CURSO SUPERIOR DE ADS

Listas, Fila & Pilha

Prof. Fernando Marlon Soares Figueiredo

Disciplina: Programação Estruturada

SECTRAS

ADS

Listas

Introdução

- Uma das formas mais comuns de manter **dados agrupados** é a lista!
- Por exemplo, Lista de Compras, Lista de amigos...
- A lista é um recurso **útil e eficiente** para as pessoas.
- Em ciência da computação a lista é **uma das estruturas de dados mais utilizadas** no desenvolvimento de programas.

Lista Linear

Aula baseada no capítulo 2 do livro Estruturas de Dados Fundamentais: conceitos e aplicações do Silvio do Lago Pereira.

Lista Linear

- É uma sequência de itens $L = \{x_1, x_2, ..., x_n\}$, n >= 0, cujas propriedades estruturais dependem apenas da posição relativa de seus itens;
- Se **n = 0**, dizemos que a lista é **vazia**.
- Quando **a lista *L* não é vazia**, valem as seguintes propriedades:
  - x_1 é o primeiro item de *L*;
  - x_n é o último item de *L*;
  - x_i é precedido pelo item $x_i - 1$ e seguido pelo item $x_i + 1$ em *L*, para $1 < i < n$.

Lista Linear

A característica fundamental de uma lista linear é sua organização **unidimensional**, que permite dizer **precisamente onde ela começa** e **onde ela termina**.

Algumas Operações de Listas Lineares

1 - Inserir um item na i-ésima posição de L...

- Pera...

- Você sabe o que quer dizer *i-ésima*?
- Em que posição está o **centésimo** elemento?
- Na posição 100!
- Então o *i-ésimo* item, é o que está na posição *i*.

i-ésimo elemento

- Diremos que o *i-ésimo* elemento de um vetor x é x[i]...
- Embora, a rigor, deveria ser x[i - 1], uma vez que os elementos são numerados a partir de 0 na linguagem C e Java;
- De modo mais geral, ao listar coisas, diremos que:
  - a 0-ésima coisa na lista é a primeira,
  - a 1-ésima coisa na lista é a segunda,
  - a 2-ésima é a terceira coisa na lista... e assim por diante.

Operações em Listas Lineares

1. **Inserir** um item na i-ésima posição de L.
2. **Remover** o item da i-ésima posição de L.
3. **Acessar** o item da i-ésima posição de L.
4. **Determinar** se um item x pertence a L.
5. **Determinar** o total de itens em L.
6. **Ordenar** os itens em L.

Tipos de Listas Lineares

- Existem casos que as operações de **inserção**, **remoção** e **acesso** são **restritas** aos **extremos de uma lista linear** *L*.
- Por exemplo, a fila de um banco. O atendimento começa por quem está no início. A entrada de pessoas é sempre pelo **fim** da fila.
- Alguns **casos especiais** envolvendo **listas** aparecem frequentemente na **modelagem de problemas** a ser resolvidos por computador, são as **Pilhas**, **Filas** e **Fila-Dupla**.


O COFRE desta agência possui
tempo programado
FILA

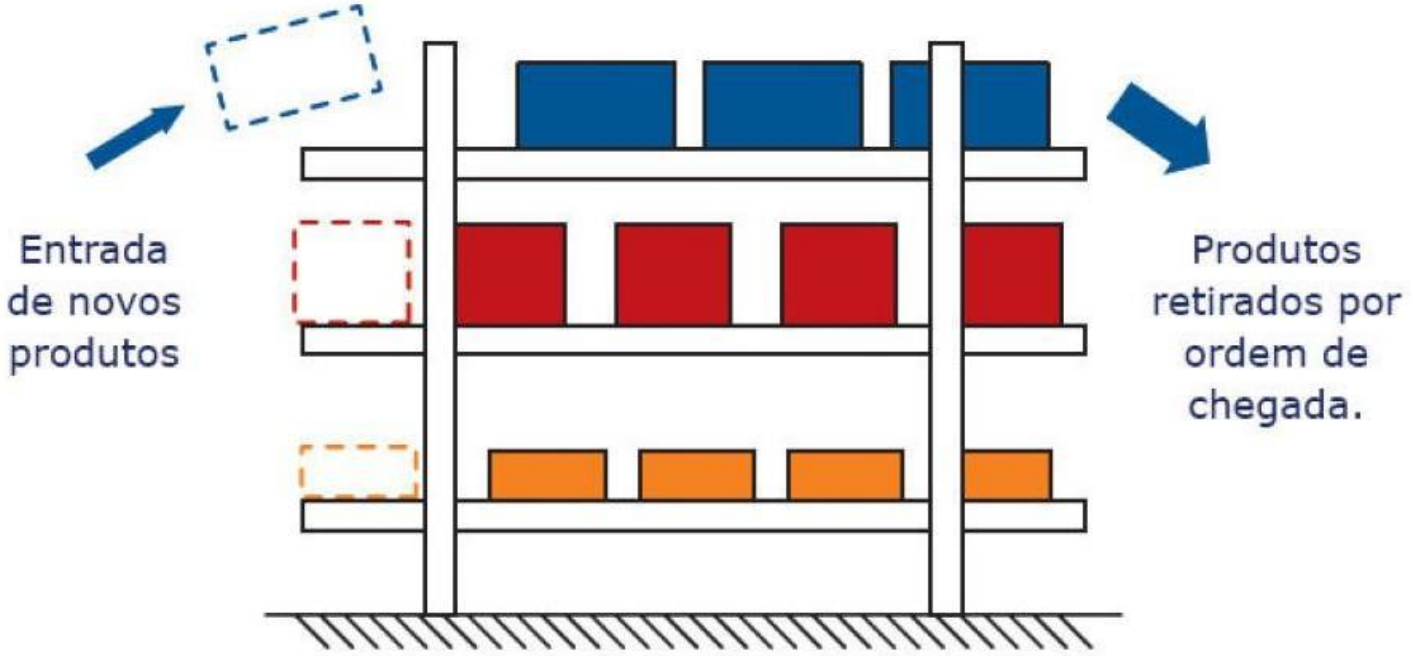
Fila

- Lista linear em que as **inserções são feitas num extremo** (denominado final) e as **remoções são feitas no extremo oposto** (denominado começo).
- Filas são também denominadas listas FIFO (*First-In/ First-Out* ou *Primeiro-que-Entra / Primeiro-que-Sai* - PEPS).

PEPS

Primeiro que entra / Primeiro que Sai

VOCÊS SABIAM QUE OPEPS É UM DOS MÉTODOS DE VALORIZAÇÃO DE ESTOQUES?

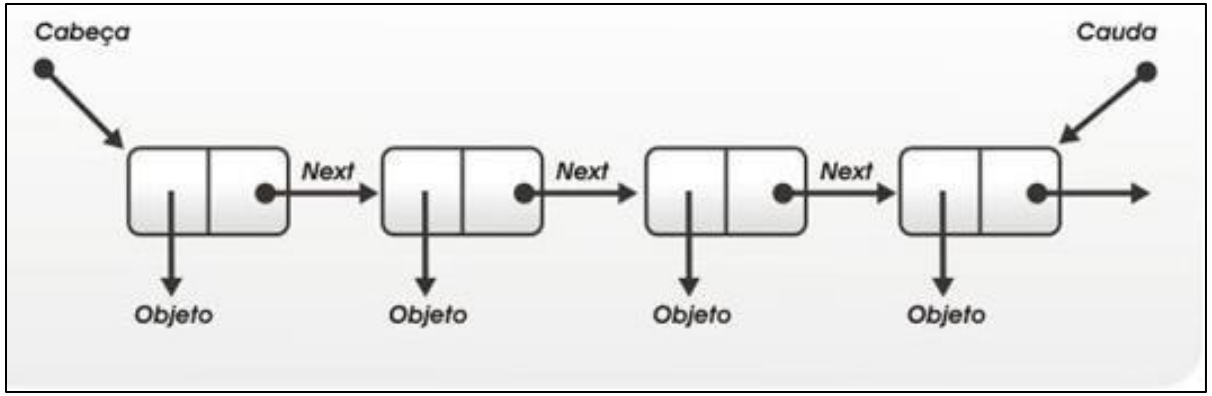


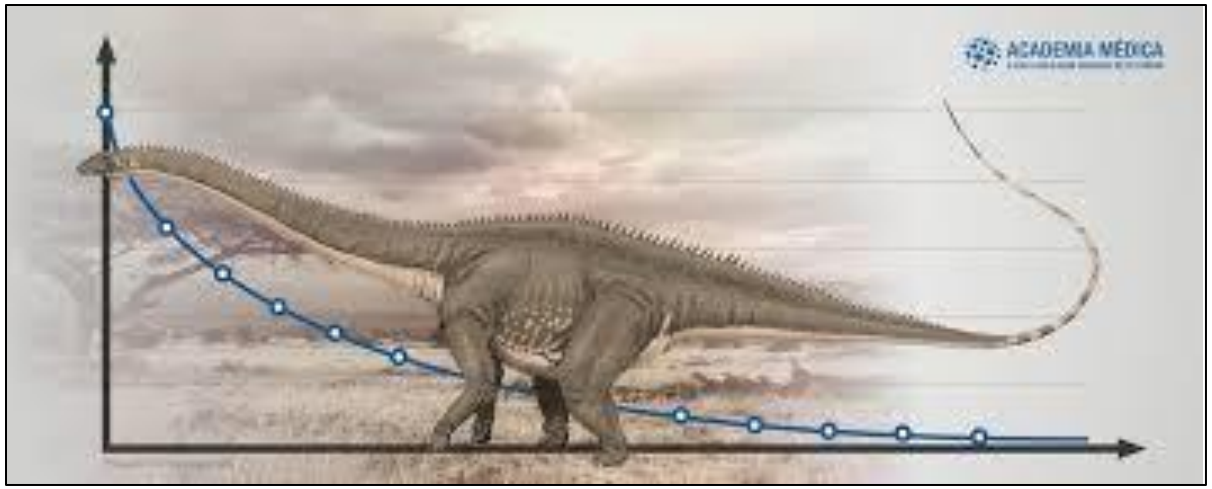
Organização de produtos num supermercado

- O que aconteceria se a reposição dos produtos fosse realizada apenas inserindo o novo produto e empurrando os mais antigos para trás?
- Os mais antigos permaneceriam lá trás e consequentemente, o prazo de validade iria vencer em algum momento.

Cabeça e Cauda

- **Cabeça** (Head): Na estrutura de uma lista, a cabeça refere-se ao primeiro elemento da lista. É o ponto de acesso inicial para a lista, a partir do qual outros elementos podem ser alcançados.
- **Cauda** (Tail): A cauda, por outro lado, é o restante da lista após a cabeça, ou seja, todos os elementos da lista, exceto o primeiro (a cabeça). Em algumas implementações de listas, a cauda pode ser uma sublista que começa após a cabeça.

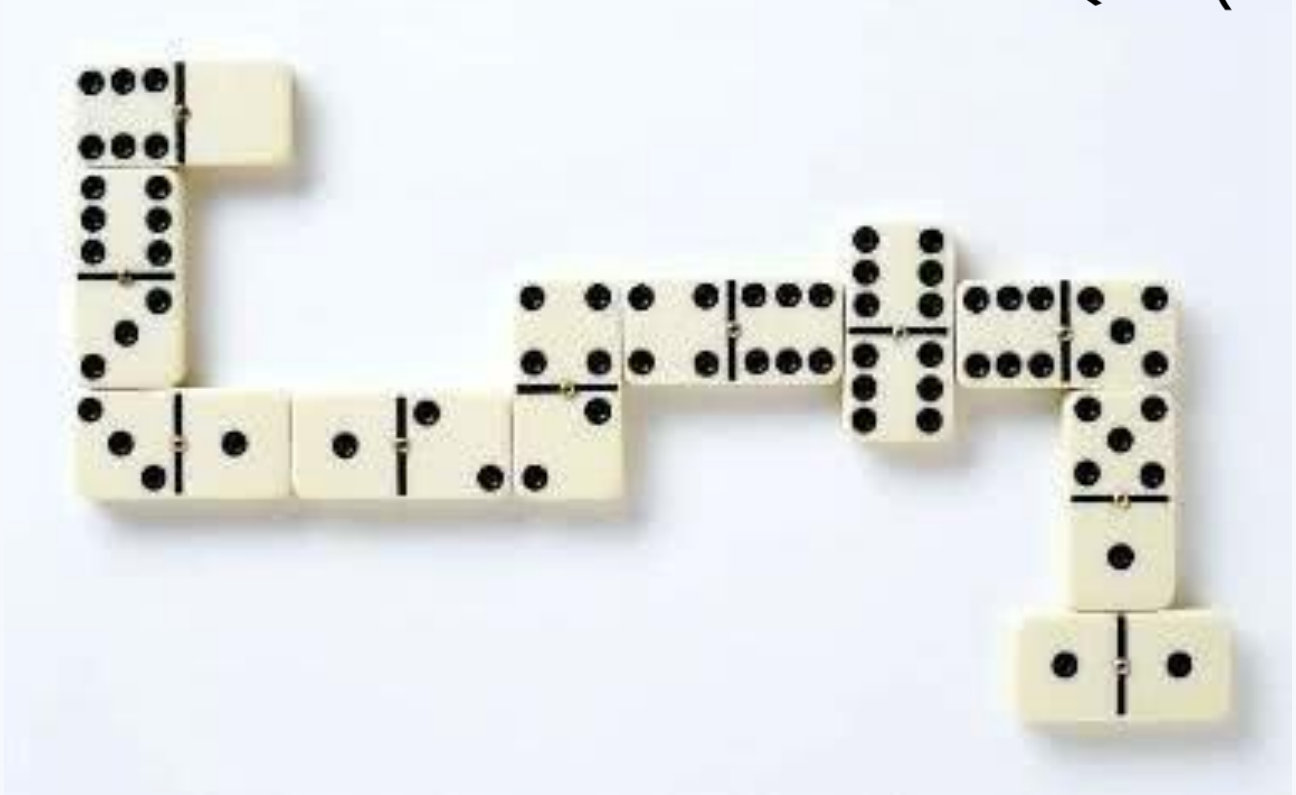





FILA DUPLA

Fila Dupla

- Lista linear em que inserções e remoções são feitas em ambos os extremos.
- Filas-duplas são também denominadas de DEQUE (Double-Ended QUEue).



Fila Dupla

Há também dois casos especiais de fila-dupla que são:

- **Fila-dupla de entrada restrita** (se a **inserção** for **restrita** a um único extremo).
- **Fila-dupla de saída restrita** (se a **remoção** for **restrita** a um único extremo).

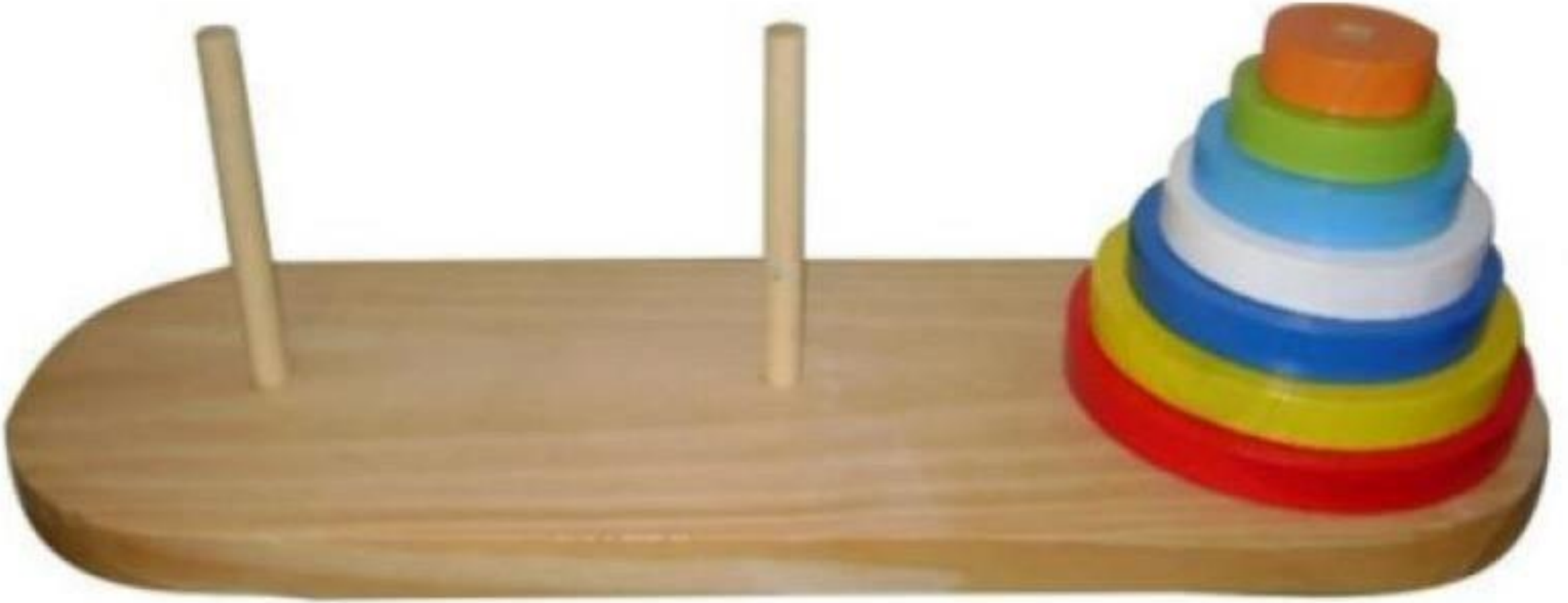
PILHA

Pilha

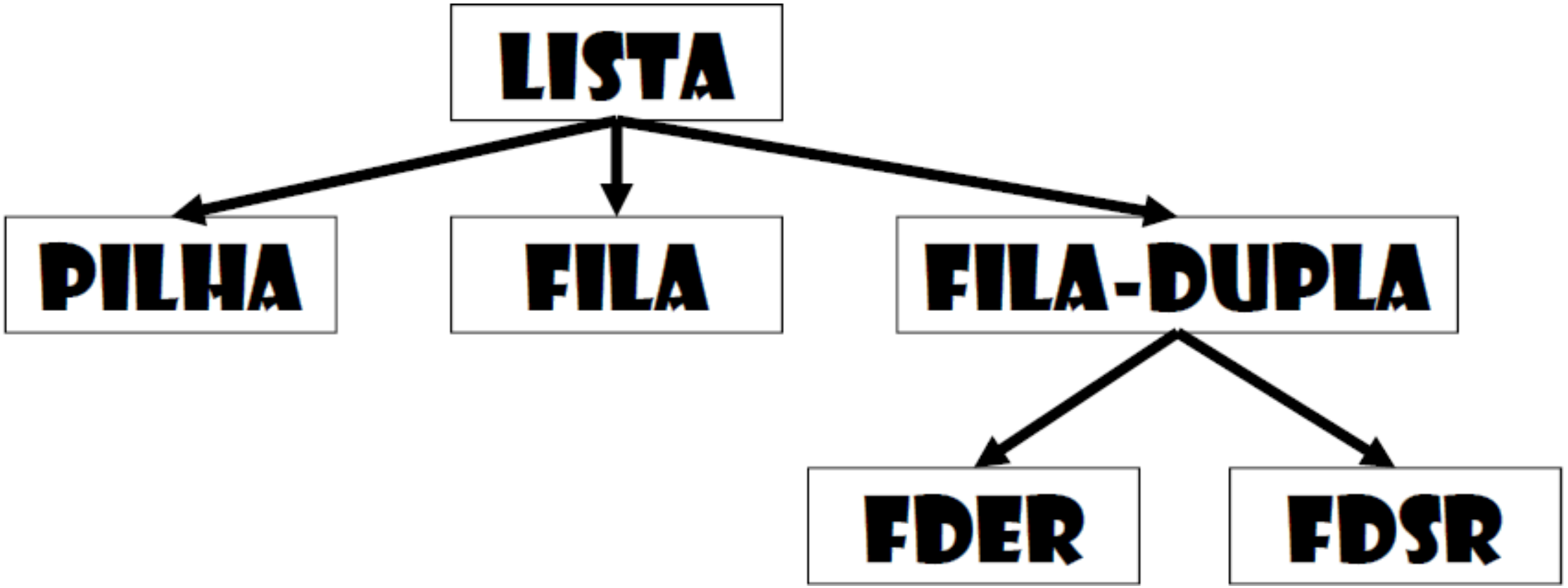
- Lista linear em que todas as inserções, remoções e acessos são feitos **num mesmo extremo**, denominado **topo**.
- Pilhas são também denominadas listas LIFO (*Last-In/ First-Out* ou *Último-que-Entra/ Primeiro-que-Sai* - UEPS).

Torre de Hanói

https://www.noas.com.br/ensino-fundamental-2/matematica/torre-de-hanoi/



Casos Especiais de Listas



Questão 1

Aplicada em: 2017 Banca: COPESE – UFPI Órgão: UFPI Prova: Analista de Tecnologia da Informação

Um conjunto ordenado de itens a partir do qual podem ser eliminados itens em uma extremidade e no qual podem ser inseridos itens na outra extremidade é denominado de...

a) fila

b) pilha

c) lista simples

d) lista encadeada

e) árvore

Questão 1

Resposta: A

Aplicada em: 2017 Banca: COPESE – UFPI Órgão: UFPI Prova: Analista de Tecnologia da Informação

Um conjunto ordenado de itens a partir do qual podem ser eliminados itens em uma extremidade e no qual podem ser inseridos itens na outra extremidade é denominado de...

a) fila

b) pilha

c) lista simples

d) lista encadeada

e) árvore

Considerações sobre a Implementação

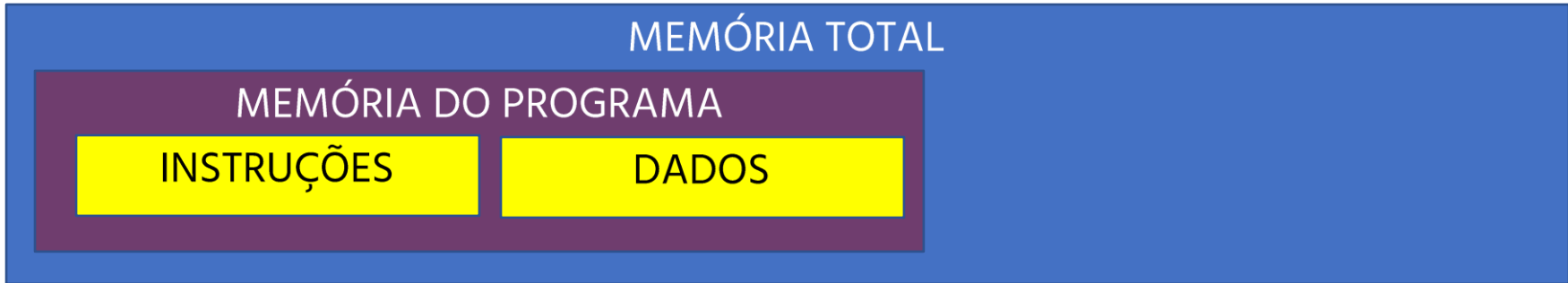
- Na **prática**, ao criar um programa que utiliza listas, **não implementamos todas as operações** (com eficiência) que podem ser realizadas numa lista linear, apenas **um conjunto reduzido** será necessário.
- É **difícil manter a eficiência** quando implementamos todas as operações.
- Por exemplo, uma implementação que **facilite o acesso a itens em posições arbitrárias** da lista certamente **dificultará** a **inserção e a remoção** de itens.

Como são armazenados os itens de uma lista na memória do computador?

Alocação de memória

Categorias de Alocação de Memória

- Antes de executar um programa, o computador precisa carregar seu código executável para a memória.
- Nesse momento **uma parte da memória total** disponível no sistema é **reservado para o uso do programa** e o restante fica livre (raramente um programa ocupa toda memória instalada no computador).
- Da área de **memória reservada, uma parte é destinada ao armazenamento de instruções** e o **restante ao armazenamento de dados**.



Categorias de Alocação de Memória

- Quem determina **quanto de memória** será usado para **instruções** é o **compilador**.
- Agora, a alocação de **áreas para armazenar dados (variáveis)**, é responsabilidade do **programador**.

Compilador

Programador

Categorias de Alocação de Memória

- Do ponto de vista do programador, a **alocação** de memória pode ser **classificada** nas **categorias** estática ou dinâmica.
- A partir dessas categorias, o **esquema de alocação das estruturas de dados** pode ser sequencial ou encadeado.

A alocação de memória estática e dinâmica estão relacionados à gestão de memória para armazenamento de dados.

Estático

Sem movimento, parado, imóvel

Alocação Estática

Refere-se ao processo de alocar espaço de memória durante a **fase de compilação do programa**. Isso significa que o tamanho da memória alocada é determinado em tempo de compilação e **permanece fixo durante toda a execução do programa**. Variáveis alocadas estaticamente são definidas em um escopo global ou local e existem durante toda a vida do programa.

- O tamanho da memória é conhecido antes do tempo de execução.
- A memória é alocada e liberada automaticamente pelo sistema.
- O escopo e a duração da variável estão determinados por onde ela é declarada (global ou local).
- Variáveis estáticas têm um espaço fixo na memória durante toda a execução do programa.

Dinâmico

Que se modifica continuamente.

Alocação Dinâmica

Envolve a alocação e liberação de memória **durante a execução do programa**, em tempo de execução. Isso permite que você aloque memória conforme necessário e libere essa memória quando não for mais necessária. A alocação dinâmica é frequentemente usada para criar estruturas de dados flexíveis, como listas encadeadas, árvores e matrizes de tamanho variável.

- O tamanho da memória pode ser determinado em tempo de execução.
- O programador é responsável por alocar e liberar a memória.
- É mais flexível, permitindo criar estruturas de dados dinâmicas.
- Requer cuidados para evitar vazamentos de memória ou acessos inválidos.

Questão 2

Em relação a tipos abstratos de dados, é correto afirmar que:

A) O TAD não encapsula a estrutura de dados para permitir que os usuários possam ter acesso a todas as operações disponibilizadas sobre esses dados.

B) Algumas pilhas admitem serem declaradas como tipos abstratos de dados.

C) Filas não permitem declaração como tipos abstratos de dados.

D) Os tipos abstratos de dados podem ser formados pela união de tipos de dados primitivos, mas não por outros tipos abstratos de dados.

E) São tipos de dados que escondem a sua implementação de quem o manipula; de maneira geral as operações sobre estes dados são executadas sem que se saiba como isso é feito.

Questão 2

Resposta: E

Em relação a tipos abstratos de dados, é correto afirmar que:

A) O TAD não encapsula a estrutura de dados para permitir que os usuários possam ter acesso a todas as operações disponibilizadas sobre esses dados.

B) Algumas pilhas admitem serem declaradas como tipos abstratos de dados.

C) Filas não permitem declaração como tipos abstratos de dados.

D) Os tipos abstratos de dados podem ser formados pela união de tipos de dados primitivos, mas não por outros tipos abstratos de dados.

E) São tipos de dados que escondem a sua implementação de quem o manipula; de maneira geral as operações sobre estes dados são executadas sem que se saiba como isso é feito.

Exercício

Implementação de uma Lista Sequencial com Alocação Dinâmica

Exercício

- Você foi designado para criar um programa em C que implemente uma lista sequencial utilizando alocação dinâmica de memória.
- A lista deve ser capaz de realizar operações básicas de uma lista sequencial, como inserir no início, no final e em uma posição específica, além de remover elementos e exibir o conteúdo da lista.

1. Estrutura da Lista Sequencial

- Crie uma estrutura para representar a lista sequencial.
- Essa estrutura deve conter um ponteiro que será usado para alocar dinamicamente um array de elementos (por exemplo, números inteiros) e um inteiro que representará a capacidade máxima da lista.

Parte 1 - Resposta

```c
#include <stdio.h>
#include <stdlib.h>

// Estrutura da lista sequencial
struct ListaSequencial {
    int *elementos;
    int capacidade;
    int tamanho;
};
```

2. Operações na Lista

Implemente funções para as seguintes operações:

2.1 Inicialização da lista sequencial.

2.2 Inserir um novo elemento no final da lista.

2.3 Exibir os elementos da lista.

2.4 Liberar a memória alocada para a lista.

2.1. Inicialização da lista sequencial

```
// Função para inicializar a lista
void inicializarLista (struct ListaSequencial *lista, int capacidade) {
    lista->elementos = (int *)malloc(capacidade * sizeof(int));
    lista->capacidade = capacidade;
    lista->tamanho = 0;
}
```

2.2. Inserir um novo elemento no final da lista

```
// Função para inserir um elemento no final da lista
void inserirNoFinal(struct ListaSequencial *lista, int elemento) {
    if (lista->tamanho < lista->capacidade) {
        lista->elementos[lista->tamanho] = elemento;
        lista->tamanho++;
    } else {
        printf("Lista cheia, incapaz de adicionar mais elementos.\n");
    }
}
```

2.3. Exibir os elementos da lista

```
// Função para exibir os elementos da lista
void exibirLista(struct ListaSequencial *lista) {
    printf("Elementos da lista: ");
    for (int i = 0; i < lista->tamanho; i++) {
        printf("%d ", lista->elementos[i]);
    }
    printf("\n");
}
```

2.4. Liberar a memória alocada para a lista

```
// Função para liberar a memória alocada para a lista
void liberarLista(struct ListaSequencial *lista) {
    free(lista->elementos);
    lista->capacidade = 0;
    lista->tamanho = 0;
}
```

3. Menu Interativo

Crie um menu interativo que permita ao usuário escolher e realizar as operações na lista sequencial.

3.1 Menu Interativo

```
int main() {
    struct ListaSequencial lista;
    int capacidade;
    printf("Digite a capacidade da lista: ");
    scanf("%d", &capacidade);

    inicializarLista(&lista, capacidade);

    int opcao, elemento;
```

3.2 Menu Interativo

```
do {
        printf("\nOpções:\n");
        printf("1. Inserir elemento no final da lista\n");
        printf("2. Exibir a lista\n");
        printf("3. Sair\n");
        printf("Escolha a operação: ");
        scanf("%d", &opcao);
```

3.3 Menu Interativo

```
switch (opcao) {
        case 1:
            printf("Digite o elemento a ser inserido: ");
            scanf("%d", &elemento);
            inserirNoFinal(&lista, elemento);
            break;
        case 2:
            exibirLista(&lista);
            break;
        case 3:
            break;
        default:
            printf("Opção inválida\n");
    }
} while (opcao != 3);
```

```
liberarLista(&lista);
    return 0;
}
```

4. Finalização do Programa

O programa deve liberar toda a memória alocada antes de encerrar.

Liberar a memória alocada para a lista

```
// Função para liberar a memória alocada para a lista
void liberarLista(struct ListaSequencial *lista) {
    free(lista->elementos);
    lista->capacidade = 0;
    lista->tamanho = 0;
}
```