Feature 2: controle de fluxo

Para esta etapa, o projeto de vocês deverá demonstrar a capacidade de reagir dinamicamente a diferentes situações e entradas do usuário, executando lógicas específicas com base em condições ou repetindo ações quando necessário. É o coração da interatividade e da automação do seu sistema.

1. Tomada de decisão condicional: no desenvolvimento do seu projeto, é fundamental que o programa possa tomar decisões. Para isso, vocês implementarão as seguintes estruturas:

- **Estruturas `if`, `else if` e `else`:** utilizem essas construções para guiar o fluxo de execução do programa, avaliando condições e executando blocos de código específicos.
- **Operadores lógicos:** os operadores `&&` (AND lógico), `||` (OR lógico) e `!` (NOT lógico) serão essenciais para criar condições mais complexas e refinadas, permitindo que vocês combinem múltiplas verificações em uma única decisão. Por exemplo, verificar se um usuário está autenticado *e* tem permissão para acessar uma funcionalidade específica.

2. Escolha múltipla com `switch-case`: quando o projeto exigir que o usuário selecione entre várias opções discretas (como um menu principal ou tipos de operação), a estrutura `switch-case` é a escolha ideal para organizar o código de forma mais limpa e eficiente.

- **Implementação:** utilizem o `switch-case` para lidar com múltiplos comandos ou escolhas, mapeando cada opção a um bloco de código correspondente. Isso é particularmente útil para menus interativos, onde o usuário insere um número ou caractere para selecionar uma ação.

3. Laços de repetição para fluxos dinâmicos: a capacidade de repetir operações é crucial para a maioria dos softwares. Vocês deverão empregar os seguintes laços de repetição em seu projeto:

- **`for`:** ideal para situações onde o número de repetições é conhecido antecipadamente ou quando se precisa iterar sobre uma sequência.
- **`while`:** perfeito para cenários onde a repetição continua enquanto uma condição específica for verdadeira, e o número de iterações não é fixo.
- **`do-while`:** semelhante ao `while`, mas garante que o bloco de código seja executado pelo menos uma vez antes que a condição seja avaliada. Útil para quando uma ação inicial é sempre necessária antes de verificar se a repetição deve continuar.

4. Controle de fluxo interno dos laços: dentro dos laços de repetição, vocês precisarão de mecanismos para alterar o comportamento padrão de iteração:

- **`break`:** para sair imediatamente de um laço de repetição (ou de um `switch-case`) antes que sua condição de término natural seja atingida. Isso é útil para quando uma condição excepcional ocorre e o restante das iterações se torna desnecessário.
- **`continue`:** pulem a iteração atual de um laço e passem para a próxima. É útil para ignorar processamentos de itens que não atendem a certos critérios dentro de um ciclo, continuando com os demais itens sem sair do laço.