



USP - ICMC - SSC SSC 0300 - 2o. Semestre 2013

Disciplina de Linguagem de Programação e Aplicações [Eng. Elétrica / Automação]

Prof. Dr. Fernando Santos Osório / PAE: Rafael Klaser (LRM / ICMC)

LRM - Laboratório de Robótica Móvel do ICMC / CROB-SC

Email: fosorio@icmc.usp.br ou fosorio@gmail.com

Página Pessoal: <http://www.icmc.usp.br/~fosorio/>

Material on-line:

Wiki ICMC - <http://wiki.icmc.usp.br/index.php>

Wiki SSC0300 - [http://wiki.icmc.usp.br/index.php/SSC-300-2013\(fosorio\)](http://wiki.icmc.usp.br/index.php/SSC-300-2013(fosorio))

Agenda:

- **Listas Encadeadas:**
 - **Fila / Queue (FIFO)**
 - **Pilha / Stack (LIFO)**
 - **Deque (Double Ended Queue)**
- **Funções para Manipular Listas Encadeadas.
Exemplos Práticos.**
- **Exercícios**

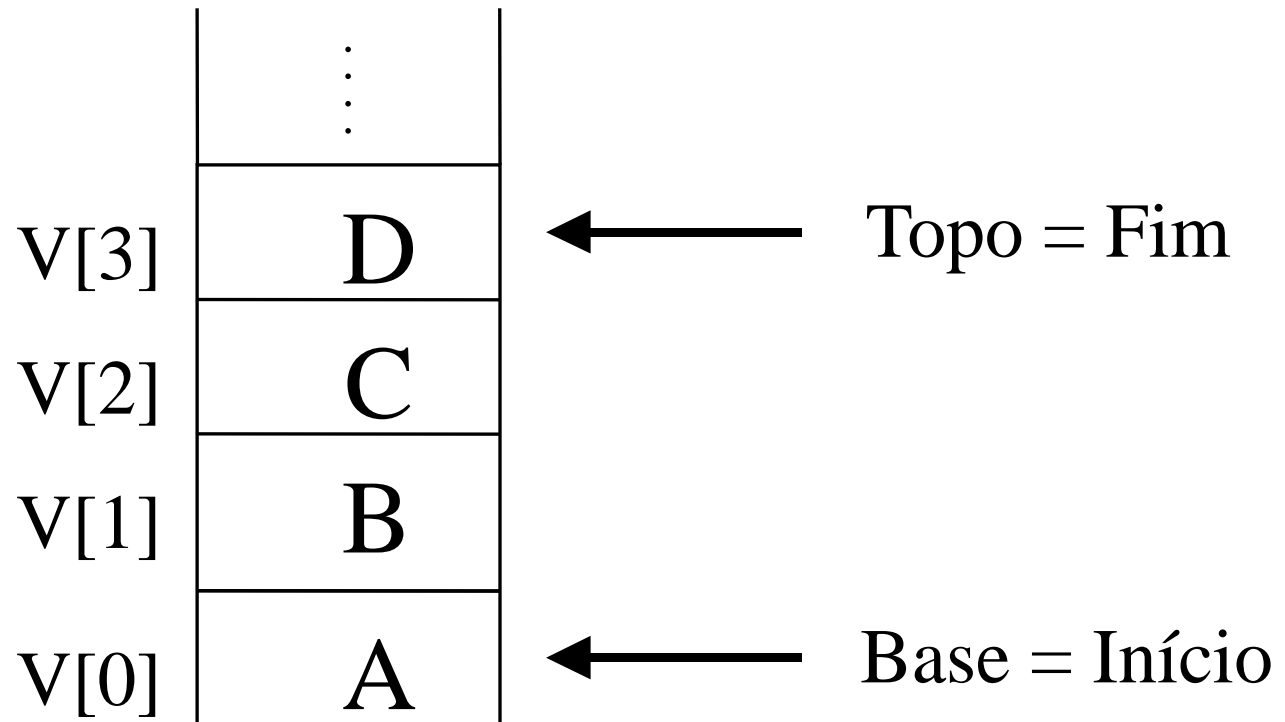
Informações Complementares a Atualizadas:

Consulte REGULARMENTE o material disponível na WIKI

[http://wiki.icmc.usp.br/index.php/SSC-300-2013\(fosorio\)](http://wiki.icmc.usp.br/index.php/SSC-300-2013(fosorio))

Listas Lineares Sequenciais (Vetores)

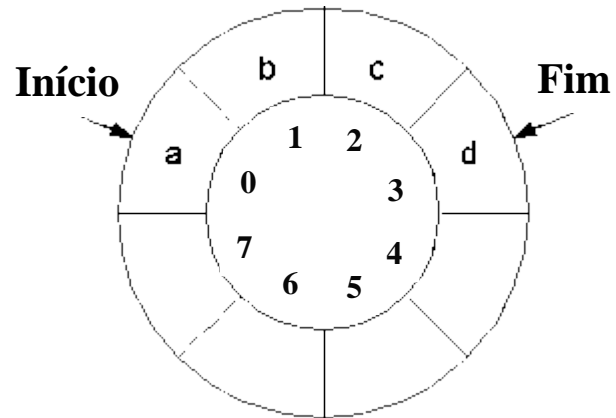
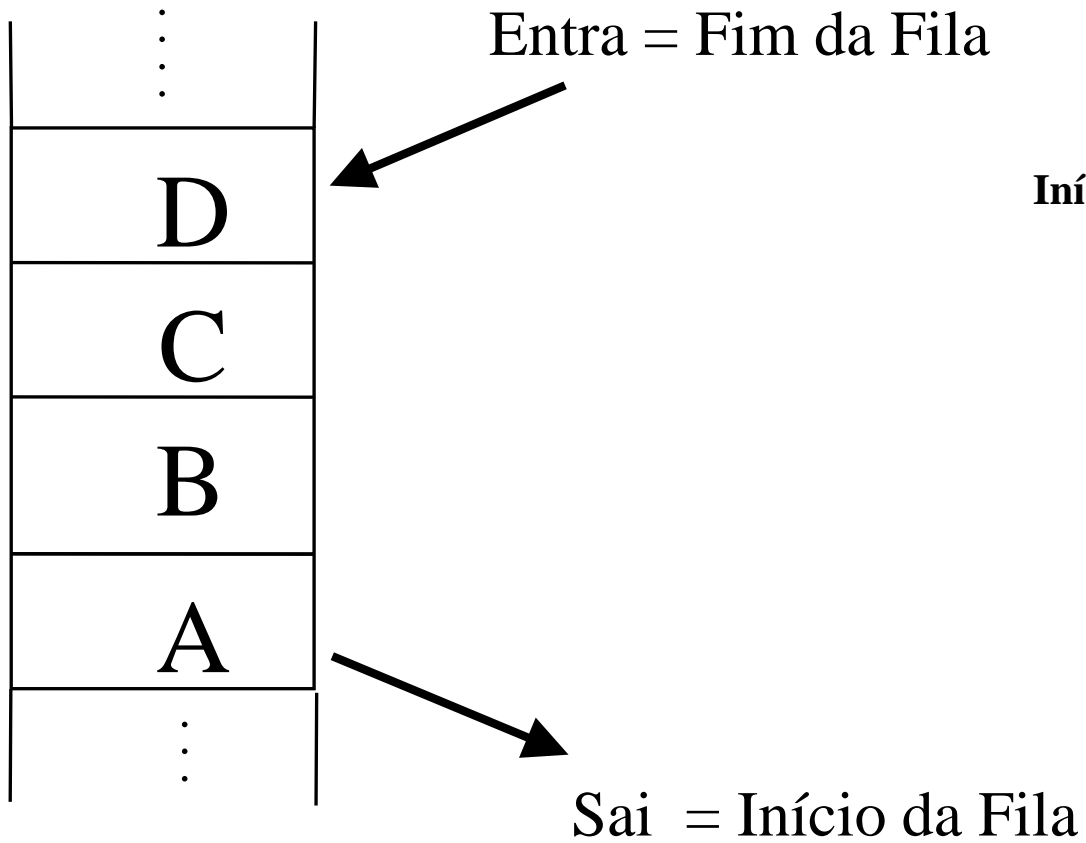
Alocação Estática - Vetores : Listas Lineares Sequenciais



Listas Lineares Sequenciais (Vetores)

Alocação Estática - Vetores : Filas - FIFO (First In, First Out)

Fila / Queue

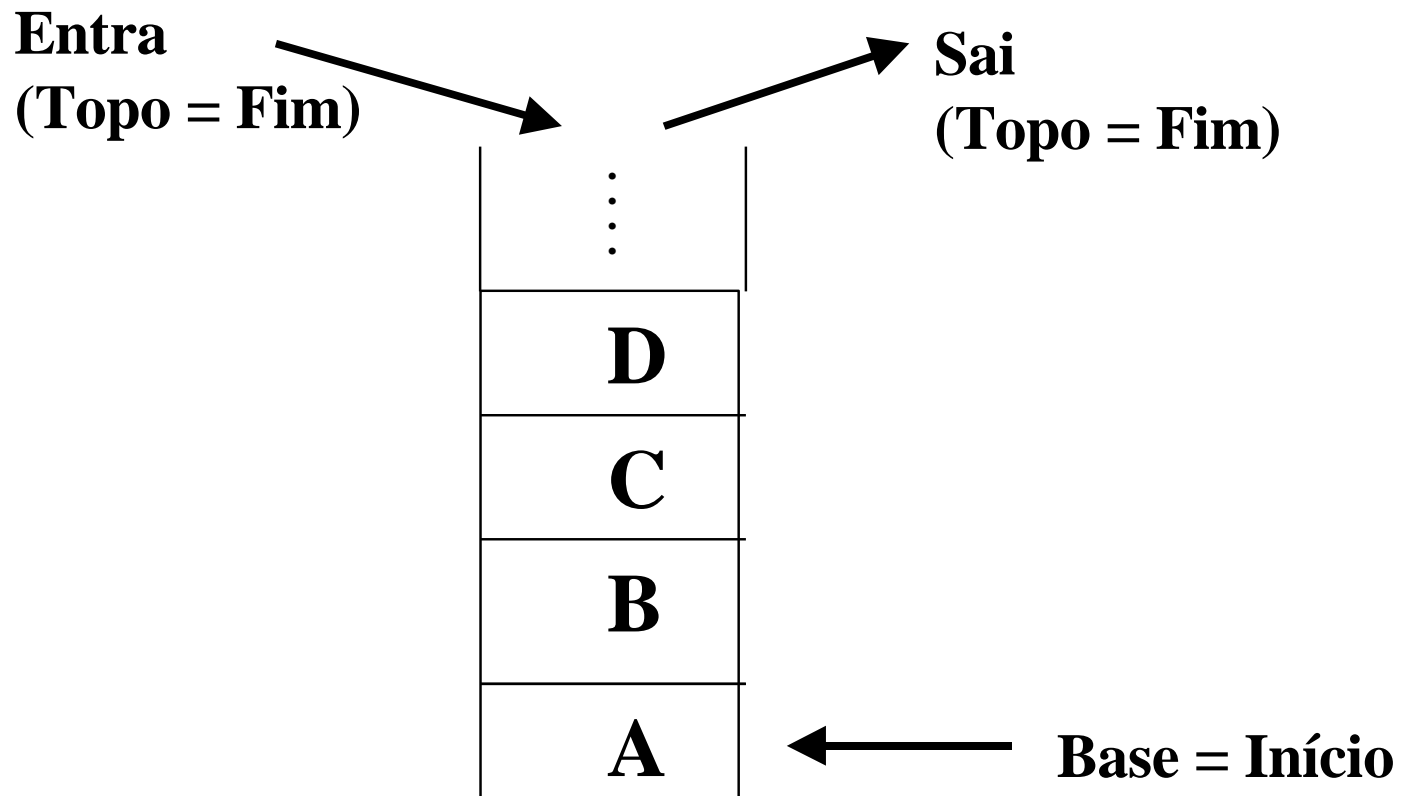


“Fila Circular”

Listas Lineares Sequenciais (Vetores)

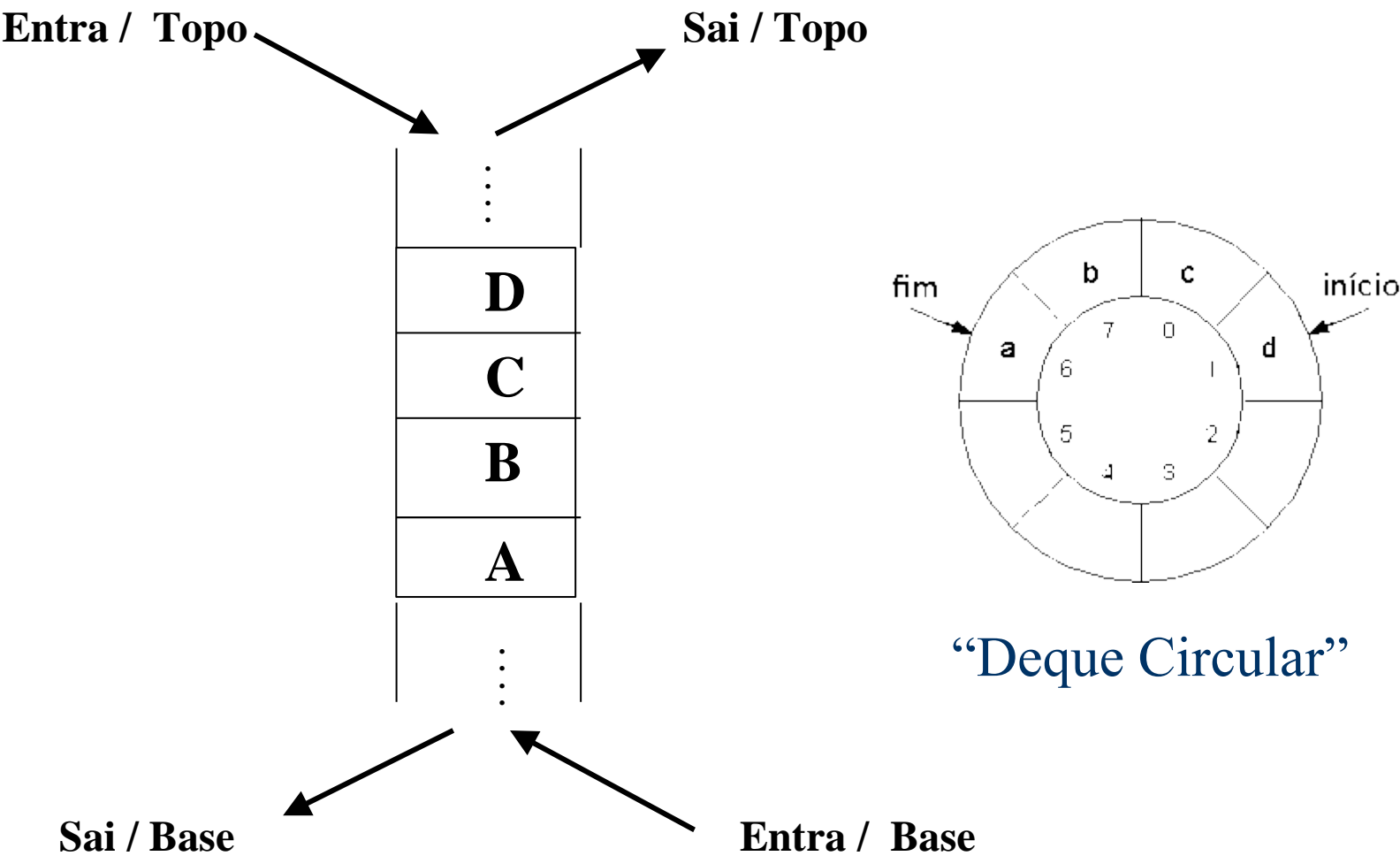
Alocação Estática - Vetores : Pilhas - LIFO (Last In, First Out)

Pilha / Stack



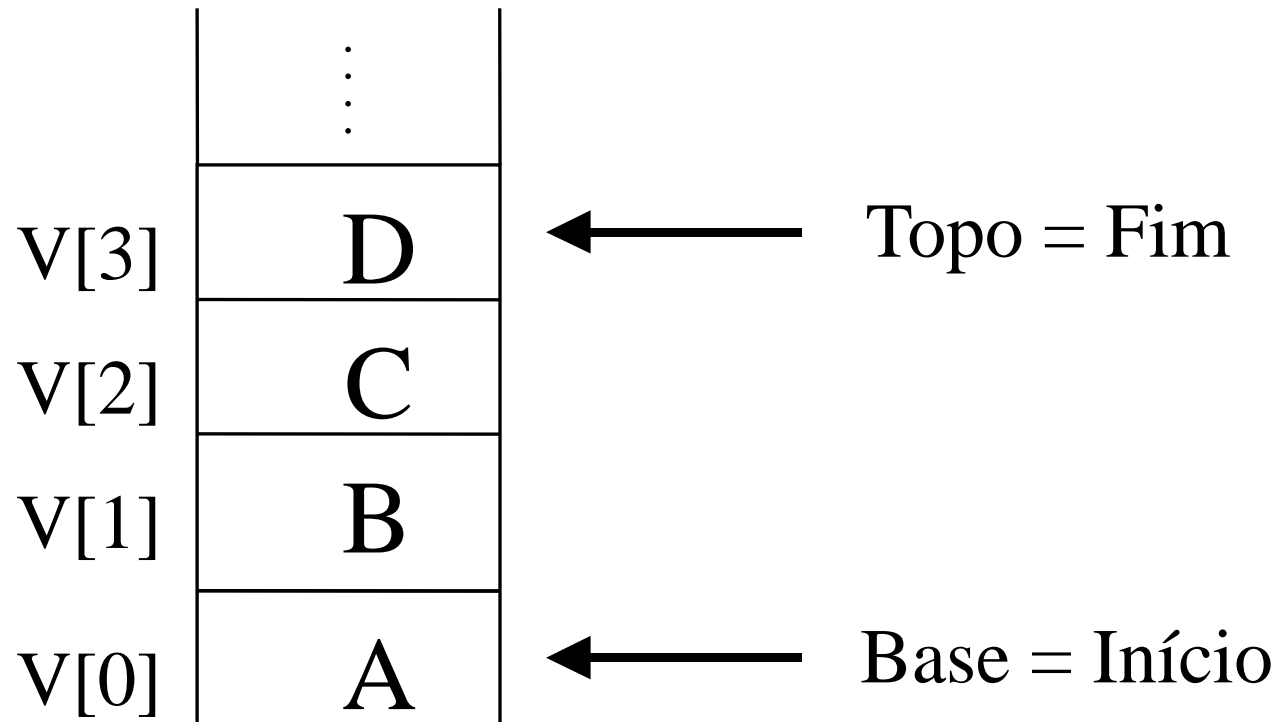
Listas Lineares Sequenciais (Vetores)

Alocação Estática - Vetores : **Deque** - Double Ended Queue



Listas Lineares Sequenciais (Vetores)

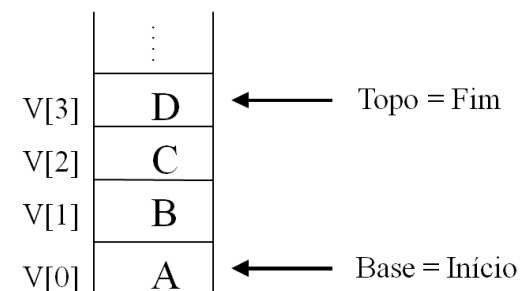
Alocação Estática - Vetores : Listas Lineares Sequenciais



Alocação Estática - Vetores : Listas Lineares Sequenciais

TIPOS DE DADOS:

```
typedef int Tipo_Dado;  
  
typedef struct {  
    Tipo_Dado    *Dado;  
    int          *Excluido;  
    int          Tamanho;  
    int          Inicio, Fim;  
} Tipo_Vetor;
```



ROTINAS:

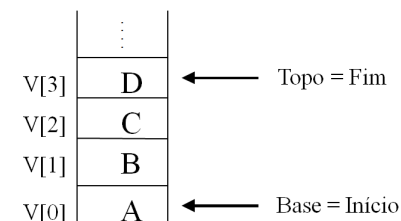
```
void inicializa_vetor (Tipo_Vetor *V, int Qtde);  
int  insere_vetor    (Tipo_Vetor *V, Tipo_Dado Dado);  
int  consulta_vetor  (Tipo_Vetor *V, int Indice, Tipo_Dado *Dado);  
int  acha_vetor      (Tipo_Vetor *V, Tipo_Dado Dado, int *Indice);  
void lista_vetor     (Tipo_Vetor *V);  
int  exclui_vetor    (Tipo_Vetor *V, int Indice);  
int  vazio_vetor     (Tipo_Vetor *V);  
int  quantidade_vetor (Tipo_Vetor *V);
```

Outras rotinas: posiciona_inicio, avanca_proximo, consulta_atual, ...

Alocação Estática - Vetores : Listas Lineares Sequenciais

```
void inicializa_vetor (V, Qtde)
Tipo_Vetor *V;
Int Qtde;
{
    V->Dado=(Tipo_Dado *)calloc(Qtde,sizeof(Tipo_Dado));
    V->Excluido=(int *)calloc(Qtde,sizeof(int));
    V->Tamanho=Qtde;
    V->Inicio=0;           /* Igual a: (*V).Inicio=0; */
    V->Fim=0;
}
```

```
int insere_vetor (V, Dado)
Tipo_Vetor *V;
Tipo_Dado Dado;
{
    if (V->Fim < V->Tamanho)           /* Vetor nao esta cheio ? */
    {
        V->Dado[V->Fim]=Dado;
        V->Excluido[V->Fim]=FALSO;      /* Falso: dado nao excluido */
        (V->Fim)++;
        return(OK);
    } else
        return(ERRO);
}
```



Alocação Estática - Vetores : Listas Lineares Sequenciais

```
#include "vetor.h"

main()
{
    Tipo_Vetor vet;
    Tipo_Dado  valor;
    int        qual;

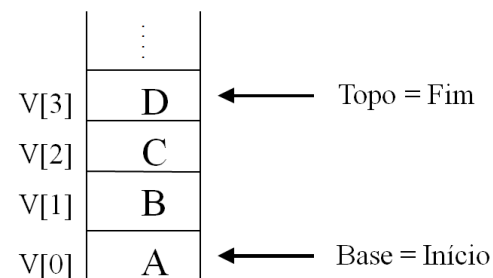
    inicializa_vetor(&vet,10);

    if (vazio_vetor(&vet))
        printf("=> Vetor vazio...\n");

    insere_vetor(&vet,2);
    insere_vetor(&vet,4);
    insere_vetor(&vet,6);

    printf("=> Elementos do vetor... %d\n",quantidade_vetor(&vet));
    lista_vetor(&vet);

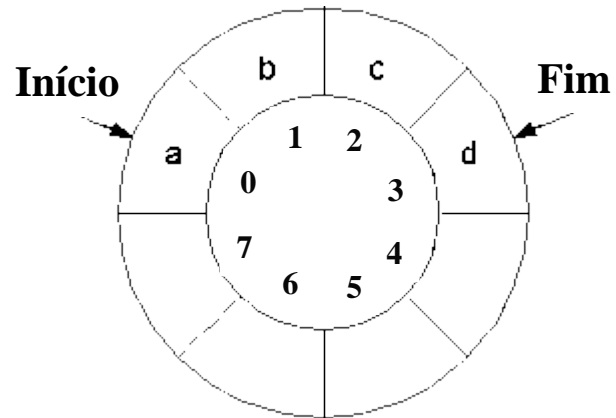
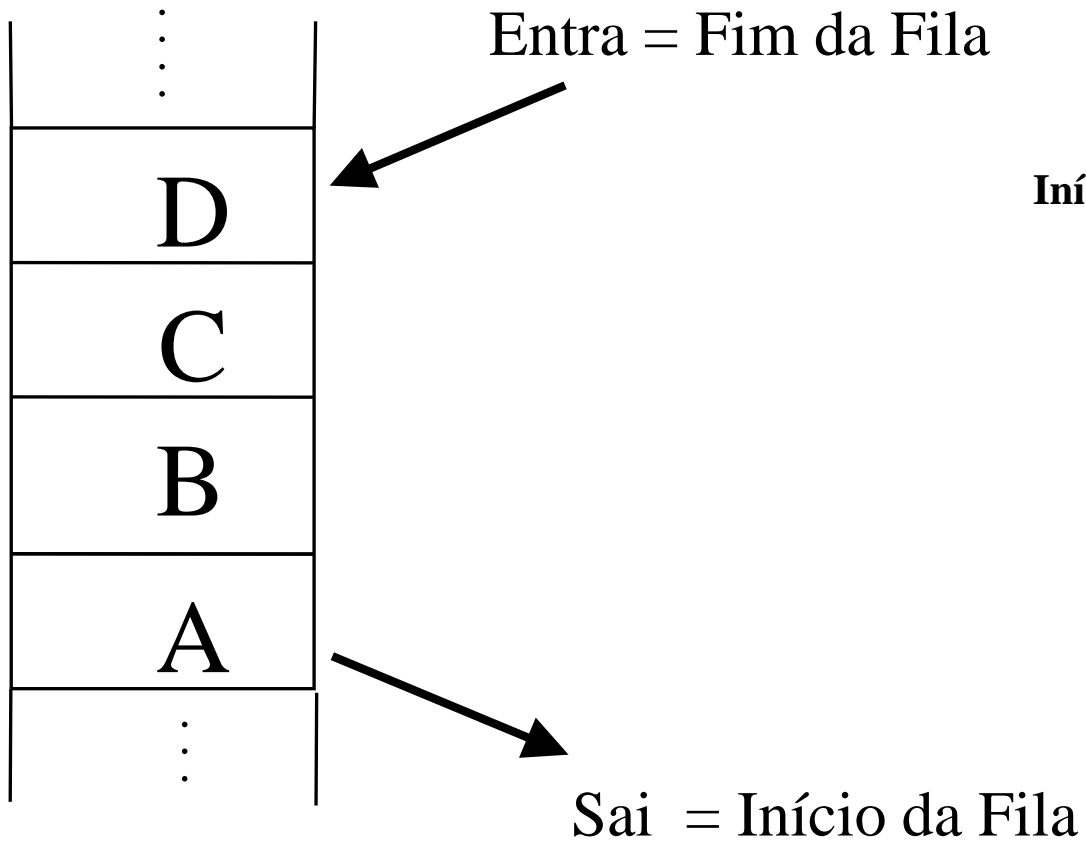
    if (acha_vetor(&vet,4,&qual))
        printf("=> O valor 4 esta no vetor na posicao %d\n",qual);
}
```



Listas Lineares Sequenciais (Vetores)

Alocação Estática - Vetores : Filas - FIFO (First In, First Out)

Fila / Queue



“Fila Circular”

Alocação Estática - Vetores : Filas - FIFO (First In, First Out)

Fila / Queue

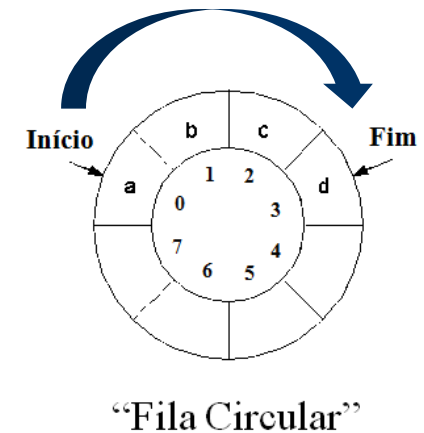
TIPOS DE DADOS:

```
typedef int Tipo_Dado;
```

```
typedef struct { Tipo_Dado *Dado;  
                int      Tamanho;  
                int      Inicio, Fim;  
            } Tipo_Fila;
```

ROTINAS:

```
void inicializa_fila (Tipo_Fila *F, int Qtde);  
int  insere_fila    (Tipo_Fila *F, Tipo_Dado Dado);  
int  retira_fila    (Tipo_Fila *F, Tipo_Dado *Dado);  
void lista_fila     (Tipo_Fila *F);  
int  cheia_fila     (Tipo_Fila *F);  
int  vazia_fila     (Tipo_Fila *F);  
int  quantidade_fila (Tipo_Fila *F);  
int  acha_fila      (Tipo_Fila *F, Tipo_Dado Dado, int *Indice);
```



Alocação Estática - Vetores : Filas - FIFO (First In, First Out)

Fila / Queue

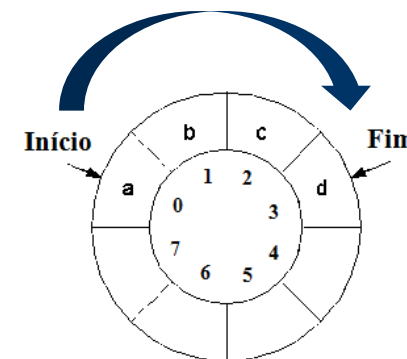
ROTINA INSERE:

```
int insere_fila (F, Dado)
Tipo_Fila *F;
Tipo_Dado Dado;
{
    int prox;

    prox=F->Fim+1;
    if (prox == F->Tamanho)
        prox=0;
    if (prox == F->Inicio)
        return(ERRO);
    else
    {
        F->Dado[F->Fim]=Dado;
        F->Fim=prox;
        return(OK);
    }
}
```

ROTINA RETIRA:

```
int retira_fila (F,
Tipo_Fila *F;
Tipo_Dado *Dado;
{
    if (F->Fim == F->Inicio)
        return(ERRO);
    else
    {
        *Dado= F->Dado[F->Inicio];
        (F->Inicio)++;
        if (F->Inicio >= F->Tamanho)
            F->Inicio=0;
        return(OK);
    }
}
```

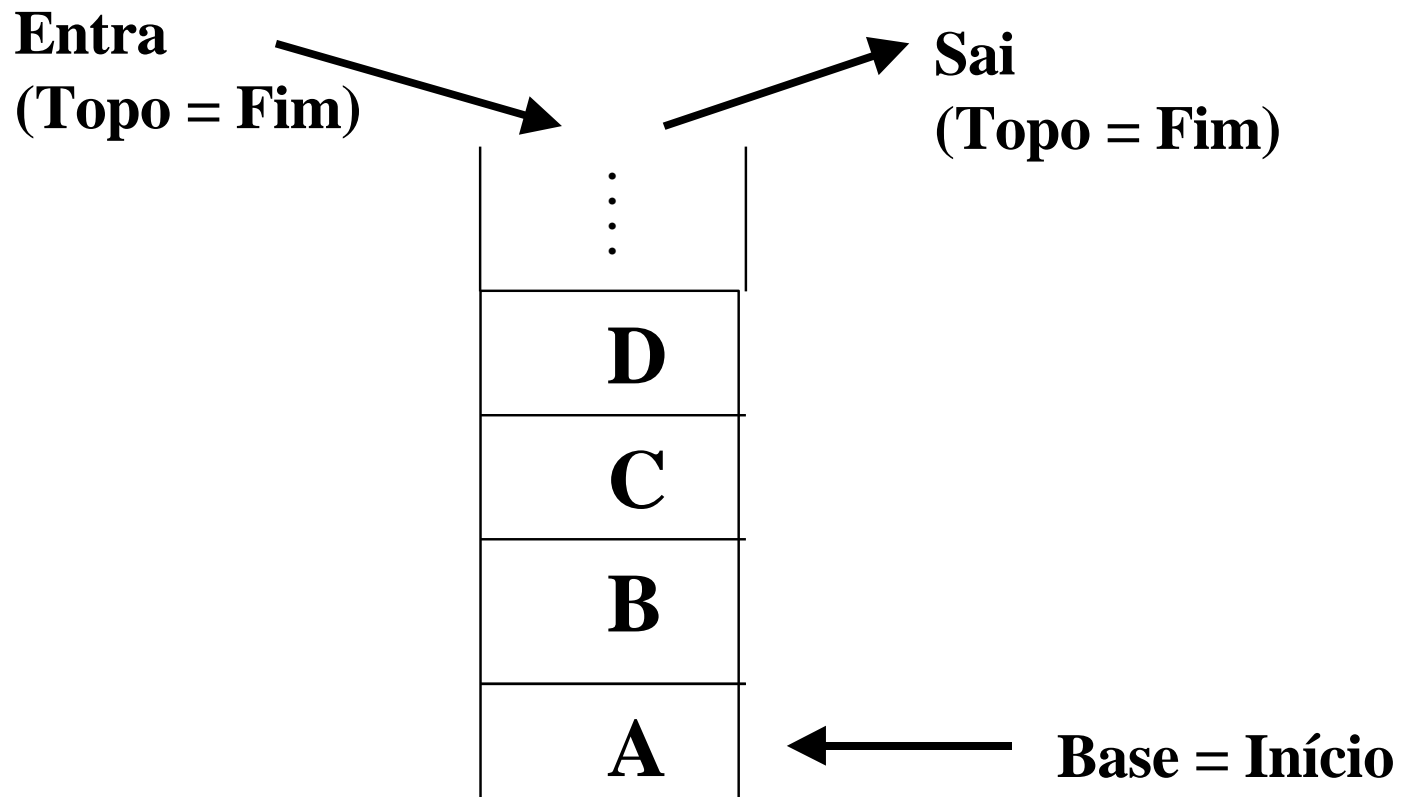


“Fila Circular”

Listas Lineares Sequenciais (Vetores)

Alocação Estática - Vetores : Pilhas - LIFO (Last In, First Out)

Pilha / Stack



Alocação Estática - Vetores : Pilhas - LIFO (Last In, First Out)

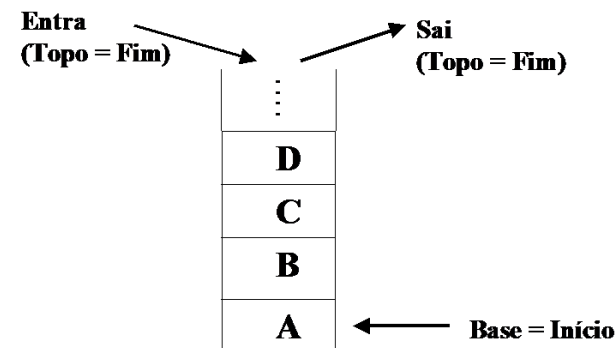
Pilha / Stack

TIPOS DE DADOS:

```
typedef int Tipo_Dado;  
  
typedef struct {  
    Tipo_Dado *Dado;  
    int Tamanho;  
    int Base, Topo;  
} Tipo_Pilha;
```

ROTINAS:

```
void inicializa_pilha (Tipo_Pilha *P, int Qtde);  
int insere_pilha (Tipo_Pilha *P, Tipo_Dado Dado);  
int retira_pilha (Tipo_Pilha *P, Tipo_Dado *Dado);  
void exibe_pilha (Tipo_Pilha *P);  
int quantidade_pilha (Tipo_Pilha *P);  
int cheia_pilha (Tipo_Pilha *P);  
int vazia_pilha (Tipo_Pilha *P);  
void esvazia_pilha (Tipo_Pilha *P);
```

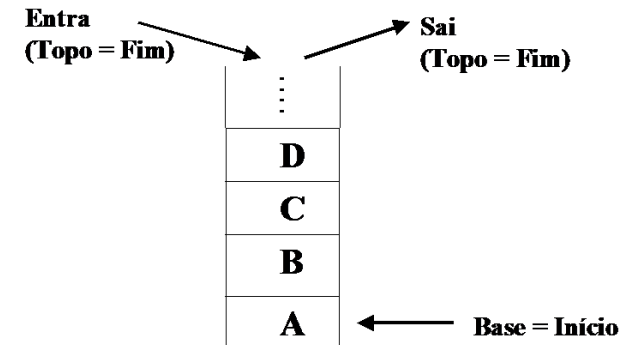


Alocação Estática - Vetores : Pilhas - LIFO (Last In, First Out)

Pilha / Stack

ROTINA EMPILHA:

```
int insere_pilha (P, Dado)
Tipo_Pilha *P;
Tipo_Dado Dado;
{
    /* Vetor nao esta cheio ? */
    if (P->Topo < P->Tamanho)
    {
        P->Dado[P->Topo]=Dado;
        (P->Topo)++;
        return(OK);
    }
    else
        return(ERRO);
}
```

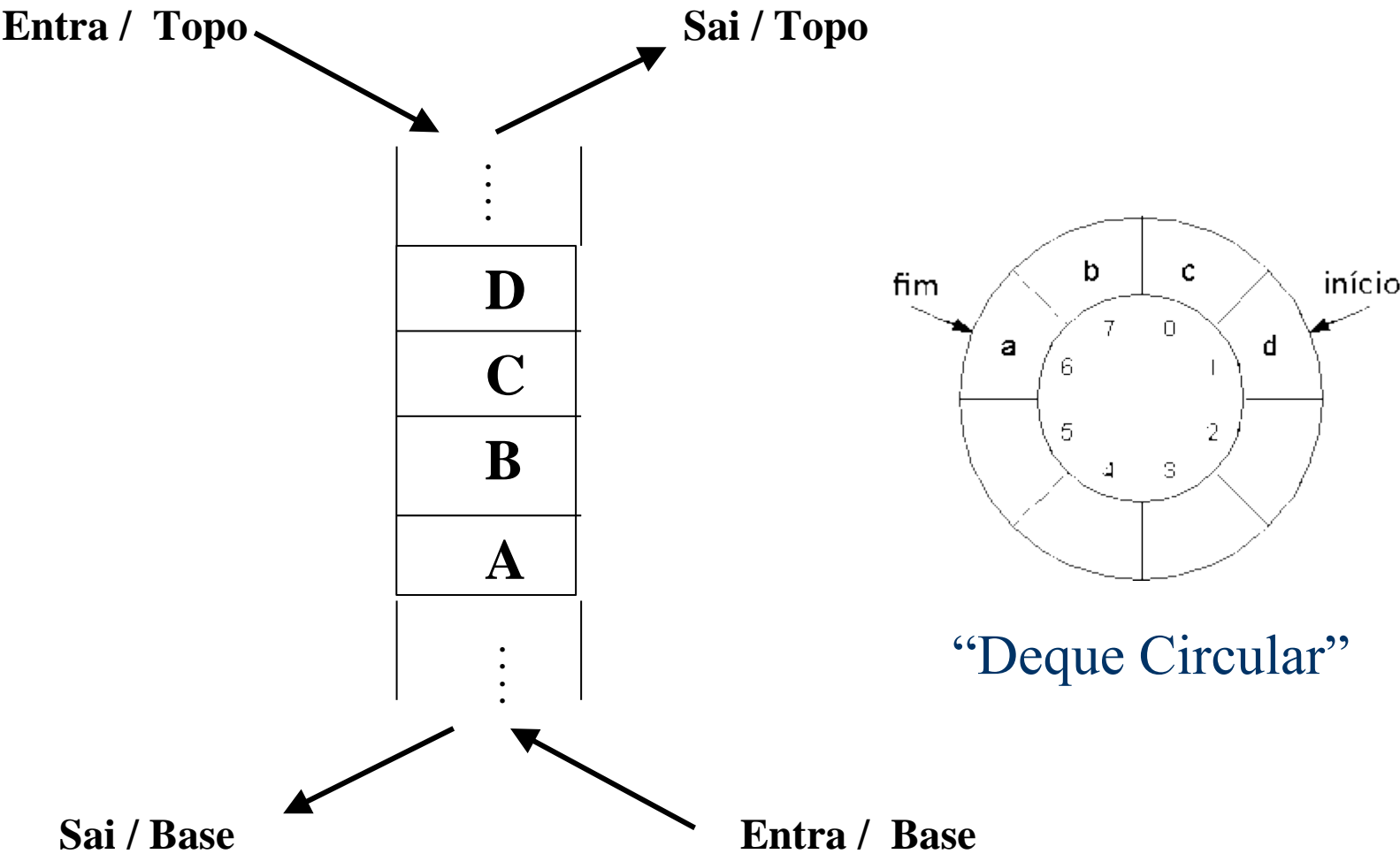


ROTINA DESEMPILHA:

```
int retira_pilha (P, Dado)
Tipo_Pilha *P;
Tipo_Dado *Dado;
{
    if (P->Topo == P->Base)
        return(ERRO);
    else
    {
        (P->Topo) --;
        *Dado=P->Dado[P->Topo];
        return(OK);
    }
}
```


Listas Lineares Sequenciais (Vetores)

Alocação Estática - Vetores : **Deque** - Double Ended Queue



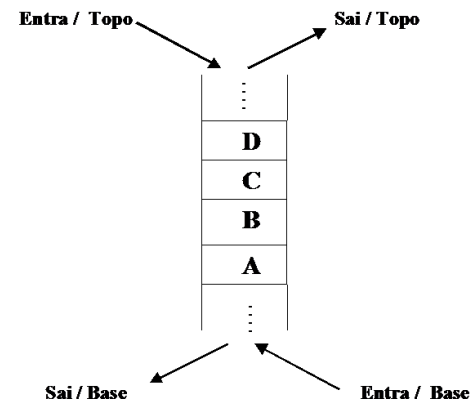
Alocação Estática - Vetores : **Deque** - Double Ended Queue

TIPOS DE DADOS:

```
typedef int Tipo_Dado;  
  
typedef struct {  
    Tipo_Dado *Dado;  
    int Tamanho;  
    int Inicio, Fim;  
} Tipo_Deque;
```

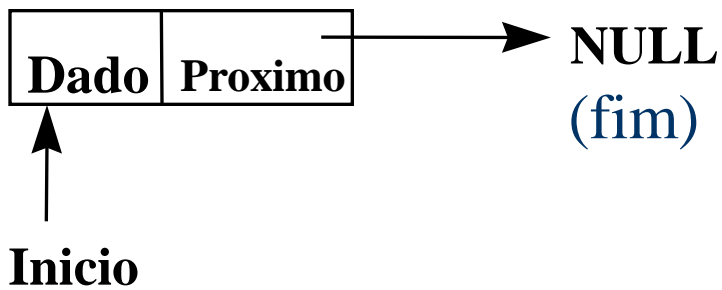
ROTINAS:

```
void inicializa_deque (Tipo_Deque *D, int Qtde);  
int insere_inicio_deque (Tipo_Deque *D, Tipo_Dado Dado);  
int insere_final_deque (Tipo_Deque *D, Tipo_Dado Dado);  
int retira_inicio_deque (Tipo_Deque *D, Tipo_Dado *Dado);  
int retira_final_deque (Tipo_Deque *D, Tipo_Dado *Dado);  
void lista_deque (Tipo_Deque *D);  
int acha_deque (Tipo_Deque *D, Tipo_Dado Dado, int *Indice);  
int cheio_deque (Tipo_Deque *D);  
int vazio_deque (Tipo_Deque *D);  
int quantidade_deque (Tipo_Deque *D);  
void apaga_deque (Tipo_Deque *D);
```

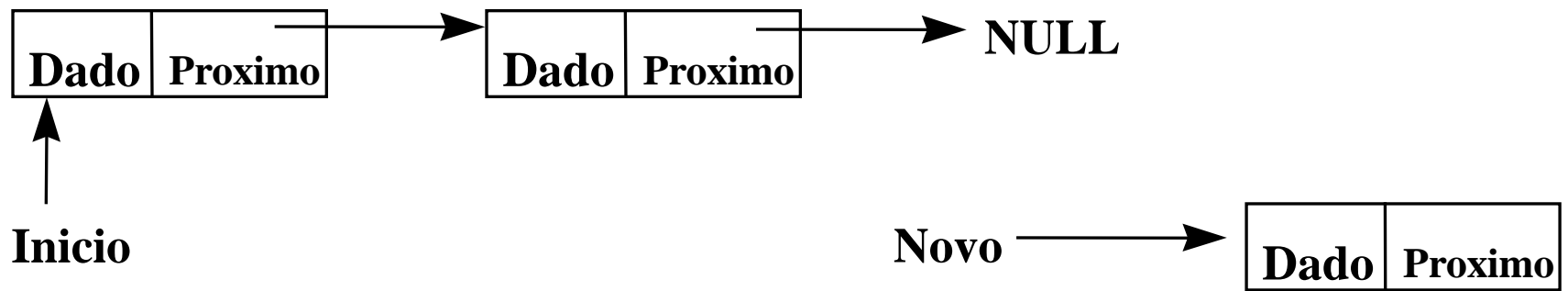


Listas Encadeada Simples (Ponteiros)

Alocação Dinâmica - Ponteiros: **Lista Encadeada Simples**

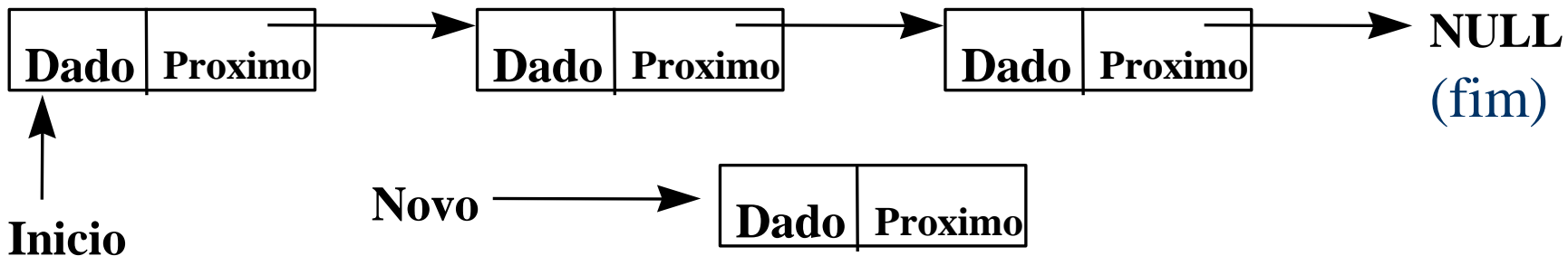


Inserção de Novo Nodo

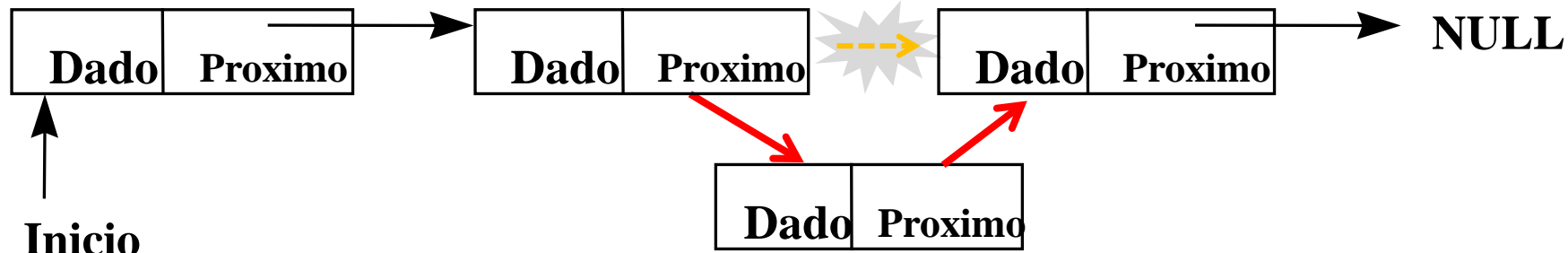


Listas Encadeada Simples (Ponteiros)

Alocação Dinâmica - Ponteiros: **Lista Encadeada Simples**



Inserção de Novo Nodo

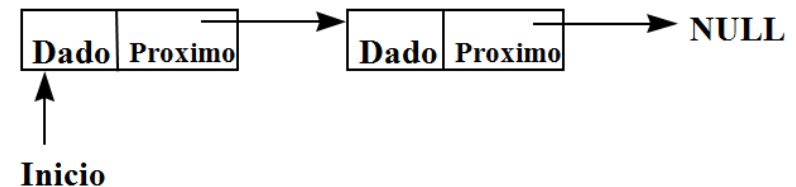


Alocação Dinâmica - Ponteiros: **Lista Encadeada Simples**

TIPOS DE DADOS:

```
typedef int Tipo_Dado;

typedef struct bloco {
    Tipo_Dado Dado;
    struct bloco *Proximo;
} Nodo;
```

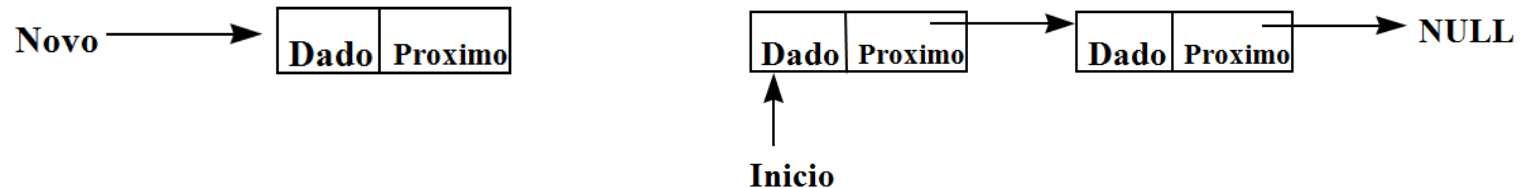


ROTINAS:

```
void inicializa_lista      (Nodo **N);
int insere_inicio_lista   (Nodo **N, Tipo_Dado Dado);
int insere_fim_lista      (Nodo **N, Tipo_Dado Dado);
int insere_ordenando_lista(Nodo **N, Tipo_Dado Dado);
int remove_inicio_lista   (Nodo **N, Tipo_Dado *Dado);
int remove_fim_lista      (Nodo **N, Tipo_Dado *Dado);
int remove_elemento_lista (Nodo **N, Tipo_Dado Dado);
int quantidade_lista      (Nodo **N);
void exibe_lista           (Nodo **N);
int pesquisa_lista        (Nodo **N, Tipo_Dado Dado);
int percorre_lista        (Nodo **N, Tipo_Dado *Dado);
void apaga_lista          (Nodo **N);
```

Listas Encadeada Simples (Ponteiros)

Alocação Dinâmica - Ponteiros: **Lista Encadeada Simples**

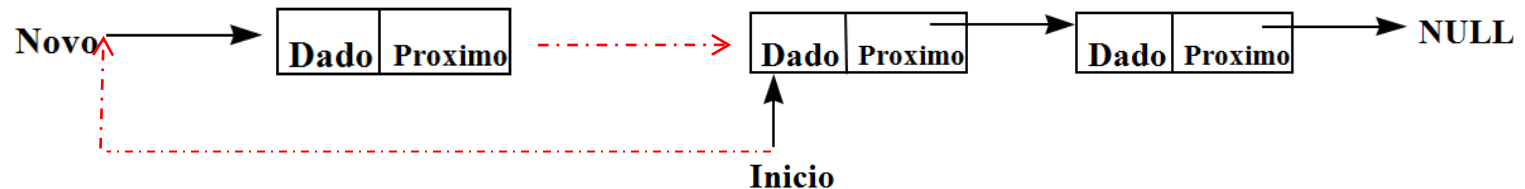


ROTINA INSERIR NO INICIO:

```
int insere_inicio_lista (N, Dado)
Nodo **N;
Tipo_Dado Dado;
{
    Nodo    *novo;

    novo = (Nodo *) calloc ( 1, sizeof (Nodo) );
    if ( novo == NULL )
        return (ERRO);          /* Não conseguiu alocar na memória */
    novo -> Dado = Dado;
    novo -> Proximo = *N;
    *N = novo;
    return (OK);
}
```

Alocação Dinâmica - Ponteiros: **Lista Encadeada Simples**



ROTINA INSERIR NO INICIO:

```
int insere_inicio_lista (N, Dado)
Nodo **N;
Tipo_Dado Dado;
{
    Nodo    *novo;

    novo = (Nodo *) calloc ( 1, sizeof (Nodo) );
    if ( novo == NULL )
        return (ERRO);          /* Não conseguiu alocar na memória */
    novo -> Dado = Dado;
    novo -> Proximo = *N;
    *N = novo;
    return (OK);
}
```

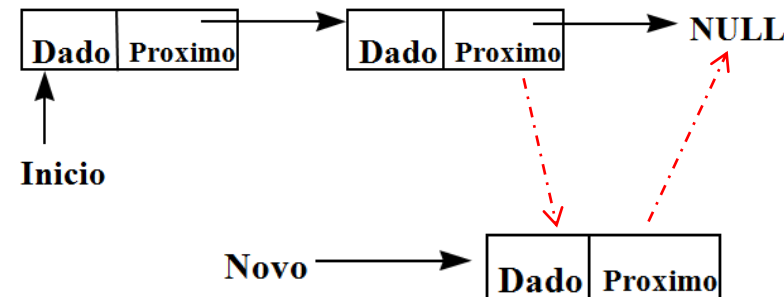
Alocação Dinâmica - Ponteiros: **Lista Encadeada Simples**

ROTINA INSERIR NO FINAL:

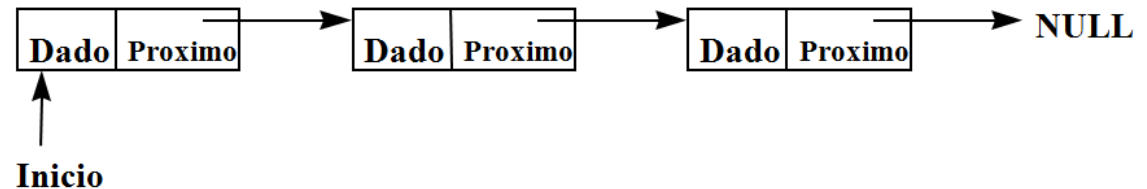
```
int insere_fim_lista (N, Dado)
Nodo **N;
Tipo_Dado Dado;
{
    Nodo  *aux,  *novo;

    novo = (Nodo *) calloc ( 1, sizeof (Nodo) );
    if ( novo == NULL ) return(ERRO);
    novo -> Dado = Dado;
    novo -> Proximo = NULL;

    if ( *N == NULL )    /* 1o. da lista? */
        *N = novo;
    else {
        aux = *N;
        while ( aux -> Proximo != NULL )
            aux = aux -> Proximo;
        aux -> Proximo = novo;
    }
    return (OK);
}
```



Alocação Dinâmica - Ponteiros: **Lista Encadeada Simples**



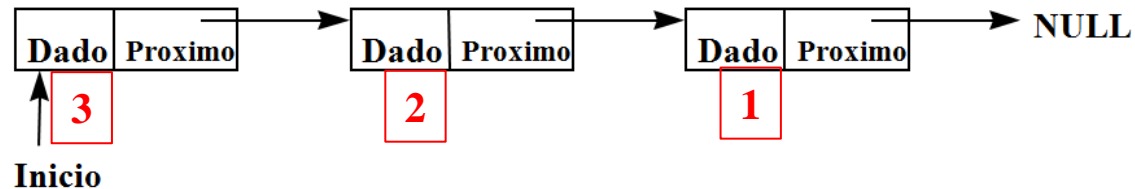
ROTINA EXIBIR LISTA:

```
int exibe_lista (N)
Nodo **N;
{
    Nodo  *aux;

    aux = *N;
    if (aux != NULL)
    {
        while ( aux !=  NULL )
        {
            printf("Dado: %d\n",aux->Dado);
            aux = aux -> Proximo;
        }
    }
    else    printf("Lista vazia!\n");
}
```

Listas Encadeada Simples (Ponteiros)

Alocação Dinâmica - Ponteiros: **Lista Encadeada Simples**



```
#include "listsimp.c"
```

```
main()
```

```
{
```

```
    Nodo *Inicio;
```

```
    printf("\n ROTINAS DE MANIPULACAO DE LISTAS SIMPLES ENCADEADAS \n\n");
```

```
    inicializa_lista(&Inicio);
```

```
    insere_inicio_lista(&Inicio,1);
```

```
    insere_inicio_lista(&Inicio,2);
```

```
    insere_inicio_lista(&Inicio,3);
```

```
    exibe_lista(&Inicio);
```

```
    system("pause");
```

```
}
```

Exercícios - Uso de Listas (Fila, Pilha, Deque):

1. Faça um programa que simule uma **fila** de um banco: (usando as rotinas vistas na aula)

O programa deve ter um laço principal que solicita a operação a executar, sendo

- 1: Insere na fila
- 2: Retira da fila
- 3: Exibe a fila

Quando a operação for inserir na fila, ler um valor numérico e inserir na fila.

Quando a operação for retirar da fila, exibir na tela o valor que estava na fila.

Quando a operação for exibir a fila, exibir todos os dados armazenados na fila.

2. Faça um programa que simule uma **pilha** de valores: (usando as rotinas vistas na aula)

O programa deve ter um laço principal que solicita a operação a executar, sendo

- 1: Insere na pilha
- 2: Retira da pilha
- 3: Exibe a pilha

Quando a operação for inserir na pilha, ler um valor numérico e inserir na pilha.

Quando a operação for retirar da pilha, exibir na tela o valor que estava na pilha.

Quando a operação for exibir a pilha, exibir todos os dados armazenados na pilha.



INFORMAÇÕES SOBRE A DISCIPLINA

USP - Universidade de São Paulo - São Carlos, SP
ICMC - Instituto de Ciências Matemáticas e de Computação
SSC - Departamento de Sistemas de Computação

Prof. Fernando Santos OSÓRIO

Web institucional: <http://www.icmc.usp.br/>

Página pessoal: <http://www.icmc.usp.br/~fosorio/>

Página do Grupo de Pesquisa: <http://www.lrm.icmc.usp.br/>

E-mail: fosorio [at] icmc. usp. br ou fosorio [at] gmail. com

Disciplina de Linguagem de Programação e Aplicações SSC300

WIKI - [http://wiki.icmc.usp.br/index.php/SSC-300-2013\(fosorio\)](http://wiki.icmc.usp.br/index.php/SSC-300-2013(fosorio))

> Programa, Material de Aulas, Critérios de Avaliação,

> Trabalhos Práticos, Datas das Provas, Notas