Lista 6 - Recursividade e Ponteiros

1. Para cada problema a seguir defina uma função recursiva, faça a simulação por substituição e desenhe o fluxo de chamadas e retornos:

   a) Calcular o fatorial de um número natural.
   b) Calcular o resto da divisão inteira usando subtração.
   c) Calcular o quociente da divisão inteira usando subtração.
   d) Calcular a soma de dois naturais usando as funções `suc(n)` e `pred(n)` que devolvem, respectivamente, o sucessor e o predecessor de um natural n.

8. Defina os seguintes procedimentos recursivos:
   a)
   `regr(n)`, que exibe uma contagem regressiva a partir de n.

   b) `bin(n)`, que exibe o número natural n em binário.

3. Escreva uma função recursiva que calcule o Máximo Divisor Comum (MDC) de dois números usando o algoritmo de Euclides.

   Entrada: Dois inteiros aa e bb.

   Saída: O MDC de aa e bb.

   Dica: A fórmula para o MDC é:

   `MDC(a,b)=MDC(b,a%b) com MDC(a,0)=a`

4. Implemente uma função recursiva para resolver o problema da Torre de Hanói. O problema consiste em mover todos os discos de uma torre para outra usando uma torre auxiliar, respeitando as regras do problema.

   - **Entrada**: O número de discos.
   - **Saída**: A sequência de movimentos necessários para resolver o problema.

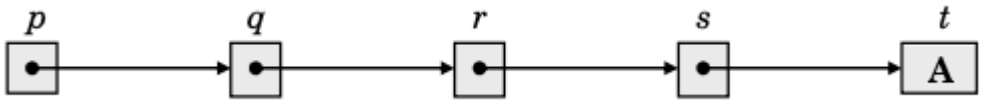
   Dica: Mova n−1 discos para a torre auxiliar, depois mova o maior disco para a torre destino, e por fim, mova os n−1 discos da torre auxiliar para a torre

destino.

5. Escreva uma função recursiva que conte quantos dígitos um número inteiro possui.
   - **Entrada**: Um número inteiro positivo n.
   - **Saída**: A quantidade de dígitos de n.
6. Implemente uma função recursiva que calcule a soma dos dígitos de um número inteiro positivo.
   - **Entrada**: Um número inteiro positivo n.
   - **Saída**: A soma dos dígitos de n.
7. Implemente uma função recursiva que verifique se um número é par ou ímpar.
   - **Entrada**: Um número inteiro n.
   - **Saída**: Retorne 1 se o número for par, e 0 se for ímpar.
8. Explique o significado de cada ocorrência de `*` no fragmento de código a seguir e indique qual a saída exibida na tela.

```
int *p, x=5;
*p *= 2**p;
printf("%d", x);
```

5. Codifique um programa para criar a configuração representada na figura abaixo e exibir a letra 'A' a partir de cada uma das variáveis.



10. Com ponteiros, 1000+1 não é necessariamente 1001! Qual será a saída do código abaixo?

```
#include <stdio.h>
void main(void) {
    char   *a = (char *) 0x1000;
```

```
  int    *b = (int *) 0x1000;
  float  *c = (float *) 0x1000;
  double *d = (double *) 0x1000;
    printf("%p %p %p %p", a+1, b+1, c+1, d+1 );
}
```

11. Operadores de adição e subtração com ponteiros e inteiros. Mostre as saídas do código abaixo:

```
#include <stdio.h>
void main(void) {
    int *p = (int *) 0x1000;
   p++;  printf("%p ", p );
   p-=3; printf("%p ", p );
   p+=2; printf("%p ", p );
   p--;  printf("%p ", p );
}
```

12. Escreva uma função que receba três números inteiros por referência e troque seus valores de forma cíclica: o valor do primeiro vai para o segundo, o valor do segundo vai para o terceiro, e o valor do terceiro vai para o primeiro.
    - **Entrada**: Três inteiros.
    - **Saída**: Os valores trocados cíclicamente.

    ```
    void trocarCiclicamente(int *a, int *b, int *c);
    ```

13. Implemente uma função que receba um número inteiro por referência e realize dois cálculos diferentes: incremente o valor em 1 e depois decremente o valor em 2. Retorne o resultado de cada operação por referência.
    - **Entrada**: Um número inteiro.
    - **Saída**: O valor incrementado em 1 e o valor decrementado em 2 (via ponteiros).

```
void incrementarDecrementar(int *valor, int *incrementad
o, int *decrementado);
```

14. Implemente uma função que receba um número inteiro por referência e retorne o número de divisores inteiros positivos desse número. Use ponteiros para retornar o resultado.
    - **Entrada**: Um número inteiro.
    - **Saída**: A quantidade de divisores do número (via ponteiro).

```
void contarDivisores(int *num, int *totalDivisores);
```

15. Escreva uma função que receba dois números inteiros por referência e retorne o maior e o menor desses dois números por referência.
    - **Entrada**: Dois números inteiros.
    - **Saída**: O maior e o menor número (via ponteiros).

```
void compararNumeros(int *a, int *b, int *maior, int *me
nor);
```

16. Implemente uma função que receba um número real (float) por referência e calcule o dobro e a metade desse número. Retorne os resultados por referência.
    - **Entrada**: Um número real.
    - **Saída**: O dobro e a metade do número (via ponteiros).

```
void calcularDobroMetade(float *num, float *dobro, float
*metade);
```

17. Implemente uma função que receba dois números inteiros por referência, calcule a divisão inteira e o resto da divisão (módulo) e retorne ambos os resultados por referência.
    - **Entrada**: Dois números inteiros.
    - **Saída**: O quociente e o resto da divisão (via ponteiros).

```
void dividir(int *dividendo, int *divisor, int *quocient
e, int *resto);
```