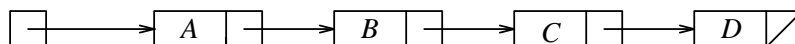
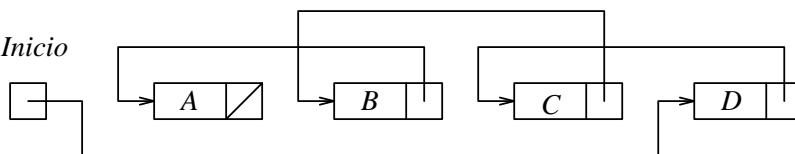


MAC 121 – Algoritmos e Estruturas de Dados I

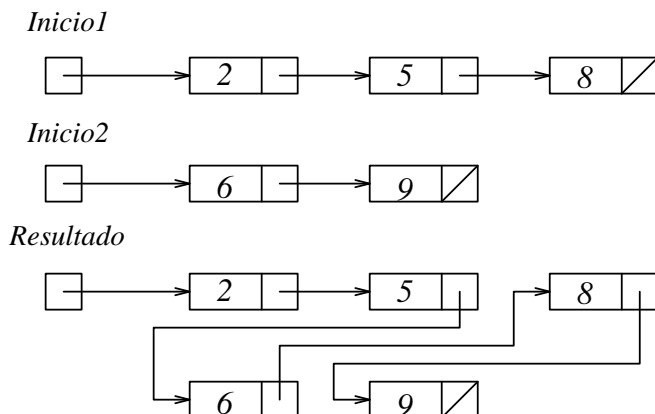
Segundo semestre de 2018

Lista de exercícios – Listas Ligadas.

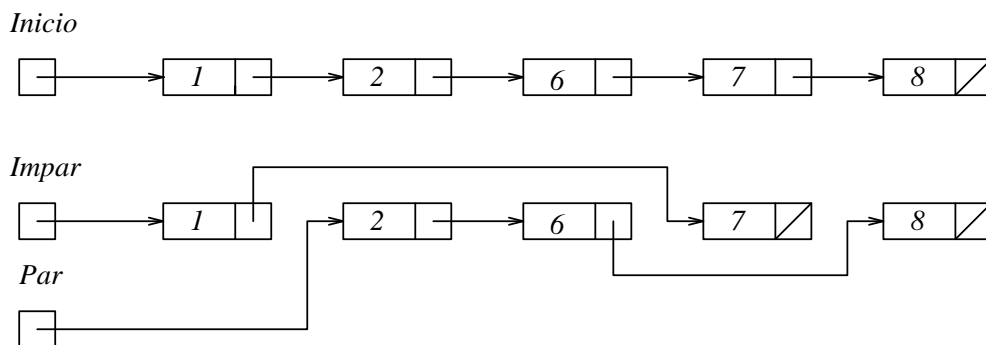
1. Escreva uma função que recebe uma lista ligada ordenada e remove da lista os elementos repetidos, deixando apenas uma cópia de cada elemento.
2. Escreva funções para as seguintes operações:
 - (a) verifica se um dado x ocorre numa lista ligada;
 - (b) acrescenta um elemento no fim de uma lista ligada;
 - (c) imprima os conteúdos de todos os elementos de uma lista ligada.
3. Faça uma função que devolve um apontador para o elemento do meio de uma lista ligada (se o número de elementos da lista for par, devolve o $\frac{n}{2}$ -ésimo elemento) **sem contar o número de elementos da lista**.
4. Discuta vantagens e desvantagens de vetores (arrays) em relação a listas ligadas. Dê atenção especial às questões de quantidade de memória, velocidade de inserção, remoção e acesso.
5. Escreva uma função que inverte uma lista ligada dada (o primeiro elemento da nova lista é o último da lista dada, o segundo é o penúltimo da lista dada, e assim por diante. Faça manipulando apenas os apontadores. Exemplo:

Inicio*Inicio*

6. Escreva uma função para intercalar duas listas ligadas cujas informações estão arranjadas em ordem crescente. Exemplo:



7. Dada uma lista ligada escreva uma função que transforma a lista dada em duas listas ligadas: a primeira contendo os elementos cujos conteúdos são pares e a segunda com os elementos cujos conteúdos são ímpares. Sua função deve manipular somente os apontadores e **não** o conteúdo das células (i.e. não vale ficar copiando os conteúdo de um lado para o outro, só vale alterar os apontadores). Exemplo:



8. Dizemos que uma lista ligada é **circular** se o último elemento aponta para o primeiro. Uma lista ligada circular normalmente é referenciada com um apontador para o último elemento (que aponta, então, para o primeiro). Considere uma lista ligada circular, apontada por **fim**, como descrito acima, e um elemento **x** e faça uma função que remove da lista todas as ocorrências de **x** (cuidado para não entrar em loop...).
9. Dizemos que uma lista é **duplamente ligada** se cada elemento tem um apontador para o próximo elemento da lista e um apontador para o elemento anterior.
- Escreva a definição de tipos de uma lista duplamente ligada.
 - Mostre como fica uma lista duplamente ligada com um único elemento
 - Como ficaria uma lista duplamente ligada circular?
 - Escreva funções para inserção e remoção de elementos em uma lista duplamente ligada circular.

10. Dizemos que uma lista ligada (duplamente ligada, circular) tem **cabeça de lista** se ela tem um elemento extra que está presente mesmo quando a lista não tem elementos válidos. Escreva funções de inserção e remoção para:
 - a. listas ligadas com cabeça de lista.
 - b. listas ligadas circulares com cabeça de lista.
 - c. listas duplamente ligadas com cabeça de lista.
 - d. listas duplamente ligadas circulares com cabeça de lista.
11. Faça o exercício 8 considerando, agora, uma lista ligada circular com cabeça de lista.
12. Faça uma implementação de pilhas (**empilha**, **desempilha**, **pilhaVazia**, etc) usando:
 - a. listas ligadas;
 - b. listas ligadas circulares;
 - c. listas duplamente ligadas;
 - d. listas duplamente ligadas circulares com cabeça de lista.
13. Faça o exercício anterior implementando agora filas.
14. Considere uma implementação de polinômios esparsos usando listas ligadas circulares com cabeça de lista.
 - a. Faça uma função que recebe dois polinômios e devolve a soma dos dois;
 - b. Faça uma função que recebe um polinômio e devolve o polinômio derivada;
 - c. Faça uma função que recebe um polinômio $p(x)$ e um real a e devolve o polinômio obtido do quociente de $p(x)$ por $x - a$ e devolve também um real r , que é o resto desta divisão. (Dica: algoritmo de Briot-Ruffini)
15. Podemos representar uma matriz esparsa $A_{m \times n}$ usando a seguinte estrutura:

```
typedef struct cel {
    int lin, col;
    double coef;
    struct cel *proxNaMesmaLin, *antNaMesmaLin,
               *proxNaMesmaCol, *antNaMesmaCol;
} celula;
typedef celula * apontador;

typedef struct {
    int m, /* número de linhas */
        n, /* número de colunas */
        r; /* número de elementos não nulos*/
    apontador *vetorDeLinhas, *vetorDeColunas;
} matrizEsparsa;
```

Ou seja, cada elemento não nulo da matriz está em uma lista duplamente ligada circular com cabeça de lista dos elementos na mesma linha (cujas cabeças estão em **vetorDeLinhas**) e em

uma lista ligada circular com cabeça de lista dos elementos na mesma coluna.

a. Desenhe a estrutura correspondente à matriz esparsa $A_{10 \times 10}$ abaixo:

$$\begin{pmatrix} 0 & 0 & -1 & 0 & 0 & 3 & 0 & 0 & 1 & 13 \\ 7 & 0 & 0 & 0 & 0 & 0 & -8 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & -5 & 0 & 0 \\ 0 & -3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 23 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 19 & 0 \\ 0 & 0 & 0 & 0 & 12 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

b. Faça uma função `criaMatrizEsparsa()` que leia m , n e r elementos não nulos de uma matriz esparsa e constroi a estrutura conforme descrito acima.

c. Faça uma função que recebe duas matrizes esparsas A e B e devolve a matriz esparsa dada pela soma das matrizes.