Programação e Desenvolvimento de Software II – Turma TM2

UNIVERSIDADE FEDERAL DE MINAS GERAIS INSTITUTO DE CIÊNCIAS EXATAS

DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

**Nome: Klysman Rezende Alves | Matricula: 2017108779**

# Trabalho Prático II – Máquina de Vender Bilhetes

# C++ e Paradigma em orientação de objetos

C++ possibilita ao desenvolvedor separar a interface de uma classe de sua implementação. Esta padronização, além de facilitar a modificação dos programas, permite com que qualquer futuro programador que queira usar uma determinada classe apenas inclua em sua interface, não expondo o acesso direto ao código-fonte.

Provavelmente, o paradigma mais conhecido é o procedural (presente na linguagem de programação C e C++). Esse paradigma dentro de seus aspectos pode ser observado no código do programa que será desenvolvido para este trabalho.

Para obter esse entendimento, é necessário conhecer alguns dos pilares da Orientação a Objetos que são: Abstração, Encapsulamento, Herança e Polimorfismo, os quais foram utilizados no código.

Na prática de programação orientada a objetos estaremos atentos em nossos programas para pontos como:

* Procedural
* Funcional
* Lógico
* Orientado a objeto
* -Compatibilidade, portabilidade.
* -Segurança.
* Reusabilidade
* Facilidade de integração.
* Facilidade de extensão.
* Eficiência.

Definições importantes para o desenvolvimento de orientação a objetos.

* *Classes:*  modelos para a criação de objetos.
* *Atributos*:  dados dos objetos.
* *Métodos:*  rotinas que acessam os dados da classe.
* *Objetos:*  instâncias de uma classe.

## Descrição do trabalho e mETODOLOGIA

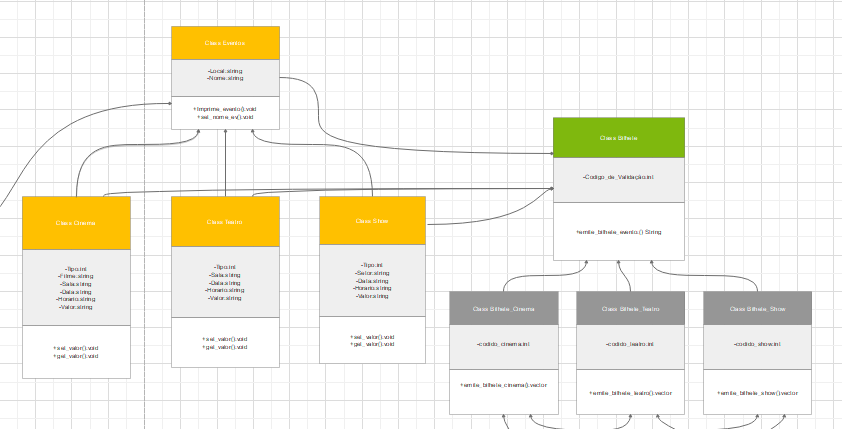
A proposta deste trabalho é a criação de maquina virtual para compra de ingressos de eventos.

Por se tratar de uma ferramenta de ampla utilização, essa máquina foi restrita a uma tecnologia para um shopping, de endereço fixo, o qual tem como atrações: cinema, shows de música, peças de teatro e concertos.

De maneira visual, o fluxograma abaixo exemplifica como foi estruturada o caminho do cliente, desde o cadastro de seus dados pessoais até a impressão do bilhete para o evento.

Cada evento tem sua propria classe, respeitando o paradigma de orientação a objetos exporto na introdução. Para tanto, cada uma das classes foi caracterizada com atributos como nome, data, horario, sala, tipo e valores proprios. Para complementar, cada uma dessas classes estão sujeitas a uma classe superior, chamada de classe Evento, e como subclasse desta a class bilhete para cada tipo de evento.

Essa estrutura de eventos pode ser melhor entendida no diagrama UML apresentado abaixo.



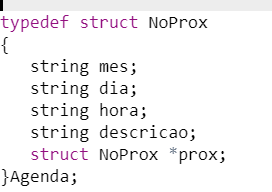
## ABORDAGEM E FUNÇÕES UTILIZADAS

Sendo o mais prático e objetivo possível dentro do programa, foi construído um menu de opção para o usuário que o usuário interaja com a agenda de forma simples.

Todas as funções, exceto a *void menu ()* foram instanciadas dentro da biblioteca *funções.h.* Lá, como será descrito logo abaixo, está escrita as funções responsáveis pelo funcionamento do programa.

Ainda no arquivo *main.cpp,* na função principal, foi instanciada a *Agenda* com a seguinte estrutura:  *Agenda \* Inicio = NULL, \* Fim = NULL, add;* Essa linha de comando diz respeito a um ponteiro para o inicio e fim de um lista encadeada, que no caso será a lista para a agenda do programa. Em seguida, para tornar o menu funcional, tratando-se de uma agenda, que pode ser inserido compromissos com frequência, utilizou-se uma função *do* que faz o menu retornar quando uma das escolhas é finalizada.

Cada uma das opções testa se o valor de entrada corresponde ao esperado para uma agenda de compromissos. Por exemplo, *while ( ( int(add.mes.length()) > 12) || add.mes.find(';')!= string::npos);*

**

*Definição da struct para as strings necessárias ao programa.*

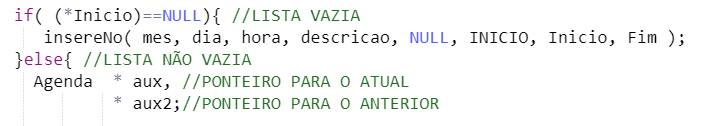
Na tabela abaixo, uma tabela simplificada de como o menu da agenda funciona e suas opções disponíveis.

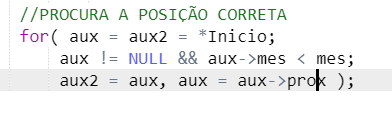
Para compor todas essas opções temos as seguintes funções implementadas:

* FUNÇÃO PARA INSERERI ORDENADO
  + void insereOrdenado( string mes, string dia, string hora, string descricao, Agenda\*\* Inicio, Agenda\*\* Fim )
* FUNÇÃO PARA ABRIR O ARQUIVO CSV COM OS DADOS DA AGENDA
  + void abrirArquivo (Agenda\*\* Inicio, Agenda\*\* Fim)
* FUNÇÃO LIBERA A MEMORIA ALOCADA UTILIZADA
  + void libera( Agenda \* Inicio )
* FUNÇÃO QUE MOSTRA OS COMPROMISSOS DA LISTA NO DIA
  + int buscar(Agenda\* Inicio, char \* escolha)
* FUNÇÃO QUE MOSTRA OS COMPROMISSOS DA LISTA
  + void mostra( Agenda \* Inicio, int opcao )

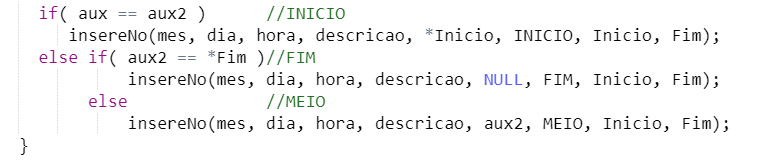
## DESCRIÇÃO DA ESTRUTURA PRINCIPAL

Temos como macro do programa a lista encadeada que faz a inserção das informações que o usuário fornece de maneira ordenada. Na função *void insereOrdenado( string mes, string dia, string hora, string descricao, Agenda\*\* Inicio, Agenda\*\* Fim )*  temos que ela recebe os dados de maneira que o mês seja a primeira entrada, e assim por em diante.





Dentro do *for* acima temos que a percorre a lista dos meses para então inserir corretamente, de maneira ordenada, a nova entrada de dado.



## aVALIAÇÕES E RESULTADOS

Para uma agenda simples, o proposto pelo trabalho foi aplicado. As implementações seguem como esperado de uma lista encadeada. A escrita do código foi pautada dentro das boas práticas de programação, estilo, identificadores, escopo, comentários, clareza e simplicidade lógica e modularidade, como sugerido, foram levados em consideração.

O programa funciona dentro do esperado, mas poderia ser aperfeiçoado com mais funções e restrições para entradas equivocadas por parte do usuário.