- Resolver diversos exercícios sobre recursão destacando o processo de construção da
solução

Exercício 1 - Sequência de Fibonacci
Comentários: fórmulas matemáticas a aplicação da recursão é um mapeamento direto da fórmula.
F(0) = 0 \ Condições
F(1) = 1 / iniciais
F(n) = F(n-1) + F(n-2) -> Chamada recursiva

```
/* sequência de Fibonacci:
resultado   0  1  1  2  3  5  8  13
n           0° 1° 2° 3° 4° 5° 6° 7°

#include <stdio.h>

int Fib(int);

int main()
{
    int n;

    scanf("%i", &n);
    printf("Fib: %i", Fib(n));
}

int Fib(int n)
{
    if (n == 0) return 0;
    else if (n == 1) return 1;
    else return(Fib(n-1) + Fib(n-2));
}
```

Exercício 2 - Imprimir dígito a dígito, os dígitos de um número natural n
Comentários: exercício sem muita utilidade prática, somente para mostrar que a recursão vai de "trás para frente"!
- Permite não ter que usar gotoxy()

1ª VERSÃO
Vai imprimir o número com um 0 (zero) na frente.
Ex.: n = 123. Resultado: 0123
     n = 1.   Resultado: 01
     n = 0.   Resultado: 0

```
#include <stdio.h>

void Digitos(int);
int main()
{
    int n;

    scanf("%i", &n);
    Digitos(n);
}

void Digitos(int n)
{
     if (n > 0) Digitos(n / 10);
     printf("%i", n % 10);
}
```

2ª VERSÃO
Modificar o programa anterior para que o 0 (zero) não seja mais impresso

```
---------
#include <stdio.h>

void Digitos(int);
int main()
{
    int n;

    scanf("%i", &n);
    Digitos(n);
}

void Digitos(int n)
{
     if (n / 10 > 0) Digitos(n / 10);
     printf("%i", n % 10);
}
--------------------------------------



--------------------------------------
Exercício 3 - Imprimir os bits de um número decimal
Comentários: Fazer a divisão "na mão" e destacar o processo.
Ex.: 13
13 / 2 = 6 resto 1
6  / 2 = 3 resto 0
3  / 2 = 1 resto 1
1  / 2 = 0 resto 1
número em binário é o resto de cada uma das operações de trás para frente:
1101 na base binária = 13 na base dez
--------------------------------------
1ª VERSÃO
Vai imprimir o número com um 0 (zero) na frente.
Ex.: n = 13. Resultado: 01101
     n = 1.   Resultado: 01
     n = 0.   Resultado: 0
--------------------------------------
#include <stdio.h>

void Binario(int);

int main()
{
    int n;

    scanf("%i", &n);
    Binario(n);
}

void Binario(int n)
{
     if (n > 0) Binario (n / 2);
     printf("%i", n % 2);
}
--------------------------------------
2ª VERSÃO
Modificar o programa anterior para que o 0 (zero) não seja mais impresso
---------
#include <stdio.h>

void Binario(int);

int main()
{
    int n;

    scanf("%i", &n);
```

```
    Binario(n);
}

void Binario(int n)
{
     if (n / 2 > 0) Binario (n / 2);
     printf("%i", n % 2);
}
```

Exercício 4 - Imprimir em octal um número decimal
- comentários: igual o anterior. Só substituir a base de 2 por 8

Exercício 5 - Imprimir em hexadecimal um número decimal
- comentários: igual ao exerc. 2. Só substituir a base de 10 por 16 e a linha 'printf("%i", n % 10) ' por 'printf("%X", n % 16)'
- Fazer o exemplo para n = 65535 que corresponde a FFFF
- Fazer o exemplo para n = 65531 = 65535 - 4 que corresponde a FFFB que foi o que
eu havia perguntado na aula anterior quanto que dava FFFF - 4 e ninguém soube dizer
com certeza

Exercício 6 - Imprimir em qualque base de 2 a 9.
- comentários: igual o exerc. 2. Só acrescentar um parâmetro.

```
#include <stdio.h>


void Base(int, int);

int main()
{
    int n, base;

    scanf("%i", &n);
    scanf("%i", &base);
    Base(n, base);
}

void Base(int n, int base)
{
     if (n / base > 0) Base(n / base, base);
     printf("%i", n % base);
}
```

Exercício 7 - Contar número de dígitos de um número decimal.
- comentários: análogo ao exercício 2.

```
#include <stdio.h>

int QtdeDigitos(int);
int main()
{
    int n;

    scanf("%i", &n);
```

```
    printf("Qtde digitos: %i",QtdeDigitos(n));
}

-------------------------------------
Versão COM STATIC - 1 VERSÃO!
-------------------------------------
/* NÃO FUNCIONA PARA n == 0, pois o "numero_digitos" não é incrementado! /
int QtdeDigitos(int n)
{
    static int numero_digitos = 0;

    if (n > 0) QtdeDigitos(n / 10);

    return (numero_digitos++);
}

-------------------------------------
Versão COM STATIC - 2 VERSÃO !
-------------------------------------
int QtdeDigitos(int n)
{
    static int numero_digitos = 0;

    if (n / 10 > 0) QtdeDigitos(n / 10);

    return (++numero_digitos);

}


-------------------------------------
Versão SEM STATIC
-------------------------------------
int QtdeDigitos(int n)
{
    int numero_digitos = 0;

    if (n / 10 == 0) return 1;
    else
        numero_digitos = QtdeDigitos(n / 10);

    return (++numero_digitos);
}

-------------------------------------


-------------------------------------
Exercício 8 - Contar número de dígitos de um número binário.
- comentários: análogo ao exercício anterior.
- Só substituir a base 10 por 2.
- OBS.: o número n será entrado em decimal e será feita a conta de quantos bits
têm a representação desse número em binário.
-------------------------------------


-------------------------------------
Exercício 9 - Calcular a potência de a elevado a b.
- comentários: mostrar que "a elevado a b" é o mesmo que fazer "a * a * ... *a",

b vezes.
-------------------------------------
#include <stdio.h>

int Pot(int, int);
int main()
{
    int a, b;
```

```
    scanf("%i %i", &a, &b);
    printf("Qtde digitos: %i",Pot(a, b));
}

int Pot(int a, int b)
{
    if (b == 1) return a;
    else return (a * Pot(a, b-1));

}
```

Exercício 10 - Contar número de dígitos de qualquer base.
- comentários: igual ao exercício 6.
- só acrescentar um parâmetro que será a base a ser calculada.

Exercício 11 - Encontrar o maior elemento de um vetor.
- comentários:

```
#include <stdio.h>

#define max(a, b) ((a > b)? (a): (b))

int Maior(int [], int);

int main()
{
    int v[5] = {13, 7, 12, 2, -1};

    printf("Maior: %i",Maior(v, 4));
}

/* n é o índice do último elemento do vetor */
int Maior(int vetor[], int indice)
{
    int maior;

    if (indice == 0) return vetor[indice];
    else maior = max(Maior(vetor, indice - 1), vetor[indice]);
    return maior;
}

/* OU usando STATIC*/
/* n é o índice do último elemento do vetor */
int Maior(int vetor[], int indice)
{
    static int maior;

    if (indice == 0) return vetor[indice];
    else return(max(Maior(vetor, indice - 1), vetor[indice]));
    return maior;
}

/* OU sem usar STATIC*/
/* n é o índice do último elemento do vetor */
int Maior(int vetor[], int indice)
{

    if (indice == 0) return vetor[indice];
    else return(max(Maior(vetor, indice - 1), vetor[indice]));

}
```