RECURSIVIDADE	RECURSIVIDADE
Como podemos definir estruturas Recursivas?	Já foi visto que repetições podem ser obtidas escrevendo-se laços, tais como laços for ou laços while. Outra forma de se obter repetição é por meio da recursão. Um objeto é recursivo se ele é definido parcialmente em termos de si próprio. A recursão é considerada uma técnica poderosa em definições matemáticas. O poder da recursão está na possibilidade de definir elementos com base em versões anteriores deles mesmos.
Que 2 características definem uma Estrutura como Recursiva?	As 2 características são: - **Uma condição de parada**, isto é, algum evento que encerre a autochamada consecutiva. No caso do fatorial, isso ocorre quando a função é chamada com parâmetro (n) igual a 1. Um algoritmo recursivo precisa garantir que esta condição será alcançada; - **Uma mudança de "estado"** a cada chamada, isto é, alguma "diferença" entre uma chamada e a próxima. No caso do fatorial, o parâmetro n é decrementado a cada chamada;
Quais são algumas operações que exigem Recursividade?	Dentre funções e métodos que exigem Recursividade podemos contar com 3 exemplos básicos: **- Potêncialização** (Quando fazemos um número "n" vezes sobre ele mesmo); **- Fatorial** (Quando operamos um número vezes o seu anterior, vezes o seu anterior até chegar em 1, o resultado é o fatorial daquele número. Por exemplo, fatorial de 5 é 5 x 4 x 3 x 2 x 1 que dá 120); **- Sequência de Fibonacci** (É uma sequência padrão partindo do número 1 onde o número 1 é somado a ele mesmo e o resultado deve ser somado ao número antecessor vez após vez, por exemplo: 1 + 1 = 2, 1 + 2 = 3, 2 + 3 = 5, 3 + 5...);
Como se dá uma uma Função Fatorial?	Quando queremos expressar uma variável como fatorial usamos a notação "nomeDaVariavel!" (nome seguido pelo sinal de exclamação). Em precisamos criar uma função onde o número recebido na variável seja operado da seguinte forma: n * (n-1) dentro de um laço que fará a conta vez após vez até chegar em 1. Lembrando que, quando desejamos calcular o fatorial de "0", o valor será sempre "1". Porém, na Linguagem C existe uma função fatorial() que já calcula automáricamente para nós, devemos usá-la da seguinte maneira: ***int fatorial (int n){*** ***if (n == 0)*** ***return (1);*** ***else return (n*fatorial (n-1));*** ***}*** **OBS: porém, podemos optar por criar nós mesmos uma função de chamada recursiva, para isso temos que tomar um cuidado especial de garantir a parada de uma recursão, do contrário, os laços iriam se repetir infinitamente ou trazer resultados errados.**
- reservada para a questão acima -	
- reservada para a questão acima -	
Como se dá uma uma Função Seguindo o Padrão de Fibonacci?	A Seqüência de Fibonacci é 0, 1, 1, 2, 3, 5, 8,… Abaixo, segue a função Fibonacci definida recursivamente. ***inf Fibonacci (int n){*** ***if (n==0)*** ***return (0);*** ***else if (n==1)*** ***return (1);*** ***else return (Fibonacci(n-1) + Fibonacci(n-2));*** ***}***
- reservada para a questão acima -	
Quando devemos usar a Recursividade e quando não devemos usar?	**Use recursão quando:** - O problema é naturalmente recursivo (clareza) e a versão recursiva do algoritmo não gera ineficiência evidente se comparado com a versão iterativa do mesmo algoritmo. - O algoritmo se torna compacto sem perda de clareza ou generalidade. - É possível prever que o número de chamadas ou a carga sobre a pilha de passagem de parâmetros não irá causar interrupção do processo. **Não use recursão quando:** - A solução recursiva causa ineficiência se comparada com uma versão iterativa. - A recursão é de cauda. - Parâmetros consideravelmente grandes têm que ser passados por valor. - Não é possível prever se o número de chamadas recursivas irá causar sobrecarga da pilha de passagem de parâmetros.
- reservada para a questão acima -	