# 🧠 Nível 1: Desafios com Laços de Repetição (O poder do `for` e `while`)

*Excelente trabalho até aqui! Agora que você domina as decisões com `if` e `else`, vamos aprender a repetir tarefas. Laços de repetição são a base para processar listas de dados, validar entradas de usuário e muito mais.*

**1. Tabuada com `for`:**

- **Objetivo:** Solicitar um número inteiro ao usuário e exibir a tabuada de multiplicação desse número, do 1 ao 10.

**Exemplo de Saída:** Para a entrada `7`, o programa deve exibir:
7 x 1 = 7
7 x 2 = 14
...
7 x 10 = 70

- 
- **Tópicos:** Laço `for`, variáveis (int), entrada/saída de dados, operadores aritméticos.

**2. Soma com `while`:**

- **Objetivo:** Criar um programa que solicite números ao usuário e os some. O programa deve parar de solicitar números quando o usuário digitar `0` e, no final, exibir a soma total.
- **Tópicos:** Laço `while`, variáveis, entrada/saída, operadores aritméticos e relacionais, estrutura condicional.

**3. Validação de Senha:**

- **Objetivo:** Simular uma tela de login. O programa deve ter uma senha secreta (ex: "1234"). Peça ao usuário para digitar a senha. Enquanto a senha estiver incorreta, mostre "Senha incorreta. Tente novamente." e peça a senha de novo. Quando a senha estiver correta, mostre "Acesso permitido!".
- **Tópicos:** Laço `do-while` (ideal para este caso, pois executa ao menos uma vez), comparação de strings, variáveis.

**4. Contagem de Números Pares:**

- **Objetivo:** Pedir ao usuário para inserir 10 números inteiros. No final, o programa deve informar quantos desses números eram pares.
- **Tópicos:** Laço `for`, operador módulo (`%`), variáveis de contagem (acumulador), condicionais `if`.

---

# 🚀 Nível 2: Desafios com Funções e Vetores (Organizando o código)

*Parabéns! Você já sabe tomar decisões e repetir tarefas. Agora, vamos organizar nosso código em blocos lógicos e reutilizáveis (Funções) e aprender a trabalhar com coleções de dados (Vetores/Arrays).*

**1. Calculadora com Funções:**

- **Objetivo:** Pegar o exercício da "Calculadora com Menu" da lista anterior e refatorá-lo. Crie uma função para cada operação: `somar(a, b)`, `subtrair(a, b)`, `multiplicar(a, b)` e `dividir(a, b)`. O `main` ficará responsável apenas por ler os dados, chamar a função correta e exibir o resultado.
- **Tópicos:** Criação e chamada de funções, passagem de parâmetros, retorno de valores, modularização de código.

**2. Maior Elemento de um Vetor:**

- **Objetivo:** Crie uma função chamada `encontrarMaior` que recebe um vetor de números inteiros como parâmetro e retorna o maior número encontrado nesse vetor. No `main`, crie um vetor, preencha-o e chame a função para exibir o resultado.
- **Tópicos:** Declaração de vetores, passagem de vetores para funções, laços `for` para percorrer o vetor.

**3. Verificador de Número Primo com Função:**

- **Objetivo:** Crie uma função `bool ehPrimo(int numero)` que retorna `true` se o número for primo e `false` caso contrário. Um número é primo se for divisível apenas por 1 e por ele mesmo. No `main`, peça um número ao usuário e use a função para informar o resultado.
- **Tópicos:** Funções com retorno booleano, lógica de programação com laços e condicionais, reaproveitamento de código.

**4. Inversor de Palavras:**

- **Objetivo:** Crie uma função `string inverter(string palavra)` que recebe uma palavra e a retorna invertida. Ex: se a entrada for "amor", a função deve retornar "roma".
- **Tópicos:** Funções com strings, manipulação de strings (ou vetores de char), laços `for` com contador regressivo.

---

# 🏆 Nível 3: Desafios de Orientação a Objetos (Modelando o Mundo Real)

*Bem-vindo(a) ao pilar da programação moderna! Agora vamos usar tudo o que aprendemos para criar nossos próprios tipos de dados com a ajuda de `Classes` e `Objetos`. Vamos modelar conceitos do mundo real.*

**1. Classe `Aluno`:**

- **Objetivo:** Crie uma classe `Aluno` com os seguintes atributos `private`: `nome` (string), `matricula` (int) e `notas` (um vetor de 3 floats).
    - Crie um **construtor** que receba o nome e a matrícula.
    - Crie um método público `void adicionarNota(int indice, float nota)` para adicionar as 3 notas do aluno.
    - Crie um método público `float calcularMedia()` que retorna a média das notas.
    - Crie um método público `void mostrarStatus()` que exibe o nome, a média e se o aluno está "Aprovado" (média >= 7.0) ou "Reprovado".
- **Tópicos:** Classes, objetos, `private`, `public`, construtor, métodos, atributos, vetores como atributos.

**2. Classe `Retangulo`:**

- **Objetivo:** Crie uma classe `Retangulo` com atributos `private` `largura` e `altura`.
    - Crie um **construtor** para inicializar esses atributos.
    - Crie um método público `float calcularArea()` que retorna `largura * altura`.
    - Crie um método público `float calcularPerimetro()` que retorna `2 * (largura + altura)`.
    - No `main`, crie dois retângulos diferentes e exiba a área e o perímetro de cada um.
- **Tópicos:** Encapsulamento, construtores, métodos que retornam valores calculados.

**3. Classe `Livro`:**

- **Objetivo:** Crie uma classe `Livro` com atributos `private`: `titulo`, `autor` e `anoPublicacao`.
    - Crie um **construtor** para inicializar todos os atributos.
    - Crie um método público `void exibirFichaCatalografica()` que imprime todos os dados do livro de forma organizada.
- **Tópicos:** Prática de construtores, encapsulamento, métodos para exibição de dados.

**4. Desafio Final: Classe `Jogador` de RPG:**

- **Objetivo:** Crie uma classe `Jogador` com os atributos `private`: `nome` (string), `pontosDeVida` (int, começa com 100) e `nivel` (int, começa com 1).
    - Crie um **construtor** que receba apenas o `nome`.
    - Crie um método `void sofrerDano(int dano)` que subtrai o dano dos pontos de vida. Os pontos de vida não podem ficar abaixo de 0.
    - Crie um método `void receberCura(int cura)` que adiciona cura aos pontos de vida. Os pontos de vida não podem passar de 100.
    - Crie um método `void ganharExperiencia(int xp)` que, a cada 100 de xp, aumenta o `nivel` do jogador em 1 e restaura sua vida para 100.
    - Crie um método `void exibirStatus()` que mostra nome, nível e pontos de vida atuais.
- **Tópicos:** Lógica de negócio dentro da classe, métodos que modificam o estado do objeto, encapsulamento.