Visão Geral do Código

[PROBLEMA 1] Múltiplos de 3 e 5:

Retirado do site: https://projecteuler.net/problem=1
Autor: github/josequilhermefmoura

LINGUAGEM UTILIZADA: C

DUAS SOLUÇÕES: [1] SOLUÇÃO SIMPLES; [2] SOLUÇÃO APERFEIÇOADA.

ENUNCIADO:

Se listarmos todos os números naturais abaixos de 10 que são múltiplos de 3 ou 5, obteremos 3, 5, 6 e 9. A soma desses múltiplos é 23.

Encontre a soma de todos os múltiplos de 3 ou 5 abaixo de 1000.

RESPOSTA:

A soma é igual a 233168.

CÓDIGO [1]:

[1] SOLUÇÃO SIMPLES:

```
const int MAX = 999;
int i, sum = 0;

for (i = 1; i <= MAX; i++) {
     if (i % 3 == 0 || i % 5 == 0){
          sum += i;
    }
}
```

COMENTÁRIO [1]:

Para determinar se um número é ou não múltiplo de 3 ou 5, precisamos apenas fazer a operação módulo (ou seja, verificar se existe qualquer resto depois da divisão) com ambos os números.

Cuidados válidos para a velocidade de execução do código são: 1) a condição `i % 3 == 0` deve vir primeiro que a condição `i % 5 == 0`, pois é mais provável que o número analisado seja múltiplo de 3 do que 5. Pois existem mais números múltiplos de 3 do que de 5, evidentemente. Caso fosse ao contrário, o código realizaria mais verificações. 2) O valor 999 pode ser alocado numa variável inteira constante `MAX`, pois o compilador interpreta constantes como sendo variáveis apenas para leitura, sendo processada mais rapidamente. Não foi feito o mesmo para o 3 e o 5 por objetivos de leitura de código.

CÓDIGO [2]:

[2] SOLUÇÃO APERFEIÇOADA:

```
const int MAX = 999;
int sumOfTheDivisibles (int num) {
    const int MAX_DIVISIBLE = (int) (MAX / num);
    return (int) (num * (MAX_DIVISIBLE * (MAX_DIVISIBLE + 1))) / 2;
}
```

```
printf("SUM = %d.\n", sumOfTheDivisibles(3) + sumOfTheDivisibles(5)
sumOfTheDivisibles(15));
```

COMENTÁRIO [2]:

Este é o modo mais rápido de resolver o problema, mesmo para casos mais extremos (divisores abaixo de 1,000.000.000 por exemplo).

Outra forma de calcular os múltiplos de 3 e 5 abaixo de 1000 é somar os divisíveis de 3 (CONJUNTO A) e 5 (CONJUNTO B) (ambos abaixo de 1000) e depois subtrair a interseção (A ∩ B) (os divisíveis de 15, também abaixo de 1000), pois ao somar os dois primeiros conjuntos, a interseção estará duplicada.

Ou seja: **D(A U B) = A + B - C(A ∩ B).**

Como os divisíveis de 3 são: 3, 6, 9, … , 999;
é o mesmo que 3 * (1 + 2 + 3, … , 333).

Como os divisíveis de 5 são: 5, 15, 20, … , 995;
é o mesmo que 5 * (1 + 2 + 3, … , 199).

O último divisível (ou dividendo) é sempre o valor máximo dividido pelo divisor, arredondado para inteiro.
Então `const int MAX_DIVISIBLE = (int) (MAX / num);`

Como a fórmula da soma dos termos de uma progressão aritmética é:

$$Sn = \frac{(a_1 + a_2) * n}{2}$$

Basta implementarmos `(int) (num * (MAX_DIVISIBLE * (MAX_DIVISIBLE + 1))) / 2;` no retorno da função de tipo `int`.