Gerenciador De Cinema

Dupla:Aryelson Gonçalves Messias

Gabriel Lopes Rodrigues Silva

Data: 17 de outubro de 2021

1. Descrição do sistema

Um sistema para gerenciamento de um cinema, projeto proposto como última nota na disciplina de POO - Programação orientada objeto, sistema implementado usando a linguagem C++. O programa permite vender ingressos e calcular o valor total, emitir bilhete, cancelar bilhete da compra, verificar Forma de pagamento , verificar o tipo de bilhete (inteiro ou estudante), verificar assentos disponíveis, verificar qual filme se encontra em cartaz , verificar informações gerais da sessão.

2. Especificação do Sistema

Sala.h

```
#ifndef SALA_H
#define SALA_H
#include<iostream>
#include<vector>
using namespace std;

typedef struct assento
{
 bool ocupado; // vazio ou livre
 int coluna;// coluna do assento
 int fila;// fila do assento
}Assento;

class Sala
{
  protected:
   int IDsala;
   int filas;
   int colunas;
   vector<vector<Assento>> Assentos;
  public:
   Sala(int IDsala,int filas,int colunas); // Construtor
```

```
    bool verificar_livre(int &fila,int &colunas); // verifica assento
livre
    int ocupar(int &filas,int &colunas); // preenche assento
    int desocupar(int &filas,int &colunas); // libera assento
    void info_sala(); // imprime informações da sala
};
#endif
```

Sessão.h

```
#ifndef SESSAO_H
#define SESSAO_H
#include "Sala.h"
#include "Programaçao.h"
#include "Ingresso.h"

class Sessao : virtual public Sala , Programaçao
{
 protected:
   int IDsessao;
   bool encerrada;
   float valor_inteira;
   vector<Ingresso>Ingressos;
 public:
        Sessao(int IDsessao,bool encerrada,float valor_inteira,int
IDsala,int filas,int colunas,int dia,int mes,int ano,string
horario,string filme,string idioma,bool legenda,bool tresD);//
construtor classe sessao
   ~Sessao(); //destrutor classe sessao
   void info_sessao(); // imprime informações da sessao
   void info_sala();  // imprime informações da sala
   int get_IDsessao(); // Get ID sessao
   int get_IDsala(); // Get ID sala
   string get_filme(); // Get Filme
   float get_valor(); // Get Valor
   void set_encerrada(bool encerrada); // encerra sessao
          void comprar_ingresso(string cpf, int &IDsessao,bool
meia_entrada,int &fila,int &coluna); // comprar um ingresso
   bool getencerrada(); // Get encerra
   void cancelar_ingresso(string IDingresso); // Cancelar ingresso
   string vendas(); // resumos de vendas
   vector<string> getIngresso(string IDingresso); // retorna Ingresso
```

```
   bool verificaingresso(string IDingresso); // verificar Ingresso
};
 #endif
```

Ingresso.h

```
#ifndef INGRESSO_H
#define INGRESSO_H
#include <iostream>
#include <vector>
using namespace std;

class Ingresso
{
 private:
   bool meia_entrada;
   string cpf;
   string IDingresso;
   float valor;
   int fila;
   int coluna;
 public:
          Ingresso(string   IDingresso,string   cpf,float   valor,bool
meia_entrada,int fila,int coluna);// construtor ingresso
   float getValor(); // Get Valor
   vector<string> get_ingresso();// Get Ingresso
   string getIDingresso(); // Get ID do ingresso
};
#endif
```

Programacao.h

```
#ifndef PROGRAMAÇAO_H
#define PROGRAMAÇAO_H
#include<iostream>
#include<vector>
using namespace std;

class Programaçao
{
 protected:
   int dia;
   int mes;
```

```
   int ano;
   string horario;
   string filme;
   string idioma;
   bool legenda;
   bool tresD;
 public:
        Programaçao(int  dia,int  mes,int  ano,string  horario,string
filme,string idioma,bool legenda,bool tresD);
   // Construtor programação
   void info_programaçao(); // Imprime informações da programação
   vector<string> get_programaçao();// Get programação
};
#endif
```

GerenteCinema.h

```
#ifndef GERENTECINEMA_H
#define GERENTECINEMA_H
#include "Sessao.h"
#include<iostream>
using namespace std;

class GerenteCinema
{
 private:
 int IDcinema;
 string senha;
 vector<Sessao> Sessoes;
 public:
   GerenteCinema(int IDcinema,string senha);
   // Construtor GerenteCinema
   void CriarSessao(int &IDsessao,bool encerrada,float
&valor_interia,int &IDsala,int &filas,int &colunas,int &dia,int
&mes,int &ano,string horario,string filme,string idioma,bool
legenda,bool tresD);
   // Criar uma nova sessão
   bool AbrirSessao(int &IDsessao);
   // Abrir Sessão
   bool FecharSessao(int &IDsessao);
   // Encerra a sessão
   void ComprarIngresso(string cpf,int &IDsessao,bool meia_entrada,int
```

```
&fila,int &coluna);
    // Comprar um novo Ingresso
    bool verificaSessao(int &IDsessao);
    // Verificar Sessão
    void CancelarIngresso(string IDingresso,int &IDsessao);
    // Cancelar Ingresso
    vector<string> EmitirIngresso(string IDingresso,int &IDsessao);
    // Imprime o ingresso
    bool verifica_ingresso(string IDingresso,int &IDsessao);
    // Verifica o ingresso
    void CalcularVendapSessao(int &IDsessao);
    // Calcula e retorna as vendas por sessão
    void CalcularVendas();
    // Calcula o faturamento geral
    void InfoSessao(int &IDsessao);
    // Imprime informações da sessão
    void InfoSala(int &IDsala);
    // Imprime Informações da sala
    void InfoProgramaçao(string filme);
    // Imprime Informações da programação
};

#endif
```

3. Conteúdos usados no projeto

- Separando a interface da classe de sua implementação;

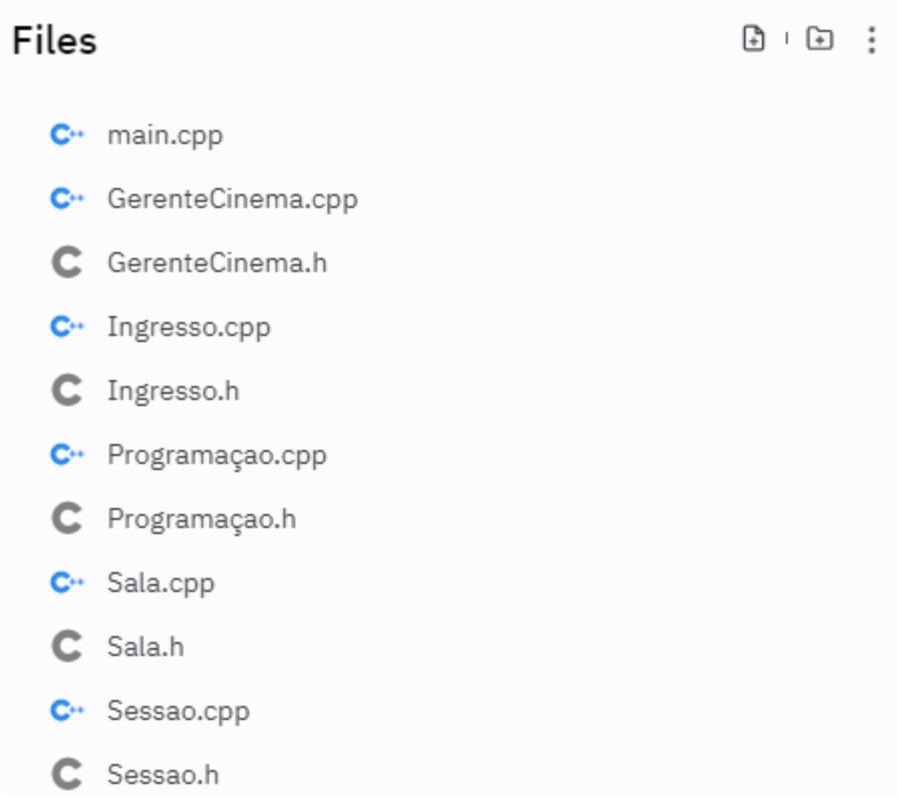

Figura 1 - Separação do .h e .cpp

- Uso de construtores;

```
Sala::Sala(int numSala, int nFileiras, int nAssentosFileira){
    estado = "disponivel";
    this->numSala = numSala;
    this->nFileiras = nFileiras;
    this->nAssentosFileira = nAssentosFileira;
    char letra = 'a';
    f.resize(nFileiras);

    for(int i = 0; i < this->nFileiras; i++){
        f[i].setIdFileira(letra);
        f[i].iniciaAssentos(nAssentosFileira);
        letra++;
    }
}
```

Figura 2 - Utilização de construtor

- Passando objetos por referência;

```
bool AbrirSessao(int &IDsessao);
bool FecharSessao(int &IDsessao);
void ComprarIngresso(string cpf,int &IDsessao,bool meia_entrada,int &fila,int &coluna);
void CancelarIngresso(int &IDingresso,int &IDsessao);
void EmitirIngresso(int &IDingresso);
void CalcularVendas();
void CalcularVendapSessao(int &IDsessao);
void InfoSessao(int &IDsessao);
void InfoSala(int &IDsala);
void InfoProgramacao(int &IDsessao);
```

Figura 3 - Passando objetos por referência

- Herança;

```
#include "Sessao.h"
#include<iostream>
using namespace std;

class GerenteCinema
{
  private:
  int IDcinema;
  string senha;
  vector<Sessao> Sessoes;
```

Figura 4 - Utilização de herança

- Sobrecarga de operadores;

```
void ComprarIngresso(string cpf,int &IDsessao,bool meia_entrada,int &fila,int &coluna);
// Comprar um novo Ingresso
bool verificaSessao(int &IDsessao);
// Verificar Sessao


void CancelarIngresso(string IDingresso,int &IDsessao);
// Cancelar Ingresso


vector<string> EmitirIngresso(string IDingresso,int &IDsessao);
// Imprime o ingresso
```

Figura 5 - SobreCarga - GerenteCinema.h

```
void comprar_ingresso(string cpf, int &IDsessao,bool meia_entrada,int &fila,int &coluna); // comprar um
ingresso
bool getencerrada(); // Get encerra

void cancelar_ingresso(string IDingresso); // Cancelar ingresso


string vendas(); // resumos de vendas
```

Figura 6 - SobreCarga - Sessao.h

- Uso da biblioteca STL.

```
#include <vector>
#include "Fileira.h"
using namespace std;


class Sala{

    private:

        vector<Fileira> f;
```

Figura 7 - Utilização de vector da biblioteca STL.

- Uso da biblioteca iostream - CIN e COUT;

```
int IDsessao,IDsala,filas,colunas,dia,mes,ano;
bool encerrada,legenda,tresD;
float valor_inteira;
string horario,idioma,filme;
cout<<"Digite ID da Sessao: "<<endl;
cin >>IDsessao;
cout<<"Digite ID da Sala: "<<endl;
cin >>IDsala;
cout<<"Digite Dia da Sessao: "<<endl;
cin >>dia;
cout<<"Digite Mes da Sessao: "<<endl;
cin >>mes;
```

Figura 8 - Utilização da biblioteca iostream - Entrada e Saída.

A primeira dificuldade foi comunicação e principalmente a formação da dupla para o projeto, já a segunda dificuldade encontrada é o curto tempo de desenvolvimento do projeto, porque é extenso. Além disso, a relação entre sala.h e GerenteCinema.h é muito complicada, aumentando a dificuldade do método de implementação de um assento em uma determinada sala. porque é grande e contém muitas classes de vetores, leva mais tempo Implementação.

4. Conclusão

Com isso temos que a princípio o programa funciona corretamente, mas não foram testadas entradas diferentes das esperadas, como por exemplo, o uso de character quando é esperado um inteiro e vice-versa. E como sugestão de 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 devido ao curto período de tempo se tornou inviável tratar todas as possíveis entradas, este seria outro ponto a se melhorar.