Centro Universitário Barão de Mauá

Estrutura de Dados
Laboratório de Estrutura de Dados

Trabalho I

O trabalho consiste em implementar um jogo chamado FreeCell em C++. Este é um jogo que utiliza as 52 cartas de um baralho:

- 13 cartas de cada naipe [*A (ás - 1), 2, 3, 4, 5, 6, 7, 8, 9, 10, J (valete - 11), Q (dama - 12) e K (rei - 13)*].
- Naipes: Pretos - *Espadas* e *Paus;* Vermelhos - *Ouro e Copas*.

Existem 8 pilhas de jogo, onde no início do jogo 4 dessas pilhas contêm 7 cartas, e as outras 4 pilhas contêm 6 cartas cada, totalizando as 52 cartas do baralho. Essas cartas são distribuídas aleatoriamente e visíveis ao jogador.

Neste sentido, é criada uma ordem para as cartas atribuídas aleatoriamente em cada uma das pilhas de jogo. Consequentemente **as únicas cartas que podem ser movimentadas são as que estão no topo.**

Objetivo do jogo: Mover todas as cartas das pilhas de jogo para as pilhas de saída.

Existem 4 pilhas de saída, inicialmente vazias. Nas pilhas de saída as cartas devem estar em ordem (ás, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q e K).

Para mover as cartas das pilhas de jogo para as pilhas de saída, o jogador poderá fazer uso de 4 posições auxiliares (as free cells) para guardar temporariamente uma carta. Importante: cada *free cell* pode ter no máximo uma carta em qualquer momento do jogo e qualquer carta pode ser colocada nessas posições.

O jogador fará quantos movimentos quiser, entre as pilhas que quiser, respeitando as seguintes regras:

- Para uma *free cell* desocupada: qualquer carta do topo de uma pilha de jogo;
- Para uma das pilhas de saída: qualquer carta de uma *free cell* ou do topo de uma pilha de jogo. Movimentos para uma pilha de saída devem ser feitos em ordem do menor para o maior, sempre de mesmo naipe. Assim, temos uma pilha de saída para cada um dos 4

naipes. Ases sempre podem ser movidos para uma pilha de saída vazia;

- Para o topo de uma pilha de jogo: qualquer carta de uma das *free cells* ou do topo de uma outra pilha de jogo. Movimentos para uma pilha de jogo devem ser feitos em ordem do maior para o menor, alternando a cor do naipe. Por exemplo: se o topo de uma pilha de jogo contém um 4 de paus (que é preto), podemos mover para o topo dessa pilha um 3 de copas ou de ouro (que são vermelhos).

O fim do jogo é alcançado quando todas as cartas forem movidas para as pilhas de saída ou quando não há movimento que permita mais alguma carta ser movida para uma das pilhas de saída.

Sua tarefa é implementar um programa em C++ que permita ao usúario jogar FreeCell como descrito acima, com a possibilidade de jogar várias partidas. A decisão de como implementar suas pilhas é sua, mas deve seguir as instruções abaixo.

Importante: Você pode utilizar no máximo 6 pilhas para controlar todas as 12 pilhas necessárias para o funcionamento do jogo (pilhas de jogo e pilhas de saída). Cada pilha utilizada em sua implementação deve ter capacidade fixa para no máximo 26 elementos. Implementações com menos pilhas e com pilhas menores ganham pontos extras.

Seu programa deve imprimir o conteúdo das pilhas a cada jogada, assim como uma mensagem dizendo que movimento o usuário realizou. As pilhas de jogo deverão ser impressas em forma de pilhas, uma ao lado da outra. Faça consistência de dados, ou seja, o seu programa deve avisar quando o usuário tenta fazer um movimento de cartas inválido e permitir que ele tente uma nova jogada.

Atenção!

- Um mesmo trabalho poderá ser feito por até **3** estudantes matriculados na disciplina;
- Entrega: 17/09/2023 contendo todos os códigos fontes, devidamente documentados, até às 23:59h via Portal (SAV da disciplina LED);
- Não serão aceitos trabalhos e modificações após a data e horário de entrega;

- **Não serão aceitos trabalhos que utilizem *containers* (*vector, queue, stack, priority_queue, list, set, map...*), bem como declarações de variáveis/ponteiros com auto type.**
- Não serão aceitos códigos em outra linguagem de programação;
- O código deve ser organizado usando divisão das classes entre arquivos de cabeçalho (.h) e implementação (.cpp);
- Obviamente seu programa deverá ser bem documentado, e a forma de utilizá-lo também será avaliada.
- Os programas resultantes do trabalho poderão serem testados na presença do professor durante o horário da aula de LED seguinte à data de entrega;
- Todo arquivo de código fonte deve conter, nas suas primeiras linhas, um campo de comentário com o nome e o número institucional dos responsáveis pelo trabalho;
- Trabalhos reconhecidos como 'muito semelhantes' pela sua estrutura de programação serão desconsiderados. Lembrem-se, variáveis com nomes diferentes, mas em códigos com a mesma estrutura, são considerados 'muito semelhantes';
- Ressalto que estas propostas de trabalhos podem envolver alguns conhecimentos da linguagem de programação C++ que não foram cobertos pelos exemplos ou pelos exercícios realizados em aula. No entanto, estes conhecimentos estão disponíveis nos livros referenciados como material de apoio na ementa da disciplina.

Bom trabalho!