UNISINOS - **UNIVERSIDADE DO VALE DO RIO DOS SINOS**
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS (C6/6) – Curso: Informática

LABORATÓRIO II – AULA 04

Disciplina: Linguagem de Programação PASCAL
Professor responsável: ***Fernando Santos Osório***
Semestre: 99/2
Horário: 21 e 41

E-mail: ***osorio@exatas.unisinos.br***
Web:
http://www.inf.unisinos.br/~osorio/lab2.html
Xerox : *Pasta 54* – LAB. II (Xerox do C6/6)

Lista Lineares – Lista Seqüencial com Alocação Estática

1. Alocação de Memória:

Ao criar um programa em Pascal usualmente temos que especificar, antes de começar a executar o programa, as variáveis que vamos usar, reservando assim um espaço na memória. As variáveis que são alocadas em posições fixas da memória são chamadas de *variáveis estáticas*, e as variáveis que não possuem uma posição fixa, e que são criadas e destruídas durante a execução do programa, são chamadas de *variáveis dinâmicas*.

A alocação de memória no computador pode ser dividida em dois grupos principais:

- Alocação Estática: os dados tem um tamanho fixo e estão organizados seqüencialmente na memória do computador. Um exemplo típico de alocação estática são as variáveis globais, e os vetores.
- Alocação Dinâmica: os dados não precisam ter um tamanho fixo, pois podemos definir para cada dado quanto de memória que desejamos usar. Sendo assim vamos alocar espaços de memória (blocos) que não precisam estar necessariamente organizados de maneira seqüencial, podendo estar distribuídos de forma esparsa na memória do computador. Na alocação dinâmica, vamos pedir para alocar/desalocar blocos de memória, de acordo com a nossa necessidade, reservando ou liberando blocos de memória durante a execução de um programa. Para poder "achar" os blocos esparsos na memória usamos as variáveis do tipo *Ponteiro* (indicadores de endereços de memória). . Um exemplo típico de alocação dinâmica são as variáveis locais e parâmetros.

Vamos estudar aqui a construção de estruturas de dados em Pascal, usando estes dois tipos de métodos de alocação de memória: estática e dinâmica. O primeiro tipo de estrutura de dados a ser abordado, que é muito usado em Pascal, são as listas. As listas servem para armazenar um conjunto de dados, sejam eles agrupados (vetores = *arrays*) ou não (listas encadeadas).

1.1. Conjunto de dados com alocação estática:

A) *Listas lineares seqüenciais – Vetores Simples*
B) *Filas*
C) *Pilhas*
D) *Deques*

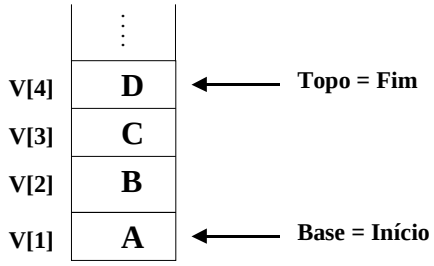
1.2. Conjuntos de dados com alocação dinâmica:

A) *Listas encadeadas simples*
B) *Filas*
C) *Pilhas*
D) *Listas duplamente encadeadas*
E) *Árvores*

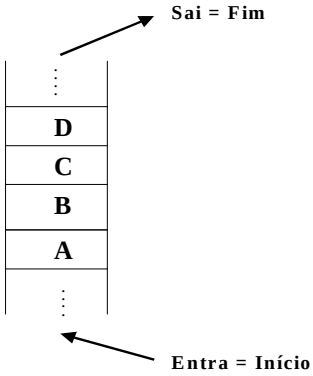
2. Alocação Estática de Memória:

As estruturas de dados para armazenar um conjunto de dados (exemplo: um cadastro com vários registros) podem ser organizadas de formas diferentes, de acordo com a maneira que os dados são inseridos e retirados da memória do computador. As formas mais usuais são: as listas lineares seqüenciais (vetores simples), as filas, as pilhas e os deques.

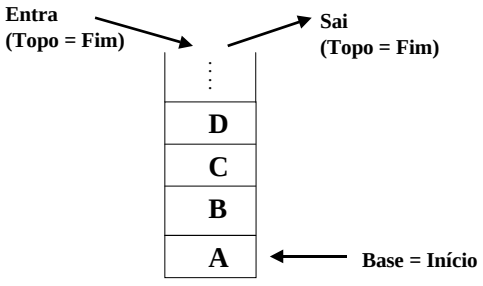
2.1. Listas lineares seqüenciais



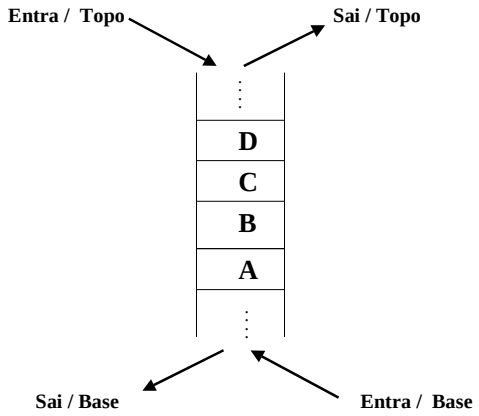
2.2. Filas – **FIFO** = "First In, First Out"



2.3. Pilhas – **LIFO** = "Last In, First Out"



2.4. Deques



EXERCÍCIOS

1. Faça um programa para a manipulação de listas lineares seqüenciais que utilize as seguintes rotinas genéricas de manipulação dos dados:

   Definições elementares:

```
Const
   Maximo = 10;
Type
   Tipo_dado = Integer;
   Tipo_vetor = Record
                    Vetor     : Array [1..Maximo] of Tipo_dado;
                    Excluido : Array [1..Maximo] of Boolean;
                     Inicio, Fim: Integer;
                End;
```

   Rotinas:

   Procedure Inicializa_Vetor (Var V: Tipo_vetor);

   Function Insere_Vetor (Var V: Tipo_vetor; Dado: Tipo_dado) : Boolean;

   Function Consulta_Vetor (V: Tipo_vetor; Qual: Integer; Var Dado: Tipo_dado): Boolean;

   Procedure Lista_Vetor (V: Tipo_vetor);

   Function Exclui_Vetor (Var V: Tipo_vetor; Qual: Integer): Boolean;

   Function Acha_Vetor (V: Tipo_vetor; Dado: Tipo_dado; var Qual: Integer): Boolean;

   Procedure Compacta_Vetor (Var V: Tipo_vetor);

   Function Atualiza_Vetor (Var V: Tipo_vetor; Qual: Integer; Novo_dado: Tipo_Dado): Booolean;

   Function Quantidade_Vetor (V: Tipo_vetor): Integer;