Relatório Projeto Entregável 02 - Potência

João Pedro Neves e Juan Pablo Tomba

11 de setembro de 2025

1 Introdução

Neste relatório, explicaremos o processo de criação de três versões de código que visam resolver o mesmo problema: implementar a operação de potenciação a partir de dois números inteiros n ($0 < n \leq 99$) e k ($0 < k \leq 10^9$). A saída deve conter apenas os três últimos dígitos do resultado.

O objetivo da criação dessas diferentes versões é realizar um comparativo de eficiência entre elas: uma baseada no método de iteração, outra na recursão e, por fim, uma versão iterativa gerada por IA (Inteligência Artificial).

Serão discutidos aspectos como tempo de execução, manutenibilidade, simplicidade e repertório.

2 Metodologia

O primeiro passo do projeto foi criar um repositório público no GitHub onde ambos os membros do grupo poderiam subir os códigos. Dessa forma, poderíamos desenvolver simultaneamente os diferentes códigos e ao final ter uma forma prática de juntá-los e/ou acessá-los.

https://github.com/joaopneves1570/Lab-ICC2/tree/main/aula3

Em seguida, criados os arquivos para o projeto, utilizamos o ambiente de desenvolvimento do Visual Studio Code para escrever e compilar os mesmos.

3 Códigos

Após isso, confeccionamos os respectivos códigos:

3.1 Iterativo Manual

```c
#include <stdio.h>

int main(){
    int n, k, rfinal = 1;
    scanf("%d %d", &n, &k);
    if (n == 0){
        printf("0");
        return 0;
    }
    // Caso a potencia seja nula (N^0 = 1)
    if (k == 0){
        printf("%d", rfinal);
        return 0;
    }
    // So utiliza os 3 ultimos numeros
    for (int i = 0; i < k; i++){
        rfinal = (rfinal * n)%1000;
    }


    printf("%d", rfinal);

    return 0;
}
```

3.2 Iterativo por IA

```c
#include <stdio.h>

int main() {
    int n, k;
    int resultado = 1;

    scanf("%d %d", &n, &k);

    n = n % 1000; // ja reduz a base para evitar overflow

    // Exponenciacao rapida iterativa
    while (k > 0) {
        if (k % 2 == 1) { // se k e impar
            resultado = (resultado * n) % 1000;
        }
        n = (n * n) % 1000; // quadrado da base
        k /= 2;
    }

    // Saida sem zeros a esquerda
    printf("%d\n", resultado);

    return 0;
}
```

3.3 Recursivo Manual

```c
#include <stdio.h>

int potencia(int n, int k){
    if (k == 1)
        return n;
    if ((k > 1) && (k%2 == 0)){
        int r = potencia(n, k/2);
        return (r * r)%1000;
    }
    else{
        int r = potencia(n,k/2);
        return (r*r*n)%1000;
    }
}

int main(){
    int n, k;

    scanf("%d %d", &n, &k);

    int resultado = potencia(n,k);
    printf("%d\n", resultado);
}
```

4 Resultados

Comparando esses três códigos, obtivemos os seguintes resultados:

4.1 Tempo de Execução

Para medir os diferentes tempos de execução, fizemos o upload dos três diferentes códigos no RunCodes e de lá retiramos o tempo de execução fornecido pela própria plataforma. Para cada um deles, iremos discutir aqui o resultado do caso teste 5, (52 1000000000, que seria calcular $52^{1000000000}$), considerado o caso mais demorado.

- **Iterativo manual:** Caso Teste 5: 1.0007 s
- **Iterativo por IA:** Caso Teste 5: 0.0014 s
- **Recursivo manual:** Caso Teste 5: 0.0011 s

4.2 Manutenibilidade

A manutenibilidade avalia a clareza e a facilidade de adaptar cada abordagem em casos de futuras alterações ou otimizações:

- **Iterativo manual:** possui implementação direta com um laço `for`. É simples de entender e modificar, mas pouco eficiente para valores grandes de k, pois a complexidade cresce linearmente. Qualquer ajuste no algoritmo exigiria mudanças estruturais.
- **Iterativo por IA:** utiliza o método de exponenciação rápida. Apesar de mais elaborado, o código é compacto, faz bom uso da aritmética modular e pode ser facilmente reutilizado em outros contextos. É a versão mais equilibrada entre clareza e desempenho.
- **Recursivo manual:** implementa a mesma ideia de exponenciação rápida, mas de forma recursiva. O uso da recursão torna o raciocínio menos direto e pode dificultar a manutenção, especialmente para entradas grandes.

4.3 Simplicidade

A simplicidade está relacionada à legibilidade e à facilidade de compreensão da lógica adotada:

- **Iterativo manual:** é a versão mais simples de compreender, pois segue a lógica básica de multiplicar n repetidamente. Contudo, é pouco eficiente, com complexidade $O(k)$.
- **Iterativo por IA:** ligeiramente mais complexo conceitualmente (exponenciação rápida), mas ainda compacto e fácil de seguir, com complexidade $O(\log k)$. Traz um bom equilíbrio entre clareza e desempenho.
- **Recursivo manual:** menos simples, pois envolve chamadas recursivas e raciocínio sobre divisão do problema. Apesar de elegante, é a versão mais difícil de acompanhar para quem não está acostumado com recursão, mas ainda mantem a complexidade $O(\log k)$, igual a iterativa por IA.

5 Conclusões

A partir da análise realizada, observamos que cada abordagem apresenta pontos positivos e negativos.

O código **iterativo manual** é o mais simples de implementar e compreender, mas foi também o menos eficiente, com tempo de execução significativamente maior devido à sua complexidade linear em relação a k. Dessa forma, é útil apenas como exercício didático, visto que em casos onde o expoente é muito grande, seu tempo de execução acaba sendo muito maior que do seus concorrentes.

O **iterativo por IA**, por outro lado, implementa a exponenciação rápida de forma clara e compacta. Seu desempenho foi excelente e sua manutenção é facilitada pela estrutura iterativa. Embora não tenha sido o mais veloz nos testes, combina eficiência com legibilidade, tornando-se a opção mais prática para aplicações reais.

Por último, o **recursivo manual** também utiliza a exponenciação rápida e, nos testes realizados, apresentou o melhor tempo de execução. No entanto, seu uso de chamadas recursivas pode acabar dificultando a leitura para iniciantes e trazer riscos de stack overflow no caso de entradas muito grandes.

Em resumo, a versão recursiva manual se destacou como a mais eficiente em termos de desempenho, enquanto a versão iterativa por IA foi a mais equilibrada em clareza e aplicabilidade. Já a versão iterativa manual cumpriu um papel melhor no quesito de introdução didática e exercício, apesar de ter sido o mais lento.