Aeroporto

Projecto 1 Gestão do embarque dos passageiros num aeroporto (check-in / embarque)

Carlos Costa - 090509097
Daniela Cardeano - 090509085
Vítor Moreira- 060509009

Índice

Introdução

No âmbito da disciplina de Algoritmos e Estruturas de Dados desenvolvemos uma aplicação em C++ para gestão do embarque de passageiros num aeroporto.

Um aeroporto é uma área que possui uma infraestrutura concebida para aterragens e descolagens de aviões destinados a transporte de passageiros. As companhias aéreas que possuam base num determinado aeroporto são responsáveis pela gestão dos aviões, direcionando para cada viagem um da sua frota, composta por veículos com diferentes características. Para um voo será destacada uma tripulação, o voo será ainda composto pelos passageiros que terão de fazer o check-in antes de se dirigirem à porta de embarque destinada, na hora especificada para o efeito. Os dados de cada passageiro serão guardados assim, como, da tripulação por questões de segurança. Este código desenvolvido permite a gestão de todas estas tarefas necessárias para o bom funcionamento de um aeroporto, permitindo a interacção dos elementos externos com o sistema, estabelecendo as ligações entre os vários intervenientes do sistema.

Assim, no momento que uma pessoa proceder ao check-in estará disponível a hora de partida, a companhia aérea com que irá viajar, o número de voo, o tipo de avião que irá realizar o voo, a hora de chegada, a porta de embarque, o seu lugar no avião.

Qualquer aeroporto seria um possível usuário deste trabalho.

Concepção e implementação de uma solução

O projecto que nos era pedido consistia na gestão do embarque de passageiros no aeroporto, além do que nos era pedido inicialmente resolvemos tornar o trabalho mais substancial, acrescentando mais classes de forma a que esta implementação se aproxima-se o mais possível da situação real de um aeroporto. Um dos exemplos, é o caso da tripulação composta por pilotos e hospedeiros, que possuem um classe que guarda as suas informações e calcula o salário conforme a sua categoria.

Para reutilizar o código e promover uma maior coerência lógica no modelo de implementação recorremos à herança, uma parte importante da programação orientada a objectos (POO). Assim classes como, por exemplo, piloto e hospedeira que têm características mútuas mas não são iguais, usufruíram desta possibilidade, derivando da classe base tripulação herdaram algumas características e depois sofreram especificações.

A captura e localização de erros inadequados é um do problemas em software. Quando um erro é detectado numa aplicação, deve-se poder fixar a causa do problema e voltar a tentar processar a operação que produziu o erro, ou seja, um procedimento encontra uma condição de erro e volta ao procedimento originário, executa as operações de limpeza necessárias para puder continuar. Por isso, por todo o programa recorremos à implementação de excepções que sinalizam quando um elemento que se está a aceder está fora do domínio do contentor, um vector se encontra vazio, o utilizador introduziu algum dado no formato errado, um dado ficheiro de dados está corrupto ou num formato desconhecido.

Foi usado o polimorfismo na função de impressão de informações acerca de tripulantes, dado que num vector de tripulantes era chamada uma função imprimeInfo(), diferente consoante o objecto fosse um Piloto ou Hospedeiro.

Foi feita a estrutura para uma base de dados de TipoAvião que o utilizador final poderá utilizar para poupar tempo ao introduzir novos aviões e memória, porque evita muita repetição de dados.

Estruturas de classes

Diagrama UML em anexo.

Implementação das classes

Para a resolução deste projecto recorremos à implementação de classes, que são as seguintes: Aeroporto, Avião, Companhia, TipoAvião, Atraso, Comandos, tipoAviãoDB, DataHora, PortaEmbarque, Pessoa, Passageiro, Tripulante, Voo, Menu, Piloto, Hospedeiro. Não estando a contar com a classe das Excepções.

Segue em seguida, uma breve descrição de cada uma das classes:

Aeroporto:

A informação acerca de um aeroporto (nome, localização, país) está armazenada nesta classe. Esta classe é a principal do programa é nela que estão guardas as várias

companhias, portas de embarque e passageiros do aeroporto, ou seja, a gestão dos recursos internos do aeroporto será aqui por esta efectuada.

Avião:

Dentro de uma dada companhia aérea existem vários aviões, o governo destes veículos é feito por esta classe, que permite guardar a informação acerca de cada avião, tal como, os dados que o identificam (nome ,matrícula, peso de bagagem), a data da última revisão a que foi submetido e a da próxima revisão, permite ainda comparar a matrícula de dois aviões para saber se são equivalentes. Esta classe está ligada a outras classes através de apontadores, pois cada avião pode ser de um tipo diferente, terá um voo atribuído e pertencerá a uma dada companhia.

Companhia:

A informação de uma determinada companhia de um aeroporto é armazenada nesta classe. Cada companhia distingue-se pelo seu nome, sigla, frota de aviões, tripulação. A gestão dos voos cabe a cada companhia fazer, atribuindo a cada voo o respectivo avião e tripulação. São ainda identificados os casos em que existam companhias com o mesmo nome, mesma sigla ou ambos.

TipoAvião:

Cada avião tem as suas características, a informação de todas estas variáveis que atribuem a um avião um determinado tipo é feita por esta classe. Aqui pudemos identificar que aviões são de um certo tipo e comparar dois aviões para saber se são do mesmo tipo.

Atraso:

Possui a informação de um atrasado de um voo na sua hora de chegada ou partida, possui funções para leitura e escrita das horas e minutos do atraso.

Comandos e TipoAviãoDB:

Bases de dados.

DataHora:

Classe onde está armazenada a informação da data e hora, contêm funções para leitura e escrita destas.

PortaEmbarque:

Para cada voo é destacada um porta de embarque, que terá como informações importantes o seu número, localização, para que tipo de aviões está direccionada. Através de apontadores pudemos relacionar à uma dada porta de embarque o voo respectivo.

Pessoa:

Esta é uma superclasse que guarda a informação acerca de uma pessoa, pessoa esta que pode pertencer aos tripulantes ou passageiros. È feita a comparação dos nomes de diferentes pessoas para saber se são iguais caso sejam é feita de seguida a comparação do número do BI.

Passageiro:

Esta classe contém a informação associada a um passageiro de um determinado voo, as suas necessidades especiais, o peso da sua bagagem, o seu lugar no avião. É lhe atribuído o voo respectivo.

Tripulante:

Dentro de uma mesma tripulação existem pessoas com diferentes cargos, esta subclasse permite guardar a informação de cada um dos tripulantes identificando o seu nome, cargo. Esta classe derivada está em interacção com as classes voo e companhia, pois um tripulante pertence a uma determinada companhia e é lhe atribuído um certo voo, caso esteja destacado para tal. São também comparados os tripulantes para saber se têm o mesmo nome, caso o sejam compara o seu número.

Voo:

Esta classe guarda as especificações de cada voo e atribui a este a respectiva porta de embarque e o avião para o efeito. São, também, guardadas as informações acerca dos seus tripulantes e passageiros.

Menu:

Funciona como a interface entre o utilizador e os dados internos. Permitindo acrescentar, consultar, modificar e eliminar os dados.

Piloto e Hospedeiro:

São classes derivadas da classe tripulante, que apesar de herdarem informação da sua classe base têm certas particularidades, que serão acrescentadas na respectiva classe. Uma das diferenças é o salário atribuído a um tripulante, que difere consoante a categoria em que este se enquadra (piloto, hospedeiro), sendo então desenvolvida uma função na classe base que é herdada e modificada pelas suas derivadas.

Conclusão

O projecto desenvolvido implementou todas as especificações pedidas e acrescentou algumas que achamos adequadas para facilitar e melhorar a utilização do programa por parte do utilizador final.

A meio do projecto fizemos uma remodelação da estrutura do projecto de vectores de objectos para vectores de apontadores para facilitar a associação de objectos através de pointers, e optimizar o uso da memória.

Isso atrasou consideravelmente o desenvolvimento do projecto porque tivemos que alterar a estrutura de grande parte da implementação que tínhamos no inicio e levou a que o programa não tenha o nível de documentação e teste que desejaríamos.

A carga de trabalho do projecto foi distribuída de forma equivalente entre os elementos do grupo.