

## Exercícios referentes ao Capítulo 4: Introdução à Recursão

### 1. Exercícios de programação:

- (a) Ao contrário de muitas linguagens, C não possui um operador primário para calcular a potência de algum número, por exemplo, calcular  $n^k$ . Como uma solução parcial para o problema, defina a função recursiva cuja declaração é

```
int potencia (int n, int k);
```

e que calcula  $n^k$ . Talvez seja útil você se lembrar da propriedade matemática de que:

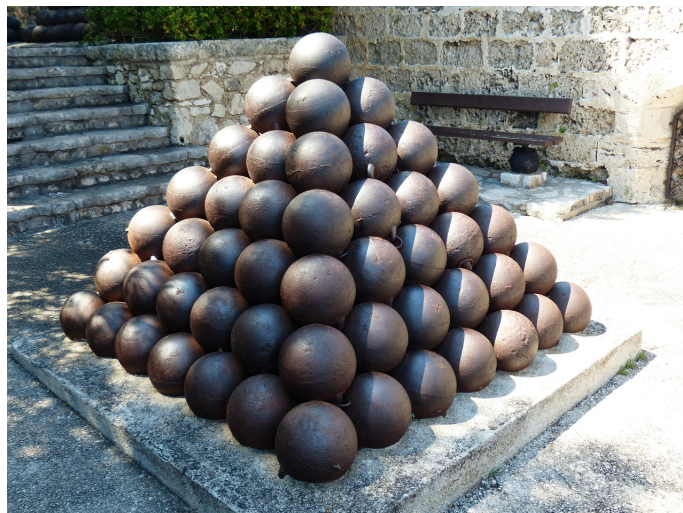
$$n^k = \begin{cases} 1 & \text{se } k = 0 \\ n \times n^{k-1} & \text{caso contrário} \end{cases} \quad (1)$$

- (b) O **maior divisor comum** (mdc) de dois números inteiros não negativos é o maior número inteiro que divide uniformemente os outros dois números. Em 300 a.C. o matemático grego Euclides descobriu que o mdc de  $x$  e  $y$  poderia ser obtido da maneira como se segue:

- Se  $x$  for uniformemente divisível por  $y$ , então  $y$  é o mdc;
- Caso contrário, o mdc entre  $x$  e  $y$  será sempre igual ao mdc entre  $y$  e o resto da divisão de  $x$  por  $y$ .

Utilize o método de Euclides para definir a função “`int gcd (int x, int y);`”, que calcula o maior divisor comum entre  $x$  e  $y$ .

- (c) Objetos esféricos, como bolas de canhão, podem ser empilhadas em uma pirâmide com 1 bola no topo, sobre uma base quadrada composta de 4 bolas, sobre uma base quadrada composta de 9 bolas, e assim por diante. Escreva uma função recursiva chamada de “`bolas_de_canhao`” que recebe como argumento a altura  $a$  da pirâmide e retorna o número de bolas de canhão que essa pirâmide contém.



Fonte: [Hans](#), no [Pixabay](#)

- (d) Para cada uma das duas implementações da função “fib(n)” que estudamos nesse capítulo (uma primeira implementação que não era eficiente e uma segunda implementação mais eficiente), escreva funções recursivas chamadas de “contagem\_fib1” e “contagem\_fib2” que conta o número de chamadas recursivas feitas durante a execução de cada função na obtenção de um determinado número de Fibonacci. Escreva um programa que utiliza essas funções para mostrar uma tabela com o número de chamadas recursivas que cada algoritmo faz para diversos valores de  $n$ , conforme ilustrado abaixo:

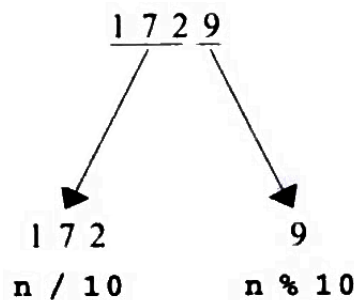
Este programa compara a performance de dois algoritmos para o cálculo da Sequência de Fibonacci:

Número de chamadas recursivas:

N	Fib1	Fib2
0	1	2
1	1	2
2	3	3
3	5	4
4	9	5
5	15	6
6	25	7
7	41	8
8	67	9
9	109	10
10	177	11
11	287	12
12	465	13
13	753	14
14	1219	15
15	1973	16

- (e) Defina a função recursiva “int somar\_digitos (int n);”, que recebe um número inteiro não negativo e retorna a soma de seus dígitos. Por exemplo: chamar a função com o argumento 1729 deve retornar 19, pois é igual à  $1 + 7 + 2 + 9$ .

A implementação recursiva dessa função se deve ao fato de que é muito fácil quebrar um inteiro em seus dígitos componentes usando a divisão e o resto da divisão do número por 10. Por exemplo, para o inteiro 1729, você pode dividir esse número em duas partes como se segue:



Cada um dos inteiros resultantes é estritamente menor do que o original e representa um caso mais simples.



- (h) Escreva uma função recursiva que recebe uma string como argumento e retorna o reverso dessa string. O protótipo para essa função deve ser

```
string reverter (string str);
```

e a sentença

```
printf("%s\n", reverter("programa"));
```

deve retornar

```
amargorp
```

- (i) A biblioteca `strlib.h` contém a função `IntegerToString`, que é implementada usando a função `sprintf`, como se segue:

```
#define MaxDigits 30

string IntegerToString(int n)
{
    char buffer[MaxDigits];

    sprintf(buffer, "%d", n);
    return (CopyString(buffer));
}
```

Esta implementação, entretanto, esconde todo o trabalho dentro da função `sprintf` e não fornece nenhuma explicação sobre como o computador realmente executa o processo de converter um número inteiro em sua representação como string. Ocorre que a maneira mais fácil para implementar essa função é usar a decomposição recursiva de um inteiro conforme você fez no exercício “1. (e)” anterior. Reescreva a implementação da função `IntegerToString` para que ela opere de maneira recursiva. Sua função pode chamar qualquer outra função da biblioteca `strlib.h`, exceto as funções `IntegerToString` e `RealToString`, mas também não pode utilizar outras funções de outras bibliotecas tais como a `sprintf`. Lembre-se: sua função deve operar recursivamente e não pode usar nenhuma estrutura iterativa.

## 2. Exercícios conceituais:

- Defina os termos **recursivo** e **iterativo**. É possível que uma função utilize as duas abordagens?
- Explique o que é o paradigma recursivo.
- Quais as duas propriedades que um problema deve ter para que a recursão seja uma possibilidade como estratégia de solução?
- Costumamos dizer que a recursão é um exemplo de estratégia **dividir para conquistar**. Explique.
- O que queremos dizer por **salto de fé recursivo**? Por que esse conceito é importante para você que é programador?

- (f) Durante nosso estudo do fatorial recursivo nós fizemos o rastreamento de todo o processo das chamadas, mostrando todos os *stack frames* empilhados, as chamadas feitas e os retornos executados para o cálculo do fatorial de 4. Usando esse mesmo modelo de rastreamento, faça a demonstração detalhada do rastreamento do cálculo de `fib(4)`, desenhando cada *stack frame* que é criado ao longo do processo.
- (g) Modifique o problema dos coelhos de Fibonacci introduzindo uma regra adicional: cada par de coelhos pára de reproduzir após dar à luz 3 ninhadas. Como essa regra afetaria a relação de recorrência? Que alterações você precisaria fazer aos casos simples?
- (h) O que é uma função *wrapper* (invólucro)? por que elas são úteis na implementação de funções recursivas?
- (i) O que é recursão mútua?