MINISTÉRIO DA EDUCAÇÃO
Universidade Federal do Piauí - UFPI
Campus Senador Helvídio Nunes de Barros - CSHNB
Curso de Sistemas de Informação
Disciplina: Estrutura de Dados I 2024.2

Lista Circular

1. Dada uma lista encadeada circular de elementos onde cada elemento tem uma posição inicial armazenada, escreva um programa que remove elementos de forma circular. O programa deve ler um número n, que representa a posição do elemento a ser removido em cada iteração (seguindo uma contagem circular). A remoção continua até que sobre apenas 1 elemento na lista. O programa deve então imprimir a posição inicial do último elemento que sobrou na lista.
   Especificações:
   - A lista deve ser implementada como uma lista encadeada circular.
   - O nó da lista pode ser uma struct.
   - A cada iteração, um elemento é removido da lista, seguindo a contagem circular de n elementos.
   - O programa deve continuar removendo até que reste apenas um elemento, e a posição inicial desse último elemento deve ser exibida.

   Entrada:
   LIsta Inicial: 10 → 20 → 30 → 40 → 50 → (volta para 1)
   Número n (posição de remoção): 2
   Iterações:
   Primeiro, remove o elemento **30** (Lista atual 10 → 20 → 30 → 40 → 50 → [volta para 1])
   Depois, remove o elemento **50** (Lista atual 10 → 20 → 30 → 40 → 50 → [volta para 1])
   Então, remove o elemento **20** (Lista atual 10 → 20 → 30 → 40 → 50 → [volta para 1])
   Finalmente, remove o elemento **10** (Lista atual 10 → 20 → 30 → 40 → 50 → [volta para 4])
   Saída:
   Posição inicial do último elemento: 4

2. Escreva um programa que leia um número n e gere uma lista encadeada circular contendo os n primeiros números primos. Cada nó da lista deve conter um número primo, e a lista deve ser circular. Após gerar a lista, percorra e imprima os números primos armazenados na lista.
   Entrada:
   Número de primos (n): 5
   Saída:
   Lista circular com os primeiros 5 números primos:
   2 → 3 → 5 → 7 → 11 → (volta para 2)

3. Escreva uma função que receba uma lista encadeada circular de nomes e inverta a ordem dos nomes na lista. A função deve garantir que, após a inversão, a lista continue sendo circular, ou seja, o último nome da lista deve apontar para o primeiro.
   Entrada:
   Lista inicial: Ana → João → Pedro → Maria → (volta para Ana)

Saída:
Lista invertida: Maria → Pedro → João → Ana → (volta para Maria)

4. Implemente uma função para remover todos os nós de uma lista encadeada circular que possuem um valor igual a X. A função deve garantir que a lista permaneça circular após as remoções. Caso o valor X não seja encontrado, exiba uma mensagem informando.
   Entrada:
   Lista inicial: 3 → 7 → 3 → 9 → 3 → (volta para 3)
   Valor a remover: 3
   Saída:
   Lista após remoção: 7 → 9 → (volta para 7)

5. Escreva uma função que remova todos os elementos duplicados de uma lista encadeada circular. A função deve garantir que, após a remoção das duplicatas, a lista continue sendo circular e ordenada da mesma forma. Considere que a lista terá pelo menos 2 valores diferentes, assim, para não acontecer de remover as duplicatas e a lista ficar vazia.
   Entrada:
   Lista inicial: 1 → 2 → 2 → 3 → 4 → 4 → (volta para 1)
   Saída:
   Lista após remoção de duplicatas: 1 → 2 → 3 → 4 → (volta para 1)

6. Crie uma função que insira um novo número em uma lista encadeada circular de forma ordenada. A lista já está ordenada em ordem crescente, e a função deve garantir que a nova inserção não quebre essa ordenação e que a lista continue circular.
   Entrada:
   Lista inicial: 10 → 20 → 30 → 40 → (volta para 10)
   Novo elemento: 25
   Saída:
   Lista após inserção ordenada: 10 → 20 → 25 → 30 → 40 → (volta para 10)

7. Escreva uma função que receba duas listas encadeadas circulares e intercale seus elementos. A primeira lista deve fornecer o primeiro elemento, seguida pelo primeiro elemento da segunda lista, depois o segundo elemento da primeira lista, e assim por diante. Ambas as listas podem ter tamanhos diferentes. A nova lista resultante deve ser circular.
   Entrada:
   Lista 1: 1 → 3 → 5 → (volta para 1)
   Lista 2: 2 → 4 → 6 → 8 → (volta para 2)
   Saída:
   Lista intercalada: 1 → 2 → 3 → 4 → 5 → 6 → 8 → (volta para 1)

8. Escreva uma função que verifique se todos os elementos de uma lista encadeada circular *Lista1* estão contidos em uma lista encadeada circular *Lista2*, mantendo a mesma sequência. A função deve retornar verdadeiro (ou imprimir "Sim") se Lista1

estiver contida em Lista2, e falso (ou imprimir "Não") caso contrário. As listas podem ter tamanhos diferentes e são circulares.
Entrada:
Lista1: 20 → 30 → (volta para 20)
Lista2: 10 → 20 → 30 → 40 → 50 → (volta para 10)
Saída:
Sim (Lista1 está contida em Lista2)

9. Implemente uma função que reorganize uma lista encadeada circular separando os números pares dos ímpares. Todos os números pares devem vir antes dos números ímpares, mantendo a ordem relativa dos elementos dentro de cada grupo. A lista deve permanecer circular após a reorganização.
Entrada:
Lista inicial: 3 → 8 → 5 → 10 → 1 → 6 → (volta para 3)
Saída:
Lista reorganizada: 8 → 10 → 6 → 3 → 5 → 1 → (volta para 8)

10. Você foi designado para implementar um sistema de gerenciamento de uma **playlist musical** usando uma **lista encadeada circular**. Cada música da playlist tem um título e um contador de quantas vezes ela foi reproduzida. O sistema deve permitir as seguintes operações complexas:

- **Inserir uma nova música na playlist**:
  - A inserção só deve acontecer se o título da música não estiver duplicado na lista (ou seja, títulos duplicados não são permitidos).
  - Caso o título já exista, aumente o contador de reproduções dessa música.
  - O sistema deve garantir que a lista continue circular após a inserção.
- **Remover músicas que foram reproduzidas menos de um certo número X de vezes**:
  - A função deve remover todas as músicas da playlist cujo contador de reproduções seja menor que um valor X fornecido pelo usuário.
  - Após a remoção, a lista deve continuar circular.
- **Imprimir a playlist**:
  - A função deve imprimir todos os títulos da playlist, junto com o número de vezes que cada música foi reproduzida, na ordem atual da lista.
- **Especificações da struct Música:**

```
struct Musica {
    char titulo[100];
    int reproducoes;
    struct Musica* prox;
};
```

OBS: IMPLEMENTE TODAS AS OPERAÇÕES EM FUNÇÕES, FAÇA UM MENU COM A OPERAÇÕES CITADAS ACIMA E FAÇA UMA FUNÇÃO MAIN QUE EXECUTE O ALGORITMO EM LOOP ATÉ QUE A OPÇÃO ”SAIR” DO MENU SEJA SELECIONADA.

Entrada:

Inserir músicas: "Shape of You", 10 reproduções
Inserir músicas: "Blinding Lights", 5 reproduções
Inserir músicas: "Shape of You", 15 reproduções
Inserir músicas: "Save Your Tears", 3 reproduções
Inserir músicas: "Levitating", 8 reproduções
Remover músicas com menos de **6** reproduções.
Imprimir playlist.
SAIR

Saída:
Musica "Shape of You" inserida … Música "Levitating" inserida… (Print com nome da música que foi inserida. Isso para cada inserção)
Músicas após remoção de músicas com menos de 6 reproduções:
1. Shape of You - 25 reproduções
2. Levitating - 8 reproduções
(volta para Shape of You)

Explicação do Exemplo:
A música "Shape of You" foi inserida duas vezes, logo a quantidade de reproduções vou somada. As demais músicas foram inseridas uma vez, então elas permanecem com suas respectivas reproduções. Ao remover todas com menos de 6 reproduções restou apenas "Shape of You" e "Levitating". Elas são organizadas na mesma ordem que foram inseridas.

ANEXO

Dica 1: Inserir Elementos de Forma Ordenada

Ao inserir elementos de forma ordenada em uma lista encadeada circular, você deve percorrer a lista e encontrar a posição correta onde o novo elemento deve ser inserido. Aqui está o processo básico:

1. Verificar se a lista está vazia: se estiver vazia, insira o novo elemento como o único nó e faça com que ele aponte para si mesmo (já que a lista é circular).
2. Percorrer a lista: se a lista não estiver vazia, comece do primeiro nó e percorra a lista até encontrar a posição onde o novo elemento deve ser inserido (ou seja, o ponto onde o valor atual é maior que o valor anterior).
3. Inserir no meio ou final: se o novo valor for maior que o atual nó, continue percorrendo a lista até encontrar o ponto correto. Certifique-se de ajustar os ponteiros para manter a lista circular.
4. Manter a circularidade: após inserir o novo nó, garanta que o último nó continue apontando para o primeiro nó da lista.

Exemplo: Ao inserir "Beatriz" na lista circular "Ana → João → Pedro", você deve percorrer a lista até encontrar a posição correta (entre "Ana" e "João") e ajustar os ponteiros para manter a circularidade.

Dica 2: Comparando e Ordenando Nomes (Strings)

Para ordenar nomes (strings) em uma lista encadeada circular, você precisa comparar as strings em ordem alfabética. Em C, isso é feito utilizando a função strcmp() da biblioteca <string.h>:

1. **Usando strcmp()**: A função strcmp() compara duas strings. Se o valor de retorno for:
    - Menor que 0: A primeira string é menor (vem antes na ordem alfabética).

    - Igual a 0: As duas strings são iguais.
    - Maior que 0: A primeira string é maior (vem depois na ordem alfabética).
2. **Percorrer a lista:** Para inserir um novo nome em uma lista circular, percorra a lista e utilize strcmp() para comparar o nome atual com o nome que deseja inserir.
3. **Inserção ordenada:** Se strcmp() indicar que o novo nome vem antes do nome do nó atual (retorno menor que 0), insira o novo nome antes do nó atual. Caso contrário, continue percorrendo a lista até encontrar a posição correta.

Exemplo: Ao inserir o nome "Beatriz" na lista circular "Ana → João → Pedro", você deve usar strcmp() para comparar "Beatriz" com "Ana", "João", e "Pedro". Como "Beatriz" vem depois de "Ana", mas antes de "João", ela será inserida entre eles, mantendo a ordem alfabética.