Principais Diferenças e Eixos Adotados

1. Nomes Significativos

Código legado: nomes de variáveis abreviadas e ambíguas (n, op) e de funções genéricas (func, func_it).
Código refatorado: nomes claros e diretos: number, option, recursive, iterative.

2. Clareza e Intenção

Código legado: if-else if para seleção de opção (possivelmente criando um ninho em caso futuro de adição de novas condições). Muitos comentários eram necessários para explicar a lógica ou funcionamento.
Código refatorado: uso de switch-case, que é mais legível para múltiplas escolhas. Comentários se tornaram desnecessários.

3. Remoção de Comentários

Código legado: comentários que descreviam o que o código já fazia.
Código refatorado: código agora autoexplicativo, eliminando a necessidade de comentários óbvios e focando em nomes descritivos.

4. Estrutura do Projeto

Código legado: Todo o código (lógica principal e funções) estava em um único arquivo.
Código refatorado: O código foi modularizado: a função main ficou em seu próprio arquivo, e as funções de refatoração e interatividade foram separadas, promovendo a reutilização e o Princípio da Responsabilidade Única (SRP).

5. Testes automatizados

Código legado: o código não tinha nenhum tipo de teste automatizado, não abrangendo todos os detalhes de funcionalidade.
Código refatorado: adicionado testes unitários, testando de fato se as funções estão funcionando como deveriam.

Lições Aprendidas

Um código limpo proporciona melhor legibilidade e, pensando futuramente, melhor manutenção. Mesmo o código sendo algo simples, como a que usei de exemplo para refatoração, foi possível ver

como a não utilização de eixos prejudicou significativamente no entendimento do que o código fazia, sendo necessários comentários para explicar o que ocorria. Um leigo no assunto, que esteja começando agora, ou apenas queira dar uma olhada, não entenderia facilmente o que o código estava fazendo. Especialmente nas funções, caso elas fossem “separadas” do código principal, entender o que elas fazem seria uma tremenda dificuldade por conta dos nomes nada diretos e significativos que elas tinham. Caso fosse um código maior e mais complexo, seguindo o mesmo padrão que estava anteriormente, a sua manutenção seria extremamente cara, difícil e problemática.

Dificuldades Enfrentadas

A configuração do ambiente de testes exigiu um tempo de aprendizado para entender como estruturar os testes, compilar e vincular os arquivos corretamente, pois nunca havia feito na linguagem C.