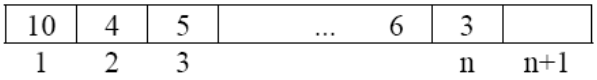
EXERCÍCIOS SOBRE ESTRUTURAS DE DADOS

Filas e Pilhas implementadas com vetores:

1. Construa uma função **retire_cinco(p)** que retire os 5 primeiros elementos da fila F e devolva um valor lógico que indica se foi possível concluir a operação sem que a lista fique vazia.
2. Construa uma função que determine o número de elementos existentes numa fila **F**.
3. Desenvolva uma função **ultimo(F)** que devolva o último elemento da fila **F**.
4. Escreva um procedimento **insere_vetor(vet,x)** para inserir o elemento **x** na 2ª posição do vetor **vet**. Considere que o vetor possui **n** elementos e que, inicialmente, foi reservado espaço, na declaração, para **n+1** elementos.



Estruturas de Dados Lineares implementados com estruturas ligadas (encadeadas):

5. Construa uma função **concatenar(L1,L2)** que faça a concatenação de duas listas, sendo L1 e L2 os apontadores para os seus primeiros elementos.

6. Construa uma nova versão do exercício anterior em que tem que criar uma cópia das duas listas concatenadas.

7. Escreva uma função **pertence(L,x)** que indica se o elemento **x** pertence à lista encadeada iniciada por **L.**

8. Faça uma rotina que inverta a ordem dos elementos de uma lista encadeada.

9. Escreva uma função que devolva o último elemento de uma lista.

10. Faça um procedimento **elimina(L,n)** que apaga o nó da lista colocado na posição **n**.

11. Faça um procedimento **elimina(L,n)** que apaga o nó da lista **L** cujo campo de informação é x (por exemplo do tipo inteiro).

12. Construa um procedimento que apague uma lista duplamente encadeada da memória, sendo fornecido apenas o apontador para o início da lista. A memória ocupada pelos nós da lista deve ser totalmente libertada.

13. Faça uma função que combine duas listas encadeadas ordenadas numa lista única também ordenada.

14. Descreva resumidamente as vantagens e desvantagens dos vetores relativamente às listas encadeadas.

15. Escreva outro procedimento **insere_L(L,x)** para inserir o elemento **x** na 2ª posição da lista encadeada simples em que L é o apontador para o 1º nó.

16. Construa um procedimento que recebe dois apontadores para o início de duas listas encadeadas ordenadas de números inteiros e cria uma nova lista, também ordenada, que é a intersecção das duas, isto é, a nova lista deve conter apenas os elementos que estiverem simultaneamente em ambas as listas.

Exercícios Pilhas

1. Construa uma função **elemento(P,n)** que devolva o elemento da pilha colocado na posição **n**, a contar do topo.

2. Construa uma função **pop_cinco(P)** que retire os 5 primeiros elementos da pilha P e devolva um valor lógico que indica se foi possível concluir a operação sem que a pilha fique vazia

3..Construa um procedimento **inverte(x,n)**, que, com o auxílio de uma pilha, inverte um vetor x que contem n números reais.

4. Dada uma pilha construa uma rotina que altere o elemento situado na posição **n** (a contar do topo) para o valor **x**. No caso de a pilha ter menos de **n** elementos deve ser alterado o último. Os restantes elementos da pilha não devem ser alterados, apesar de poderem ser movimentados. No caso de ser necessário pode ser utilizada uma estrutura auxiliar.

Exercícios Filas

1. Construa uma função **retire_cinco(P)** que retire os 5 primeiros elementos da fila F e devolva um valor lógico que indica se foi possível concluir a operação sem que a lista fique vazia
2. Construa uma função que determine o número de elementos existentes numa fila **F**.
3. Desenvolva uma função **ultimo(F)** que devolva o último elemento da fila F.

Exercícios Listas Encadeadas

1. Construa uma função **concatenar(L1,L2)** que faça a concatenação de duas listas, sendo L1 e L2 os apontadores para os seus primeiros elementos.

2. Escreva uma função **pertence(L,x)** que indica se o **x** pertence à lista encadeada iniciada por **L.**

3. Faça uma rotina que inverta a ordem dos elementos de uma lista encadeada.

4. Escreva uma função que devolva o último elemento de uma lista.

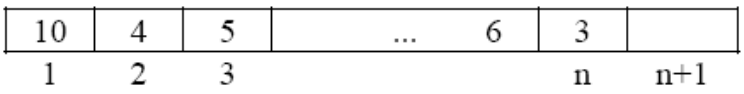
5. Faça um procedimento **elimina(L,n)** que apaga o nó da lista colocado na posição **n**.

6. Construa um procedimento que apague uma lista duplamente encadeada da memória, sendo fornecido apenas o apontador para o início da lista. A memória ocupada pelos nós da lista deve ser totalmente libertada.

7. Faça uma função que combine duas listas encadeadas ordenadas numa lista única também ordenada.

8. Descreva resumidamente as vantagens e desvantagens dos vetores relativamente às listas encadeadas.

9. Escreva um procedimento **insere_vetor(vet,x)** para inserir o elemento x na 2ª posição do vetor **vet**. Considere que o vetor possui **n** elementos e que, inicialmente, foi reservado espaço, na declaração, para **n+1** elementos.



10. Escreva outro procedimento **insere_L(L,x)** para inserir o elemento x na 2ª posição da lista encadeada simples em que L é o apontador para o 1º nó.

11. Construa um procedimento que recebe dois apontadores para o início de duas listas encadeadas ordenadas de números inteiros e cria uma nova lista, também ordenada, que é a intersecção das duas, isto é, a nova lista deve conter apenas os elementos que estiverem simultaneamente em ambas as listas.