Recursividade

Crie um programa em C que peça um número inteiro ao usuário e retorne a soma de todos os números de 1 até o número que o usuário introduziu ou seja: 1 + 2 + 3 + ... + n

Vamos criar uma função soma(int n).

Se n=5, essa função deve retornar: soma(5) = 5 + 4 + 3 + 2 + 1

Se n=4, essa função deve retornar: soma(4) = 4 + 3 + 2 + 1

Se n=3, essa função deve retornar: soma(3) = 3 + 2 + 1

Veja que:

soma(5) = 5 + 4 + 3 + 2 +1 = 5 + soma(4)

O mesmo para:

soma(4) = 4 + 3 + 2 + 1 = 4 + soma(3)

Ou seja, temos a fórmula geral:

soma(n) = n + soma(n-1)

Concorda?

Ou seja:

soma(n) = n + soma(n-1) = n + (n-1) + soma(n-2) = n + (n-1) + (n-2) + soma(n-3)...

E onde essa soma para? Para quando o último elemento dessa soma for 1.

Então:

soma(n) = n +(n-1) + (n-2) + (n-3) + ... + 1

Agora vamos traduzir essa fórmula em termos de programação.

A função recebe um número, e a primeira coisa que ela deve fazer é ver se esse valor é 1.

Se for, deve retornar 1, afinal:

soma(1) = 1

E se não for 1, deve retornar:

n + soma(n-1)

```c
#include <stdio.h>

int soma(int n)
{
    if(n == 1)
        return 1;
    else
        return ( n + soma(n-1) );
}

int main()
{
    int n;
    printf("Digite um inteiro positivo: ");
    scanf("%d", &n);

    printf("Soma: %d\n", soma(n));
}
```

Crie um programa que calcule o fatorial de um número fornecido pelo usuário através da recursividade.

O fatorial de 'n' é representado por n!, onde:

n! = n * (n-1) * (n-2)*..1

O raciocínio desse exemplo é análogo ao do exemplo anterior, porém, vamos usar a multiplicação ao invés da soma.

Antes de resolvermos, vamos ver a idéia matemática por trás do fatorial.

Como dissemos na questão, para n=5:

5! = 5 * 4 * 3 * 2 * 1

Para n=4:

4! = 4 * 3 * 2 * 1

Para n=3:

3! = 3 * 2 * 1

E assim sucessivamente.

Porém, note que:

5! = 5 * 4 * 3 * 2 * 1 = 5 * 4!

E também:

4! = 4 * 3 * 2 * 1 = 4 * 3!

Podemos formular uma fórmula geral:

n! = n * (n-1)!

Abrindo esse produto, devemos parar somente quando o último elemento do produto for 1:

n! = n * (n-1)! = n * (n-1) * (n-2)! = n * (n-1) * (n-2) * ... * 1

Para programar isso, criamos uma função fatorial(int n) que retorna 1 se for passado 1 como argumento (pois fatorial(1) = 1) e caso o argumento seja maior que 1:

fatorial(n) = n * fatorial(n-1)

```c
#include <stdio.h>

int fatorial(int n)
{
    if(n == 1)
        return 1;
    else
        return ( n * fatorial(n-1) );
}

int main()
{
    int n;
    printf("Digite um inteiro positivo: ");
    scanf("%d", &n);

    printf("%d! = %d\n", n, fatorial(n));
}
```

Crie um programa que calcule o enésimo número da sequencia de Fibonacci através da recursividade.

0, 1, 2, 3, 5, 8, 13, 21, 34

Formula Geral:

Fn = Fn-1 + Fn-2 para n>2.

```c
#include <stdio.h>

int fibonacci(int n)
{
    if(n == 1 || n == 2)
        return 1;
    else
        return ( fibonacci(n -1) +  fibonacci(n-2) );
}

int main()
{
    int n;
    printf("Digite um inteiro positivo: ");
    scanf("%d", &n);

    printf("%d! = %d\n", n, fibonacci(n));
}
```

Exercício

1) Faça uma função recursiva, em linguagem C, que calcule o valor da série S descrita a seguir para um valor n>0 a ser fornecido como parâmetro para a mesma: S = 1 + 1/1! + 1/2! + 1/3! + 1 /n!

2) Faça uma função recursiva, em linguagem C, que calcule o valor da série S descrita a seguir para um valor n>0 a ser fornecido como parâmetro para a mesma.

$$S = 2 + \frac{5}{2} + \frac{10}{3} + \ldots + \frac{1+n^2}{n}$$