Estruturas de Dados I

Prof. Roney Pignaton da Silva

Exercícios sobre Listas , Pilhas e Filas

1. Crie um TAD em linguagem C de uma estrutura de dados chamada pilha, contendo as seguintes funções: empilhar, desempilhar, altura, topo;
2. Crie um TAD em linguagem C de uma estrutura de dados chamada fila, contendo as seguintes funções: enfileirar, desenfileirar, comprimento, próximo;
3. Crie um TAD em linguagem C para manipular uma lista duplamente encadeada (LDE), contendo as seguintes funções:
   a) cria lista,
   b) busca ordenada,
   c) busca desordenada,
   d) inserção e remoção,
   e) intercala lista,
   f) ordena lista com ordenação crescente e decrescente,
   g) Remover todos os nós da lista que possuem informações duplicadas (deixe apenas uma ocorrência de cada número e dar free nos nós com valores repetidos).
4. Crie um programa que faça uso das funções implementadas no item anterior. Considere que este programa deverá:
   a) realizar a leitura de duas listas de 10 elementos, cada;
   b) imprimir as duas listas criadas;
   c) solicitar ao usuário a escolha de 2 elementos da primeira lista que deverão migrar para a segunda lista;
   d) efetuar a ordenação em ordem decrescente da segunda lista;
   e) intercalar os elementos das duas listas em uma nova lista;
   f) imprimir a nova lista.
   Obs.: implemente novas funções internas aos TADs, caso seja necessário.

5. Considere uma lista duplamente encadeada circular e implemente um programa que contém as seguintes funções:
   a) Inserção no início da lista.
   b) Remoção (free) do primeiro elemento da lista.
   c) Inserção na última posição da lista.
   d) Remoção (free) do último elemento da lista.
   e) Remoção (free) de uma chave informada.
   f) Impressão da lista.

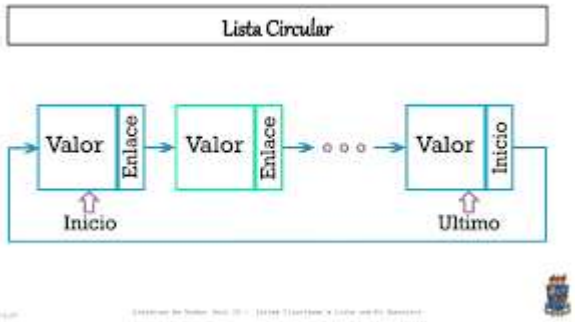
6. Considere as estruturas seguintes:

```
typedef struct no{
  int ch;
  struct no *prox, *ant;
}No;

typedef struct lista{
  No *inicio, *fim;
}Lista;
```

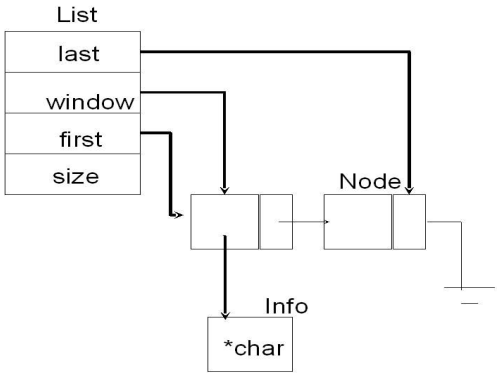
Implemente funções que:
a) Cria uma lista vazia.
b) Insere no início da lista.
c) Insere no final da lista.
d) Remove uma dada chave da lista.
e) Recebe duas listas, L1 e L2, e duas chaves, c1 e c2. Busca c1 na lista L1, busca c2 em L2. Caso as duas chaves existam nas respectivas listas, realiza a troca das chaves entre as listas. Ou seja, retira os nós que contém c1 e c2 de L1 e L2, respectivamente. Insere c1 em L2 na posição onde estava c2. Insere c2 em L1 na posição onde estava c1. Sua função deve ter complexidade linear
f) Calcule a função de complexidade de tempo da função do item anterior.

7. Considere uma lista simplesmente encadeada circular sem cabeça, conforme mostrado abaixo.



Esta lista mantém um ponteiro para o primeiro e último elemento da lista e não possui cabeça.
Implemente um programa que contém as seguintes funções:
a) Inserção no início da lista.
b) Remoção (free) do primeiro elemento da lista.
c) Inserção na última posição da lista.
d) Remoção (free) do último elemento da lista.
e) Remoção (free) de uma chave informada.
f) Impressão da lista.

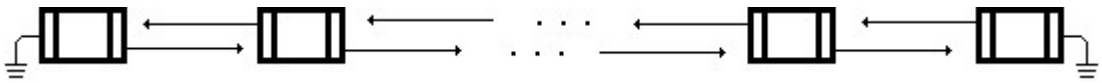
8. Considere o modelo TAD List mostrado abaixo:



Implemente os seguintes itens:
a) O TAD List;
b) void posList(List *lst, int pos): posiciona a janela (window) da lista no pos-ésimo elemento da lista.
c) Info infoList(List lst): retorna o elemento de informação da janela (window) eliminando-o da lista. O acesso aos elementos da lista **SOMENTE** pode ser feito através dessa função.
d) Considerando que o TAD List acima definido é modificado para manter uma lista duplamente encadeada. Altere o TAD e implemente a seguinte função:

-List ordenaList(List lst): ordena a lista em ordem crescente. Atenção, a troca de posição entre dois elementos deve ser feita via reapontamento de ponteiros.

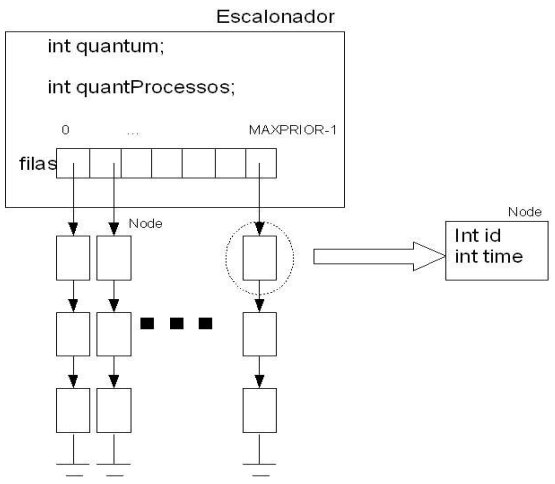
9. Considere o seguinte método para codificar mensagens de texto:
   • Primeira etapa- todas sequências não-vogais, incluindo o espaço em branco e os caracteres de pontuação são invertidas;
   • Segunda etapa- a mensagem inteira resultante é invertida dando origem à mensagem codificada.
   Exemplo: Dada a mensagem:
   ESTRUTURAS DE DADOS E MUITO LEGAL.
   Após a primeira etapa, teremos:
   ERTSUTURAD SED ADO SEM UITOL EGA.L
   E depois da segunda etapa teremos:
   L.AGE LOTIU MES ODA DES DARUTUSTRE
   Modele o TAD List mostrado abaixo e implemente uma função para decodificar as mensagens assim codificadas. Use uma lista duplamente ligada para representar a mensagem onde cada elemento contém um caractere no campo de informação.
   Assinatura da função: List* decodifica(List *lst)
   Modelo da lista:



   Sugestão: Implemente uma sub-função "List *inverte(List *p,List *q)" que inverte uma lista entre os limites apontados por p e q.

10. Codifique as seguintes implementações de filas (Queue)
    Implemente uma fila (queue) com duas pilhas (stack) e analise a complexidade das operações InsertQueue e RemoveQueue.

11. Em computadores multiprogramados, vários processos (programas) competem pelo uso da CPU na tentativa de executar suas operações. Nesse contexto, os SOs necessitam um elemento capaz de gerenciar o uso da CPU pelos processos. Esse elemento é chamado "**Escalonador**". O escalonador implementa uma política determinada de filas através de algoritmos que podem ser classificados segundo uma das seguintes categorias: 1) Lote; 2) Interativo ou 3) Tempo Real. Na classe dos escalonadores Interativos está o escalonador de "Multiplas Filas". A figura abaixo mostra um TAD para esse escalonador.



Esse modelo funciona da seguinte forma: o escalonador determina um quantum (slot de tempo), em milisegundos, que determina o tempo que cada processo pode utilizar a CPU, uma vez que lhe for dado o controle da mesma. Perceba que esse quantum pode ser suficiente ou não para executar um processo por completo. Isso depende do atributo time do processo id (identificador do processo), que indica o tempo de CPU necessário por processo para ser completado. Além disso, o escalonador mantém uma fila de prioridades para os processos, como mostrado acima. O escalonador funciona da seguinte forma: em cada rodada de execução do escalonador, todos os processos presentes no escalonador receberão um tempo de CPU, segundo a seguinte política: para a fila de prioridade 1 (menor prioridade), um quantum de CPU é dado a cada processo dessa fila; para a fila de prioridade 2, dois quanta de CPU são dados a cada processo dessa fila; para a fila de prioridade 3, 4 quanta de CPU são dados a cada processo dessa fila; para a fila de prioridade 4, 16 quanta de CPU são dados a cada processo dessa fila; e assim por diante, até a última prioridade (mais alta). Após processados, aqueles processos cujo time necessário for menor ou igual a zero devem ser retirados de suas filas. Considerando esse modelo de escalonador, implemente a seguinte função:

a) ***void roundScheduler(Escalonador *esc)***: executa uma rodada de uso da CPU para todos os processos de todas as prioridades, retirando de suas filas àqueles processos que exaurirem seu tempo de execução.

OBS: Considere que a seguinte função já foi implementada:

void executeProcessCPU(int id, int timeCPU): essa função determina a execução por parte da CPU do processo identificado por id por um tempo igual a timeCPU.

12. Considere a estrutura de dados ME (Matriz Esparsa) apresentada em sala de aula. Lembre-se que nela é usada uma célula com os seguintes campos:

a- lin: indice da linha da matriz;
b- col: indice da coluna da matriz;
c- lprox ou direita: apontador para a próxima célula da linha;
d- cprox ou abaixo: apontador para a próxima célula da coluna;
e- valor: valor do elemento contido na célula. Considere o valor como sendo um inteiro.

Implemente as seguintes funções:

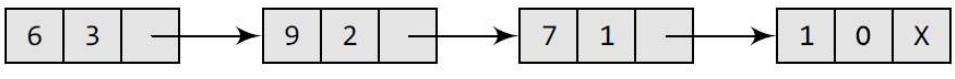
int produtoEscalarME(la,cb): função que recebe um apontador para a cabeça da lista de uma linha de uma matriz A e o apontador cb para a cabeça de lista de uma coluna de uma matriz B, e devolve o valor do produto escalar entre essa linha e essa coluna.

13. Considere uma lista duplamente encadeada circular e implemente um programa que contém as seguintes funções:

a) Inserção no início da lista.
b) Remoção (free) do primeiro elemento da lista.
c) Inserção na última posição da lista.
d) Remoção (free) do último elemento da lista.
e) Remoção (free) de uma chave informada.
f) Impressão da lista.

14. Considere que uma lista simplesmente ligada é usada para representar um polinômio, conforme o exemplo abaixo:

$$6x^3 + 9x^2 + 7x + 1.$$



Como se pode ver, cada termo individual do polinômio consiste de duas partes: 1) coeficiente e 2) a potência do termo.

Nesse exemplo, 6, 9, 7, and 1 são os coeficientes dos termos de potência 3, 2, 1, e 0, respectivamente.

Considerando o exposto e que a estrutura de dados usada para representar o polinômio é uma lista sem cabeça cuja estrutura é mostrada abaixo:

```
struct node
{
        int num;
        int coeff;
        struct node *next;
};
```

implemente as funções que realizam as seguintes operação sobre polinômios:

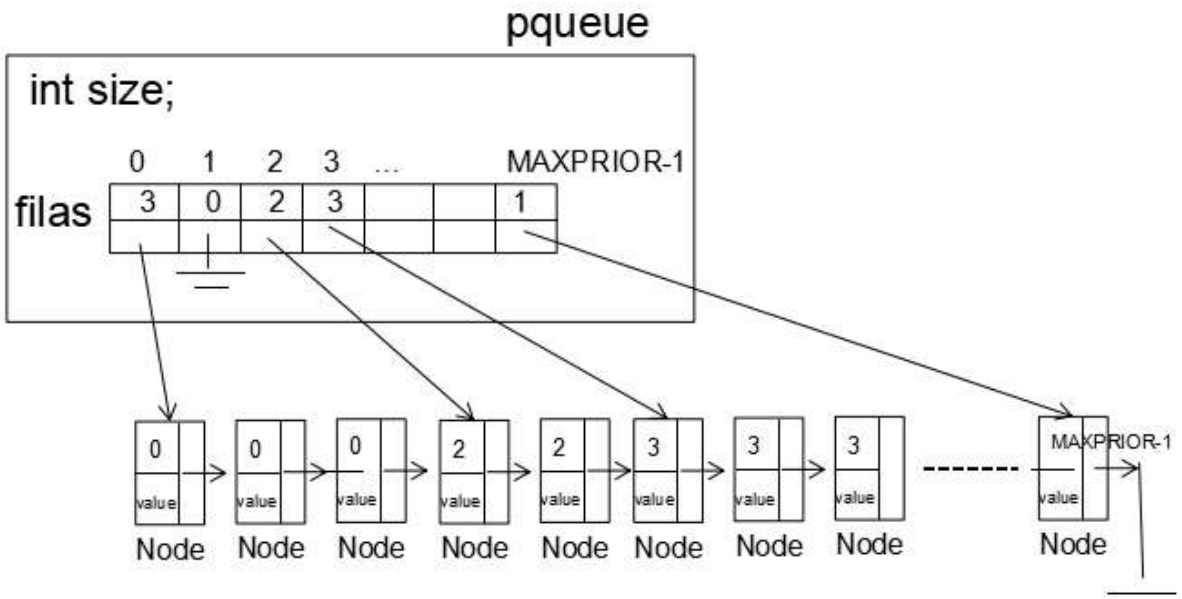
1) struct node *add_poly(struct node *start1, struct node *start2, struct node *start3): função que soma dois polinômios (start1 e start2) retornando o polinômio resultado (start3)

2) struct node *sub_poly(struct node *start1, struct node *start2, struct node *start3): função que subtrai dois polinômios (start1 e start2) retornando o polinômio resultado (start3)

1) struct node *mult_poly(struct node *start1, struct node *start2, struct node *start3): função que multiplica dois polinômios (start1 e start2) retornando o polinômio resultado (start3)

2) struct node *div_poly(struct node *start1, struct node *start2, struct node *start3): função que divide dois polinômios (start1 e start2) retornando o polinômio resultado (start3)

15. Considere uma implementação de uma fila de prioridades conforme a estrutura mostrada abaixo:



Considerando que esta fila de prioridades mantém um apontador no campo filas para cada sublista representando os elementos pertencentes às diferentes prioridades, implemente:

1) a estrutura de dados pqueue;

2) as seguintes funções

a) void insertPQueue(Pqueue *pq, int prior, int inf): função que insere um elemento de informação (inf) na fila (porque), em sua sublista definida pela prioridade (prior)

b) removePQueue(Pqueue *pq, int prior, int quant): função que retira uma quantidade (quant) de elementos de informação (inf) da fila (porque), removendo tais elementos da sublista definida pela prioridade (prior)

16.
17. uma lista duplamente encadeada circular e implemente um programa que contém as seguintes funções:
    a) Inserção no início da lista.
    b) Remoção (free) do primeiro elemento da lista.
    c) Inserção na última posição da lista.
    d) Remoção (free) do último elemento da lista.
    e) Remoção (free) de uma chave informada.
    f) Impressão da lista.
18. Consi