Trabalho Prático

BCC 221 - Programação Orientada a Objetos (POO)

Sistema de Controle de Voo em Aeroporto

Instruções gerais

- O problema deve ser resolvido por meio de um programa em C++ e a STL;
- O código-fonte deve estar devidamente comentado;
- Não serão aceitos trabalhos que caracterizem cópia (mesma estrutura e algumas pequenas modificações) de outro ou de códigos da internet;
- Eventualmente, após a entrega dos trabalhos serão marcadas entrevistas com cada um dos aluno para apresentação dos mesmos para o professor.

Entrega

- A entrega do código-fonte será feito pelo Moodle
- Deve ser entregue um arquivo *.zip* com:
    - Relatório
    - Código fonte com comentários explicativos
    - Um arquivo `README.md` com instruções de compilação e uso
    - Vídeo mostrando a compilação, execução e uso do código. Além de uma explicação do projeto e dos códigos desenvolvidos.

Avaliação

- Funcionamento adequado do programa:
    - Códigos que não compilarem e tiveram *warnings* diminuirão a nota;
    - Corretude (independente se gerado por IDE ou manualmente).
- Atendimento ao enunciado do trabalho;
- Comentários elucidativos e bem elaborados;
- Identação do código e boas práticas de programação;
- Boa organização do código fonte em geral (modularidade, organização dos arquivos .h e .cpp do projeto, nomes de funções e variáveis, etc.);
- Bom uso da biblioteca STL.

Enunciado do trabalho

Objetivo

Desenvolver um sistema em C++ que simule o controle básico de voos em um aeroporto. O sistema deverá gerenciar informações sobre voos, aeronaves, passageiros e pilotos, e permitir o registro de embarques, com **interface por linha de comando** e **persistência de dados em arquivos**.

Este trabalho tem como objetivo aplicar de forma prática os seguintes conceitos:

- Classes e objetos
- Encapsulamento
- Associação entre classes
- Herança e polimorfismo
- Manipulação de arquivos
- Utilização da biblioteca STL

Requisitos Funcionais

O sistema deverá permitir:

1. **Cadastro de aeronaves**
    - Código, modelo, capacidade, velocidade média (em milhas por hora), autonomia de voo (distância em milhas).
2. **Cadastro de pessoas**, sendo:
    - **Pilotos**: nome, matrícula, brevê, horas de voo
    - **Passageiros**: nome, CPF, número do bilhete
3. **Criação de voos**, com:
    - Código, origem, destino, distância da viagem
    - Aeronave associada
    - Hora de saída prevista
    - Número de escalas estimado
        i. com base na distância do voo e na autonomia da Aeronave
    - Tempo estimado de voo
        i. com base na distância e na velocidade da Aeronave
        ii. cada escala leva 1 hora
    - Comandante e Primeiro Oficial (ambos pilotos)

    - Lista de passageiros (respeitando a capacidade)

4. **Associação de passageiros a voos**

5. **Listagem de voos**: exibir o código do voo, o código e o modelo da aeronave, matrícula do piloto comandante, origem e destino, número de passageiros, hora de saída e de chegada previstas.
6. **Listagem de passageiros em voo**: Ao informar o código do voo, exibir o código e o modelo da aeronave, o nome de cada passageiro no voo.

7. **Salvar e carregar dados automaticamente** dos seguintes arquivos (formato CSV):
    - `aeronaves.csv`
    - `pessoas.csv`
    - `voos.csv`

Menu Interativo (Terminal)

Implemente um **menu textual simples** com opções numeradas, como o do exemplo abaixo:

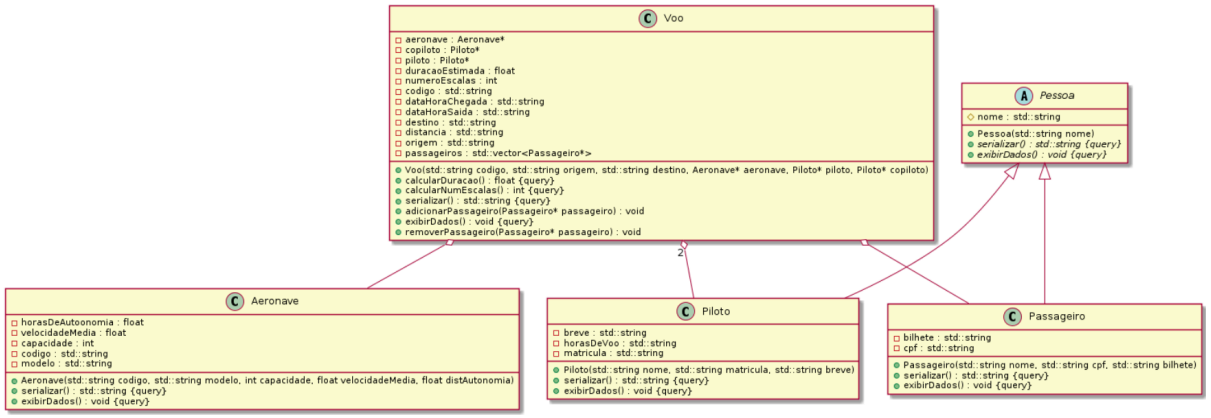
```
======= SISTEMA DE CONTROLE DE VOOS =======

1. Cadastrar aeronave
2. Cadastrar piloto
3. Cadastrar passageiro
4. Criar voo
5. Embarcar passageiro em voo
6. Listar voos
7. Listar passageiros de um voo
8. Gerar relatórios e estatísticas
9. Salvar dados e sair

============================================
Escolha uma opção: _
```

Exemplo de Estruturas de Classes Iniciais

O diagrama de classes abaixo apresenta um exemplo de estrutura inicial que pode ser usado para definir o projeto do programa a ser implementado. Utilize como base para desenvolver a sua própria solução.



Lembre-se de definir os métodos *getters* e *setters* e outros métodos necessários para corresponder aos requisitos do seu programa. Implemente novas classes conforme precise, mas tente não deixar o projeto muito complexo ou extenso.

Tarefa Extra (Ponto Adicional)

Relatórios e Estatísticas do Sistema

Implemente uma funcionalidade que permita ao usuário gerar relatórios estatísticos, com informações como:

- Número total de voos cadastrados
- Média de passageiros por voo
- Lista de aeronaves mais utilizadas
- Passageiros que participaram de mais de um voo
- Voos que atingiram pelo menos 90% da capacidade máxima
- Distância total percorrida por cada aeronave

Essa funcionalidade deve ser acionada como uma nova opção no menu. Defina um sbomenu para os diefrentes tipos de estatísticas e as imprima no terminal e permita salvar em arquivos .txt.

📄 Orientações para o Relatório Técnico

Cada grupo/aluno deverá entregar um relatório em PDF que documente o desenvolvimento do sistema proposto. Este relatório será avaliado junto com o código-fonte e contribuirá para a nota final.

Estrutura Sugerida do Relatório

---

1. Capa

- Nome da instituição
- Nome da disciplina
- Nome(s) do(s) aluno(s) (em ordem alfabética)
- Título: *Sistema de Controle de Voo em Aeroporto*
- Professor
- Data

---

2. Introdução

- Apresentação geral do problema abordado
- Objetivos do trabalho
- Escopo do sistema desenvolvido

---

3. Descrição do Sistema

- Explicação das principais funcionalidades
- Quais entidades o sistema manipula (ex: passageiros, voos, aeronaves)
- Como o usuário interage com o sistema via terminal

---

4. Decisões de Projeto

- Explicação da modelagem orientada a objetos escolhida
- Justificativa para o uso de herança, polimorfismo, encapsulamento
- Quais estruturas da STL foram utilizadas e por quê (`vector`, `map`, etc.)
- Como foi feita a persistência de dados (formato dos arquivos, lógica de leitura/escrita)

---

5. Arquitetura do Projeto

- Descrição dos arquivos `.h` e `.cpp` e suas responsabilidades
- Explicação do funcionamento geral do `main.cpp`
- Como as classes se relacionam entre si
- Se desejar, inclua o caminho de execução do programa (diagrama de fluxo)

---

6. Diagramas UML

- Inclua pelo menos um diagrama de classes UML com as seguintes informações:
    - Nome das classes
    - Atributos e métodos principais
    - Relacionamentos (associação, herança)
    - Pode ser feito com ferramentas como: Lucidchart, draw.io, StarUML, ou outros

---

7. Testes Realizados

- Descrever os testes manuais executados (cenários criados)
- Quantas aeronaves, voos, passageiros, pilotos foram testados
- Relatar se houve tratamento de erros (ex: voo cheio, código duplicado)

---

8. Conclusão

- O que foi aprendido durante a implementação
- Dificuldades encontradas
- Melhorias que poderiam ser feitas

---

9. Anexos (opcional)

- Trechos relevantes do código (opcional se já estiver bem comentado)
- Prints do terminal funcionando
- Estrutura de diretórios do projeto