Universidade Estadual de Maringá
Departamento de Informática

Disciplina: Estrutura de Dados – 9893
Curso: Informática
Professora: Juliana Keiko Yamaguchi

2º trabalho prático

Objetivo

Consolidar os conceitos de filas, pilhas e árvores, por meio da aplicação dessas estruturas nas resoluções de problemas.

Instruções

1. O trabalho pode ser desenvolvido individualmente ou em equipe de, **no máximo**, **duas** pessoas.
2. Data de entrega: **18/01/20**.
3. O trabalho deve ser desenvolvido na linguagem C.
   - Deve ser implementado o conceito de TAD, portanto, devem ser implementados:
     - o arquivo de **declaração** dos subprogramas – *header* (.h)
     - o arquivo com a **definição** dos subprogramas declarados no *header* (.c)
     - e o arquivo contendo o programa principal (.c).
   - Os dois últimos arquivos (.c) devem incluir chamada ao arquivo .h.
4. Os arquivos do trabalho devem ser compactados em um único arquivo que deve ser nomeado de acordo com o nome dos integrantes da equipe seguido de seu R.A.
   - Exemplos:
     - NomeAluno123456.zip
     - FulanodeTal123456_OutroSicrano678910.zip.
5. O trabalho deve ser entregue via Moodle, por meio de um link para *upload* de arquivos disponível até as 23:55 h da data da entrega.
6. É permitida a interação entre as equipes, mas **cópias** de códigos implicará automaticamente na atribuição de **nota zero a todos** os trabalhos envolvidos.
7. A ordem de apresentação dos trabalhos se dará por ordem alfabética de acordo com os nomes dos integrantes das equipes.

Descrição do trabalho

1. (7,0) Faça um programa que simule o jogo de cartas Paciência de modo simplificado. O jogo consiste de um baralho de 52 cartas. Cada carta possui os seguintes campos:
   - valor: 1 a 13, ou seja, de Às a Rei
   - naipe: copas, espadas, ouros, paus.

   O objetivo do jogo é distribuir as 52 cartas em 4 pilhas, em que cada pilha possui 13 cartas ordenadas por naipe e em ordem crescente.

   O programa deve conter as seguintes estruturas de dados:

   a) Uma estrutura (*struct*) para definir a carta;

   b) Uma pilha para guardar as cartas por naipe. Você pode decidir se usará uma pilha estática ou dinâmica.

   c) Uma fila para guardas as 52 cartas do baralho. Você pode decidir se usará uma fila estática ou dinâmica.

   Além das funções elementares de pilha (empilhar, desempilhar, pilha_vazia, liberar) e das

funções elementares de fila (enfileirar, desenfileirar, fila_vazia, liberar), o programa deve conter as seguintes funções/procedimentos:

a) fila *criar_baralho(): cria uma variável do tipo fila inicialmente vazia. Depois, cria uma variável do tipo carta, preenche os campos da carta e a insere na fila (enfileirar a carta). A função retorna a fila preenchida com as 52 cartas.

b) pilha *criar_pilha():cria uma variável pilha vazia que armazenará as cartas.

c) void exibir_fila(fila *F): recebe uma fila e exibe suas cartas.

d) void exibir_pilha(pilha *P): recebe uma pilha e exibe suas cartas.

e) fila *embaralhar(fila *F): recebe a fila F(baralho com 52 cartas) e a divide em duas filas internas, f1 e f2. Isto é, enquanto F não esvaziar, desenfileira 26 cartas em f1 e as 26 cartas restantes em f2. Desse modo, cada fila (f1 e f2) terá 26 cartas. Em seguida, a função retorna a fila F contendo a intercalação das filas f1 e f2. A intercalação das filas f1 e f2 consiste da chamada da função desenfileirar de f1 e desenfileirar de f2. Ou seja, a carta desenfileirada de f1 é enfileirada em F e, em seguida, a carta desenfileirada de f2 é enfileirada em F, não necessariamente nesta ordem. Ao final, a fila F é retornada com as cartas embaralhadas (intercaladas).

f) int movimento_valido(carta *C, pilha *P): deve retornar verdadeiro (1), se a carta pode ser empilhada na pilha P, falso (0), caso contrário. Uma carta pode ser empilhada se for do mesmo naipe da carta do topo, e se o valor da carta a ser empilhada é maior em apenas 1 unidade. Isto é, se a pilha estiver vazia, só pode ser inserido uma carta com o valor 1 (Às). Se a pilha não estiver vazia e no topo estiver a carta 2, só poderá ser empilhado a carta com valor 3 de mesmo naipe, e assim sucessivamente.

Funcionamento do jogo:

- Criar uma fila de 52 cartas; essa fila deve ser embaralhada.
- Criar 4 pilhas secundárias vazias, uma para cada naipe.
- O jogador deve receber uma carta da fila (desenfileirar) e escolher em qual pilha deve colocar a carta (empilhar a carta em uma das pilhas secundárias).
- A cada jogada, o programa deve verificar se o jogador escolheu a pilha correta para a carta (chamada da função movimento_valido)
- Se a jogada não for válida, a carta volta para a fila (enfileirar) e o jogador perde 2 pontos. Se a jogada for válida, o jogador ganha 1 ponto.
- O jogo termina quando a fila estiver vazia e as 4 pilhas secundárias estiverem completas, com as cartas agrupadas por naipe e ordenadas por número.

2. (3,0) Implemente uma árvore binária de busca que armazena valores inteiros e que contenha as seguintes funções:

a) no *criar();

b) int arvore_vazia(no *raiz);

c) no *alocar(int dado);

d) no *inserir(no *raiz, no *novo);

e) no *buscar(no *raiz, int dado);

f) void mostrar_pre_ordem(no *raiz);

g) void mostrar_in_ordem(no *raiz);

h) void mostrar_pos_ordem(no *raiz);

i) int altura(no *raiz) → retorna a altura da árvore;

j) int contar_nos(no *raiz) → retorna quantos nós existem na árvore;

k) void mostrar_largura(no *raiz);

l) no *remover_copia_antecessor(no *raiz, int dado);

m) no *remover_fusao_antecessor(no *raiz, int dado);

O que será avaliado

1. Implementação correta das estruturas e respectivas operações conforme a descrição deste trabalho.
2. Organização do código (uso correto do conceito de TAD, endentação, coesão dos subprogramas).
   - coesão dos subprogramas = o subprograma tem um único propósito; desse modo, fazer com que o programa principal contenha a maior parte das mensagens de interação com o usuário.
3. Usabilidade do programa:
   - facilidade de utilização proporcionada pelas telas (menus do console) de interação com o usuário;
   - mensagens de erro para situações como: estrutura vazia, valores inválidos, etc.