UNIVERSIDADE FEDERAL DE CAMPINA GRANDE

CENTRO DE ENGENHARIA E ELÉTRICA E INFORMÁTICA

CURSO DE ENGENHARIA ELÉTRICA

DISCIPLINA TÉCNICAS DE PROGRAMAÇÃO – PERÍODO 2019.2

PROF. DR. MARCUS SALERNO DE AQUINO

PROJETO

Controle de venda de ingresso em cinema

Julia Ramalho Costa Souza

Maria Gabriela Abreu de Souza

Moisés de Araújo Oliveira

Samara Lima Cardoso

CAMPINA GRANDE/PB

DEZEMBRO/2019

SUMÁRIO

I. Resumo

Para realização deste trabalho foi feita uma pesquisa explicativa e descritiva, a qual teve seu enfoque voltado para a tentativa de compreender, descrever e aplicar os assuntos abordados em sala de aula durante as aulas e os temas dos seminários da disciplina de técnicas de programação. Com o entendimento dos referidos assuntos, foi possível desenvolver o projeto em relação ao controle de venda de ingresso no cinema.

II. Introdução

O programa consiste, basicamente, na elaboração de um programa para o controle de vendas em um cinema. Para isso, foram utilizados métodos eficazes de programação, como manipulação de arquivo, tratamento de erro, alocação dinâmica e herança. Além disso, foi feito um diagrama de classes, a fim de mostrar de forma clara as relações de herança entre as classes do projeto.

III. Revisão Bibliográfica

Para o desenvolvimento do projeto foram necessárias ferramentas que abrangem as mais diversas áreas da informática, incluindo a área de programação, conhecimento esse, adquirido nas disciplinas de introdução a programação e técnicas de programação. Para o desenvolvimento do projeto utilizamos a linguagem C++ e para criação dos diagramas de classes e de caso de uso, usamos a ferramenta Astah Community.

- **C++:**

O desenvolvimento dessa linguagem começou na década de 80, é uma linguagem de programação de nível médio e baseada na linguagem C. O C++ é uma linguagem de programação com vários paradigmas, incluindo linguagem imperativa, orientada a objetos e genérica e de uso geral.

- **ASTAH COMMUNITY:**

Software para a criação do diagrama de caso de uso e diagrama de classe, que especifica a atuação de cada indivíduo no processo de desenvolvimento, ou seja, essa ferramenta é utilizada para o desenvolvimento dos diagramas necessários para representar o projeto dos requisitos.

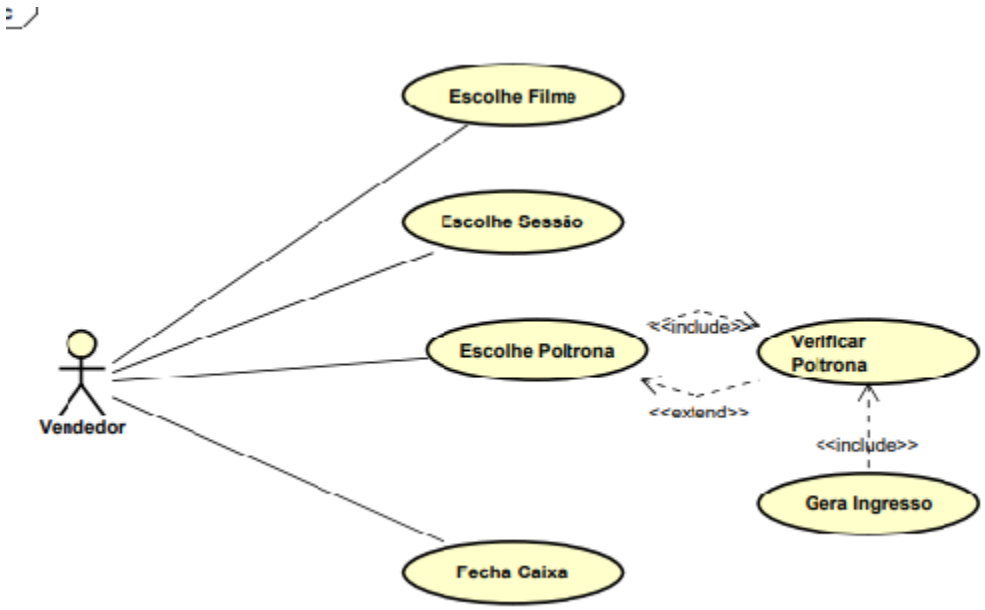
Além disso, outros recursos foram usados para que o desenvolvimento do projeto fosse executado com êxito, tais como: alocação dinâmica, manipulação de arquivo, tratamento

de erro, diagrama de classe e de caso de uso, container e o Falcon (programa para a escrita do código).

O projeto é uma simulação de venda de ingressos no cinema, apresentando duas opções: uma para vender ingresso e outra para o controle de ingressos vendidos no dia. Ao selecionar a opção vender ingresso, o programa lhe fornece cinco opções de filmes, apresentando quatro sessões para cada filme com quatro tipos de sala, podendo escolher sua poltrona e o tipo de ingresso. E por fim, selecionando a segunda opção, o programa mostrará a quantidade de ingressos vendidos ao final do dia.

Com a ideia do projeto pronta criamos o diagrama de caso de uso, mostrado na figura 1, para auxiliar nos requisitos funcionais do sistema, descrevendo a funcionalidade e as interações com os elementos.

Figura 1. Diagrama de caso de uso

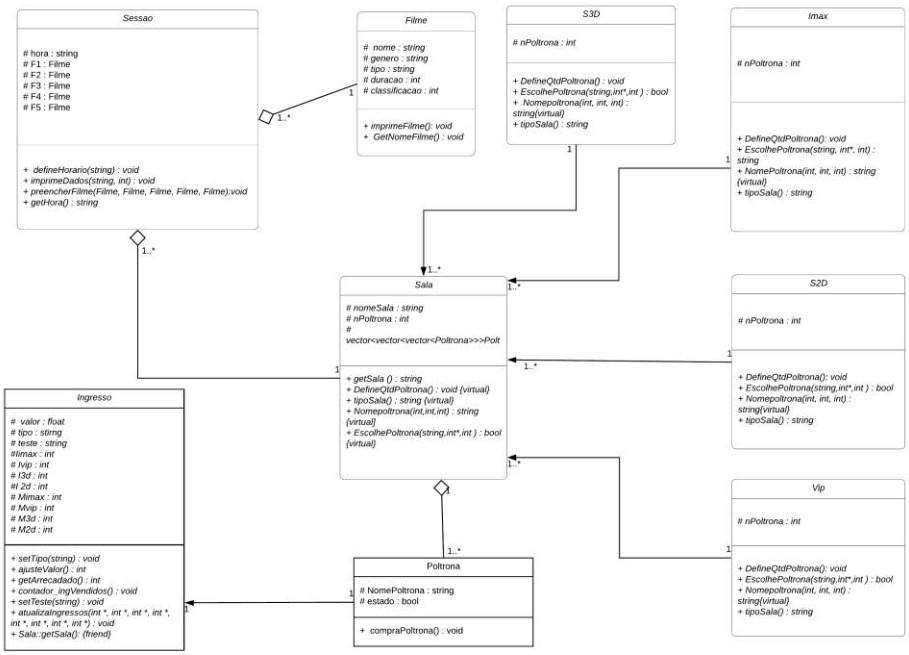


Fonte: autores (2019)

explicar o funcionamento

Com o diagrama de caso de uso concluído, elaboramos o diagrama de classes, figura 2, para mostrar de forma clara e objetiva a estrutura do projeto, evidenciando a relação entre as classes com seus respectivos métodos e atributos.

Figura 2. Diagrama de classes



Fonte: autores(2019)

O diagrama de classes acima mostra todas as classes do projeto e todas as relações entre elas. Primeiramente, é importante salientar a relação de herança entre algumas classes, as classes S3D, S2D, Imax e Vip são filhas da classe “Sala”. Além disso, podem existir sessões de várias salas, e como Sala possui vector tridimensional, ela será “renovada” a cada sessão estabelecida. Esse relacionamento com a classe sala vale para as classes :Sessao, Imax, S2D, S3D e Vip.

O relacionamento da classe poltrona com a classe ingresso é a seguinte : Um ingresso só pode ter uma poltrona, onde uma poltrona só poderá ter um ingresso, ou seja, os ingressos são individuais. A classe Ingresso e a classe Sala são friends, para que os métodos e atributos de sala possam ser acessados em Ingresso. Finalmente, temos o relacionamento entre as classes Filme e Sessao, onde um filme pode ter várias sessões, no entanto uma sessão só pode ter um único filme.

Com a base do projeto concluída, a escrita do código foi inicializada, implementando as classes com seus respectivos atributos e métodos, resultando em 11 arquivos.h, 10 arquivos.cpp e a main. A lógica do programa é que ao escolher a opção comprar ingresso, haverá uma sequência na compra, inicialmente o filme será escolhido, após isso a sessão será

escolhida, posteriormente a poltrona e por fim, seleciona o tipo de ingresso. Com base nisso, será explicado a funcionalidade do código referente os principais arquivos do projeto, tais como:

- Filme.h/Filme.cpp;
- Sessao.h/Sessao.cpp;
- Poltrona.h/Poltrona.cpp;
- Ingresso.h/Ingresso.cpp;

Quando escolhemos o filme que desejamos assistir no cinema, olhamos algumas características do mesmo, por exemplo: nome do filme, gênero, diretor, tempo de duração e classificação indicativa. A classe Filme, apresentada na figura 3, apresenta métodos e atributos com o objetivo de mostrar ao usuário as características descritas acima, ao total são cinco filmes para o usuário escolher qual deseja assistir.

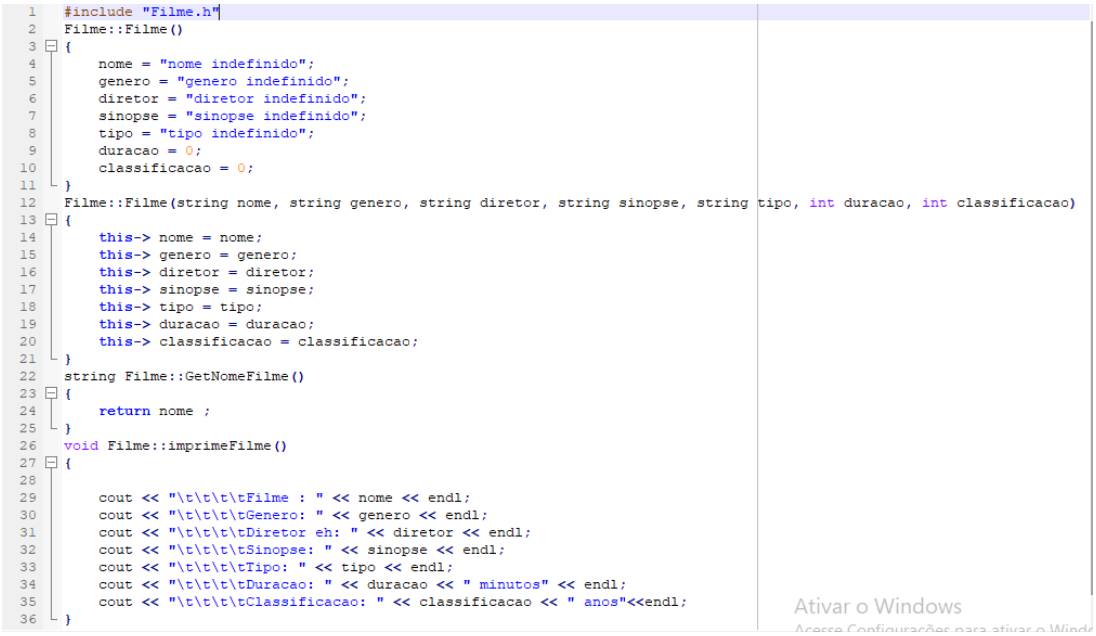
Figura 3. Arquivo Filme.h

```
1  #ifndef FILME_H
2  #define FILME_H
3  #include <string>
4  #include <iostream>
5  using namespace std;
6
7  class Filme
8  {
9  protected:
10     string nome, genero, diretor, sinopse, tipo;
11     int duracao, classificacao; // DURAÇÃO É EM MINUTOS
12 public:
13     Filme();
14     Filme(string, string, string, string, string, int, int);
15     void imprimeFilme();
16     string GetNomeFilme();
17 };
18
19 #endif
```

Fonte: Autores (2019)

O arquivo apresentado acima, mostra os atributos, responsáveis por armazenar as características do filme e os métodos que serão implementados no arquivo Filme.cpp ,mostrado na figura 4, para que as características do mesmo sejam armazenadas com êxito. Há a criação de cinco variáveis do tipo “string”, destinadas a armazenar o nome do filme, gênero,diretor,sinopse e tipo(dublado/legendado). Além disso, declaração de duas variáveis do tipo “int” para armazenar o tempo de duração e a classificação indicativa do filme, inicializando o construtor default da classe “Filme”, onde as funções foram declaradas para serem executadas no arquivo Filme.cpp .

Figura 4. Arquivo Filme.cpp



```cpp
#include "Filme.h"
Filme::Filme()
{
    nome = "nome indefinido";
    genero = "genero indefinido";
    diretor = "diretor indefinido";
    sinopse = "sinopse indefinido";
    tipo = "tipo indefinido";
    duracao = 0;
    classificacao = 0;
}
Filme::Filme(string nome, string genero, string diretor, string sinopse, string tipo, int duracao, int classificacao)
{
    this-> nome = nome;
    this-> genero = genero;
    this-> diretor = diretor;
    this-> sinopse = sinopse;
    this-> tipo = tipo;
    this-> duracao = duracao;
    this-> classificacao = classificacao;
}
string Filme::GetNomeFilme()
{
    return nome ;
}
void Filme::imprimeFilme()
{

    cout << "\t\t\t\tFilme : " << nome << endl;
    cout << "\t\t\t\tGenero: " << genero << endl;
    cout << "\t\t\t\tDiretor eh: " << diretor << endl;
    cout << "\t\t\t\tSinopse: " << sinopse << endl;
    cout << "\t\t\t\tTipo: " << tipo << endl;
    cout << "\t\t\t\tDuracao: " << duracao << " minutos" << endl;
    cout << "\t\t\t\tClassificacao: " << classificacao << " anos"<<endl;
}
```


Fonte: Autores (2019)

Com a declaração do construtor e a inicialização das variáveis feita no mesmo, os métodos já declarados no arquivo “Filme.h”,mostrado na figura 3, estão aptos para que seus códigos sejam escritos. O método Filme tem a finalidade de atribuir todas as características do filme, as armazenando nos atributos destinados a cada uma. Já o método “GetNomeFilme” tem a responsabilidade de “guardar” o nome do filme respectivo,método bastante útil quando a impressão do ingresso for realizada. Enquanto o método “imprimeFilme”, mostrará todas as peculiaridades do filme declaradas anteriormente no método Filme.

Com o filme escolhido, o próximo passo é a escolha da sessão, onde constará as opções de horários e o tipo de sala (Vip/ Imax/2D/3D). Com base nisso, o arquivo Sessao.h,mostrado na figura 5, nos fornece os atributos e métodos necessários para que os mesmos sejam implementados no arquivo Sessao.cpp para que as informações(horário/tipo de sala) da sessão sejam implementadas.

Figura 5. Arquivo Sessao.h

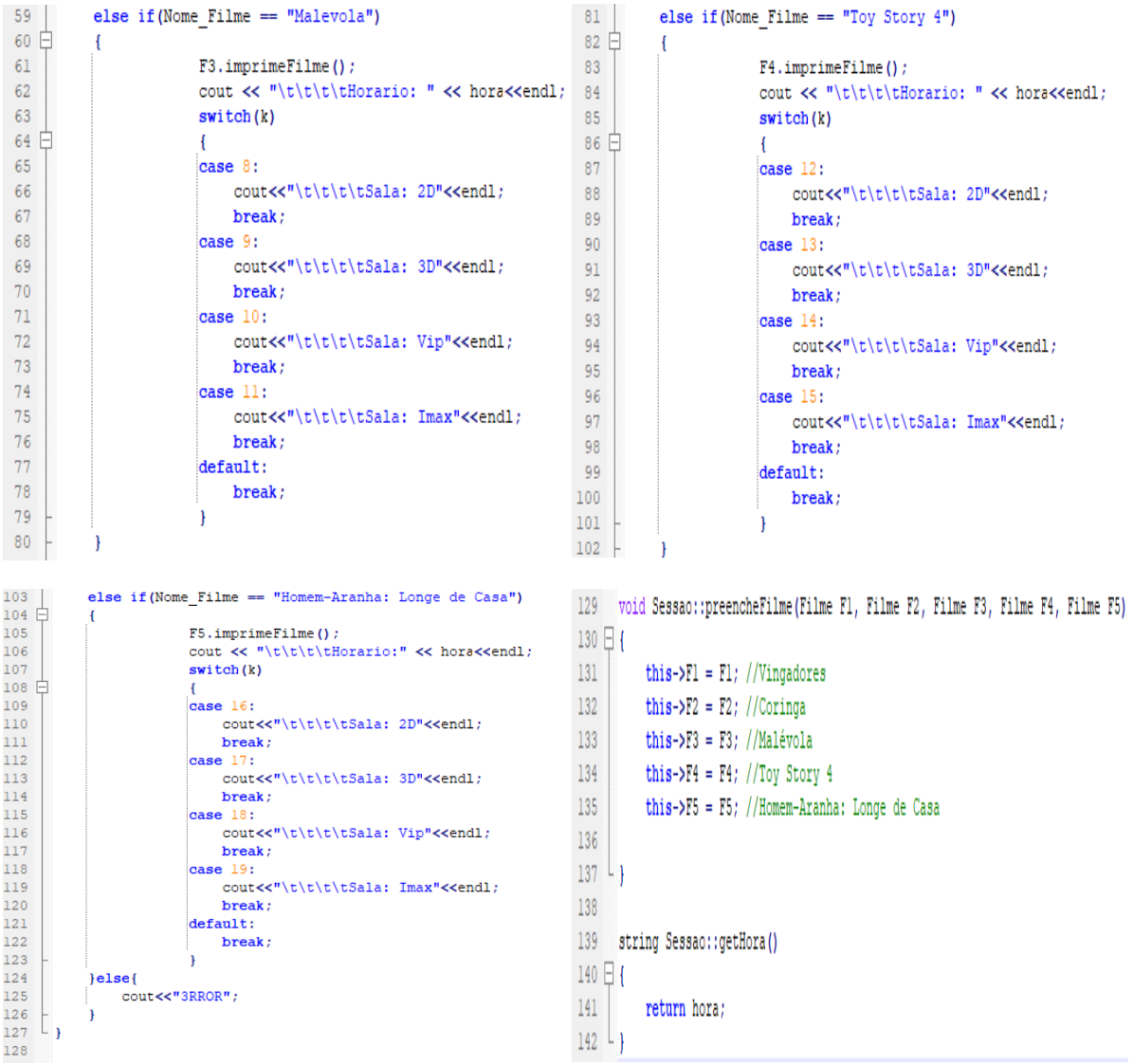
```
#ifndef SESSAO_H
#define SESSAO_H
#include "Filme.h"
#include "Sala.h"
#include "Horario.h"
#include "S3D.h"
#include "S2D.h"
#include "Imax.h"
#include "Vip.h"

class Sessao
{
protected:
    Filme F1,F2,F3,F4,F5;
    string hora;
public:
    Sessao();
    void DefiniHorario(string);
    void imprimeDados(string, int);
    void preencheFilme(Filme,Filme,Filme,Filme,Filme);
    string getHora();
};

#endif
```

Fonte: Autores (2019)

A classe “Sessao”,mostrada na figura 5, foi implementada para que houvesse a possibilidade de definir o horário e o tipo de sala nas sessões, onde a mesma possui métodos e atributos que torna essa funcionalidade possível. A classe “Sessao” apresenta uma variável do tipo “string” chamada hora, armazenando as horas no formato do tipo “15:00 hrs”,sendo estes valores pré-estabelecidos. Além disso, cinco objetos de filme foram declarados como protected, possibilitando que fossem alterados por outras classes, para os respectivos cinco filmes oferecidos no cinema. A partir disso, o construtor default foi inicializado, assim como os métodos,prontos para serem executados no arquivo Sessao.cpp .

Figura 6. Arquivo Sessao.cpp

```
#include "Sessao.h"
Sessao::Sessao()
{
    Sala*s2d = new S2D;
    Sala*s3d = new S3D;
    Sala*vip1 = new Vip;
    Sala*Imax1 = new Imax;
}
void Sessao::DefiniHorario(string hora)
{
    this->hora = hora;
}
void Sessao::imprimeDados(string Nome_Filme, int k)
{
    if(Nome_Filme == "Vingadores")
    {
                F1.imprimeFilme();
                cout << "\t\t\t\tHorario: " << hora<<endl;
                switch(k)
                {
                case 0:
                    cout<<"\t\t\t\tSala: 2D"<<endl;
                    break;
                case 1:
                    cout<<"\t\t\t\tSala: 3D"<<endl;
                    break;
                case 2:
                    cout<<"\t\t\t\tSala: Vip"<<endl;
                    break;
                case 3:
                    cout<<"\t\t\t\tSala: Imax"<<endl;
                    break;
                default:
                    break;
                }
    }
    else if(Nome_Filme == "Coringa")
    {
                F2.imprimeFilme();
                cout << "\t\t\t\tHorario: " << hora<<endl;
                switch(k)
                {
                case 4:
                    cout<<"\t\t\t\tSala: 2D"<<endl;
                    break;
                case 5:
                    cout<<"\t\t\t\tSala: 3D"<<endl;
                    break;
                case 6:
                    cout<<"\t\t\t\tSala: Vip"<<endl;
                    break;
                case 7:
                    cout<<"\t\t\t\tSala: Imax"<<endl;
                    break;
                default:
                    break;
                }
    }
```



```
else if(Nome_Filme == "Malevola")
{
            F3.imprimeFilme();
            cout << "\t\t\t\tHorario: " << hora<<endl;
            switch(k)
            {
            case 8:
                cout<<"\t\t\t\tSala: 2D"<<endl;
                break;
            case 9:
                cout<<"\t\t\t\tSala: 3D"<<endl;
                break;
            case 10:
                cout<<"\t\t\t\tSala: Vip"<<endl;
                break;
            case 11:
                cout<<"\t\t\t\tSala: Imax"<<endl;
                break;
            default:
                break;
            }
}
```

```
else if(Nome_Filme == "Toy Story 4")
{
            F4.imprimeFilme();
            cout << "\t\t\t\tHorario: " << hora<<endl;
            switch(k)
            {
            case 12:
                cout<<"\t\t\t\tSala: 2D"<<endl;
                break;
            case 13:
                cout<<"\t\t\t\tSala: 3D"<<endl;
                break;
            case 14:
                cout<<"\t\t\t\tSala: Vip"<<endl;
                break;
            case 15:
                cout<<"\t\t\t\tSala: Imax"<<endl;
                break;
            default:
                break;
            }
}
```

```
else if(Nome_Filme == "Homem-Aranha: Longe de Casa")
{
            F5.imprimeFilme();
            cout << "\t\t\t\tHorario:" << hora<<endl;
            switch(k)
            {
            case 16:
                cout<<"\t\t\t\tSala: 2D"<<endl;
                break;
            case 17:
                cout<<"\t\t\t\tSala: 3D"<<endl;
                break;
            case 18:
                cout<<"\t\t\t\tSala: Vip"<<endl;
                break;
            case 19:
                cout<<"\t\t\t\tSala: Imax"<<endl;
                break;
            default:
                break;
            }
    }else{
        cout<<"3RROR";
    }
}
```

```
void Sessao::preencheFilme(Filme F1, Filme F2, Filme F3, Filme F4, Filme F5)
{
    this->F1 = F1; //Vingadores
    this->F2 = F2; //Coringa
    this->F3 = F3; //Malévola
    this->F4 = F4; //Toy Story 4
    this->F5 = F5; //Homem-Aranha: Longe de Casa

}

string Sessao::getHora()
{
    return hora;
}
```

Fonte: Autores (2019)

Ao começar a escrita do código do arquivo Sessao.cpp,mostrado na figura 6 , um dos pré requisitos foi a utilização de alocação dinâmica, ferramenta esta, presente na inicialização do construtor, já que a palavra chave “new” é um operador que faz a memória ser armazenada durante o tempo de execução para os referidos apontadores. Além disso, o método “DefineHorario”, tem a responsabilidade de armazenar o horário da sessão no formato definido(YY:00 hrs) e o método “getHora” para “guardar”a hora estabelecida no método dito anteriormente.

Já o método imprimeDados,mostrará a sessão do respectivo filme,e para que isso seja possível, foi necessário utilizar a ferramenta ‘if/if else”,havendo cinco casos para os cincos filmes disponíveis, pois dependendo do tipo de filme, a sessão apresentará horário e tipo de sala diferente, sendo assim para que o programa “leia” qual o filme pertende a referida sessão,

chama-se o objeto criado de filme (F1,F2,F3,F4,F5) no arquivo Sessao.h junto com a função "imprimeFilme". E por fim, o método "preencheFilme", faz com que os endereços dos objetos (F1,F2,F3,F4,F5) sejam armazenados por meio do apontador "this".

Na escolha da sessão do filme, além de apresentar os horários disponíveis, o tipo de sala (Imax,Vip,2D,3D) também constará. Com isso definido, o projeto apresenta uma classe chamada "Sala", possuindo atributos e métodos de suma importância, pois as classes derivadas necessitam ter acesso as informações da classe "sala", para exercer suas funções da forma correta. Apresentando quatro tipos de sala, serão quatro classes, uma para cada tipo de sala, todas derivadas de classe "Sala", além disso, cada tipo de sala apresenta capacidades diferentes, por exemplo: sala Imax apresenta 100 poltronas, sala Vip comn36 poltronas, enquanto as salas 2D e 3D apresentam 64 poltronas.

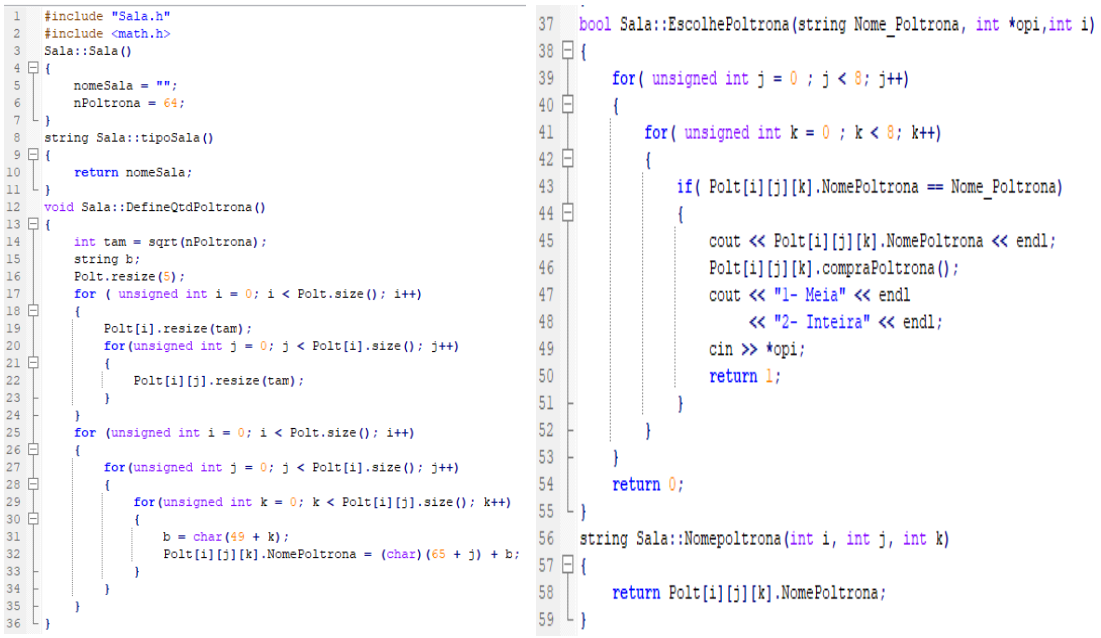
Figura 7. Arquivo Sala.h

```
#ifndef SALA_H
#define SALA_H
#include "Poltrona.h"
#include <vector>
#include <iostream>

using namespace std;

class Sala
{
protected:
    string nomeSala;
    int nPoltrona;
    vector <vector< vector<Poltrona> > >Polt;
public:
    Sala();
    string getSala();
    virtual void DefineQtdPoltrona();
    virtual string tipoSala();
    virtual string Nomepoltrona(int,int,int);
    virtual bool EscolhePoltrona(string,int*,int );
    friend class Ingresso;

};

#endif
```

Fonte: Autores (2019)

EXPLICAR ESSE ARQUIVO

Figura 8. Arquivo Sala.cpp



```cpp
#include "Sala.h"
#include <math.h>
Sala::Sala()
{
    nomeSala = "";
    nPoltrona = 64;
}
string Sala::tipoSala()
{
    return nomeSala;
}
void Sala::DefineQtdPoltrona()
{
    int tam = sqrt(nPoltrona);
    string b;
    Polt.resize(5);
    for ( unsigned int i = 0; i < Polt.size(); i++)
    {
        Polt[i].resize(tam);
        for(unsigned int j = 0; j < Polt[i].size(); j++)
        {
            Polt[i][j].resize(tam);
        }
    }
    for (unsigned int i = 0; i < Polt.size(); i++)
    {
        for(unsigned int j = 0; j < Polt[i].size(); j++)
        {
            for(unsigned int k = 0; k < Polt[i][j].size(); k++)
            {
                b = char(49 + k);
                Polt[i][j][k].NomePoltrona = (char)(65 + j) + b;
            }
        }
    }
}
bool Sala::EscolhePoltrona(string Nome_Poltrona, int *opi,int i)
{
    for( unsigned int j = 0 ; j < 8; j++)
    {
        for( unsigned int k = 0 ; k < 8; k++)
        {
            if( Polt[i][j][k].NomePoltrona == Nome_Poltrona)
            {
                cout << Polt[i][j][k].NomePoltrona << endl;
                Polt[i][j][k].compraPoltrona();
                cout << "1- Meia" << endl
                     << "2- Inteira" << endl;
                cin >> *opi;
                return 1;
            }
        }
    }
    return 0;
}
string Sala::Nomepoltrona(int i, int j, int k)
{
    return Polt[i][j][k].NomePoltrona;
}
```

Fonte: Autores (2019)

EXPLICAR ESSE CÓDIGO

Visto que o filme e a sessão já foram escolhidos, será necessário selecionar a poltrona. Para que essa operação seja possível, o arquivo Poltrona.h, mostrado na figura 9, apresenta atributos e métodos que serão implementados no arquivo Poltrona.cpp, para que o programa forneça ao usuário o estado (ocupada/disponível) da poltrona, caso esteja ocupada , será pedido ao usuário para escolher outra poltrona.

Figura 9. Arquivo Poltrona.h

```
#ifndef POLTRONA_H
#define POLTRONA_H

#include <string>
#include <iostream>

using namespace std;

class Poltrona
{
protected:

public:
    string NomePoltrona;
    bool estado; // 0-Livre
    Poltrona();
    void compraPoltrona();

};
#endif
```

Fonte: Autores (2019)

O objetivo da classe “Poltrona”, mostrado na figura9, é informar no ato da compra do ingresso, se a poltrona encontra-se ocupada ou disponível. Para que isso seja possível, uma variável do tipo “string”chamada NomePoltrona, foi declarada para mostrar ao usuário se a poltrona encontra-se disponível ou não, além disso, a declaração da variável “ bool”, afim de tornar o código mais simples no momento de verificação do estado da poltrona. Sendo notório a inicialização do construtor default e do método “compraPoltrona”, aptos para serem desenvolvidos no arquivo “Poltrona.cpp” .

Figura 10. Arquivo Poltrona.cpp

```
#include "Poltrona.h"

Poltrona::Poltrona()
{
    NomePoltrona="";
    estado=0;
}
void Poltrona::compraPoltrona()
{
    if(estado){
        cout<<"OCUPADA";
    }else{

        estado=1;
        NomePoltrona="--";
    }
}
```

Fonte: Autores (2019)

No arquivo “Poltrona.cpp”, o construtor é declarado, assim como os atributos são inicializados no mesmo, visto que a variável do tipo”bool” chamada estado, já se inicializa com o valor 0, possuindo o estado “ocupada”. Dessa forma, será pedido para que o usuário escolha outra poltrona, quando a variável estado apresentar o valor 1, a poltrona está disponível e poderá ser comprada.

Com a poltrona escolhida, o último passo é pagar o valor do ingresso, dependendo do tipo de ingresso (meia entrada/ingresso normal), o valor irá variar. É justamente isso que os atributos e métodos do arquivo Ingresso.h fazem, possibilitam que quando os mesmos forem implementados no arquivo Ingresso.cpp, o programa entenda que o valor do ingresso irá variar e que o usuário poderá escolher o tipo de ingresso.

Figura 11. Arquivo Ingresso.h

```
#ifndef INGRESSO_H
#define INGRESSO_H
#include "Poltrona.h"
#include "Sala.h"


class Ingresso : public Poltrona
{
protected:
    float valor;
    string tipo,teste;
    int Iimax,Ivip,I3d,I2d,Mimax,Mvip,M3d,M2d; // contador para vendas
    float arrecadado;
public:
    Ingresso();
    void setTipo(string); // adicionar ao carrinho
    int ajusteValor(); // para cada tipo de sala usando classe friend
    int getArrecadado();
    void contador_ingVendidos();
    void setTeste(string);
    void atualizaIngressos(int *, int *, int *, int *, int *, int *, int *, int *);
    friend string Sala::getSala();

};

#endif
```

Fonte: Autores (2019)

O arquivo “Ingresso.h”, mostrado na figura 11, apresenta a classe Ingresso com seus respectivos métodos e atributos, que irão possibilitar que o valor do ingresso seja impresso, dependendo do tipo de ingresso(meia entrada/normal) que o usuário escolher, como também o tipo de sala que irá afetar no valor final. Os atributos são declarados, apresentando uma variável do tipo “float” chamada valor (referente ao valor real do ingresso),duas variáveis do tipo “string” chamada de tipo(refere-se ao tipo de ingresso:meia entrada ou normal) e teste (verifica qual o tipo de sala que o usuário escolheu), cinco variáveis do tipo “int” com o objetivo e fazer uma contagem da quantidade de ingressos vendidos de cada tipo de sala (Vip,

2D, 3D, Imax) e por fim uma variável do tipo float chamada “arrecadado”, onde é a soma de todos os ingressos. Além disso, a declaração do construtor default e dos métodos, para que possam ser executados no arquivo “Ingresso.cpp” e mostrar o valor do ingresso.

Figura 12. Arquivo Ingresso.cpp

```cpp
#include "Ingresso.h"
Ingresso::Ingresso()
{
    valor = 0;
    tipo = "";
    arrecadado = 0;
    Iimax = 0;
    Ivip = 0;
    I3d = 0;
    I2d = 0;
    Mimax = 0;
    Mvip = 0;
    M3d = 0;
    M2d = 0;
}
void Ingresso::setTipo( string tipo)  // adicionar ao carrinho
{
    this->tipo = tipo;
}
int Ingresso::ajusteValor()  // para cada tipo de sala usando classe friend
{
    if( teste == "VIP")
    {
        if(tipo == "meia")
        {
            valor = 21;
            return 21;
        }
        else
        {
            valor = 42;
            return 42;
        }
    }
    else if(teste == "Imax")
    {
        if(tipo == "meia")
        {
            valor = 18;
            return 18;
        }
        else
        {
            valor = 36;
            return 36;
        }
    }
    else if(teste == "3D")
    {
        if(tipo == "meia")
        {
            valor = 15;
            return 15;
        }
        else
        {
            valor = 30;
            return 30;
        }
    }
    else if(teste == "2D")
    {
        if(tipo == "meia")
        {
            valor = 12;
            return 12;
        }
        else
        {
            valor = 24;
            return 24;
        }
    }else{
        return 0;
    }
}
int Ingresso::getArrecadado()
{
    arrecadado = (Mvip * 21) + (Mimax * 18) + (M3d * 15) + (M2d * 12) + (Ivip * 42) + (Iimax * 36) + (I3d * 30) + (I2d * 24);
    return arrecadado;
}

void Ingresso::setTeste(string teste){
    this->teste=teste;
}

```

```
void Ingresso::contador_ingVendidos()
{
    if(teste == "vip")
    {
        if(tipo == "meia")
        {
            Mvip++;
        }
        else
        {
            Ivip++;
        }
    }
    else if(teste == "IMAX")
    {
        if(tipo == "meia")
        {
            Mimax++;
        }
        else
        {
            Iimax++;
        }
    }
    else if(teste == "3D")
    {
        if(tipo == "meia")
        {
            M3d++;
        }
        else
        {
            I3d++;
        }
    }
   else if(teste == "2D")
    {
       if(tipo == "meia")
       {
           M2d++;
       }
       else
       {
           I2d++;
       }
    }
}

void Ingresso::atualizaIngressos(int *pIimax, int *pIvip, int *pI3d, int *pI2d, int *pMimax, int *pMvip, int *pM3d, int *pM2d)
{
    *pIimax = Iimax;
    *pIvip = Ivip;
    *pI3d = I3d;
    *pI2d = I2d;
    *pMimax = Mimax;
    *pMvip = Mvip;
    *pM3d = M3d;
    *pM2d = M2d;
}
```


Fonte: Autores (2019)

O arquivo “Ingresso.cpp”,mostrado na figura 12, temos declaração do construtor, inicializando os atributos no mesmo. O método “ajusteValor’ irá configurar todos os valores do ingresso para cada tipo de sala e para cada tipo de ingresso correspondente a cada sala. Como também, a função “getArrecadado” irá retornar o valor total arrecadado com a venda dos ingressos e o método “contador_ingVendidos” irá retornar a quantidade total de ingressos vendidos. Por fim, atualiza-se a nova quantidade de ingressos com o método “atualizaIngressos”,utilizando apontadores.

Ao escolher a opção 2 do menu principal (controle de vendas) faz-se necessário contabilizar o quanto foi arrecadado ao longo do dia, afim de para controlar as finanças do cinema. O código apresenta a opção de controle de vendas, que mostrará em detalhes o

ingresso vendido, exibindo as características do mesmo, tais como: nome do filme, sessão, sala, poltrona e tipo de ingresso escolhido.

Figura 13. Controle de vendas

```
1729            case 2:
1730                opcao = 0;
1731                Il.atualizaIngressos(&iiimax, &iivip, &ii3d, &ii2d, &mmimax, &mmvip, &mm3d, &mm2d);
1732
1733                cf << "\t-INGRESSOS VENDIDOS-" << endl << endl
1734                   << "\tIMAX: " << iiimax << " inteira\t" << mmimax << " meia" << endl
1735                   << "\tVIP: " << iivip << " inteira\t" << mmvip << " meia" << endl
1736                   << "\t3D: " << ii3d << " inteira\t" << mm3d << " meia" << endl
1737                   << "\t2D: " << ii2d << " inteira\t" << mm2d << " meia" << endl << endl
1738                   << "\tVALOR ARRECADADO: " << "R$ " << Il.getArrecadado() << ",00" << endl;
1739
1740                cf.close();
1741
1742                break;
1743            default:
1744                break;
1745        }
1746    }
1747
1748 }
```

Fonte: autores (2019)

A escrita do código referente ao controle de vendas, consta na main. Ao escolher a referida opção, a função “atualizaIngresso” é chamada juntamente com o objeto de ingressos, possibilitando que haja a atualização da quantidade de ingressos vendidos de cada tipo de sala e qual o tipo de ingresso vendido(meia entrada ou inteira).

Por fim, o programa principal consiste, primeiramente, em três opções: vender ingresso, controle das vendas e sair do programa. Caso a opção escolhida seja venda de ingressos, os objetos das classes são instanciados, mostrando os filmes, sessões, e horários disponíveis, a partir dos métodos feitos para imprimir os dados de cada classe. Feito isso, o usuário deverá escolher o filme, tipo de sessão, tipo de sala (Imax,Vip,2D,3D), o tipo de ingresso (meia ou inteira) e a poltrona. Para a escolha da poltrona, foram feitos tratamentos de erro em todos os casos nos quais o usuário insere a poltrona desejada, pois dessa forma, caso seja inserida alguma poltrona inválida o programa assimila o erro, envia uma mensagem de erro, “Poltrona Inválida”, e repete a operação até que seja inserido um valor válido, garantido o funcionamento adequado do programa.

Finalmente, com todas as opções escolhidas que o usuário escolheu, por meio da manipulação de arquivo, o programa disponibiliza os ingressos em “ingresso.txt”, contendo todos os dados (filme, sessão, tipo de sala, tipo de ingresso, poltrona). Escolhendo a opção 2(controle de vendas), também por meio da manipulação de arquivos, a partir de “fechacaixa.txt”, é possível o acesso à quantidade de ingressos vendidos para cada filme, cada

sessão, cada tipo de sala e cada tipo de ingresso assim como valor total arrecadado com as vendas.

IV. Considerações Finais

Ao concluir o projeto proposto, teve-se como conclusão que os assuntos abordados em sala de aula, como também os temas abordados nos seminário, foram de suma importância para a conclusão do projeto.

Contribuindo para o conhecimento a programação, visto que foram utilizados ferramentas que ajudaram a aprimorar os conhecimentos em C++, favorecendo o aprendizado sobre boas técnicas de programação.

V. Referências Bibliográficas

FSTREAM – C++. Disponível em: http://www.cplusplus.com/reference/fstream/fstream/ . Acesso em 29 de Novembro de 2019.

C++. Disponível em: https://www.infoescola.com/informatica/cpp/ . Acesso em 28 de Novembro de 2019.

ALOCAÇÃO DINÂMICA EM C++. Disponível: https://www.guj.com.br/t/alocacao-dinamica-em-c/377301 . Acesso em 28 de Novembro de 2019.