GERMINARE
Escola de Negócios

# Lógica de Programação

## Estruturas de Dados e Complexidade dos Algoritmos

Introdução a Estruturas e Complexidade

**Germinare Tech**

Carlos Santi e Marcelo Modolo



GerminaTECH

GERMINARE
Escola de Negócios

GerminaTECH

# Introdução a Estruturas e Complexidade

Carlos Santi e Marcelo Modolo

# Conceitos Iniciais

- **Tipo de Dado**: define o conjunto de valores que uma variável pode assumir e as operações sobre esses valores.
  Exemplos: int, double, char
- **Tipo Abstrato de Dados (TAD)**: define o conjunto de valores e as operações sobre os valores, mas não define sua implementação.
  Exemplos: lista, pilha, fila, árvore
- **Estrutura de Dados (ED)**: é uma maneira particular de se implementar um Tipo Abstrato de Dados.
  Exemplos: sequencial, encadeado

GERMINARE
Escola de Negócios

# Tipos Abstratos de Dados

GERMINARE
Escola de Negócios

# Tipo Abstrato de Dados (TAD)

- Define o conjunto de valores e as operações que atuam sobre esses valores
- Não considera sua implementação computacional
- Não considera questões de eficiência em relação a tempo de execução e espaço na memória
- TADs que estudaremos: lista, pilha e fila

GERMINARE
Escola de Negócios

# Lista

- Conjunto de itens interligados cujas operações de inserção e remoção podem ser feitas em qualquer parte
- O conjunto mínimo de regras de negócio necessárias a funcionalidade de uma lista são:
  - Inserir um item a lista
  - Remover um item da lista
  - Encontrar um item na lista
  - Contar quantos itens pertencem à lista
  - Descobrir qual item está em uma dada posição

GERMINARE
Escola de Negócios

# Pilha

- Conjunto de itens interligados cujas operações de inserção e remoção só podem ser feitas no topo Inserção e remoção são baseadas no princípio LIFO (*last in, first out*) ou, em português, UEPS (último a entrar, primeiro a sair)
- Operações que se aplicam a todas as pilhas:
  - Inserir um item no topo da pilha
  - Remover um item do topo da pilha

GERMINARE
Escola de Negócios

# Fila

- Conjunto de itens interligados cuja operação de inserção é feita no final e a operação de remoção é feita no início
- Inserção e remoção são baseadas no princípio FIFO (*first in, first out*) ou, em português, PEPS (primeiro a entrar, primeiro a sair)
- Operações que se aplicam a todas as filas:
  - Inserir um item no final da fila
  - Remover um item do início da fila

GERMINARE
Escola de Negócios

# Estrutura de Dados

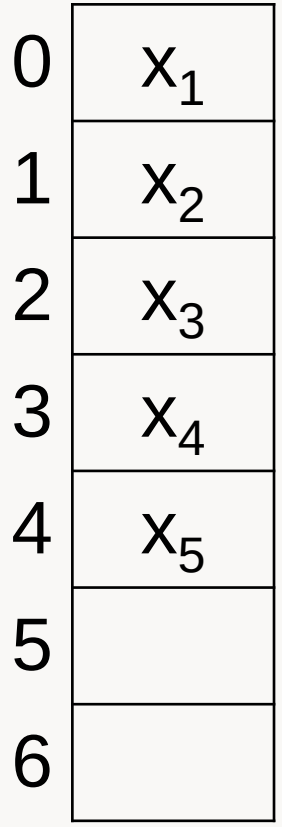
GERMINARE
Escola de Negócios

# Estrutura de Dados (ED)

- Maneira de organizar e representar computacionalmente os Tipos Abstratos de Dados
- A eficiência de um programa está diretamente relacionada à estrutura de dados utilizada
- Estruturas estudadas nesse módulo:
  - Sequencial
  - Encadeada

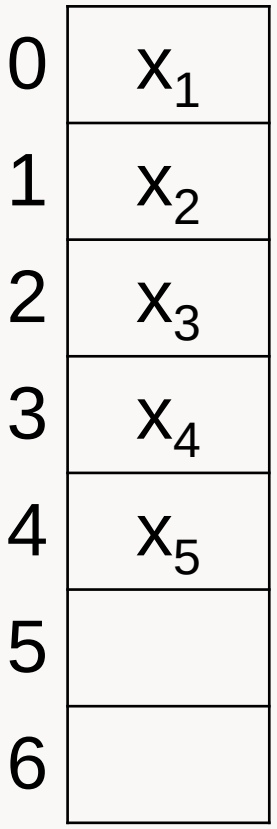
# Estrutura Sequencial

Exemplo de Representação Gráfica



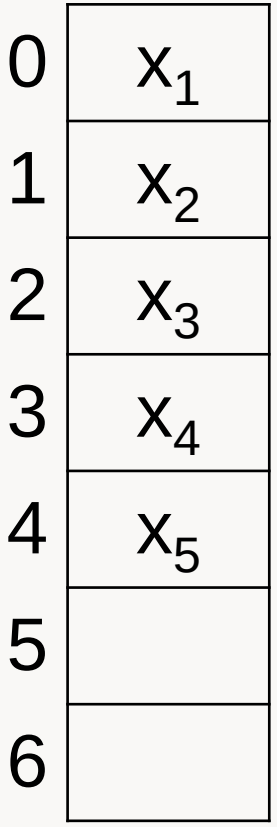
# Estrutura Sequencial

- Os itens são armazenados em posições contíguas da memória
- O tamanho da estrutura deve ser definido na sua criação
- A estrutura pode ser percorrida em qualquer direção
- Qualquer item pode ser acessado diretamente



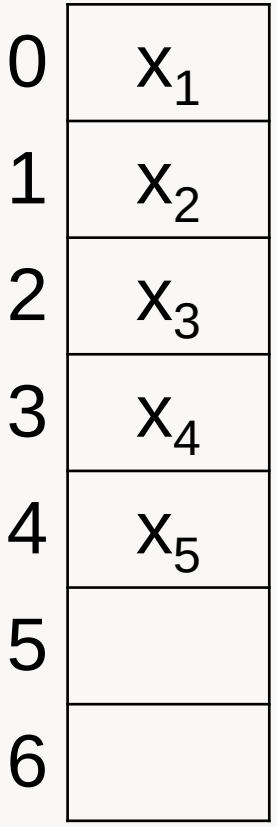
# Estrutura Sequencial

- **Pontos positivos**
  - Facilidade de acesso a cada item
  - Custo baixo e constante para inserção ao final dos itens ou remoção do último item



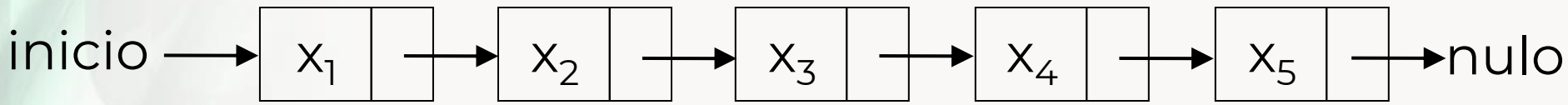
# Estrutura Sequencial

- **Pontos negativos**
  - Alto custo para inserir ou remover itens no meio, pois implica em deslocamento de outros itens
  - O tamanho fixo pode implicar em alocação de memória não utilizada ou falta de espaço



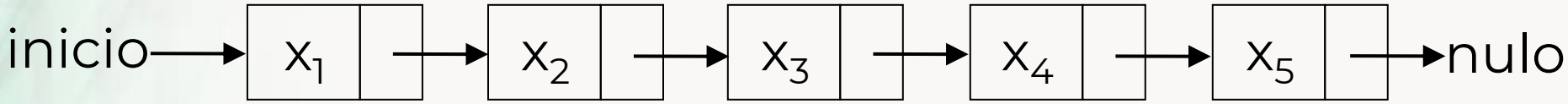
# Estrutura Encadeada

## Exemplo de Representação Gráfica



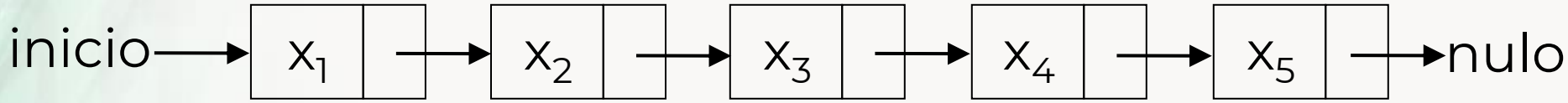
# Estrutura Encadeada

- Cada item é encadeado por um ponteiro que aponta para o item seguinte da estrutura
- Permite armazenamento em posições não contíguas de memória
- O acesso a um determinado item é feito passando por todos os itens que o precedem



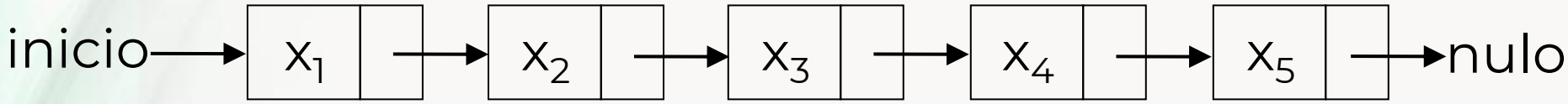
# Estrutura Encadeada

- **Pontos positivos**
  - Não tem desperdício de memória, pois a memória alocada pela estrutura é aquela utilizada pelos itens
  - Custo baixo para inserção ou remoção de itens



# Estrutura Encadeada

- **Ponto Negativo**
  - Custo alto para acessar um determinado item porque implica em passar por todos os itens que o precedem na estrutura



GERMINARE
Escola de Negócios

# Complexidade dos Algoritmos

GERMINARE
Escola de Negócios

# Complexidade dos Algoritmos

- A complexidade do algoritmo fornece a medida do trabalho envolvido na execução de um determinado algoritmo
- Uma possibilidade é considerar um conjunto de dados e seguir a execução do algoritmo para esses dados
- Neste caso, complexidade passa a ser uma função (f) do número de elementos (n) a serem processados
- Formato da função:

$$f(n) = \text{complexidade}$$

GERMINARE
Escola de Negócios

# Exemplo 1

- Determinar quantas vezes o laço abaixo é repetido

```
public static int exemplo1(){
    int cont = 0;
    for (int i = 1; i <= 1000; i++){
        cont++;
    }
    return cont;
}
```

GERMINARE
Escola de Negócios

# Exemplo 2

- Determinar quantas vezes o laço abaixo é repetido

```
public static int exemplo2(int n){
    int cont = 0;
    for (int i = 1; i <= n; i++){
        cont++;
    }
    return cont;
}
```

GERMINARE
Escola de Negócios

# Função nos Exemplos 1 e 2

- Número de repetições do exemplo 1 = 1000
- Número de repetições do exemplo 2 = n
- O número de repetições é proporcional a n
- Como a complexidade desse algoritmo está diretamente ligada ao número de repetições do laço podemos definir a função de complexidade como:

$$f(n) = n$$

GERMINARE
Escola de Negócios

# Exemplo 3

- Determinar quantas vezes o laço abaixo é repetido

```
public static int exemplo3(){
    int cont = 0;
    for (int i = 1; i <= 1000; i = i * 2){
        cont++;
    }
    return cont;
}
```

# Exemplo 4

- Determinar quantas vezes o laço abaixo é repetido

```
public static int exemplo4(int n){
    int cont = 0;
    for (int i = 1; i <= n; i = i * 2){
        cont++;
    }
    return cont;
}
```

GERMINARE
Escola de Negócios

# Função nos Exemplos 3 e 4

- Número de repetições do exemplo 3 = 10
- Se $2^9 = 512$ e $2^{10} = 1024$ e sabemos que a 10ª repetição faz sair do laço, podemos afirmar que:
- Número de repetições do exemplo 3 = $\log_2 1000$
- Número de repetições do exemplo 4 = $\log_2 n$
- O número de repetições é proporcional a $\log_2 n$
- Como a eficiência desse algoritmo também está diretamente ligada ao número de repetições do laço:

$$f(n) = \log_2 n$$

# Exemplo 5

- Determinar quantas vezes o laço abaixo é repetido

```
public static int exemplo5(int n){
        int cont = 0;
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= n; j++) {
                cont++;
            }
        }
        return cont;
 }
```

GERMINARE
Escola de Negócios

# Função do Exemplo 5

- Número de repetições do laço externo = n
- Número de repetições do laço interno = n
- Número de repetições do algoritmo = n x n

$$f(n) = n \times n$$

$$f(n) = n^2$$

# Exemplo 6

- Determinar quantas vezes o laço abaixo é repetido

```
public static int exemplo6(int n){
        int cont = 0;
        for (int i = 1; i <= n; i++) {
            cont++;
        }
        for (int j = 1; j <= n; j++) {
            cont++;
        }
        return cont;
}
```

GERMINARE
Escola de Negócios

# Função do Exemplo 6

- Número de repetições do laço 1 = n
- Número de repetições do laço 2 = n
- Número de repetições do algoritmo = n + n

$$f(n) = n + n$$

$$f(n) = 2n$$

GERMINARE
Escola de Negócios

# Perguntas???

GERMINARE
Escola de Negócios

# Exercícios

- Encontrar a função que representa a complexidade dos programas a seguir

GERMINARE
Escola de Negócios

# Exercício 1

```
public static int exercicio1(int n){
        int cont = 0;
        for (int i = 1; i <= n; i++) {
            cont++;
        }
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= n; j++) {
                cont++;
            }
        }
        return cont;
}
```

GERMINARE
Escola de Negócios

# Exercício 2

```
public static int exercicio2(int n){
        int cont = 0;
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= n; j++) {
                for(int k = 1; k <= n; k++) {
                    cont++;
                }
            }
        }
        return cont;
}
```

GERMINARE
Escola de Negócios

# Exercício 3

```
public static int exercicio3(int n){
        int cont = 0;
        for (int i = 1; i <= n; i++) {
            for (int j = i; j <= n; j++) {
                cont++;
            }
        }
        return cont;
}
```

# Exercício 4

```
public static int exercicio4(int n){
    int cont = 0;
    for (int i = 1; i <= n; i = i * 2) {
        cont++;
    }
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j++) {
            cont++;
        }
    }
    for (int i = 1; i <= n; i = i + 10) {
        cont++;
    }
    return cont;
}
```