

CLASSES & NAMESPACES

DISCIPLINA: PROGRAMAÇÃO ORIENTADA A OBJETOS

Data de entrega: até 15 de dezembro de 2024.

Professor: Delano Beder

1 Enunciado

A atividade T1 consiste em implementar em C++ as classes conforme descritas abaixo. Os atributos das classes devem ser **privados**. Organizem suas classes no *namespace* **poo**.

1. Definam a classe **Pessoa** cujos objetos representam pessoas. Cada objeto dessa classe deve guardar os seguintes dados de uma pessoa: **nome** e **CPF**.

Escrevam as seguintes funcionalidades desta classe:

Nome	Descrição
Pessoa(string, string)	Construtor capaz de setar os atributos do objeto. Esse construtor deve ser único.
~Pessoa()	Destrutor da classe.
string getNome()	Método responsável por retornar o nome da pessoa.
string getCPF()	Método responsável por retornar o CPF da pessoa.
void imprime()	Método responsável por imprimir as informações de uma pessoa (nome, CPF).

2. Definam a classe **Estudante** (subclasse da classe **Pessoa**) cujos objetos representam estudante matriculados em uma disciplina. Cada objeto dessa classe deve guardar os seguintes dados de um estudante: **nome**, **CPF**, **RA**, **notas da 1ª e 2ª provas**, **notas do 1º e 2º trabalhos**.

Escrevam as seguintes funcionalidades desta classe:

Nome	Descrição
Estudante(string, string, int, double, double, double, double)	Construtor capaz de setar os atributos do objeto. Esse construtor deve ser único.
~Estudante()	Destrutor da classe.
int getRA()	Método responsável por retornar o RA do estudante.
double media()	Método responsável por calcular a média final (MF) do estudante. $MF = \frac{MP \times 8 + MT \times 2}{10}$, $MP = \frac{P1 + P2}{2}$, $MT = \frac{T1 + T2}{2}$
bool aprovado()	Método responsável por retornar verdadeiro se o estudante foi aprovado ($MF \geq 6.0$) e falso, caso contrário.
bool sac()	Método responsável por retornar verdadeiro se o estudante ficou em SAC – Sistema de Avaliação Complementar ($5.0 \leq MF < 6.0$) e falso, caso contrário.
double notaSAC()	Método responsável por calcular qual a nota mínima necessária, na prova de avaliação complementar (SAC), para aprovação na disciplina. Para o estudante ser aprovado após a prova de avaliação complementar (SAC) precisa atender a seguinte regra: $\frac{SAC + MF}{2} \geq 6.0$. Caso o estudante não ficou em SAC, retornar 0.
void imprime()	Método responsável por imprimir as informações de um estudante (nome, CPF, média final).

3. Definam a classe **Professor** (subclasse da classe **Pessoa**) cujos objetos representam professores universitários. Cada objeto dessa classe deve guardar os seguintes dados de um professor: **nome**, **CPF** e **nome da universidade** em que o professor trabalha.

Escrevam as seguintes funcionalidades desta classe:

Nome	Descrição
Professor(string, string, string)	Construtor capaz de setar os atributos do objeto. Esse construtor deve ser único.
~Professor()	Destrutor da classe.
string getUniversidade()	Método responsável por retornar o nome da universidade em que o professor trabalha.
void imprime()	Método responsável por imprimir as informações de um professor (nome, CPF, universidade).

4. Definam a classe **DataHorario** com os dados: **dia**, **mês**, **ano**, **hora**, **minuto** e **segundo**. A classe deverá dispor das seguintes funcionalidades:

Nome	Descrição
DataHorario(int, int, int, int, int, int)	Construtor capaz de setar os atributos. Este construtor verifica se a data/horário estão corretos, caso não esteja, a data/horário é configurada como 01/01/0001 00:00:00. Esse construtor deve ser único.
~DataHorario()	Destrutor da classe.
int getDia()	Método responsável por retornar o dia da data.
int getMes()	Método responsável por retornar o mês da data.
int getAno()	Método responsável por retornar o ano da data.
int getHora()	Método responsável por retornar a hora do horário.
int getMinuto()	Método responsável por retornar o minuto do horário.
int getSegundo()	Método responsável por retornar o segundo do horário.
void imprime(bool)	Método responsável pela impressão das informações de uma data/horário. Formato DD/MM/YYYY hh:mm:ss [AM/PM]. O parâmetro booleano indica se deve-se utilizar o sistema horário de 12 horas (AM/PM). Exemplo: 25/11/2024 15:57:10 ou 25/11/2024 3:57:10 PM.
void imprimePorExtenso()	Método responsável pela impressão das informações de uma data/horário por extenso. Exemplo: 25 de Novembro de 2024 – 15 horas, 57 minutos e 10 segundos.

5. Definam a classe **Sessao** cujos objetos representam sessões de um determinado teatro que acontecem em determinada data/horário. O teatro possui no máximo 210 poltronas (Figura 1), distribuídas em 15 fileiras de 14 poltronas, e a classe permite controlar a ocupação das poltronas.

Cada objeto dessa classe deve guardar os seguintes dados (dentre outros) da sessão de teatro: **nome da peça**, **data/horário**. A classe deve dispor das seguintes funcionalidades:

Nome	Descrição
<code>Sessao(string, DataHorario&)</code>	Construtor capaz de setar os atributos: nome da peça e data/horário (instância da classe DataHorario definida na questão anterior).
<code>~Sessao()</code>	Destrutor da classe.
<code>string proximoLivre()</code>	Método responsável por retornar o número da próxima poltrona livre (formato [A-O][1-14]). Retorna “cheio” se não houver poltrona disponível na sessão de teatro.
<code>bool verifica(string)</code>	Método responsável por verificar se a poltrona recebida (formato [A-O][1-14]) como parâmetro está ocupada.
<code>bool ocupa(string, Pessoa&)</code>	Método responsável por ocupar determinada poltrona da sessão de teatro, cujo número da poltrona é recebido (formato [A-O][1-14]) como parâmetro, e retornar verdadeiro se a poltrona não estiver ocupada (operação foi bem sucedida) e falso caso contrário. O segundo parâmetro representa um expectador (instância da classe Pessoa definida anteriormente) que “comprou” um ingresso da sessão de teatro.
<code>bool desocupa(string)</code>	Método responsável por desocupar determinada poltrona da sessão de teatro, cujo número da poltrona é recebido como parâmetro, e retornar verdadeiro se a poltrona estiver ocupada (operação foi bem sucedida) e falso caso contrário.
<code>int vagas()</code>	Método responsável por retornar o número de poltronas vagas disponíveis (não ocupadas) na sessão de teatro.
<code>void imprime()</code>	Método responsável pela impressão das informações de uma sessão de teatro (nome da peça, data/horário e lista dos expectadores com suas respectivas poltronas).

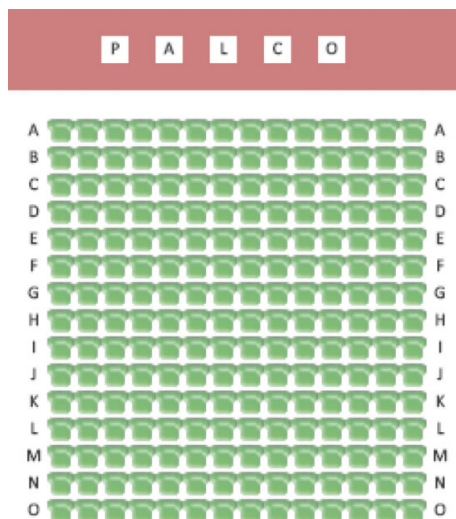


Figura 1: Teatro: Poltronas

Há 2 possibilidades na implementação do conjunto das 210 poltronas de uma sessão.

1. Utilize uma matriz `Pessoa*[15][14]`.

Considere uma poltrona (string no formato [A-O][1-14]), o seguinte código calcula a linha e coluna dessa poltrona na matriz.

```
int linha = s[0] - 'A'; // s[0] é a 1a letra da string [A-O]
// A função stoi converte de string para int.
// [stoi] string => int (substring formada apenas pela 2a letra)
int coluna = stoi(s.substr(1)) - 1;
```

Por exemplo, a poltrona A1 (linha = 0, coluna = 0) seria calculada assim:

```
linha = s[0] - 'A' => 'A' - 'A' = 0
coluna = stoi(s.substr(1)) - 1 => stoi("1") - 1 => 1 - 1 = 0
```

Enquanto a poltrona E8 (linha = 4, coluna = 7) seria calculada assim:

```
linha = s[0] - 'A' => 'E' - 'A' = 4
coluna = stoi(s.substr(1)) - 1 => stoi("8") - 1 => 8 - 1 = 7
```

2. Utilize um array `Pessoa*[210]`.

Considere uma poltrona (string no formato [A-O][1-14]), o seguinte código calcula a posição dessa poltrona no array.

```
int linha = s[0] - 'A';
int coluna = stoi(s.substr(1)) - 1;
int posicao = linha * 14 + coluna;
```

2 Observações importantes

2.1 Sobre a elaboração e entrega:

- Este exercício-programa deve ser elaborado individualmente.
- Vocês devem utilizar **apenas** os conceitos apresentados em aula.
 - Vocês devem implementar as classes em C++.
 - Os atributos das classes devem ser **privados**.
 - Organizem suas classes no *namespace* **poo**.
- Compactem o código-fonte das classes em C++ e entreguem somente este arquivo (<NroRA>.zip) no ambiente moodle.
Exemplo: 567890.zip (cuidado para não enviar arquivos errados!)
- O prazo de entrega é o dia 15 de dezembro de 2024 às 23h59.
- A entrega será feita unicamente pelo ambiente moodle (<https://ava2.ead.ufscar.br>). Não serão aceitos trabalhos enviados por email.
- Guardem uma cópia dos arquivos entregues.

2.2 Sobre a avaliação:

- Não serão toleradas cópias! Exercícios copiados (com ou sem eventuais disfarces) receberão nota ZERO. O exercício do estudante/grupo alvo da cópia também receberá nota ZERO.
- Exercícios com erros de sintaxe (ou seja, erros de compilação) receberão nota ZERO.
- Os exercícios serão avaliados segundo os seguintes critérios:
 - Soma simples dos valores obtidos nos itens de 1 a 2
 1. Atendimento às normas de boas práticas de programação (comentários, endentação, nomes de variáveis, estruturação do código, modularização, etc) [0..20]
 2. Corretude na implementação da atividade [0..80]