1. Introdução.pdf

- **Foco Principal:** Perfil do Profissional de TI e Conceitos Básicos.
- **Pontos-chave:** O mercado de TI busca profissionais com **conhecimentos técnicos e visão estratégica de negócios**. O técnico em informática tem alta procura em diversos setores (público e privado), mas a **qualificação e certificação** são cada vez mais necessárias. Apresenta conceitos iniciais de **Dados, Informação, Sistemas, Lógica e Fluxograma**.

2. Consistências (Comparações).pdf

- **Foco Principal:** Expressões Lógicas e Estruturas Condicionais.
- **Pontos Chave:**
  - Explica os **Operadores Comparativos** (>, <, >=, <=, ==, !=) usados para gerar resultados booleanos (Verdadeiro ou Falso).
  - Detalha os **Operadores Lógicos** (&& - E, || - OU, ! - NÃO) para combinar expressões.
  - Demonstra a **Estrutura Condicional** **if/else** para executar blocos de código com base na consistência das expressões. O exemplo prático é o cálculo e classificação do **Índice de Massa Corporal (IMC)**.

3. Estruturas de Repetição .pdf

- **Foco Principal:** Estruturas Repetitivas (Laços/Loops) em C#.
- **Pontos Chave:**
  - **while (Enquanto):** Repete um bloco de comandos *enquanto* a condição for verdadeira.
  - **for (Para):** Estrutura ideal quando o número de repetições é conhecido.
  - **do-while (Faça-Enquanto):** Executa o bloco *pelo menos uma vez* e, em seguida, avalia a condição. É executado uma ou mais vezes.

4. Funções _ Métodos.pdf

- **Foco Principal:** Conceitos de Funções, Métodos e Orientação a Objetos.
- **Pontos Chave:**
  - Define **Funções/Métodos** como processamentos que promovem **modularização, delegação e reaproveitamento** do código. Em Orientação a Objetos, são chamados de "métodos".
  - Introduz o conceito de **Funções Locais** (métodos privados aninhados).
  - Apresenta os pilares da **Programação Orientada a Objetos (POO)**, onde programas são arquitetados através de objetos (instâncias de classes) que interagem entre si, definindo comportamentos (métodos) e estados (atributos).
  - Explica a diferença na alocação de memória: **Stack** (Pilha) para variáveis temporárias/estáticas e **Heap** (Montão) para alocação dinâmica de variáveis.

5. Funções Matemática C#.pdf

- **Foco Principal:** Uso de Funções Matemáticas Nativas do C#.
- **Pontos-chave:** O C# oferece a classe `Math` com funções para desenvolver rotinas específicas. Exemplos de uso prático:
  - `Math.Sqrt(x)`: Raiz quadrada.
  - `Math.Pow(x, y)`: Potenciação (x elevado a y).
  - `Math.Abs(x)`: Valor absoluto (módulo, sem sinal).
  - O documento ilustra a aplicação dessas funções na solução da **Equação de 2º Grau (Bhaskara)** e no cálculo de área e preço de um terreno.

6. Vetores & Matrizes.pdf

- **Foco Principal:** Coleções de Dados Indexadas (Arrays).
- **Pontos Chave:**
  - **Vetor:** Coleção de dados de **tamanho fixo, indexada, unidimensional e homogênea** (mesmo tipo de dado). A primeira posição é sempre o índice `0`. Deve ser alocado previamente (`new`).
  - **Matriz:** Coleção de dados que pode ter **mais de uma dimensão** (bidimensional, tridimensional).

7. Desenvolvendo Utilitários (System.IO).pdf

- **Foco Principal:** Manipulação de Arquivos e Diretórios (Namespace `System.IO`).
- **Pontos-chave:** O namespace `System.IO` no .NET contém classes e métodos para operações básicas de **leitura (READ) e escrita (WRITE)** em arquivos e diretórios.
  - **Classe `File`:** Métodos **estáticos** para operações simples (criação, cópia, exclusão). Adequado para operações simples, pois realiza análise de segurança a cada ação, o que pode impactar a performance em operações complexas.
  - **Classe `FileInfo`:** Propriedades e métodos de **instância**. Requer a declaração e instância (`new`). Recomendado para operações mais complexas, dinâmicas ou de alta frequência.

8. Técnicas de Conexão em Banco de Dados.pdf

- **Foco Principal:** Evolução e Técnicas de Acesso a Banco de Dados.
- **Pontos Chave:**
  - **Evolução:** Detalha a evolução das técnicas de acesso a dados: desde a necessidade de conhecimento de APIs específicas de *drivers* (DBLIB), passando pelo padrão intermediário **ODBC** (Open DataBase Connectivity) em 1990, até o **OLEDB** e a arquitetura **UDA** da Microsoft.
  - **ADO (ActiveX Data Objects):** API de alto nível para acesso a dados.

- **Processo Básico ADO:** Criar objeto de conexão, criar objeto de conjunto de registros, abrir a conexão e, em seguida, efetuar as operações.
- **Componentes ADO.NET:** Incluem objetos como Connection (gerencia conexão), Command (executa comandos SQL), DataReader (leitura de dados somente para frente) e DataAdapter (preenche DataSet).

9. Método override (Substituir_Modificador).pdf

- **Foco Principal:** Programação Orientada a Objetos - Sobrescrita de Método (override).
- **Pontos Chave:**
  - Demonstra como **sobrescrever (override) o método padrão Object.ToString()** em uma classe C#.
  - **Objetivo:** Alterar o comportamento padrão para retornar uma representação personalizada de um objeto (Produto).
  - A implementação prática exibe os detalhes do produto, incluindo Descrição, Preço Unitário, Quantidade e Valor Total em Estoque (calculado por ValorEmEstoque()).
  - O programa também inclui métodos para manipulação de estoque (AdicionarEstoque e RemoverEstoque).