Estácio

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

ESTRUTURA DE DADOS

AULA 8

RICARDO EIJI KONDO, Me.

TÓPICOS

UNIDADE IV - Listas, Pilhas e Filas

4.1 - Fundamentos dos Tipos Abstratos de Dados (TAD)

4.1.1 - TADs Lineares (pilhas, filas, listas, sequencias, dicionários, tabelas de hash, filas de prioridade, etc.)

4.1.2 - TADs Não-Lineares: (árvores e grafos)

4.2 - Lista

4.2.1 - Listas Sequenciais

4.2.2 - Listas Encadeadas

4.2.2.1 - Listas Simplesmente Encadeadas

4.2.2.2 - Listas Duplamente Encadeadas

4.2.2.3 - Listas Circulares Simples

4.3 - Pilha

4.4 - Fila

LISTA LINEARES

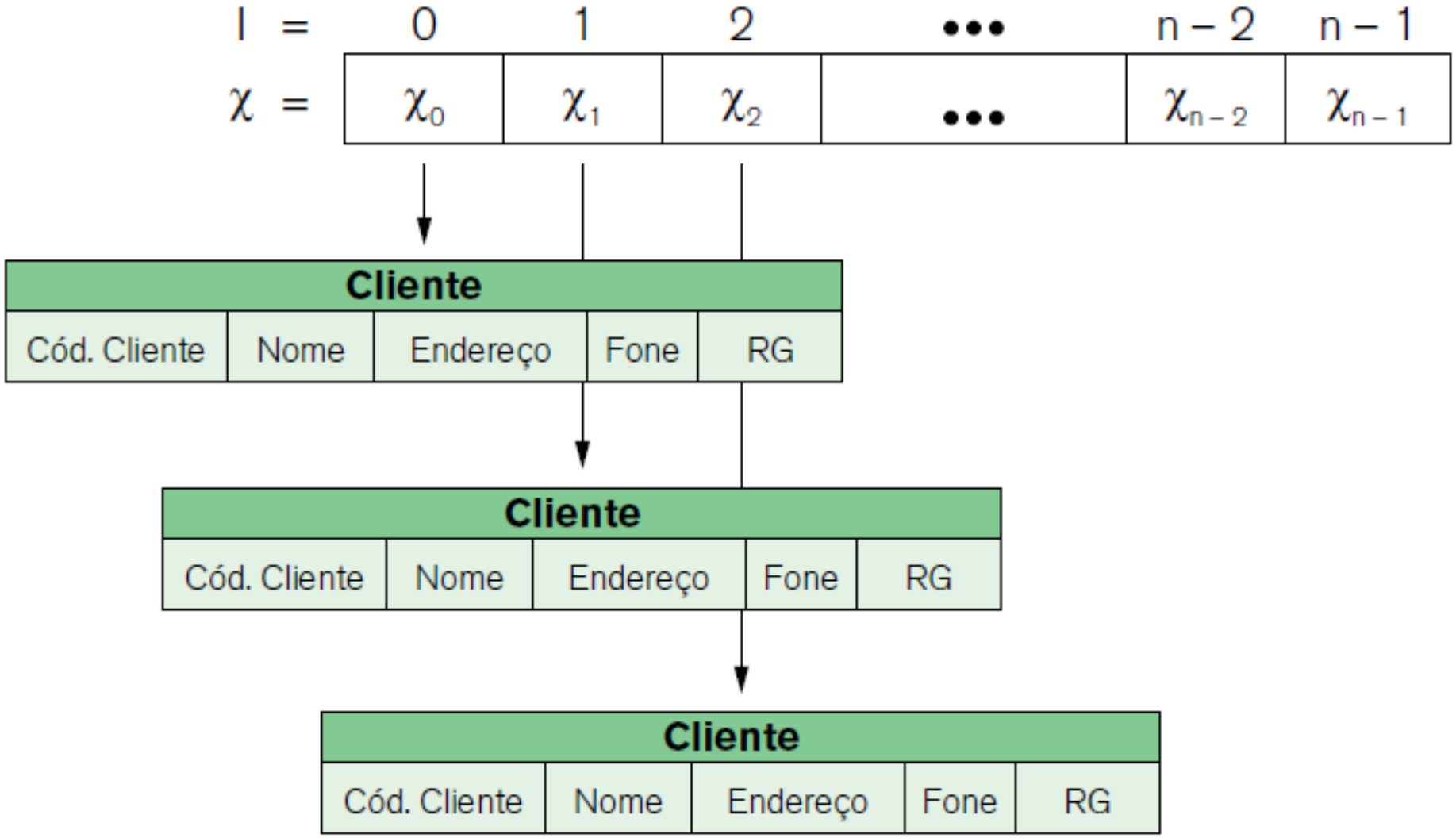
- Listas lineares são estruturas de dados que têm como objetivo armazenar um conjunto de dados, que de alguma forma se relacionam, com os elementos dispostos em sequência.

- As listas lineares podem ser representadas quanto a sua forma de armazenamento, de duas formas: sequencial (contígua) ou encadeada.

LISTA LINEAR SEQUENCIAL

- A forma contígua é considerada a maneira mais simples de armazenar uma estrutura de lista na memória. Isto porque os elementos da lista ocupam posições consecutivas na memória do computador, ou seja, um dado após o outro.
- Dessa forma temos uma estrutura estática, sendo assim a alocação de memória é feita durante a compilação do programa. Consequentemente, deve-se pré-determinar a quantidade máxima de elementos da lista.
- As listas lineares sequenciais são ideais para conjunto pequeno de dados. Os dados são armazenados nos nós da lista como primitivos ou compostos. Cada nó da lista, geralmente possui um identificador distinto chamado chave.

LISTA LINEAR SEQUENCIAL



LISTA LINEAR SEQUENCIAL

Vantagens da lista linear sequencial:

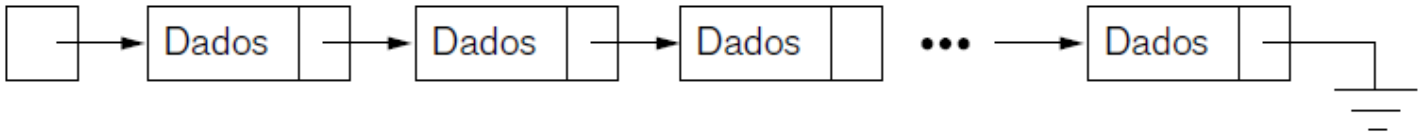
- Qualquer elemento da lista pode ser acessado diretamente.
- Tempo constante para acesso a qualquer elemento da lista.

Desvantagem da lista linear sequencial:

- Movimentação de todos os elementos da lista quando há uma inserção ou remoção de elemento.
- O tamanho máximo da lista deve ser pré-determinado.

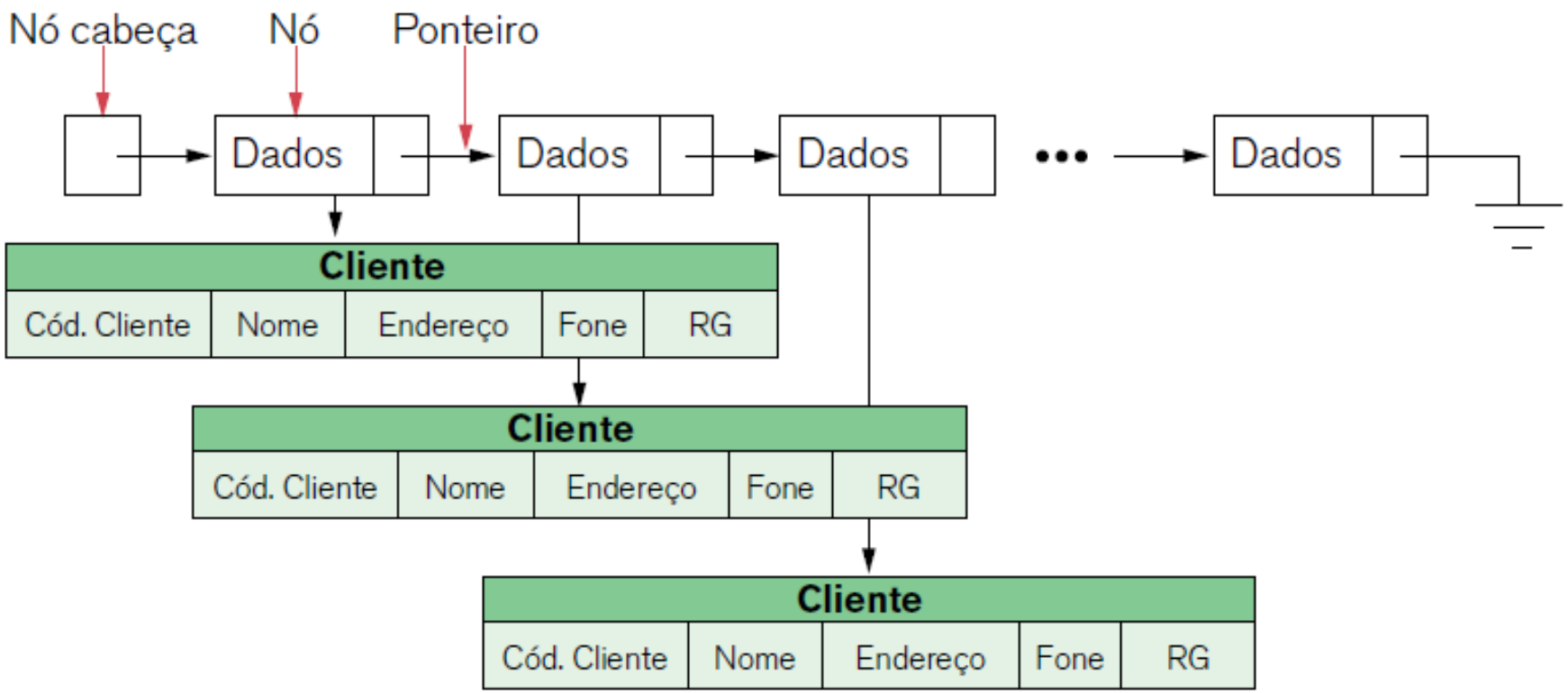
LISTA LINEAR ENCADEADA

- Nas listas lineares encadeadas não há a necessidade de pré-determinar a quantidade máxima de elementos da lista. Os elementos não são armazenados de forma contígua e ocupam qualquer posição de memória disponível.



- A ligação entre os elementos dispersos na memória são feitos através de ponteiros. Ou seja, cada nó, deve conter, além dos registros, uma referência para o próximo nó da lista.
- A quantidade máxima de elementos é determinada pela quantidade de memória disponível que pode ser alocada pelo programa. Desta forma temos uma estrutura dinâmica, ou seja, a alocação de memória é feita durante a execução do programa.

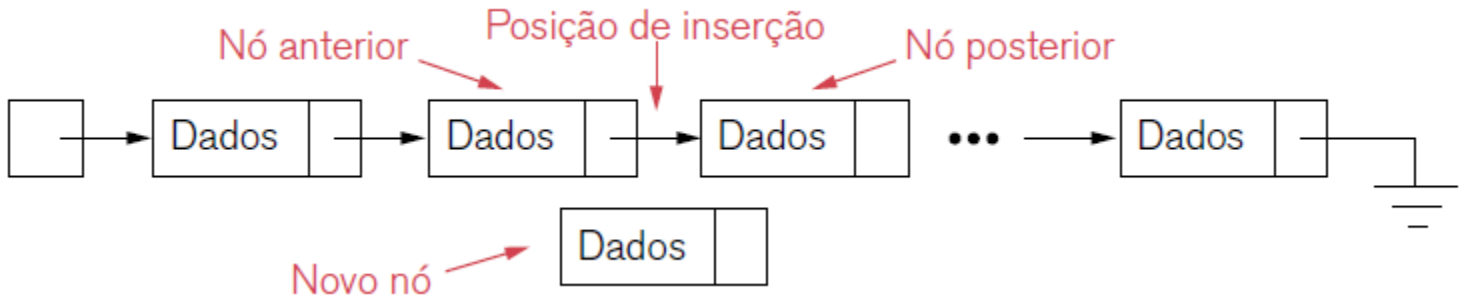
LISTA LINEAR ENCADEADA



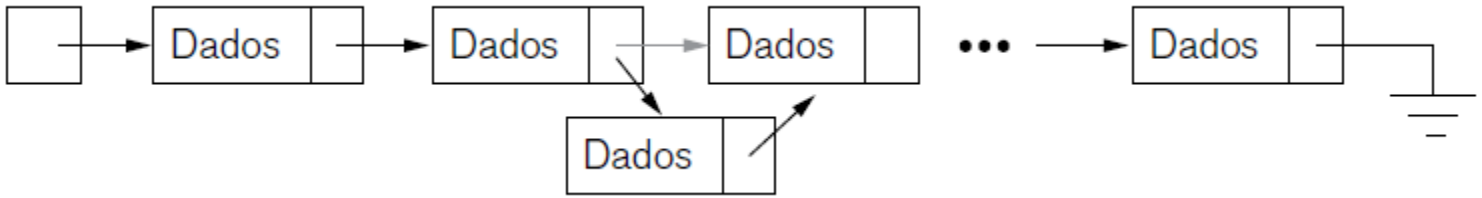
- A lista possui um nó especial, chamado nó-cabeça. O nó-cabeça indica o início da lista encadeada e nunca pode ser removido. Dados (registros) que são armazenados nos demais nós da lista, não devem ser armazenados no nó-cabeça.

Exemplo de inserção:

- A partir do nó-cabeça, efetua-se a busca da posição onde o novo nó deverá ser inserido.



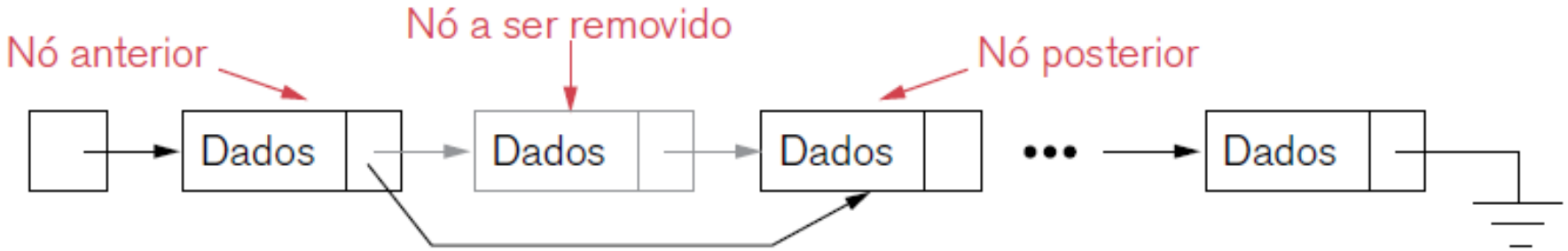
- Uma vez encontrado o local de inserção, liga-se o ponteiro do nó anterior ao novo nó e liga-se o ponteiro no novo nó ao nó posterior.



LISTA LINEAR ENCADEADA

Exemplo de remoção:

- A partir do nó-cabeça, efetua-se a busca do nó a ser removido.
- Uma vez encontrado nó a ser removido, liga-se o ponteiro do nó anterior ao nó posterior.



LISTA LINEAR ENCADEADA

Vantagens da lista linear encadeada:

- Facilidade de inserir ou remover um elemento em qualquer ponto da lista.
- Não há a necessidade de movimentar os elementos da lista quando há uma inserção ou remoção de elemento.

Desvantagem da lista linear encadeada :

- Por utilizar ponteiros, a implementação deve ser feita com cuidado para que não ocorra um mal encadeamento (ligação) dos nós e consequentemente a lista seja perdida.
- Encontrar um determinado elemento na posição n da lista é necessário percorrer os n-1 anteriores.
- Necessidade de memória extra para armazenamento dos ponteiros.

UTILIZAÇÃO DA LISTA LINEAR SEQUENCIAL

Algumas operações com listas sequenciais:

- **Criar:** cria uma lista vazia.
- **Verificar lista vazia:** verifica se há algum elemento na lista.
- **Verificar lista cheia:** verifica se a lista esta cheia.
- **Inserir:** insere um elemento numa determinada posição ou no final da lista.
- **Alterar:** alterar algum elemento da lista.
- **Remover:** remove um elemento de uma determinada posição.
- **Buscar:** acessa um elemento da lista.
- **Exibir a quantidade:** retorna a quantidade de elementos da lista.
- **Combinar:** combina duas ou mais listas em uma única.
- **Dividir lista:** dividi uma lista em duas ou mais.
- **Ordenar:** ordena os elementos da lista de acordo com algum de seus componentes.
- **Esvaziar:** esvaziar a lista

UTILIZAÇÃO DA LISTA LINEAR SEQUENCIAL

- Criação de uma lista vazia

```
#include<iostream>
using namespace std;

# define MAX_LISTA 10 // Tamanho Máximo da Lista

int main(){
    float ListaNotas[MAX_LISTA]; // Lista Linear Sequencial
    int PosUltimoElemLista = 0; // Qtde de elementos da lista
    bool Ret; // Recebe o retorno da função chamada
}
```

UTILIZAÇÃO DA LISTA LINEAR SEQUENCIAL

- Inserir elemento na lista

```cpp
#include<iostream>
using namespace std;

# define MAX_LISTA 5                // Tamanho Máximo da Lista

bool Inserir(float Lista[], float valor, int &PosElemLista) {
    if (PosElemLista == MAX_LISTA){
        cout << "ERRO: Lista cheia.";
        return false;
    }
    else {
        Lista[PosElemLista] = valor;
        PosElemLista++;             //incrementa posicao atual
    }
    return true;
}
```

```cpp
int main(){
    float ListaNotas[MAX_LISTA];    // Lista Linear Sequencial
    int PosElemLista = 0;           // Posicao do elemento da lista
    bool Ret;                       // Recebe o retorno da função chamada
    int valor;

    for (int i=0;i<10;i++){
        cout << "Digite o valor para insercao: ";
        cin >> valor;
        Ret = Inserir(ListaNotas, valor, PosElemLista);

        if(Ret == true){
            cout << "Insercao efetuada com sucesso!" << endl;
        }
    }
}
```

UTILIZAÇÃO DA LISTA LINEAR SEQUENCIAL

- Inserir elemento em posição determinada e exibir

```
#include<iostream>
using namespace std;
bool Exibir(float Lista[]);

# define MAX_LISTA 5                    // Tamanho Máximo da Lista

bool InserirPos(float Lista[], int PosIns, float valor) {
    for(int ind = MAX_LISTA; ind > PosIns; ind-- ){
        Lista[ind] = Lista[ind-1];
    }
    Lista[PosIns] = valor;
    Exibir(Lista);
    return true;
}

bool Exibir(float Lista[]){
    for(int ind = 0; ind < MAX_LISTA; ind++ ){
        cout << "Nota " << ind << ": " << Lista[ind] <<endl;
    }
    return true;
}
```

```
int main(){
    float ListaNotas[MAX_LISTA];     // Lista Linear Sequencial
    bool Ret;                        // Recebe o retorno da função chamada
    int valor, posicao;

    for (int i=0;i<MAX_LISTA;i++){
        cout << "Digite o valor para insercao: ";
        cin >> valor;
        cout << "Digite a posicao para insercao: ";
        cin >> posicao;
        Ret = InserirPos(ListaNotas, posicao, valor);

        if(Ret == true){
            cout << "Insercao efetuada com sucesso!" << endl;
        }
    }
    Ret = Exibir(ListaNotas);
    if(Ret == false){
        cout << "Não foi possível exibir a lista." << endl;
    }
}
```

UTILIZAÇÃO DA LISTA LINEAR SEQUENCIAL

- Pesquisar elemento e retornar sua posição

```
#include<iostream>
using namespace std;

# define MAX_LISTA 5                    // Tamanho Máximo da Lista

int Pesquisar(float Lista[], float valor) {
    for(int ind = 0; ind < MAX_LISTA; ind++ ){
        if (Lista[ind] == valor){
        return ind;
        }
    }
return -1;
}

bool Exibir(float Lista[]){
    for(int ind = 0; ind < MAX_LISTA; ind++ ){
        cout << "Nota " << ind << ": " << Lista[ind] <<endl;
    }
    return true;
}

bool InserirPos(float Lista[], int PosIns, float valor) {
    for(int ind = MAX_LISTA; ind > PosIns; ind-- ){
        Lista[ind] = Lista[ind-1];
    }
    Lista[PosIns] = valor;
    Exibir(Lista);
    return true;
}
```

```
int main(){
    float ListaNotas[MAX_LISTA];    // Lista Linear Sequencial
    bool Ret;                       // Recebe o retorno da função chamada
    int Pos;
    int valor, posicao;

    for (int i=0;i<MAX_LISTA;i++){
        cout << "Digite o valor para insercao: ";
        cin >> valor;
        cout << "Digite a posicao para insercao: ";
        cin >> posicao;
        Ret = InserirPos(ListaNotas, posicao, valor);

        if(Ret == true){
            cout << "Insercao efetuada com sucesso!" << endl;
        }
    }
    Exibir(ListaNotas);
    cout << "Digite o valor para busca: ";
    cin >> valor;
    Pos = Pesquisar(ListaNotas, valor);
    cout << "Valor encontrado na posicao " << Pos<<endl;
}
```

UTILIZAÇÃO DA LISTA LINEAR SEQUENCIAL

- Remover elemento e de uma posição

```
#include<iostream>
using namespace std;

# define MAX_LISTA 5                    // Tamanho Máximo da Lista
int Pesquisar(float Lista[], float valor) {
    for(int ind = 0; ind < MAX_LISTA; ind++ ){
        if (Lista[ind] == valor){
        return ind;
        }
    }
return -1;
}

bool RemoverElem(float Lista[], int valor) {
    int PosRem = Pesquisar(Lista, valor);
    for(int ind = PosRem; ind < MAX_LISTA; ind++ ){
        Lista[ind] = Lista[ind+1];
    }
    return true;
}

bool Exibir(float Lista[]){
    for(int ind = 0; ind < MAX_LISTA; ind++ ){
        cout << "Nota " << ind << ": " << Lista[ind] <<endl;
    }
    return true;
}

bool InserirPos(float Lista[], int PosIns, float valor) {
    for(int ind = MAX_LISTA; ind > PosIns; ind-- ){
        Lista[ind] = Lista[ind-1];
    }
    Lista[PosIns] = valor;
    //Exibir(Lista);
    return true;
}
```

```
int main(){
    float ListaNotas[MAX_LISTA];     // Lista Linear Sequencial
    bool Ret;                        // Recebe o retorno da função chamada
    int Pos;
    int valor, posicao;

    for (int i=0;i<MAX_LISTA;i++){
        cout << "Digite o valor para insercao: ";
        cin >> valor;
        cout << "Digite a posicao para insercao: ";
        cin >> posicao;
        Ret = InserirPos(ListaNotas, posicao, valor);

        if(Ret == true){
            cout << "Insercao efetuada com sucesso!" << endl;
        }
    }
    Exibir(ListaNotas);
    cout << "Digite o valor para remocao: ";
    cin >> valor;
    Ret = RemoverElem(ListaNotas, valor);
        if(Ret == true){
            cout << "Remocao efetuada com sucesso!" << endl;
        }
    Exibir(ListaNotas);
}
```

EXERCÍCIOS

- Refaça os mesmos exemplos utilizando uma estrutura de dados heterogênea, cos os seguintes campos:

```
struct DADOS_ALUNO{
    int CodAluno;
    char Nome[100];
    int Turma;
};
```