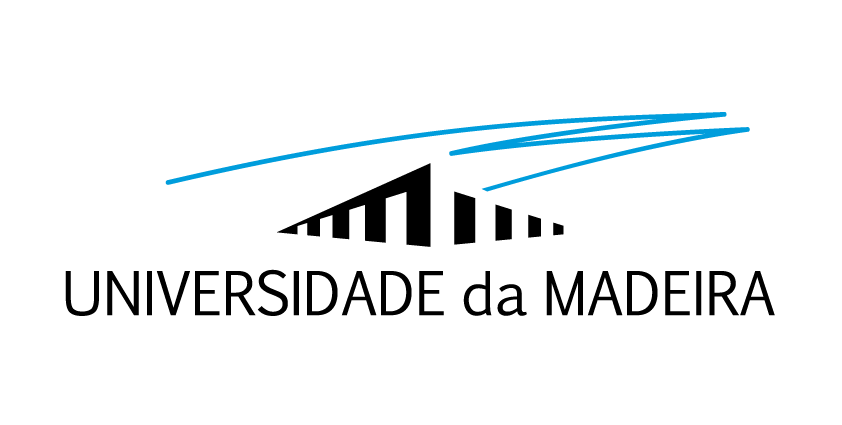
****

Estruturas de Dados e Algoritmos

Relatório do projeto prático

Aeroporto EDA

Lee Silva nº 2042016

Marcos Oliveira nº 2082116

Ricardo Lucas nº 2040416

Vítor Marques nº 2041116

**Introdução**

O objetivo deste trabalho foi criar um programa em C++ que fizesse a simulação de um aeroporto em que foi necessário implementar todas as funcionalidades básicas de um aeroporto de modo a que a simulação possa ser utilizada por entidades externas como um modelo de qualquer aeroporto.

Pontos a ter em conta:

* Representação dos aviões em aproximação;
* Representação dos aviões em pista;
* Representação dos aviões a descolar;
* Representação dos passageiros;
* Identificação/pesquisa dos aviões por voo ou modelo;
* Identificação/pesquisa dos passageiros por nome ou número de bilhete;
* Atividades de emergência: Alteração das prioridades de aterragem e descolagem.

**Problema**

Neste projeto, houve como objetivos secundários, a criação aleatória de aviões constituídos por:

* Nome de voo
* Modelo do avião
* Origem
* Destino
* Capacidade (variada entre 5 a 15)
* Conjunto de passageiros

Dentro da criação dos aviões, existe a criação de passageiros, constituídos por:

* Número do bilhete
* Primeiro nome
* Segundo nome
* Nacionalidade

Ao premir a seta direita inicia-se um ciclo. Cada ciclo, acontece os seguintes passos:

1. Remover um avião ao conjunto dos aviões a descolar.
2. Mover um avião do conjunto de aviões em pista para o conjunto de aviões a descolar
3. Mover um avião do conjunto de aviões em aproximação para

o conjunto de aviões em pista

1. Adicionar um avião ao conjunto dos aviões em aproximação.

O sistema guarda registo dos 10 próximos aviões a aterrar, 7 aviões em pista e 5 aviões a descolar.

Antes de qualquer avião aterrar o conjunto de aviões em aproximação tem de ter os 10 aviões completo e antes de qualquer avião descolar o conjunto de aviões em pista tem de estar cheio (7 aviões).

A qualquer momento o operador pode pressionar a tecla 'g' para gravar todos os dados do aeroporto.

Existe também a funcionalidade "Emergência" (que funciona quando o utilizador pressiona a tecla 'e'), que consiste em o utilizador identificar o avião em emergência e de seguida, o sistema pedirá o número do voo que deve ser descolado, para a colocação do avião em emergência.

A criação de passageiros e de aviões, como já foi referido anteriormente, deve ser feita aleatoriamente, utilizando ficheiros de texto: *modelos.txt, voo.txt, origem.txt, destino.txt e capacidade.txt* para os aviões e: *primeiro\_nome.txt, segundo\_nome.txt e nacionalidade.txt* para os passageiros. Os bilhetes dos passageiros deverão ser do tipo "TKXXXXXXXXXX" em que os X seriam qualquer número inteiro.

A cada chegada de um avião os seus passageiros serão removidos, pois estes chegaram ao destino. Mas enquanto este avião se encontra em pista um novo conjunto de passageiros deve ser criado e associado ao avião.

Os passageiros de nacionalidade estrangeira, que cheguem ao "aeroporto EDA" devem ser mantidos na lista durante 2 ciclos.

Entre cada ciclo, o utilizador poderá clicar na tecla 'o' para aceder a um menu de opções que consiste em:

* Mostrar todos os passageiros (primeiro, ultimo nome e nacionalidade);
  + Mostrar todos os passageiros, ordenados por ordem alfabética do último nome;
* Mostrar todos os voos (Modelo, voo e destino);
  + Mostrar em pista e a descolar, ordenados por ordem alfabética do destino
* Mostrar os passageiros em pista, por nacionalidade
* Pesquisa sobre os passageiros no aeroporto de origem estrangeira
* Listagem por ordem alfabética dos passageiros de origem estrangeira no aeroporto
* Pesquisa sobre os passageiros
* Editar o nome do passageiro
* Editar a nacionalidade do passageiro
* Editar destino de voo

**Solução Proposta**

Para os passageiros criou-se um struct onde implementou-se os as seguintes variáveis:

* Primeiro nome;
* Segundo nome;
* Nacionalidade;
* Numero do bilhete.

A implementação ficou da seguinte forma:

**struct Passageiro {**

**string \*primNome, \*segNome, \*Nacionalidade;**

**int bilhete;**

**};**

Para a solução dos aviões foi criado da mesma forma, mas criando um apontador para a struct Passageiro, este apontador serviu-nos para poder relacionar os passageiros que se apresentavam no avião. Esta implementação pode ser observada da seguinte maneira:

**struct aviao1 {**

**string \*nome, \*modelo, \*origem, \*destino;**

**int capacidade;**

**Passageiro \* passageiros;**

**};**

Em relação aos ficheiros de texto, mencionados no problema, criou-se uma função (**string \* carregar\_string**), esta função funciona da seguinte forma, cria um inteiro n que conta o numero de linhas que o ficheiro apresenta ter, em seguida criou-se um vetor e um apontador que aponta para a primeira linha do ficheiro e vai guardando cada linha do mesmo dentro do vetor, esta implementação pode ser observada da seguinte forma:

**string \*carregar\_strings(string nomeficheiro, int \*tamanho) {**

**ifstream file(nomeficheiro);**

**int n = 1 + count(istreambuf\_iterator<char>(file),**

**istreambuf\_iterator<char>(), '\n');**

**if (!file.is\_open()) {**

**cout << "NOT OPEN" << endl;**

**return NULL;**

**}**

**file.seekg(0, ios\_base::beg);**

**string line, \*lines = new string[n];**

**int i = 0;**

**while (i < n)**

**file >> lines[i++];**

**\*tamanho = n;**

**return lines;**

**}**

A função (**string\*load\_strings**) foi implementada noutra função (**void ficheiros**) que consiste na obtenção das linhas dos ficheiros relacionados com os aviões ("voo.txt", "modelo.txt", "origem.txt" e "destino.txt"):

**void ficheiros() {**

**nome = load\_strings("voo.txt", &n\_nome),**

**modelo = load\_strings("modelo.txt", &n\_modelo),**

**origem = load\_strings("origem.txt", &n\_origem),**

**destino = load\_strings("destino.txt", &n\_destino);**

**}**

O mesmo foi feito para os passageiros com (**string \* carregar\_strings**) e (**ficheiros2**).

**void ficheiros2() {**

**PrimNome = carregar\_strings("primeiro\_nome.txt", &n\_primNome),**

**segNome = carregar\_strings("segundo\_nome.txt", &n\_segNome),**

**Nacionalidade = carregar\_strings("nacionalidade.txt", &n\_Nacionalidade);**

**}**

Para a funcionalidade "Gravar dados", criámos uma função (**gravar1**) que consiste numa gravação das linhas presentes na consola num documento de texto:

**void gravar1(Fila& aviaoproximaçao, Fila& aviaoPista, Fila& aviaoDescolar)**

**{**

**Fila::Item \* aux = aviaoproximaçao.inicio;**

**Fila::Item\* aux2 = aviaoPista.inicio;**

**Fila::Item\* aux3 = aviaoDescolar.inicio;**

**string line;**

**ofstream myfile;**

**myfile.open("avioes.txt");**

**while (aux != 0 && aux2!=0 && aux3 != 0)**

**{**

**myfile << aux << "\n" << aux2 << "\n" << aux3 << endl;**

**aux = aux->next;**

**aux2 = aux2->next;**

**aux3 = aux3->next;**

**}**

**myfile.close();**

**}**

Após isto, desenvolveu-se outra função (**ficheiro3**) que traduz-se no cout das linhas guardadas no ficheiro "avioes.txt" criado anteriormente:

**void ficheiro3()**

**{**

**ifstream myFile("avioes.txt");**

**string line;**

**int n = 0;**

**myFile.is\_open();**

**while (getline(myFile, line)){**

**cout << line << endl;**

**n++;**

**}**

**myFile.close();**

**}**

Usamos filas para criar os aviões a ser utilizados:

**struct Fila{**

**struct Item{**

**string valor;**

**Item\* next;**

**};**

**Item\* inicio;**

**Item\* final;**

**};**

De seguida implementou-se várias funções para a manipulação filas a serem criadas (**void NovaFila, bool Vazia, string primeiroaviao, void Retira, void adicionaElemento, int Length e void escreve**):

**void Novafila(Fila& f)**

**{**

**f.final = NULL;**

**f.inicio = NULL;**

**}**

**bool Vazia(Fila& f)**

**{**

**return (f.final == 0);**

**}**

**string primeiroaviao(Fila& p)**

**{**

**return p.inicio->valor;**

**}**

**void Retira(Fila& p)**

**{**

**Fila::Item \* aux = p.inicio;**

**if (p.inicio == p.final)**

**{**

**p.inicio = NULL;**

**p.final = NULL;**

**}**

**else**

**{**

**p.inicio = p.inicio->next;**

**delete aux;**

**}**

**}**

**void adicionarElemento(Fila& f, string elemento){**

**Fila::Item\* aux = new Fila::Item();**

**aux->valor = elemento;**

**aux->next = NULL;**

**if (f.final == NULL){**

**f.inicio = aux;**

**f.final = aux;**

**}**

**else{**

**f.final->next = aux;**

**f.final = aux;**

**}**

**}**

**int Length(Fila& f)**

**{**

**Fila::Item\* aux = f.inicio;**

**int cont = 0;**

**while (aux != 0){**

**cont = cont + 1;**

**aux = aux->next;**

**}**

**return cont;**

**}**

**void escreve(Fila& f)**

**{**

**Fila::Item\* aux = f.inicio;**

**while (aux != 0){**

**cout << setw(5) << aux->valor << "\n";**

**aux = aux->next;**

**}**

**cout << endl;**

**}**

A função **Novafila** cria uma fila completamente vazia.

A função **Vazia** devolve True ou False dependendo se a fila estiver vazia ou não.

**primeiroAviao** devolve o primeiro avião presente na fila.

Na função **Retira**, diz-se que se retira o primeiro elemento da fila.

A função **adicionarElemento**, adiciona um avião à ultima posição da fila.

**Length** retorna o comprimento da fila.

**escreve** transmite a fila.

Também foi desenvolvida uma função chamada **novoaviao**:

**string novoaviao()**

**{**

**ficheiros2();**

**string p3="";**

**struct aviao1 a = randAviao();**

**p3 = "voo : " + \*a.nome + "\n";**

**p3 += "Modelo : " + \*a.modelo + "\n";**

**p3 += "origem : " + \*a.origem + "\n";**

**p3 += "Destino : " + \*a.destino + "\n";**

**p3 += "Passageiro: ";**

**p3 += "\n";**

**p3 += "-------";**

**p3 += "\n";**

**for (int i = 0; i < rand() % a.capacidade; i++){**

**p3 += (\*a.passageiros[i].primNome)+" "+(\*a.passageiros[i].segNome)+ ", ";**

**}**

**p3 += "\n";**

**p3 += "-------------------------------------";**

**p3 += "\n";**

**return p3;**

**}**

Considerando as funções acima definidas, foi criado uma função para gerar um ciclo:

**void Cicloaviao(Fila& aviaoproximaçao, Fila& aviaoPista, Fila& aviaoDescolar)**

**{**

**cout << "############################ Em aproximação ##############################" << endl;**

**adicionarElemento(aviaoproximaçao, novoaviao());**

**escreve(aviaoproximaçao);**

**if (Length(aviaoproximaçao) == 11)**

**{**

**cout << "############################ Na pista ############################" << endl;**

**adicionarElemento(aviaoPista, primeiroaviao(aviaoproximaçao));**

**escreve(aviaoPista);**

**Retira(aviaoproximaçao);**

**cout << "############################ A descolar ############################" << endl;**

**if (Length(aviaoPista) == 6)**

**{**

**adicionarElemento(aviaoDescolar, primeiroaviao(aviaoPista));**

**Retira(aviaoPista);**

**escreve(aviaoDescolar);**

**if (Length(aviaoDescolar) == 8)**

**{**

**Retira(aviaoDescolar);**

**}**

**}**

**}**

**}**

Neste código está explicado, como funcionam os ciclos, ou seja, à fila (por enquanto vazia) **aviaoproximaçao** adicionamos um novo avião e escrevemos a fila **aviaoproximaçao**. Repetimos o procedimento até a fila **aviaoproximaçao** ter Length de 11, porque quando tiver 10 aviões na fila, e o utilizador mandar executar mais um ciclo, o comprimento da fila passará para 11 e o programa avançará no código até o próximo if, que diz **if (Length(aviaoproximaçao) == 11).** Assim sendo repetir-se-á o processo anterior, mas desta vez retirando um avião da fila anterior para puder ter espaço para acrescentar um novo, de forma a preencher a fila **aviaoPista**. O mesmo acontece para a fila **aviaoDescolar.** No fim do ciclo, após as duas primeiras filas estarem completas, introduzimos **if (Length(aviaoDescolar) == 8) {Retira(aviaoDescolar);}** para retirar um aviao da fila **aviaoDescolar** e assim, deixando espaço para o próximo avião.

A emergência, consiste em pedir uma posição ao utilizador para poder retirar um avião em emergência da **fila aviaoaproximaçao**, para tal criou-se um apontador, nomeadamente **aux**, que ira apontar para o primeiro elemento da **fila aviaoproximaçao**, esse apontador ira percorrer a fila e selecionar a posição que foi introduzido pelo utilizador, podemos observar a implementação:

**int pos;**

**cout << "Digite a posiçao do aviao em emergencia " << endl;**

**cin >> pos;**

**int pos\_count = 0;**

**Fila::Item \*aux = aviaoproximaçao.inicio;**

**while (aux->next != NULL && pos\_count != pos) {**

**aux = aux->next;**

**pos\_count++;**

**}**

**cout << "remover aviao em Emergencia!" << endl;**

**Fila::Item \*aux1 = aux->next;**

**cout << "aviao: " << aux->valor << endl;**

**aux->next = aux->next->next;**

**delete aux1;**

Para poder transferir o avião que foi selecionado pelo utilizador, pede-se de seguida para inserir uma posição para transferir o avião em emergência. O código utilizado para esta transferência foi parecido ao código que se mostrou para a remoção do avião.

**Utilização da aplicação**

Ao executar o programa vai aparecer as seguintes opções:

* Emergência (clicar no E);
* Opções (clicar no O);
  + Todos os Passageiros (P);
  + Todos os Voos (V);
    - Voos em Aproximação e em Pista;
* Gravar (clicar no G);
* Ver gravação (clicar J);
* Sair (clicar no S);
* Gerar ciclo (clicar no N).

Na emergência pede para escolher a posição do avião que está em emergência e aterra-lo no aeroporto junto com os aviões em Pista, para tal pede-se ao utilizador para inserir a posição para aterra-lo.

Nas opções podemos ver todos os voos ao clicar no V, em seguida pede se deseja ver os aviões em aproximação e os que estão na pista.

Ao clicar na tecla G grava todos os aviões e seus passageiros e depois para vê-los é preciso clicar na tecla J. (Não esta 100% correto, não conseguimos gravar os dados completos)

Para gerar um novo ciclo e adicionar um avião aleatório aos aviões em aproximação é preciso clicar na tecla N (tecla N porque nós não conseguimos colocar na tecla da seta direita).

Por fim, para sair do programa basta clicar na tecla S.

**Conclusão e trabalho futuro**

Podemos concluir que o objetivo deste projeto foi parcialmente cumprido, visto que a maioria dos objetivos propostos foram cumpridos, com a utilização vários códigos de implementação obtidos através da cadeira de EDA.

Com a conclusão deste projeto ajudou-nos no entendimento da linguagem C++, mais concretamente na criação de arrays, alocação dinâmica de memória, criação de listas ligadas e operações entre listas, métodos de ordenação de listas e pesquisa de elementos de uma lista.

A finalização deste projeto poderá nos ajudar futuramente na conclusão de outras cadeiras de programação e num trabalho futuro, pois poderá servir de fundamento e aprendizagem de outras linguagens de programação, já que a base teórica é constante em varias linguagens.

**Referências**

Fóruns de C++ (cplusplus.com, stackoverflow.com, etc.)

Moodle.cee.uma.pt (através dos PowerPoints disponibilizados pelos docentes)