Lista Encadeada

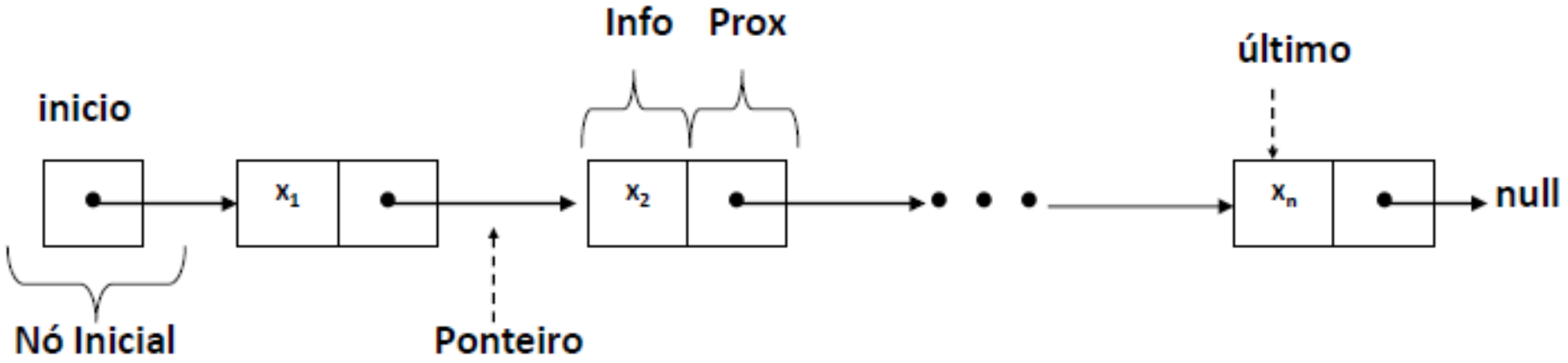
Centro Federal de Educação Tecnológica Celso Suckow da Fonseca
CEFET-RJ

Lista Encadeada

- **Recomendáveis quando o número de elementos de dados não podem ser previstos.**
- **São listas dinâmicas**
  - **comprimento da lista pode aumentar ou diminuir quando necessário.**
- **Vantagem**
  - **inserção ou remoção de um elemento em qualquer ponto da lista linear. Não é necessário movimentar nenhum elemento, basta fazer a devida atualização no campo do nó que contém o endereço apontado para o próximo elemento.**
- **Desvantagem**
  - **quando é necessário manipular um elemento específico da lista linear, diferentemente do grupamento sequencial, neste será necessário percorrer todos os elementos anteriores.**

Lista Encadeada

- **Representação gráfica de uma lista encadeada:**



  - **Apesar de fisicamente os nós NÃO estarem armazenados continuamente na memória, logicamente parecem estar contíguos.**

- **Pilhas e filas são estruturas lineares, mas são versões limitadas das listas encadeadas.**
- **Árvores são estruturas não-lineares de dados.**

Exemplo #1 – Função sem passagem de Parâmetro

```
#include <iostream>
//#include <stdlib.h>  - Para usar com a função malloc()
using namespace std;

struct no {
int info;
struct no *prox;
};
typedef struct no *noPtr;

noPtr inicio = NULL;
//Escopo das funções
int menu();
void insere();
void retira();
void listar();
bool listaVazia();
```

Exemplo #1 – continuação

```
main() {
    int op;
do {
    op = menu();
    switch (op) {
        case 1: insere(); break;
        case 2: retira(); break;
        case 3: listar(); break;
    }
    } while (op != 0);
}
```

```
int menu()
{
    int opcao;
    cout << "\n1: Insere elemento na lista" << endl;
    cout << "2: Retira elemento da lista" << endl;
    cout << "3: Listar elementos" << endl;
    cout << "0: Sair" << endl;
    cout << "\nDigite a opcao (0 - 3): ";
    cin >> opcao;
    return opcao;
}
```

```
bool listaVazia ()
{
    if (inicio)
        return false;
    else
        return true;
}
```

Exemplo #1 – continuação

```
void insere ()
{
    noPtr p;
    int x;

    p = new no;          //p = (noPtr) malloc(sizeof(struct no));
    cout << "\nDigite o valor do elemento: ";
    cin >> x;
    p -> info = x;
    p -> prox = inicio;
    inicio = p;
}
```

Exemplo #1 – continuação

```
void retira () {
    noPtr p;
    if (!listaVazia())
    {
        p = inicio;
        inicio = p -> prox;
        delete(p);        //free(p);   - Usado com malloc
        cout << "\nO elemento foi retirado!" << endl;
    }
    else
        cout << "\nLista Vazia!" << endl;
}
```

Exemplo #1 – continuação

```
void listar() {
    noPtr p;
    p = inicio;
    if (!listaVazia())
    {
        cout << "\nOs elementos da lista sao:" << endl;
        while (p != NULL)
        {
            cout << p->info << endl;
            p = p -> prox;
        }
    }
    else  cout << "\nLista Vazia!" << endl;
}
```

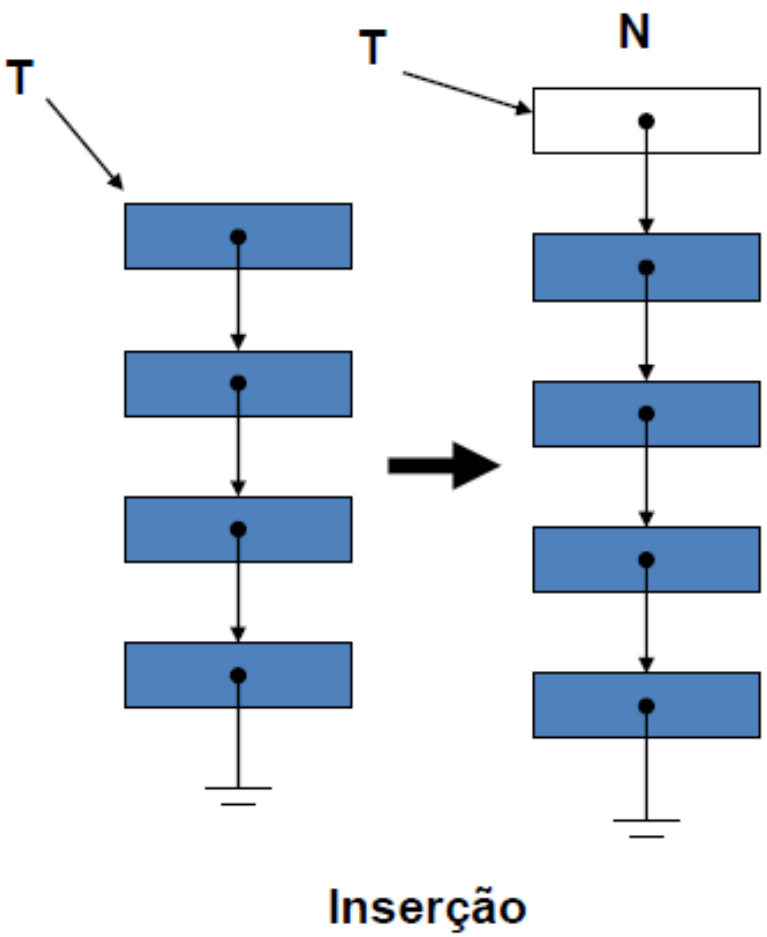
Lista Encadeada - Políticas

- **A maneira como é feita a inserção e a remoção dos dados na lista, indica o tipo de política que está sendo empregado.**
  - **Pilhas (*Stacks*) - LIFO (*last-in, first-out*)**
    - **último a entrar, primeiro a sair**
  - **Filas (*Queues*) - FIFO (first-in, first-out)**
    - **primeiro a entrar, primeiro a sair.**

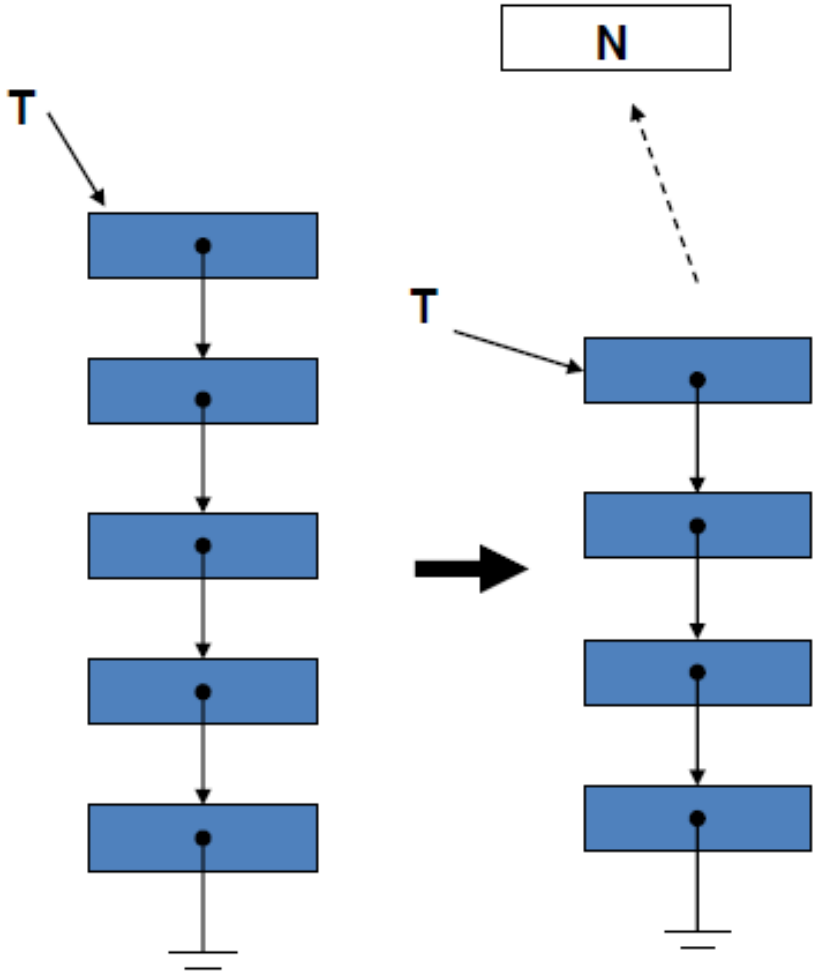
Pilha

- **Os nós só podem ser adicionados e removidos no topo da pilha.**
  - **A referência a uma pilha é feita por meio de um ponteiro para o elemento do topo da pilha.**
  - **O membro de ligação no último nó da pilha é definido como NULL para indicar o final da pilha.**
- **Principais funções**
  - **Push**
    - **cria um novo nó e o coloca no início da pilha**
  - **Pop**
    - **remove um nó do topo da pilha e libera a memória que estava alocada ao nó removido.**

Inserção e Remoção na Pilha



Inserção

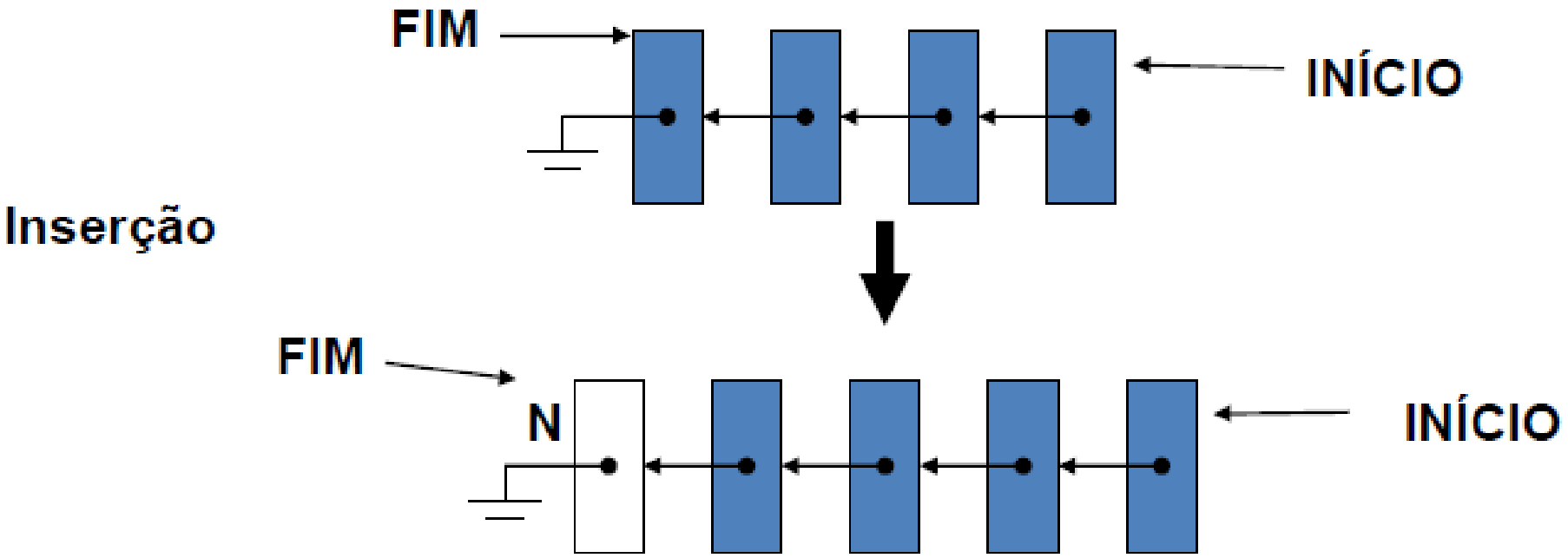


Remoção

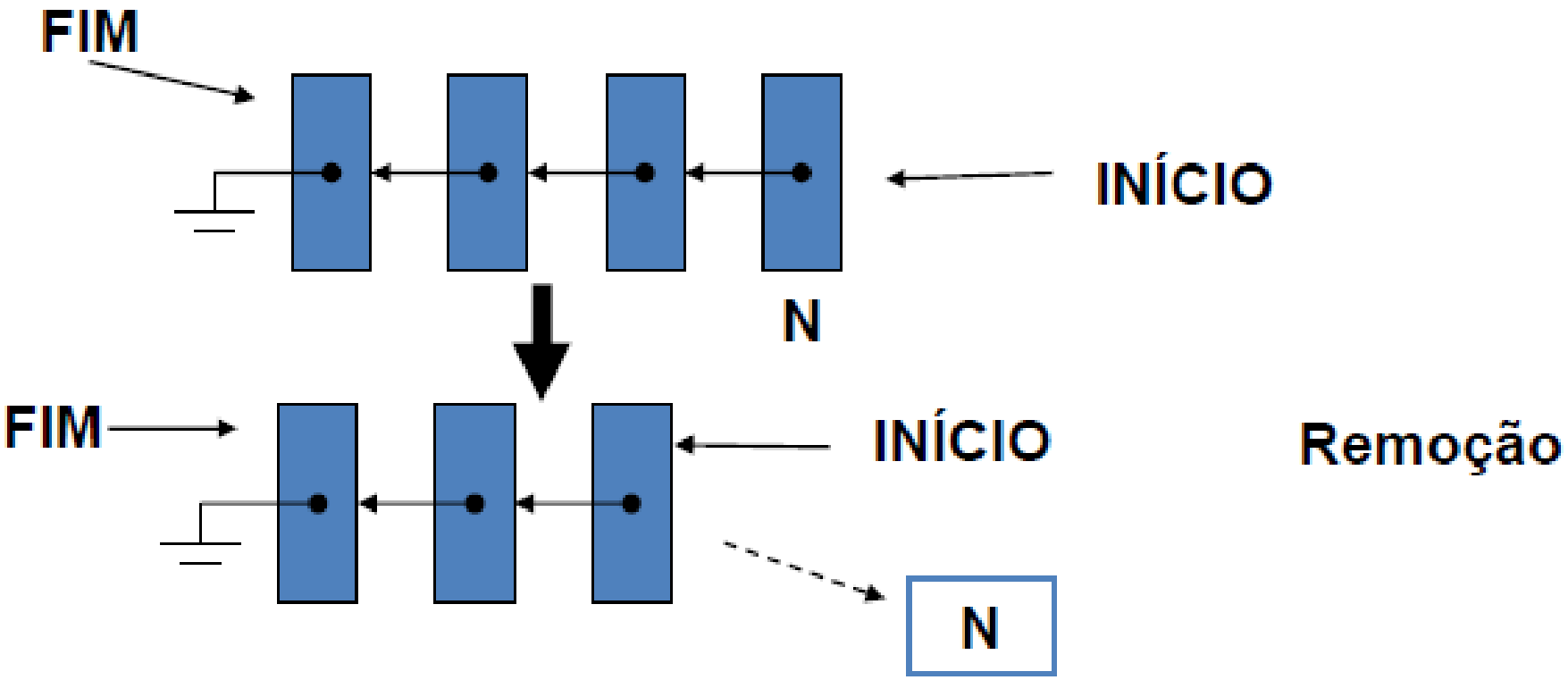
Fila

- **Os nós são removidos apenas no início da fila e são inseridos apenas em seu final.**
  - **Dois ponteiros: um apontando o início e outro apontando o final da fila.**
- **Principais funções:**
  - **enqueue (enfileirar)**
  - **dequeue (desenfileirar)**
- **Possuem muitas aplicações em sistemas computacionais:**
  - **Computadores com um único processador**
  - **Armazenamento de dados para impressão (spooling)**
  - **Redes de computadores (buffer de entrada num roteador)**

Inserção na Fila



Remoção da Fila



Exemplo – implementação de Fila

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

struct no {
    int info;
    struct no *prox;
};

typedef struct no *noPtr;

noPtr inicio = NULL;
void insere();
void retira();
void listar();
bool listaVazia(); //usar código anterior da Pilha
```

Exemplo – continuação

```
main() {
    int op;

    do {
        cout << "\n1: Insere elemento na fila"
             << "\n2: Retira elemento da fila"
             << "\n3: Listar elementos"
             << "\n0: Sair"
             << "\n\nDigite a opcao (0 - 3): " ;
        cin >> op;
        switch (op) {
            case 1: insere(); break;
            case 2: retira(); break;
            case 3: listar(); break;
        }
    } while (op != 0);
}
```

Exemplo – continuação

```
void insere () {
    noPtr aux, p = new no;

    cout << "\nDigite o valor do elemento: ";
    cin >> p->info;
    p->prox = NULL;
    if (listaVazia())
        inicio = p;
    else
    {
        aux = inicio;
        while(aux->prox != NULL)
            aux = aux->prox;
        aux->prox = p;
    }
}
```

Exemplo – continuação

```
void retira ()
{
    noPtr p;

    if (listaVazia())
        cout <<"\nFila Vazia!";
    else
    {
        p = inicio;
        inicio = p -> prox;     //inicio = inicio->prox
        delete p;
        cout << "\nO elemento foi retirado da Fila!\n";
    }
}
```

Exemplo – continuação

```
void listar()
{
    noPtr p = inicio;

    if (listaVazia())
        cout << "\nFila Vazia!";
    else
    {
        cout << "\nOs elementos da fila sao: \n";
        cout << "INICIO";
        while (p != NULL) {
            cout << " --> " << p->info;
            p = p->prox;
        }
        cout << "--> NULL\n\n";
}
```

Lista Encadeada Ordenada - Inserir

```
void inserir() {
    noPtr aux, ant, p = new no;

    cout << "Digite elemento: ";
    cin >> p->info;
    if (listaVazia())
    {
            p->prox = NULL;
            inicio = p;
  } else  {
    aux = inicio;
    while (aux != NULL && p->info > aux->info)
    {
        ant = aux;
        aux = aux->prox;
    }
    if (aux == inicio)
    {
        p->prox = aux;
        inicio = p;
     } else  {
        if (aux == NULL)
        {
            p->prox = NULL;
            ant->prox = p;
        } else {
            p->prox = aux;
            ant->prox = p;
        }
     }
  }
}
```

Lista Encadeada Ordenada - Consultar

```
void consultar() {
    noPtr p = inicio;
    int x;
    bool achei = false;

    if (!listaVazia())   {
        cout << "\nDigite o elemento: ";
        cin >> x;
        while (p != NULL && achei == false) {
            if (p->info == x)  {
                cout << "Elemento " << p->info << " possui endereco: " << p;
                achei = true;
            }
            p = p->prox;
        }
        if (!achei)
            cout << "\nElemento nao encontrado";
    }
    else cout << "\nLista Vazia!";

}
```

Lista Encadeada Ordenada - Exercício em Aula

- **Implementar as funções:**
  - **retirar( )**
    - **O usuário deve fornecer o elemento a ser retirado da lista. Caso não exista, mostrar mensagem de erro.**
  - **crescente( )**
    - **mostrar os elementos da lista em ordem crescente**
  - **decrescente( )**
    - **mostrar os elementos da lista em ordem decrescente**

Exercício #1

- **Prepare um programa para controlar os dados sobre cidades. Você deve armazenar o nome da cidade, o nome do prefeito, o nome do partido e o número de habitantes.**
  - **Inclusão de Cidades. Sem repetição de nomes de cidades.**
  - **Exclusão de Cidades. Pedir o nome da cidade.**
  - **Mudança de prefeito. Pedir o nome da cidade e o nome do novo prefeito e seu partido político.**
  - **Listagem de todos os dados em ordem alfabética de cidades.**
  - **Listar os nomes das cidades e dos prefeitos de um determinado partido. Pedir o nome do partido.**
  - **Listar todas as cidades com mais de 200.000 habitantes.**

Exercício #2

- **Deseja-se controlar os movimentos efetuados nas contas dos clientes de um banco. Para cada cliente deve ser armazenado o número da conta, nome e saldo. Sempre que um cliente fizer uma movimentação em sua conta, deve ser solicitado o número da conta, o tipo (depósito ou retirada) e o valor. O programa deve atualizar o saldo para cada movimentação e imprimir o comprovante contendo: número da conta, nome, tipo de movimento realizado, valor e saldo.**

Exercício #3

- **Você deve simular o funcionamento de um celular. Seu programa deverá apresentar um menu com as seguintes opções:**
  - **Fazer ligação: pedir o número a ser discado. Opção inválida se houver uma ligação em curso.**
  - **Encerrar ligação: serve para ligações chamadas ou recebidas.**
  - **Receber ligação: pedir o número de origem (lembrar que é uma simulação). Opção inválida se houver uma ligação em curso.**
  - **Listar as ligações realizadas: da mais recente para a mais antiga.**
  - **Listar as ligações recebidas: da mais antiga para a mais atual.**
  - **Incluir número na agenda: incluir o telefone e o nome de uma pessoa (ordenada por nome).**
  - **Excluir número da agenda: excluir o telefone e o nome de uma pessoa. Solicitar o nome.**
  - **Consultar número na agenda: consultar o telefone e o nome de uma pessoa. Solicitar o nome e mostrar o telefone ou a mensagem "Nome inválido".**
  - **Sair**

Exercício #4

- **Você deve gerenciar uma agência de modelos. Para isso armazene o nome, sexo, telefone de contato e todos os trabalhos já realizados de cada modelo. Implemente as seguintes funcionalidades:**
  - **Inclusão de um novo modelo. Sem repetição de nome.**
  - **Inclusão de serviço realizado por um modelo. Pedir o nome do modelo e o nome do cliente.**
  - **Listagem de todos os modelos.**
  - **Listagem de todos os serviços realizados por um modelo. Os trabalhos mais recentes deverão aparecer primeiro.**
  - **Listagem dos modelos que já prestaram serviços para um cliente. Pedir o nome do cliente.**
  - **Exclusão de um modelo. Solicitar o nome e excluir também todos os serviços.**

Exercício #5

- **Controle as informações sobre o desempenho de um time de basquete. As jogadas foram codificadas em: 1 – *cesta de um ponto*; 2 – *cesta de dois pontos*; 3 – *cesta de três pontos*; 4 – *assistência*; 5 – *rebote*. Implemente as seguintes funcionalidades:**
  - **Inclusão de um jogador. Solicitar o número da camisa e o nome. Não aceitar números repetidos.**
  - **Inclusão de jogada. Solicitar o número do jogador e o tipo de jogada (1 -5).**
  - **Listagem de todos os jogadores e o total de cada um dos cinco tipos de jogadas.**
  - **Exclusão de um jogador. Pedir o número do jogador.**
  - **Consulta. Pedir o número do jogador e mostrar seu desempenho em cada tipo de jogada.**

Exercício #6

- **Implemente um programa para controlar as informações sobre a utilização de uma CPU. Todos os processos que necessitam utilizar a CPU possuem uma prioridade. Além das definições relevantes, faça as seguintes funcionalidades:**
  - **Chegada de processo: recebe como parâmetro o número do processo e sua prioridade. Se dois processos possuem a mesma prioridade, o que chegou primeiro deve ser atendido primeiro.**
  - **Atendimento: retirar o processo que está chegou primeiro.**
  - **Exclusão de processo: recebe como parâmetro o número do processo.**
  - **Listagem de todos os processos.**

Exercício #7

- **Elabore um programa para gerenciar as informações sobre um tribunal. Você deve guardar o número do processo e todas as suas tramitações, ou seja, por que setores este processo já passou. Faça as seguintes funcionalidades:**
  - **Inclusão de um novo processo. Pedir o número do processo.**
  - **Inclusão de uma tramitação. Pedir o número de um processo e o nome do setor.**
  - **Consulta de um processo. Pedir o número de um processo e listar todas as suas tramitações. As tramitações mais recentes devem ser listadas primeiro.**
  - **Listagem de todos os processos e suas tramitações.**
  - **Processos em um setor. Pedir o nome do setor e listar todos os processos que estão neste setor, isto é, a última tramitação foi para este setor.**

Exercício #8

- **Controle as informações de uma agência de viagens. Para isso você deve fazer o gerenciamento de hotéis e voos para as cidades.**
  - **Inclusão de uma cidade. Pedir o nome da cidade. Sem repetição de nomes.**
  - **Inclusão de voo. Pedir o nome da cidade de origem e o nome da cidade destino. Ambas devem ter sido cadastradas. Origem e destino não podem ser iguais.**
  - **Inclusão de um hotel. Pedir o nome da cidade, o nome do hotel e sua capacidade.**
  - **Consulta. Pedir o nome da cidade e informar todas as cidades em que se pode chegar, além do nome e a capacidade de cada hotel.**
  - **Listagem Geral. O nome de todas as cidades em ordem alfabética.**

Referências

- **Moraes. *Estruturas de Dados e Algoritmos – uma abordagem didática*. Ed. Futura**
- **Markenzon e Szwarcfiter. *Estruturas de Dados e seus Algoritmos*. Ed. LTC**
- **Deitel. Como Programar em C/C++. Ed. Pearson**