Trabalho Prático 1: Sistema de Gerenciamento de Aeroportos xAero

Leandro Lázaro (3513), Mateus Pinto (3489), Vinícius Júlio (3495)

Ciência da Computação – Universidade Federal de Viçosa – Campus Florestal (UFV-Caf) – Florestal – MG – Brasil

{leandro.lazaro, mateus.p.silva, vinicius.julio}@ufv.br

Resumo. *Criamos um sistema de gerenciamento de aeroportos batizado de xAero usando, como os Tipos Abstratos de Dados principais, o TAD voo, Lista de Voos, Item Matriz e a Matriz de Voos. Como secundário, foi usado o TAD horário. Ao fim, criamos um programa Main para gerenciar os TADs.*

1. Introdução

Neste trabalho implementaremos um sistema de gerenciamento completo de aeroporto com a principais funções: criar novo voo, retirar novo voo, verificar voos diários, etc. Todo o sistema será feito em linguagem C através de Tipos abstratos de dados principais e auxiliares, e criaremos uma interface agradável via terminal para o usuário final do sistema.

2. Nomenclatura Utilizada

Para uma melhor organização e compreensão do código, adotamos a seguinte nomenclatura em relação aos TADs Principais:

TADs Principais: “NOME_DO_TAD.h” e “NOME_DO_TAD.c”

Tipos (Estruturas, Listas, Matrizes): “NOME_DO_TIPO”

Funções: “NOME_DO_TAD_nomeFuncao”

E em relação aos TADs Auxiliares:

TADs Auxiliares: “nome_do_tad.h” e “nome_do_tad.c”

Tipos (Estruturas, Listas, Matrizes): “nome_do_tipo”

Funções: “nome_do_tad_nomeFuncao”

3. Tipos Abstratos de Dados

Tipos Abstratos de Dados utilizados no sistema. Para melhor organização, os TADs Auxiliares e Principais foram separados em duas diferentes pastas: “TAD_AUXILIAR/” e “TAD/”.

3.1 Tipos Abstratos de Dados Auxiliares

Durante o desenvolvimento, notamos a necessidade de TADs auxiliares para melhor organizar os dados. Neste tópico, explicaremos o motivo da implementação de cada um deles e as curiosidades de cada dado dentro das e estruturas.

3.1.1. Horário

Como era muito recorrente a necessidade de definir o horário de acontecimentos, criamos o TAD horario.

Seu header se encontra em “TAD_AUXILIAR/horario.h” e define o tipo “horario” que é composto pelas variáveis “hora” e “min”, que são declaradas como unsigned int uma vez que horas são números naturais (inteiros e positivos), além de definir os protótipos das funções a serem executadas.

O source se encontra em “TAD_AUXILIAR/horario.c” e implementa as funções de inicialização do horário, verificação de os horários estarem em ordem crescente, estabelece o horário no TAD horario, e mostrar o horário.

3.2 Tipos Abstratos de Dados Principais

Implementação dos TADs essenciais para o funcionamento do sistema de aeroporto.

3.2.1. Voo

O TAD Voo contém as informações de cada voo.

O header é localizado em “TAD/VOO.h” e define o tipo VOO, que é composto pelo VID (que é um unsigned int, já que não é necessário que o código tenha sinal negativo), o horário de decolagem e de pouso, o aeroporto de decolagem e de pouso, e o identificador de pista de decolagem, que também é um unsigned int. O cabeçalho também define os protótipos de funções das operações a serem feitas no tipo VOO.

O source está localizado em “TAD/VOO.c” e contém a implementação das funções de inicializar a estrutura VOO, e funções “set”, “get” e “show” que, respectivamente, alteram, leem e mostram cada uma das informações contidas no tipo VOO.

Para o VID, usamos, além do random padrão do C, uma variável global int denominada randomizer, que a cada inserção de VOO tem seu valor acrescido e somado ao número gerado aleatoriamente. Esse recurso foi necessário visto que a linguagem C atualiza sua semente de números aleatórios de segundo a segundo, e o modo arquivo ficaria corrompido, visto que são necessárias milhões de alterações para que o programa rode em mais de um segundo. Em arquivos menores, todos os VOOs ficariam com o mesmo VID, problema resolvido graças ao randomizer.

3.2.2. Lista de Voos

O TAD Lista de Voos coloca os voos, definidos pelo TAD Voo, que decolaram em um determinado dia em uma lista encadeada, o que facilita a inserção e remoção de voos caso hajam cancelamentos ou mudanças de horários.

Seu header encontra-se em "TAD/LISTA_DE_VOOS.h" e define como item "ITEM_LISTA_DE_VOOS", que contém o item VOO, e a lista encadeada "LISTA_DE_VOOS". Também define o protótipo das funções de operações a serem realizadas na lista.

O source, localizado em "TAD/LISTA_DE_VOOS.c" implementa as funções de criação da lista, inserção e remoção de voos, e também as de retornar e mostrar os voos pelo identificador.

3.2.3. Item Matriz

Os voos que partirem de um determinado aeroporto em um mesmo dia serão colocados na MATRIZ_VOOS, para isso deverá ser criado o TAD Item Matriz contendo as listas de voos e outras informações necessárias.

O header é encontrado em "TAD/ITEM_MATRIZ.h" e define o item "ITEM_MATRIZ", que contém um item LISTA_DE_VOOS, o número na lista e o horário da última atualização dessa lista. Também é definido o protótipo das funções que serão implementadas no TAD ITEM_MATRIZ.

O source se encontra em "TAD/ITEM_MATRIZ.c" e tem a implementação das funções de inicialização do ITEM_MATRIZ, funções "set", "get" e "show" (alterar, ler e mostrar) para cada informação contida no ITEM_MATRIZ.

3.2.4. Matriz Voos

O TAD MATRIZ_VOOS cadastra todos os voos que partiram de um aeroporto e dia a ser definido, esses voos foram agrupados em uma lista encadeada pelo TAD LISTA_DE_VOOS e definidos como item de matriz no TAD ITEM_MATRIZ. Os voos estão organizados pelos horários de decolagem e pouso, sendo cada posição da matriz referente a uma faixa de horário, as linhas representam o horário de decolagem e as colunas o de pouso, por exemplo, MATRIZ_VOOS[0][0] representa de 0h até 1h, MATRIZ_VOOS[1][1] representa de 1h até 2h, e assim por diante.

O header fica localizado em "TAD/MATRIZ_VOOS.h" e define o tipo MATRIZ_VOOS que contém a o item_matriz com os a matriz dos voos e suas faixas de horários, também possui o tipo horário e o total de voos do dia selecionado (em forma de unsigned int). Também define o protótipo das funções a serem usadas

O source é encontrado em "TAD/MATRIZ_DE_VOOS.c" e implementa as funções prototipadas no header, que são funções de inicialização da matriz, funções para alterar, remover e leitura do voo, mostrar voo com base em horário de decolagem e de pouso, apenas horário de decolagem e apenas horário de pouso, mostrar todos os voos cadastrados na matriz, mostrar faixa de horário com maior e menor número de voos

cadastrados, mostrar faixa de horário com maior e menor número de alterações recentes, verificar se matriz é esparsa, ou seja, confere se a quantidade de posições da matriz sem voos cadastrados é, pelo menos, o dobro do número de posições com voos cadastrados, por fim, há a função que salva a matriz.

4. Programa Principal (main.c)

O Programa Principal é a implementação do sistema de gerenciamento de aeroportos xAero. O sistema pode ser utilizado em dois modos: interativo, onde o usuário informa as manualmente as informações e escolhe as operações a serem realizadas, e o modo por arquivo, onde os parâmetros serão passados por meio da leitura de um arquivo contendo os dados dos voos ou gravação. Ambos os modos realizam as mesmas operações. Para melhor organização do método "main", cada uma das operações têm a sua própria função para realizar as operações de inicialização, criação dos TADs, alocamento de memória, e as próprias operações descritas no menu.

É mostrado ao usuário a tela de boas-vindas em ASCII e o menu principal. Nele, é impresso opções de "a" a "m" que são relacionadas às operações do modo interativo, e são descritas no TAD Matriz Voos, a opção "8", que salva todo o banco de dados para posterior continuação do uso do mesmo, e a opção "9", que ativa o modo de arquivo, em que os parâmetros passados nas funções são lidos de um arquivo contendo as informações sobre os voos e as operações em que esses parâmetros serão utilizados. Ainda nas seleções de menu, há a opção "?" que imprime a descrição do sistema xAero e cita os seus desenvolvedores. Por fim, a opção "0" encerra a execução do programa, que está em um while com a condição de parada de que o usuário selecione essa opção, e mostra a mensagem de encerramento.

Foram usadas duas variáveis globais com o intuito de deixar a sintaxe mais clara para os programadores e acelerar o desempenho: a matriz do tipo MATRIZ_VOOS e a ponteiroMatriz do tipo ponteiro para o TAD anterior. Em momento algum o endereço desse ponteiro foi alterado, garantindo assim que o uso de variável global nesse programa foi feito de forma consciente.

O modo leitura de arquivo percorre o arquivo selecionado lendo o primeiro caractere de cada linha e o interpreta como a opção a ser selecionada, assim ele armazena as informações da linha e chama a função correspondente. Ao fim das operações, o arquivo é fechado para evitar corrompimento.

O modo gravação de arquivo percorre a matriz e todas as suas listas encadeadas gerando um novo arquivo de entrada com a primeira linha iniciando a matriz de voos e as instruções de inserção de todos os VOOs existentes naquela execução.

O programa também exibe mensagens de erro ou sucesso de acordo com o retorno das funções e seus motivos de erro.

5. Compilação com Makefiles

O programa conta com dois makefiles, um para Windows e outro para Linux, a fim de facilitar a compilação para o usuário;

5.1. Windows

Para compilar no Windows, basta que o usuário dê um clique duplo no arquivo “make.bat”, arquivo feito para tal. É necessário ter o GCC instalado.

5.2. Linux

Para compilar no Linux, basta que o usuário abra o terminal e digite “make”. Para rodar, o comando correto é “make run” e a instrução necessária para apagar o arquivo compilado é “make clean”. O código feito para tal está no arquivo “makefile” e usa o recurso de mesmo nome da linguagem C presente nas plataforma Unix.

6. Considerações Finais

O desenvolvimento do sistema de aeroportos nos permitiu ver uma aplicação de maior complexidade que utiliza o conceito de Tipo Abstrato de Dados e perceber as vantagens de seu uso por permitir uma melhor compreensão do código, o que facilita o seu desenvolvimento, pode ser facilmente reutilizado, e a sua manutenção é de maior praticidade, uma vez que o TAD não está vinculado com a sua implementação. Isso também contribui para um maior nível de segurança já que o usuário não possui acesso direto aos dados, apenas às operações definidas pelo desenvolvedor.

Em suma, esse foi um projeto em que foi cobrado todo o conhecimento de programação adquirido até então no curso de Ciência da Computação, foram utilizados os conceitos de matrizes, subprogramas, estruturas, ponteiros, arquivo, TADs, alocação dinâmica de memória e listas encadeadas para a implementação de um sistema de gerenciamento de aeroportos completo e funcional.