Desafio de Estrutura de Dados
"Simulador de Aeroporto"

Universidade Tecnológica Federal do Paraná (UTFPR), campus Apucarana
Curso de Engenharia de Computação
Disciplina de Estrutura de Dados - EDCO3A
Prof. Dr. Rafael Gomes Mantovani

---

1 Objetivo

O objetivo deste exercício, originalmente proposto Ziviani (1999) [2], é simular padrões de aterrissagem e decolagem em um aeroporto.

---

2 Descrição da atividade

Suponha que um aeroporto possua 3 pistas, numeradas 1, 2 e 3. **Existem 4 (quatro) filas de espera para aterrissagem**, duas para cada uma das pistas 1 e 2. Aeronaves que se aproximam do aeroporto devem se integrar a uma das filas de espera, sendo que estas filas devem procurar manter o mesmo tamanho. Assim que um avião entra em uma fila de aterrissagem, ele recebe:

- um número de identificação (id); e
- um número inteiro que indica o número de unidades de tempo que o avião pode permanecer na fila antes que ele tenha que descer. Do contrário, seu combustível acaba e ele cai.

Existem também **3 (três) filas para decolagem**, uma para cada pista. Os aviões que chegam nestas filas também recebem um id. Estas filas também devem procurar manter o mesmo tamanho.

A cada unidade de tempo, de 0 a 3 aeronaves podem chegar nas filas de decolagem, e de 0 a 3 aeronaves podem chegar nas filas de espera para pouso. A cada unidade de tempo, cada pista pode também ser usada para um pouso ou uma decolagem. A pista 3 em geral é usada apenas para decolagens, a não ser que um dos aviões nas filas de espera fique sem combustível, quando ela deve ser imediatamente usada para pouso. Se apenas uma única aeronave está com falta de combustível, ela pousará na pista 3; se mais de um avião estiver nessa situação, as outras pistas poderão ser usadas (a cada unidade de tempo, no máximo 3 aviões poderão estar nesta desagradável situação).

2.1 Como organizar a modelagem?

Use:

- números inteiros pares sucessivos para o *id* dos aviões chegando nas filas de decolagem;
- inteiros ímpares sucessivos para o *id* dos aviões chegando nas filas de aterrissagem.

A cada unidade de tempo, assuma que os aviões entram nas filas antes que aterrissagem ou decolagem ocorram. **Tente projetar um algoritmo que não permite o crescimento excessivo das filas de aterrissagem ou decolagem**. Coloque os aviões sempre no final das filas, que não devem ser reordenadas.

2.2 O que deve ser impresso periodicamente?

Periodicamente, a cada instante de tempo, o programa deverá mostrar o que ocorre com as filas. Dessa forma, para todo instante de tempo, imprima:

a) o conteúdo de cada fila;

b) o tempo médio de espera para decolagem;

c) o tempo média de espera para aterrissagem; e

d) o número de aviões que aterrissam sem reserva de combustível.

Os itens **b** e **c** acima devem ser calculados para os aviões que já decolaram ou pousaram, respectivamente. A saída do programa deve ser auto-explicativa e fácil de entender. **Dica:** A entrada poderia ser criada manualmente, mas o melhor é utilizar um gerador de número aleatórios.

Para cada unidade de tempo, a entrada deve ter as seguintes informações:

a) número de aviões (0-3) chegando nas filas de aterrissagem com respectivas reservas de combustível (de 1 a 20 em unidades de tempo);

b) número de aviões (0-3) chegando nas filas de decolagem.

2.3 O que deve ser entregue?

Para entrega da atividade, o aluno deve enviar ao professor:

a) todos os arquivos desenvolvidos em C para a resolução do problema;

b) relatório individual descrevendo a modelagem realizada, explicando todas as tomadas de decisão (estruturas de dados usadas, lógica desenvolvida, etc). Constar no relatório também casos de teste realizados (pode inserir figuras para mostrar as execuções realizadas);

c) apresentação oral para o professor em horário a ser combinado.

3 Orientações gerais

- Implementar também o controle de erros, para lidar com exceções que possam ocorrer ao longo da execução do programa;
- Para acompanhamento do desenvolvimento, criar um repositório individual com o código desenvolvido no `github Classroom`, por meio do link: https://classroom.github.com/a/p78qHqDu. Os repositórios serão privados, com acesso apenas do professor e do aluno.
- Entrega do programa final: via Moodle. O aluno deve submeter o código-fonte, e relatório com as análises na página da disciplina no Moodle.
- **Data máxima para entrega**: fim do semestre letivo corrente.//
- Os códigos desenvolvidos por cada aluno serão também verificados por ferramentas de plágio. Códigos iguais/similares terão nota zero.

---

4 Links úteis

Números aleatórios em C:

- http://linguagemc.com.br/valores-aleatorios-em-c-com-a-funcao-rand/
- https://www.ime.usp.br/~pf/algoritmos/aulas/random.html
- http://www.cplusplus.com/reference/cstdlib/rand/

Referências

[1] Thomas H. Cormen,; Ronald Rivest; Charles E. Leiserson; Clifford Stein. Algoritmos - Teoria e Prática - 3ª Ed. Elsevier - Campus, 2012.

[2] Nivio Ziviani. Projeto de algoritmos com implementações: em Pascal e C. Pioneira, 1999.

[3] Adam Drozdek. Estrutura De Dados E Algoritmos Em C++. Cengage, 2010.
