29/10/2018

COM111 - Algoritmo e Estrutura de Dados I

Profª Elisa de Cássia Silva Rodrigues

Observações:

Equipe: grupos de 2 ou 3 alunos.
Prazo de entrega: 23h59 do dia 01/11/2018 (segunda-feira).
E-mail: `profa.elisa.rodrigues@gmail.com`
Assunto do e-mail: `[COM111][T3] xxxxx e xxxxx`
Arquivos: `com111_pilhade.c`, `com111_pilhade.h`, `com111_main.c`
** Os arquivos implementam a TAD Pilha Dinâmica Encadeada Simples **

Atividade Prática T3 (Valor: 2 pontos)

1. Considere a aplicação da estrutura de dados Pilha no seguinte cenário:

   Em computação gráfica, a aplicação de uma transformação geométrica a um objeto 2D pode ser feita através da multiplicação de matrizes de transformação de dimensão 3×3 (utilizando-se coordenadas homogêneas). Dentre as transformações possíveis estão: translação (T), rotação (R) e escala (E).

   Considere que a aplicação de uma matriz de transformação T a um ponto $p(x, y)$ de um objeto é feita na seguinte ordem: $p' = T * p$. Se mais de uma transformação (por exemplo, T, R e E, nesta ordem) é aplicada ao ponto p, temos que o ponto p''' transformado pode ser obtido da seguinte forma:

   $$p' = T * p$$
   $$p'' = R * p'$$
   $$p''' = E * p''$$

   Outra forma mais simples de obter o ponto p''', é encontrar a matriz final de transformação M e aplicar ao ponto p de forma direta, isto é:

   $$p''' = M * p$$

   Para encontrar a matriz final de transformação M, multiplica-se as matrizes de cada transformação na ordem inversa da aplicação. No exemplo acima temos que M é:

   $$M = E * R * T$$

   Uma das formas de executar este procedimento, para encontrar a matriz final M, é utilizar uma **pilha** para armazenar as **matrizes de transformação** conforme forem sendo determinadas, por exemplo T, R e E. Quando for necessário executar a multiplicação, desempilha-se as matrizes duas a duas e as multiplica, de forma que a matriz resultante $M' = E * R$ possa ser multiplicada pela próxima T, até sobrar apenas uma matriz na pilha (a matriz final M). *Note que várias matrizes de transformação podem ser armazenadas na pilha.*

2. Considerando a aplicação acima, sua tarefa é executar as seguintes alterações na TAD Pilha Dinâmica Encadeada Simples:

   (a) Alterar a `struct elemento` para que o dado armazenado seja uma matriz 3×3. (***Obs**: utilize um ponteiro `*mt` para essa finalidade.*)

   ```
   struct elemento{
     double *mt;
     struct elemento *prox;
   };
   ```

   (b) Alterar a função `empilhar()` para empilhar uma matriz de transformação MT[3][3] em uma pilha M. A função deve retornar 1 se a operação foi possível ou 0, caso contrário. O protótipo da função será:

   ```
   int empilhar(Pilha *M, double MT[3][3]);
   ```

   (***Obs**: para atribuir a matriz `MT`, recebida como parâmetro, ao ponteiro `mt` do elemento `no` utilize:* `no->mt = &MT[0][0]`.)

   (c) Alterar a função `desempilhar()` para desempilhar uma matriz de transformação MT[3][3] da pilha M. A função deve retornar um ponteiro para a matriz MT se a operação foi possível ou $NULL$, caso contrário. O protótipo da função será:

   ```
   double *desempilhar_topo(Pilha *M);
   ```

   (d) Escrever uma função para desempilhar todas as matrizes da pilha M até encontrar a matriz final de transformação. Esta função deve desempilhar as duas últimas matrizes da pilha M, multiplicá-las e empilhar novamente a matriz resultante. A multiplicação deve ser feita até que reste apenas a matriz final na pilha M. A função deve retornar 1 se a operação foi possível ou 0, caso contrário. O protótipo da função será:

   ```
   int desempilhar(Pilha *M);
   ```

   (***Obs**: para acessar o elemento `i,j` da matriz `MT` (`MT[i][j]`) utilize:* `no->mt[i*3+j]`.)

3. Para testar as funções acima, crie um menu com as opções:

   1. Empilhar matriz
   2. Desempilhar
   3. Sair

   A opção 1 lê uma matriz `MT[3][3]` e chama a função `empilhar(M, MT)`, onde `M` é uma pilha (Pilha *M). A opção 2 chama a função `desempilhar(M)` e imprime a matriz resultante na pilha `M`.