FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Aeroporto

Projecto 2 - Gestão do embarque dos passageiros num aeroporto (check-in / embarque)

21/12/2010

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 infra-estrutura 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, associando a cada voo um avião da sua frota, composta por veículos com diferentes características.

Para um voo será destacada uma tripulação, constituída por piloto/co-piloto, e hospedeiras.

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.

Após o ckeck-in os passageiros passam para a porta de embarque respectiva e são chamados por ordem de prioridade.

Após todos os passageiros serem chamados para embarcar, o voo é fechado e adicionado a uma tabela de dispersão, que será posteriormente usada para a pesquisa de informações relativas a taxas de utilização dos balcões/portas de embarque, bem como para pesquisar as informações de um determinado voo ou fazer análises dos voos dessa tabela.

A atribuição das portas de embarque é feita automaticamente pelo programa, usando para tal uma bst, de forma a encontrar a porta mais adequada para cada voo. Ou seja, aquando da criação dos voos, de acordo com o número de passageiros que esse voo possui, é atribuído uma porta de embarque que possua bagas suficientes e que seja do mesmo tipo.

Posteriormente, caso seja alterado o numero de passageiros desse voo através dos menus de edição, é verificado se é necessário atribuir outra porta de embarque, (ou seja, caso a porta atribuída não tiver vagas suficientes para albergar os novos passageiros, é atribuída outra que tenha, libertando vagas na antiga).

O programa desenvolvido permite a gestão das tarefas de gestão necessárias para o bom funcionamento de um aeroporto, permitindo a interacção intuitiva do usuário com o sistema e estabelecendo as ligações entre as várias estruturas de dados, de forma a possibilitar uma gestão eficiente dos recursos do sistema (quer em termos de memória, evitando redundância de dados, quer em termos de desempenha, usando algoritmos e estruturas de dados desenvolvidos para o efeito.

Concepção e implementação

Usando o código do projecto anterior, foram feitas algumas alterações, mantendo, no entanto, a anterior api e interface com o utilizador da estrutura inicial, acrescentando as funcionalidades pedidas e optimizando algumas das já implementadas.

Sendo assim, usamos as api que já tínhamos desenvolvido e estendemos o namespace CRUD, com as rotinas necessárias para as novas funcionalidades. Como o namespace CRUD, engloba todas as funções necessárias à gestão do aeroporto, possuindo para tal as funções referentes à criação, leitura, edição, eliminação e gestão da memória de todos os recursos do programa, não foi necessário criar mais classes mas antes estender a funcionalidade deste namespace e das classes a ele associadas.

- ✓ **BST**

As portas de embarque passaram a ser implementadas por uma BST que ordena-as segundo a disponibilidade (nº de vagas na porta), e categoria dos aviões (médio ou longo alcance).

(Não foi usado o critério do horário porque a atribuição da porta de embarque é feita aquando da criação do voo. Sendo assim, tendo em conta que podem estar vários voos associados à mesma porta de embarque e a operarem sobrepostos, o critério da ordenação usando o horário não faz sentido na nossa implementação (nós não atribuímos portas de embarque num período de tempo - ex: dia - a porta de embarque é logo atribuída quando é necessário, para evitar ter voos, com apontador nulo para a porta de embarque a ele associado).

Para evitar erros e simplificar o código, as categorias dos aviões passaram a ser um inteiro de uma lista pré-definida de tipos de avião, em vez de uma string (o que corresponde a uma enumeração de prioridades).

Portanto, ao criar um novo voo, é atribuída uma nova porta de embarque a esse voo, sendo a BST actualizada (ou seja, a porta de embarque que foi atribuída ao voo vai ser retirada da BST, actualizada - diminuição do número de vagas de acordo com o numero de passageiros do voo à qual foi atribuída,

update de dos apontadores... - e de seguida introduzida novamente de forma a ficar na posição correcta da BST).

A BST possui menus para a introdução e edição das portas de embarque, bem como menus que permitem a listagem por critérios das portas de embarque lá armazenadas. Por outro lado, ainda possui rotinas que mostram os passageiros que estão numa dada porta de embarque por voo, seleccionado de acordo com critérios definidos pelo utilizador para facilitar a pesquisa do voo do qual pretende ver os passageiros a embarcar.

- ✓ **Queues (fila e fila de prioridade)**

Tal como pedido, foi criado um novo menu para simular o check-in, que é implementado por uma fila (queue), dentro da classe voo.

A simulação da chegada dos passageiros à fila de check-in é feita baralhando o vector de passageiros que está associado aquele voo e de seguida introduzindo os elementos na queue. Desta forma, simula-se a aleatoriedade da chegada dos passageiros.

Após criada a fila de passageiros, procede-se ao início do check-in. Ou seja, quem está a usar o programa vai atendendo cada um dos passageiros, registando se tem necessidades especiais, bem como o peso da bagagem e número do lugar do passageiro no avião.

Isto é efectuado para todos os passageiros, sendo que após o passageiro sair da fila de check-in, é movido para uma fila de prioridade, estando esta fila ordenada pelas necessidades especiais de cada passageiro.

Terminado o check-in os passageiros deslocam-se dos balcões para as portas de embarque, sendo chamados de acordo com a ordem na fila de prioridade.

Caso um passageiro não esteja presente aquando da sua chamada, a sua informação é guardada num vector temporário e restaurada no fim de todos os outros passageiros serem chamados. Caso todos os passageiros estiverem presentes o voo é fechado automaticamente.

Senão é perguntado ao utilizador do programa se quer fechar deliberadamente o voo, apesar de ainda haver passageiros que não embarcaram.

O processo de embarque pode ser interrompido e retornado a qualquer momento, visto que o programa possibilita o embarque individual de passageiros (e sequencial, caso o utilizador deseje).

Os voos também podem ser fechados manualmente caso o operador do programa deseje.

Fechar um voo corresponde a incluir esse voo na tabela de dispersão, e a libertar as vagas da porta de embarque que a ele estava associada, bem como terminar o check-in e o embarque de passageiros.

- ✓ **Tabela de dispersão (hash_set)**

Na tabela de dispersão estão todos os voos que foram fechados para no aeroporto.

De forma a flexibilizar esta tabela, foram criadas rotinas que permitem ao utilizador pesquisar as informações de um dado voo usando o id do voo e desta forma maximizando o potencial das hash_set (visto que foram implementadas de forma a permitir pesquisa rápida dada a key).

Por outro lado é possibilitada vários tipos de listagens por critérios, sendo alguns deles, listar as taxas de utilização por voo.

Relativamente às taxas, é possível calcular as taxas quer para os balcões de check-in quer para as portas de embarque usando 3 tipos de pesquisas.

A primeira é calcular a taxa para balcões / portas de embarque para os voos que estão entre duas datas limites dadas pelo utilizador.

A segunda é possibilitar o cálculo da taxa para um voo especifico.

E a terceira é possibilitar a listagem das taxas por voo entre dois limites (datas) dadas pelo utilizador.

Estruturas das classes

Diagrama UML em anexo.

Implementação das classes

Para estruturar a implementação e facilitar a adição de funcionalidades, bem como para criar estruturas de dados mais eficientes e mais flexíveis, o projecto assenta sobre o seguinte conjunto de classes: Aeroporto, Atraso, Avião, BST, Comandos, Companhia, DataHora, Exceptions, gestaoAeroporto, Hospedeiro, Menu, Passageiro, Pesssoa, Piloto, PortaEmbarque, TipoAviao, TipoAviaoDB, Tripulante, Voo (e ainda uma libraria de funções úteis para a interface com o utilizador "utils").

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

Aeroporto:

A informação acerca de um aeroporto (nome, localização, país, taxa de utilização das portas de embarque) está armazenada nesta classe. Esta classe é a principal do programa e é nela que estão guardas as companhias aéreas a operar no aeroporto, bem como as portas de embarque e passageiros do aeroporto. Ou seja, esta é a classe que engloba todas as estruturas de dados associados à gestão de um aeroporto.

Algumas dessas estruturas de dados são o vector de companhias e passageiros, do aeroporto e a bst de portas de embarque. Por fim também está englobado o hash_set de voos fechados.

Atraso:

Estrutura de dados que armazena um dado atraso de um voo.

Aviao:

Dentro de uma dada companhia aérea existem vários aviões, a gestão 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 máximo da bagagem permitido, o tipo de avião que a ele está associado, a companhia a que pertence, o voo ao qual está associado, e datas de construção, primeiro voo, última revisão e próxima revisão).

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.

Assim os apontadores facilitam a gestão das associações e evitam duplicação de dados.

BST:

Implementação de uma árvore binária de pesquisa, optimizada para a pesquisa de elementos.

Comandos:

Estrutura de dados que armazena os comandos / atributos que todas as outras classes possuem.

Usada para facilitar a listagem dos parâmetros que se pode editar em cada classe.

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.

É uma interface que permite a um dado balcão ter num submenu toda a informação que precisa para o seu correcto funcionamento.

DataHora:

Classe onde está armazenada a informação da data e hora, associada a um voo.

Exceptions:

Classe que colecta todas as excepções que o programa poderá lançar.

Com a documentação associada a cada tipo de erro que poderá estar associada cada uma dessas excepções.

GestaoAeroporto:

Classe que contém as funções relacionadas com a criação, leitura, edição e eliminação de qualquer uma das outras classes.

Todas as funções estão dentro de um namespace para evitar conflitos de nomes e para englobar as funções dentro de um domínio que tem um papel específico na aplicação.

Todas as funções recebem os dados que precisão por parâmetro, por isso são facilmente portadas para outras aplicações, bastando para tal fornecer os adaptadores de estruturas de dados respectivos.

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.

É aqui que é usado o polimorfismo, porque usando um vector de apontadores de tripulantes, caso o apontador seja de piloto ou hospedeiro irá chamar a função imprimeInfo() respectiva a cada uma das subclasses.

Menu:

Funciona como a interface entre o utilizador e as estruturas de dados do programa. Permitindo acrescentar, consultar, modificar e eliminar os dados.

Passageiro:

Subclasse de Pessoa, esta classe contém a informação associada a um passageiro de um determinado voo, as suas necessidades especiais, o peso da sua bagagem bem como o seu lugar no avião.

Cada passageiro está associado apenas a um voo.

Pessoa:

Esta é uma superclasse que guarda a informação acerca de uma pessoa, pessoa esta que pode pertencer aos tripulantes ou passageiros. É 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.

PortaEmbarque:

Cada voo está associado a uma porta de embarque, que terá como características o seu número, localização, para que tipo de aviões pode receber voos e a sua capacidade máximo de passageiros que suporta, que será usado, para a ordenação da BST.

Cada porta de embarque tem um vector de voos associado, o que significa que pode albergar vários voos, (desde que tenha vagas para eles).

TipoAviao:

Classe usada para evitar a repetição de dados dentro da classe avião.

Por outro lado simplifica a adição de novos voos por parte dos utilizadores do programa porque depois de ter uma base de dados formada, a não é necessário repetir imensos parâmetros repetidos.

TipoAviaoDB:

Classe que faz a gestão da base de dados dos tipos de aviões.

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, categoria, salário por hora e números de horas de trabalho.

Esta classe derivada da classe Pessoa 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.

Voo:

Esta classe guarda as especificações de cada voo e atribui a este a respectiva porta de embarque e o avião para o efeito, bem como a companhia à qual pertence.

São, também, guardadas as informações acerca dos seus tripulantes, passageiros, bem como as filas de check-in e de embarque.

Conclusão

O projecto desenvolvido implementou todas as especificações pedidas adaptando-as ao código existente.

Tentamos implementar o código usando um estilo de programação legível e claro, e focando a implementação na sua eficiência do uso dos recursos de memória e processamento, tentando reduzir a duplicação de dados.

Fomos encontrando algumas dificuldades no decorrer do projecto nomeadamente na migração da tabela de dispersão para ambos o Visual Studio e o GCC, tendo por fim usado directivas do pré-processador de forma a detectar qual o compilador. Mesmo assim, o método find não se comporta da mesma maneira em ambos, tendo nós optado por usar a versão do GCC. Apesar do código ser igual e compilar em ambos, esse método retorna valores diferentes.

Outra dificuldade encontrada foi a do uso da BST, devido à interpretação inicial do projecto e ao facto de os enunciados darem margem a interpretações várias dependendo da ideia que cada um tem sobre o funcionamento dum aeroporto.

Não metemos uma descrição detalhada dos métodos implementados no relatório devido ao facto de termos mantido a documentação actualizada, sendo de mais fácil visualização.

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

Anexos

```
***** Gestão do embarque dos passageiros num aeroporto (check-in/embarque) *****

0 - Sair
1 - Gestao de companhias aéreas
2 - Gestão de portas de embarque
3 - Gestão de passageiros
4 - Gestao dos voos fechados do aeroporto
5 - Visualização das taxas dos balcões e portas de embarque
6 - Informações acerca do aeroporto
7 - Editar Informações acerca do aeroporto

Opção: 1
```

- Menu principal

```
***** Gestao de companhias aéreas *****

0 - Retroceder
1 - Listar companhias aéreas
2 - Adicionar companhia aérea
3 - Editar companhia aérea
4 - Remover companhia aérea

Opção:
```

- Menu das Companhias

-

```
***** Gestao de Portas de Embarque *****

0 - Retroceder
1 - Listar informações das portas de embarque
2 - Lista dos passageiros a embarcar num voo
3 - Chamada dos passageiros a embarcar num voo
4 - Adicionar porta de embarque
5 - Editar porta de embarque
6 - Remover porta de embarque

Opção:
```

Menu das Portas de Embarque

```
***** Gestão de passageiros *****

0 - Retroceder
1 - Listar passageiros
2 - Adicionar passageiro
3 - Editar pssageiro
4 - Remover passageiro

Opção:
```

- Menu dos Passageiros

```
***** Gestão dos voos fechados do aeroporto *****

0 - Sair
1 - Listagem de todos os voos fechados
2 - Listagem de todos os voos fechados entre datas por critério
3 - Pesquisar voo fechado
4 - Inserir voo aos voos fechados
5 - Eliminar voo da tabela de voos fechado

Opção:
```

- Menu dos Voos fechados

```
***** Gestão da companhia aérea *****

0 - Retroceder
1 - Gestao de aviões
2 - Gestao de tipos de aviões
3 - Gestao de tripulantes
4 - Gestão de planos de Voo
5 - Editar esta companhia Aérea
6 - Informações sobre a companhia

Opção:
```

- Menu de Gestão de Companhia

```
***** Gestao de Planos de Voo *****

0 - Retroceder
1 - Selecionar plano de voo
2 - Adicionar plano de Voo
3 - Adicionar tripulantes ao voo
4 - Remover tripulantes ao voo
5 - Editar plano de voo
6 - Remover plano de voo

Opção:
```

- Menu de Planos de Voo

```
Método de ordenação:
+---+---------------------------+
| # |     Critério de ordenação |
+---+---------------------------+
| 0 |                Número voo |
| 1 |             Origem do voo |
| 2 |            Destino do voo |
| 3 |    Data/hora saída do voo |
| 4 | Data/hora cheagada do voo |
| 5 |             Atraso do voo |
| 6 |                 Cancelado |
+---+---------------------------+

Selecione o critério de ordenação que quer usar: 0


Lista de planos de voo:
+---+------------+---------------+----------------+
| # | Número voo | Origem do voo | Destino do voo |
+---+------------+---------------+----------------+
| 1 |          1 |          ddgf |          gfgfg |
+---+------------+---------------+----------------+

Introduza o índice (0 para retroceder):1
```

- Exemplo de listagem ordenada

```
***** Gestão do voo *****

0 - Retroceder
1 - Inicialização do check-in do voo
2 - Check-in dos passageiros
3 - Lista dos passageiros a embarcar num voo
4 - Chamada dos passageiros a embarcar num voo
5 - Informações sobre o voo
6 - Fechar voo

Opção:
```

- Menu de Voo

```
***** Gestão dos voos fechados do aeroporto *****

0 - Sair
1 - Listagem de todos os voos fechados
2 - Listagem de todos os voos fechados entre datas por critério
3 - Pesquisar voo fechado
4 - Inserir voo aos voos fechados
5 - Eliminar voo da tabela de voos fechado

Opção: 1
Voos fechados do aeroporto:
+-------------+---------+--------+-------------------+
| Id do avião | Destino | Origem | Data saída do voo |
+-------------+---------+--------+-------------------+
|           1 |   gfgfg |   ddgf |    2010/1/1 11:22 |
+-------------+---------+--------+-------------------+
```

- Exemplo de listagem de Voos fechados