# Resumo Completo de Estrutura de Dados (Baseado nos Arquivos Fornecidos)

## 1. Introdução e Conceitos Fundamentais

- **Estrutura de Dados:** É uma forma organizada de armazenar e gerenciar dados em um computador para que possam ser usados eficientemente. A escolha da estrutura correta afeta o desempenho do programa.
- **Variáveis e Tipos de Dados:** Fundamentais para armazenar informações (números inteiros, decimais, texto, booleanos).
- **Algoritmos:** Sequência de passos para realizar uma tarefa específica (ex: ordenar uma lista, buscar um item).

## 2. Manipulação de Arquivos em C (Aula 1)

- **Objetivo:** Ler dados de arquivos ou salvar dados em arquivos permanentemente.
- **Quando Usar:** Quando você precisa que os dados persistam após o programa ser fechado, ou quando precisa ler dados de uma fonte externa.
- **Funções Principais:**
    - `FILE *ponteiroArquivo;`: Declara um ponteiro para o arquivo.
    - `fopen("nome_arquivo.txt", "modo");`: Abre um arquivo. Retorna o ponteiro ou `NULL` se falhar.
        - **Modos Comuns:**
            - `"w"`: Write (escrita). Cria um arquivo novo ou sobrescreve um existente.
            - `"r"`: Read (leitura). Lê um arquivo existente.
            - `"a"`: Append (anexar). Adiciona dados ao final de um arquivo existente ou cria um novo.
    - `fprintf(ponteiroArquivo, "Formato %d %s\n", variavel_int, variavel_string);`: Escreve dados formatados no arquivo (similar ao `printf`).
    - `fscanf(ponteiroArquivo, "%d", &variavel_int);`: Lê dados formatados do arquivo (similar ao `scanf`). Retorna `EOF` (End Of File) quando chega ao final.
    - `fclose(ponteiroArquivo);`: Fecha o arquivo, liberando recursos. **Sempre feche os arquivos abertos!**

- **Exemplo Prático (Simplificado):**

C

```
#include <stdio.h>
#include <stdlib.h> // para exit()

int main() {
    FILE *arquivo;
    int numero = 10;
    char texto[] = "Exemplo";

    // --- Escrita ---
    arquivo = fopen("saida.txt", "w");
    if (arquivo == NULL) {
        printf("Erro ao abrir arquivo para escrita!\n");
        exit(1); // Termina o programa se não conseguir abrir
    }
    fprintf(arquivo, "Numero: %d\n", numero);
    fprintf(arquivo, "Texto: %s\n", texto);
    fclose(arquivo);
    printf("Dados gravados em saida.txt\n");

    // --- Leitura ---
    int numLido;
    char txtLido[50];
    arquivo = fopen("saida.txt", "r");
     if (arquivo == NULL) {
        printf("Erro ao abrir arquivo para leitura!\n");
        exit(1);
    }
    // Lê linha por linha (cuidado com a formatação exata)
    fscanf(arquivo, "Numero: %d\n", &numLido);
    fscanf(arquivo, "Texto: %s\n", txtLido); // Lê até o espaço ou
nova linha
    fclose(arquivo);
    printf("Dados lidos de saida.txt:\nNumero: %d\nTexto: %s\n",
numLido, txtLido);

    return 0;
}
```

## 3. Listas (`ArrayList` em Java, `List<T>` em C#, `list` em Python)

- **O que é:** Uma coleção ordenada de elementos onde o tamanho pode aumentar ou diminuir dinamicamente. Os elementos são acessados por um índice (posição).
- **Quando Usar:**
  - Quando você precisa armazenar uma coleção de itens.
  - Quando a ordem dos itens é importante.
  - Quando você não sabe quantos itens terá de antemão (tamanho dinâmico).
- **Operações Comuns:**
  - **Adicionar:** `add()` (Java/C#), `append()` (Python) - Geralmente adiciona no final.
  - **Acessar:** `get(indice)` (Java), `lista[indice]` (C#/Python).
  - **Remover:** `remove(indice)` ou `remove(objeto)` (Java/C#), `remove(valor)` ou `pop(indice)` (Python).
  - **Tamanho:** `size()` (Java), `Count` (C#), `len()` (Python).
  - **Verificar se Vazia:** `isEmpty()` (Java), `Count == 0` (C#), `len() == 0` (Python).
  - **Iterar (Percorrer):** Usando loops `for` ou `foreach`.
- **Exemplo Prático (Java - Aula 2, 3):**

Java

```
import java.util.ArrayList;
import java.util.List;
import java.util.Random;

public class ExemploLista {
    public static void main(String[] args) {
        // Criar lista de Integers
        List<Integer> numeros = new ArrayList<>();
        Random random = new Random();

        // Popular com 5 números aleatórios (0 a 99)
        for (int i = 0; i < 5; i++) {
            numeros.add(random.nextInt(100)); // Adiciona no final
        }
        System.out.println("Lista inicial: " + numeros); // Imprime
a lista toda
```

```
        // Acessar elemento no índice 2
        if (numeros.size() > 2) {
            System.out.println("Elemento no indice 2: " +
numeros.get(2));
        }

        // Remover o elemento no índice 0
        if (!numeros.isEmpty()) {
            numeros.remove(0);
        }
        System.out.println("Lista apos remover indice 0: " +
numeros);

        // Percorrer e imprimir cada elemento
        System.out.println("Elementos restantes:");
        for (Integer num : numeros) {
            System.out.println("- " + num);
        }

        System.out.println("Tamanho final da lista: " +
numeros.size());
    }
}
```

- **Exemplo Prático (C# - Aula 3):**

C#

```
using System;
using System.Collections.Generic; // Necessário para List<T>

public class ExemploListaCS
{
    public static void Main(string[] args)
    {
        // Criar lista de strings
        List<string> nomes = new List<string>();

        // Adicionar elementos
        nomes.Add("Ana");
        nomes.Add("Bruno");
        nomes.Add("Carlos");
        Console.WriteLine("Lista inicial:");
        nomes.ForEach(nome => Console.WriteLine($"- {nome}")); //
Outra forma de percorrer

        // Acessar elemento no índice 1
        if (nomes.Count > 1) {
             Console.WriteLine($"\nElemento no indice 1:
{nomes[1]}");
        }

        // Remover "Ana"
        nomes.Remove("Ana");
        Console.WriteLine("\nLista apos remover 'Ana':");
        foreach (string nome in nomes) {
            Console.WriteLine($"- {nome}");
        }

        Console.WriteLine($"\nTamanho final da lista:
{nomes.Count}");
    }
}
```

- **Listas de Listas (Aula 4):** Você pode ter uma lista onde cada elemento é, ele mesmo, outra lista. Útil para estruturas como matrizes ou dados agrupados.

Java

```
// Exemplo Conceitual Java (Lista de Listas de Inteiros)
List<List<Integer>> matriz = new ArrayList<>();

List<Integer> linha1 = new ArrayList<>();
linha1.add(1);
linha1.add(2);
matriz.add(linha1); // Adiciona a primeira linha

List<Integer> linha2 = new ArrayList<>();
linha2.add(3);
linha2.add(4);
matriz.add(linha2); // Adiciona a segunda linha

// Acessar o elemento na linha 0, coluna 1 (valor 2)
int elemento = matriz.get(0).get(1);
System.out.println("Elemento[0][1]: " + elemento);
```

## 4. Dicionários / Mapas (HashMap em Java)

- **O que é:** Uma coleção que armazena pares de **chave-valor**. Cada chave é única e usada para acessar rapidamente o valor associado. Não garante uma ordem específica dos elementos.
- **Quando Usar:**

    - Quando você precisa associar um valor a um identificador único (chave).
    - Para contagens de frequência (Ex: contar quantas vezes cada palavra aparece).
    - Para buscas rápidas baseadas em uma chave (Ex: encontrar dados de um aluno pelo seu ID).
- **Operações Comuns (Java HashMap):**
    - **Adicionar/Atualizar:** `put(chave, valor)` - Se a chave já existe, atualiza o valor.
    - **Acessar:** `get(chave)` - Retorna o valor associado à chave, ou `null` se a chave não existir.
    - **Verificar Chave:** `containsKey(chave)` - Retorna `true` se a chave existe.
    - **Remover:** `remove(chave)` - Remove o par chave-valor.
    - **Tamanho:** `size()`
    - **Iterar:** Pode-se iterar sobre as chaves (`keySet()`), sobre os valores (`values()`), ou sobre os pares (`entrySet()`).

- **Exemplo Prático (Java - Aula 4 - Contador de Disciplinas):**

Java

```
import java.util.HashMap;
import java.util.Map;
import java.util.List; // Para List.of

public class ExemploMap {
    public static void main(String[] args) {
        // Mapa para contar ocorrências de disciplinas (String ->
Integer)
        Map<String, Integer> contagemDisciplinas = new HashMap<>();
```

```
        List<String> disciplinasAlunos = List.of("Calculo",
"Fisica", "Calculo", "Algoritmos", "Fisica", "Calculo"); // Java 9+
List.of

        // Contar as ocorrências
        for (String disciplina : disciplinasAlunos) {
            // Forma mais curta usando getOrDefault (Java 8+):
            contagemDisciplinas.put(disciplina,
contagemDisciplinas.getOrDefault(disciplina, 0) + 1);
        }

        System.out.println("Contagem de Alunos por Disciplina:");
        // Iterar sobre as chaves para mostrar o resultado
        for (String chave : contagemDisciplinas.keySet()) {
            System.out.println("- " + chave + ": " +
contagemDisciplinas.get(chave) + " aluno(s)");
        }

        // Acessar uma contagem específica
        System.out.println("\nAlunos em Calculo: " +
contagemDisciplinas.get("Calculo"));
    }
}
```

## 5. Lista de Objetos (POO + Estruturas de Dados - Aula 6)

- **O que é:** Em vez de listas de tipos básicos (int, String), você cria listas que armazenam instâncias (objetos) de classes que você mesmo define.
- **Quando Usar:** Quando você precisa representar e armazenar coleções de entidades do mundo real ou estruturas de dados complexas que possuem múltiplos atributos (Ex: uma lista de Alunos, onde cada Aluno tem ID, Nome, Curso, etc.).
- **Conceito Chave:**
    - **Classe:** Molde/Modelo para criar objetos (Ex: `class Aluno { int id; String nome; }`).
    - **Objeto:** Instância de uma classe (Ex: `Aluno a1 = new Aluno(); a1.id = 1; a1.nome = "Maria";`).

    - **Chave Primária (Conceito):** Um atributo (ou conjunto de atributos) que identifica unicamente cada objeto dentro da coleção (Ex: o `id` do Aluno). Garante que não haja dois alunos com o mesmo ID na lista.
- **Exemplo Prático (Java - Aula 6):**

Java

```
import java.util.ArrayList;
import java.util.List;
import java.util.Objects; // Para equals e hashCode

// 1. Definir a Classe (Molde)
class Produto {
    int codigo; // Atuará como chave primária conceitual
    String nome;
    double preco;

    public Produto(int codigo, String nome, double preco) {
        this.codigo = codigo;
        this.nome = nome;
        this.preco = preco;
    }

    // É importante sobrescrever toString() para facilitar a
impressão
    @Override
    public String toString() {
```

```
        return "Produto [codigo=" + codigo + ", nome=" + nome + ",
preco=" + String.format("%.2f", preco) + "]";
    }

    // Para buscas ou remoções baseadas no objeto, é bom ter
equals/hashCode
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Produto produto = (Produto) o;
        return codigo == produto.codigo; // Compara apenas pelo
código (chave primária)
    }

    @Override
    public int hashCode() {
        return Objects.hash(codigo); // Usa apenas o código para o
hash
    }
}

// 2. Usar a Classe em uma Lista
public class ExemploListaObjetos {
    public static void main(String[] args) {
        List<Produto> estoque = new ArrayList<>();

        // Criar objetos (instâncias) e adicionar à lista
        Produto p1 = new Produto(101, "Teclado", 150.00);
        Produto p2 = new Produto(102, "Mouse", 80.00);
        Produto p3 = new Produto(103, "Monitor", 1200.00);

        estoque.add(p1);
        estoque.add(p2);
        estoque.add(p3);

        System.out.println("Estoque de Produtos:");
        for (Produto p : estoque) {
            System.out.println(p); // Usa o toString() da classe
Produto
        }
```

```
        // Buscar um produto pelo código (simulação, iteração
necessária)
        int codigoBusca = 102;
        Produto encontrado = null;
        for (Produto p : estoque) {
            if (p.codigo == codigoBusca) {
                encontrado = p;
                break; // Para a busca assim que encontrar
            }
        }

        if (encontrado != null) {
            System.out.println("\nProduto encontrado: " +
encontrado.nome);
        } else {
            System.out.println("\nProduto com codigo " + codigoBusca
+ " nao encontrado.");
        }

        // Remover um produto (ex: pelo objeto p1)
        // Funciona melhor se equals/hashCode estiver implementado
comparando a chave primária
        estoque.remove(p1);
        System.out.println("\nEstoque apos remover o Teclado:");
        estoque.forEach(p -> System.out.println(p)); // Outra forma
de iterar
    }
}
```

- **Exemplo Prático (C# - Aula 6):**

C#

```
using System;
using System.Collections.Generic;
using System.Linq; // Para FirstOrDefault

// 1. Definir a Classe
public class Cliente
{
    public int Id { get; set; } // Chave primária
    public string Nome { get; set; }
```

```
    public string Cidade { get; set; }

    // Construtor
    public Cliente(int id, string nome, string cidade)
    {
        Id = id;
        Nome = nome;
        Cidade = cidade;
    }

    // Sobrescrever ToString para facilitar a impressão
    public override string ToString()
    {
        return $"Cliente [Id={Id}, Nome={Nome}, Cidade={Cidade}]";
    }

    // Opcional: Sobrescrever Equals e GetHashCode para comparações
por Id
    public override bool Equals(object obj)
    {
        return obj is Cliente cliente && Id == cliente.Id;
    }

    public override int GetHashCode()
    {
        return HashCode.Combine(Id);
    }
}

// 2. Usar a Classe em uma Lista
public class ExemploListaObjetosCS
{
    public static void Main(string[] args)
    {
        List<Cliente> cadastro = new List<Cliente>();

        // Criar objetos e adicionar à lista
        cadastro.Add(new Cliente(1, "Fernanda", "Porto Alegre"));
        cadastro.Add(new Cliente(2, "Ricardo", "Sao Paulo"));
        cadastro.Add(new Cliente(3, "Mariana", "Porto Alegre"));

        Console.WriteLine("Cadastro de Clientes:");
```

```
        foreach (var cliente in cadastro)
        {
            Console.WriteLine(cliente); // Usa o ToString()
        }

        // Buscar um cliente pelo Id usando LINQ (mais avançado, mas
comum em C#)
        int idBusca = 2;
        // Busca o primeiro que satisfaz a condição ou null se não
encontrar
        Cliente encontrado = cadastro.FirstOrDefault(c => c.Id ==
idBusca);

        if (encontrado != null)
        {
            Console.WriteLine($"\nCliente encontrado:
{encontrado.Nome} de {encontrado.Cidade}");
        }
        else
        {
            Console.WriteLine($"\nCliente com Id {idBusca} nao
encontrado.");
        }

         // Remover um cliente (ex: o cliente com Id 1)
         Cliente clienteParaRemover = cadastro.FirstOrDefault(c =>
c.Id == 1);
         if(clienteParaRemover != null) {
            cadastro.Remove(clienteParaRemover); // Usa Equals para
encontrar
         }

         Console.WriteLine("\nCadastro apos remover cliente Id 1:");
         cadastro.ForEach(c => Console.WriteLine(c));
    }
}
```

- **Exemplo Prático (Python - Aula 6):**

Python

```
# 1. Definir a Classe
class Funcionario:
```

```
    def __init__(self, matricula, nome, salario):
        self.matricula = matricula # Chave primária
        self.nome = nome
        self.salario = salario

    # Método especial para representação em string (similar ao
toString)
    def __str__(self):
        return f"Funcionario [Matricula={self.matricula},
Nome={self.nome}, Salario={self.salario:.2f}]"

    # Método especial para comparação (similar ao equals)
    def __eq__(self, other):
        if not isinstance(other, Funcionario):
            return NotImplemented
        return self.matricula == other.matricula # Compara pela
matrícula

    # Opcional: def __hash__(self): return hash(self.matricula) # Se
usar em sets ou dict keys

# 2. Usar a Classe em uma Lista
if __name__ == "__main__":
    lista_funcionarios = []

    # Criar objetos e adicionar à lista
    f1 = Funcionario(10, "Joao Silva", 3500.0)
    f2 = Funcionario(11, "Beatriz Costa", 4200.50)
    f3 = Funcionario(12, "Pedro Alves", 3800.0)

    lista_funcionarios.append(f1)
    lista_funcionarios.append(f2)
    lista_funcionarios.append(f3)

    print("Lista de Funcionarios:")
    for func in lista_funcionarios:
        print(func) # Usa o __str__

    # Buscar um funcionário pela matrícula (iteração)
    matricula_busca = 11
    encontrado = None
    for func in lista_funcionarios:
```

```
        if func.matricula == matricula_busca:
            encontrado = func
            break
  if encontrado:
        print(f"\nFuncionario encontrado: {encontrado.nome}")
    else:
        print(f"\nFuncionario com matricula {matricula_busca} nao
encontrado.")

    # Remover um funcionário (ex: o objeto f1)
    try:
        lista_funcionarios.remove(f1) # Usa o __eq__ para encontrar
        print("\nLista apos remover Joao Silva:")
        for func in lista_funcionarios:
            print(func)
    except ValueError:
        print("\nFuncionario f1 nao estava na lista para ser
removido.")
```

# Mini-Consulta de Sintaxe (Java, C#, Python, C)

*(Com espaçamento duplo entre linhas para melhor visualização no Bloco de Notas)*

## Java

### Lista (`ArrayList`)

- **Importar:** `import java.util.ArrayList; import java.util.List;`
- **Criar:** `List<Tipo> lista = new ArrayList<>();` (Ex: `List<String> nomes = new ArrayList<>();`)
- **Adicionar:** `lista.add(elemento);`
- **Acessar:** `Tipo elemento = lista.get(indice);`
- **Remover (índice):** `lista.remove(indice);`
- **Remover (objeto):** `lista.remove(objeto);`
- **Tamanho:** `int tam = lista.size();`
- **Vazia:** `boolean vazia = lista.isEmpty();`
- **Percorrer (foreach):** `for (Tipo item : lista) { ... }`

- **Percorrer (índice):** `for (int i = 0; i < lista.size(); i++) { lista.get(i); ... }`

## Mapa (HashMap)

- **Importar:** `import java.util.HashMap; import java.util.Map;`
- **Criar:** `Map<TipoChave, TipoValor> mapa = new HashMap<>();` (Ex: `Map<Integer, String> alunos = new HashMap<>();`)
- **Adicionar/Atualizar:** `mapa.put(chave, valor);`
- **Acessar:** `TipoValor valor = mapa.get(chave);` (`null` se não existe)
- **Verificar Chave:** `boolean existe = mapa.containsKey(chave);`
- **Acessar Seguro:** `TipoValor val = mapa.getOrDefault(chave, valorPadrao);` (Java 8+)
- **Remover:** `mapa.remove(chave);`
- **Tamanho:** `int tam = mapa.size();`
- **Percorrer Chaves:** `for (TipoChave k : mapa.keySet()) { mapa.get(k); ... }`
- **Percorrer Entradas:** `for (Map.Entry<TipoChave, TipoValor> entry : mapa.entrySet()) { entry.getKey(); entry.getValue(); ... }`

## Foco em Arquivos (Java IO Básico)

- **Importar:** `import java.io.*;` (Ou classes específicas como `BufferedReader`, `FileReader`, `PrintWriter`, `FileWriter`, `IOException`)
- **Escrever Linhas (Try-with-resources - fecha automaticamente):**

Java
```
try (PrintWriter writer = new PrintWriter(new
FileWriter("saida.txt"))) {
    writer.println("Primeira linha.");
    writer.printf("Numero: %d\n", 10);
} catch (IOException e) {
    System.err.println("Erro ao escrever no arquivo: " +
e.getMessage());
}
```

- **Ler Linhas (Try-with-resources - fecha automaticamente):**

Java
```
try (BufferedReader reader = new BufferedReader(new
FileReader("entrada.txt"))) {
```

```
    String linha;
    while ((linha = reader.readLine()) != null) {
        System.out.println("Lido: " + linha);
        // Processar a linha...
    }
} catch (IOException e) {
    System.err.println("Erro ao ler o arquivo: " + e.getMessage());
}
```

- **Modo Anexar (Append):** `try (PrintWriter writer = new PrintWriter(new FileWriter("saida.txt", true))) { ... }` // O `true` habilita append

### Classe Simples

- **Definir:**

Java
```
class Produto {
    int codigo; String nome;
    public Produto(int c, String n) { codigo = c; nome = n; }
    @Override public String toString() { return "Produto[...]"; }
    // @Override public boolean equals(Object o) { ... }
    // @Override public int hashCode() { ... }
}
```

- **Criar Objeto:** `Produto p = new Produto(101, "Teclado");`

## C#

### Lista (`List<T>`)

- **Importar:** `using System.Collections.Generic;`
- **Criar:** `List<Tipo> lista = new List<Tipo>();` (Ex: `List<int> numeros = new List<int>();`)
- **Adicionar:** `lista.Add(elemento);`
- **Acessar:** `Tipo elemento = lista[indice];`
- **Remover (índice):** `lista.RemoveAt(indice);`

- **Remover (valor):** `lista.Remove(elemento);`
- **Tamanho:** `int tam = lista.Count;`
- **Vazia:** `bool vazia = lista.Count == 0;`
- **Percorrer (foreach):** `foreach (Tipo item in lista) { ... }`
- **Percorrer (índice):** `for (int i = 0; i < lista.Count; i++) { lista[i]; ... }`

## Dicionário (`Dictionary<TKey, TValue>`)

- **Importar:** `using System.Collections.Generic;`
- **Criar:** `Dictionary<TipoChave, TipoValor> dict = new Dictionary<TipoChave, TipoValor>();` (Ex: `Dictionary<string, double> precos;`)
- **Adicionar (Exceção se existe):** `dict.Add(chave, valor);`
- **Adicionar/Atualizar:** `dict[chave] = valor;`
- **Acessar (Exceção se não existe):** `TipoValor valor = dict[chave];`
- **Acessar Seguro:** `bool achou = dict.TryGetValue(chave, out TipoValor valor);`
- **Verificar Chave:** `bool existe = dict.ContainsKey(chave);`
- **Remover:** `dict.Remove(chave);`
- **Tamanho:** `int tam = dict.Count;`
- **Percorrer Pares:** `foreach (KeyValuePair<TipoChave, TipoValor> par in dict) { par.Key; par.Value; ... }`

## Classe Simples

- **Definir:**

C#
```
public class Cliente {
    public int Id { get; set; }
    public string Nome { get; set; }
    public Cliente(int id, string nome) { Id = id; Nome = nome; }
    public override string ToString() { return $"Cliente[...]"; }
    // public override bool Equals(object obj) { ... }
    // public override int GetHashCode() { ... }
}
```

- **Criar Objeto:** `Cliente c = new Cliente(1, "Ana");`

# Python

## Lista (`list`)

- **Criar:** `lista = []`
- **Adicionar:** `lista.append(elemento)`
- **Acessar:** `elemento = lista[indice]`
- **Remover (índice):** `del lista[indice]` ou `elem = lista.pop(indice)`
- **Remover (valor):** `lista.remove(valor)`
- **Tamanho:** `tam = len(lista)`
- **Vazia:** `vazia = not lista`
- **Percorrer:** `for item in lista: ...`
- **Percorrer (índice):** `for i in range(len(lista)): lista[i] ...`

## Dicionário (`dict`)

- **Criar:** `dicionario = {}`
- **Adicionar/Atualizar:** `dicionario[chave] = valor`
- **Acessar (KeyError se não existe):** `valor = dicionario[chave]`
- **Acessar Seguro:** `valor = dicionario.get(chave)` (None se não existe)
- **Acessar Seguro (padrão):** `valor = dicionario.get(chave, default_val)`
- **Verificar Chave:** `existe = chave in dicionario`
- **Remover (KeyError se não existe):** `del dicionario[chave]` ou `val = dicionario.pop(chave)`
- **Tamanho:** `tam = len(dicionario)`
- **Percorrer Chaves:** `for k in dicionario: dicionario[k] ...`
- **Percorrer Valores:** `for v in dicionario.values(): ...`
- **Percorrer Itens:** `for k, v in dicionario.items(): ...`

## Classe Simples

- **Definir:**

Python

```
class Funcionario:
    def __init__(self, mat, nome):
        self.matricula = mat
        self.nome = nome
    def __str__(self):
```

```
        return f"Funcionario[...]"
    # def __eq__(self, other): ...
```

- **Criar Objeto:** `f = Funcionario(100, "Carlos")`

## C (Foco em Arquivos)

- **Incluir:** `#include <stdio.h>`
- **Ponteiro:** `FILE *arquivo;`
- **Abrir (Escrita):** `arquivo = fopen("arq.txt", "w");`
- **Abrir (Leitura):** `arquivo = fopen("arq.txt", "r");`
- **Abrir (Anexar):** `arquivo = fopen("arq.txt", "a");`
- **Verificar Erro:** `if (arquivo == NULL) { /* erro */ }`
- **Escrever:** `fprintf(arquivo, "formato %d", var);`
- **Ler:** `fscanf(arquivo, "%d", &var);` (Retorna itens lidos ou EOF)
- **Fechar:** `fclose(arquivo);` // **Essencial!**