

5.1 void setTipo(string tipo);
Método que seta o tipo de ingresso (inteira ou meia).

5.2 void setFileira(char fileira);
Método que seta o caracter da fileira do ingresso.

5.3 void setAssento(int assento);
Método que seta o número do assento do ingresso.

5.4 string getTipo();
Método que retorna o tipo de ingresso.

5.5 string getDtIngresso();
Método que retorna a data do ingresso.

5.6 double getValor();
Método que retorna o valor do ingresso.

5.7 void setNSessao(int nSessao);
Método que seta o número da sessao.

5.8 void setDtIngresso(string dtIngresso);
Método que seta a data do ingresso.

5.9 void setValor(double valor);
Método que seta o valor do ingresso.

5.10 void imprimir();
Método que imprime os dados do ingresso.

5.11 int getId();
Método que recebe o identificador da sessão do ingresso.

5.12 int getAssento();
Método que recebe o assento relacionado ao ingresso.

5.13 char getFileira();
Método que recebe a fileira relacionada ao ingresso.

6. Venda

6.1 Venda(int idVenda);

Construtor que inicializa o número de identificação da venda.

6.2 double calcularValorTotal();

Método calcula o valor total da venda dos ingressos e o retorna.

6.3 void emitirIngresso();

Método que emite todos os ingressos da venda.

6.4 Ingresso addIngresso(vector<Sala> &sala, vector<Sessao> &sessao, vector<Venda> &venda);

Método para adicionar um ingresso na venda. O usuário informa uma sessão, a fileira e o assento que deseja e então verifica-se se o assento esta disponível. Caso esteja, a adição é realizada, caso não esteja disponível, o usuário deve informar um novo assento.

6.5 void removeIngresso();

Método que remove um ingresso.

6.6 imprimeIngressos();

Método que mostra os ingressos escolhidos.

Descrição das Condições de Contorno

Em toda alocação de memória é testada se a alocação foi realizada com sucesso, com o tratador de exceções try, que lança um bad_alloc caso a alocação tenha falhado. O catch faz o tratamento para o bad_alloc.

Em todo “menu” de opções é tratado valores incorretos.

Uma sala só poderá ter uma sessão caso a sala esteja como disponível.

Uma sessão não pode estar em uma mesma sala e em um mesmo horário que uma outra sessão.

Um assento só pode ser comprado caso esteja disponível.

Descrição dos Testes Realizados

O programa funciona corretamente, mas não foram testas entradas diferentes das esperadas, como por exemplo, o uso de character quando é esperado um inteiro e vice-versa.

Dificuldades Encontradas

A principal dificuldade encontrada foi o pouco tempo para o desenvolvimento do projeto, por ser extenso. Além disso, o relacionamento entre sala, fileira e assento foi muito complexo, aumentando o grau de dificuldade na implementação de métodos para achar um assento em uma determinada sala. Também na classe Venda, por ser grande e conter vetores de muitas classes, demorou um pouco mais de tempo para ser implementada.

Possíveis Extensões e Melhorias do Programa

Uma melhoria que pode ser feita é verificar a quantidade de ingressos que a pessoa deseja comprar ao invés de comprar um ingresso por vez. Também se tornou inviável tratar todas as possíveis entradas, este seria outro ponto a se melhorar.

Testes

Teste Salas:
Passos: Gerenciar Salas → Ver Salas → Remover Sala → Sala 3 → Interditar Sala → Sala 2 → Manutenção Equipamento → Voltar ao Menu

Resultado Esperado: Sala 3 removida e sala 2 interditada

Resultado do Teste: Satisfeito

Teste Sessão
Passos: Gerenciar Sessão → Ver Sessões → Encerrar Sessão → Sessão 3 → Encerrar Sessão → Sessão 2 → Inserir Sessão → Filme Gladiador → Hora 22 → Sala 2 → Ver Sessões → Voltar ao menu

Resultado Esperado: Apresentar sessões, encerrar sessões 2 e 3, nova sessão no lugar da sessão 2.

Resultado do Teste: Satisfeito

Teste Venda:
Passos: Gerenciar Venda → Inserir Ingresso 02/07/2014 → Inteira → Sessão 0 → Fileira a Assento 0 → Inserir Ingresso 02/07/2014 → Inteira → Sessão 0 → Fileira a Assento 0 → Fileira a Assento 1 → Remover Ingresso → Ingresso 1 → Remover Ingresso → Ingresso 0 → Inserir Ingresso → Data 02/07/2014 → Intera → Sessão 0 → Fileira a Assento 0 → Calc. Valor Total → Emitir → Voltar → Encerrar

Resultado Esperado: Inserir um ingresso, dizer que o lugar está ocupado e pedir para escolher outro livre para inserir, inserir ingresso, remover os dois ingressos, inserir um ingresso novamente, mostrar o preço de um ingresso inteiro (10), salvar em um arquivo .txt, voltar ao menu principal e encerrar.

Resultado do Teste: Satisfeito.

Diagrama de Classes