Exercício Recursividade – AEDs PUC Minas Barreiro

Raí Átila Cavalcante

Ex 1 – FATORIAL

```
using System;
class Program{
    public static void Main(string[] args){
        // Entrada de dados
        int entrada;
        entrada = int.Parse(Console.ReadLine());
        // Declarando variável para condição de parada
        int parada = entrada - 1;
        // Imprimindo resultado
        Console.WriteLine(fatorialRecursivo(entrada,parada));
    }
    // Criando Função Recursiva
    static int fatorialRecursivo(int entrada,int parada){
        // Condição de parada da recursividade
        if(parada == 1){
            return entrada;
        }
        // Recursividade
        else{
            entrada *= parada;
            return fatorialRecursivo(entrada,parada-1);
        }
    }
}
```

Ex 2 – SOMA DE INTEIROS

```
using System;
class Program{
    public static void Main(string[] args){
        // Entrada de dados
        int entrada;
        entrada = int.Parse(Console.ReadLine());
        // Declarando variável para condição de parada
        int parada = entrada - 1;
        // Imprimindo resultado
        Console.WriteLine(somaRecursiva(entrada,parada));
    }
    // Criando Função Recursiva
    static int somaRecursiva(int entrada,int parada){
        // Condição de parada da recursividade
        if(parada == 0){
            return entrada;
        }
        // Recursividade
        else{
            entrada += parada;
            return somaRecursiva(entrada,parada-1);
        }
    }
}
```

Ex 3 – FIBONACCI

```
using System;
class Program{
    public static void Main(string[] args){
        // Declarando variáveis
        int numero, percorre = 0, resp = 1, proxNum = 0, numAnterior = 1
;
        // Entrada de dados do n-ésimo numero
        numero = int.Parse(Console.ReadLine());
        // Chamando a função que calcula o número de fibonacci e imprime
o resultado
        Console.WriteLine(fibonacciRecursivo(numero,percorre,resp,proxNum
,numAnterior));
    }

    // Criando a função recursiva que calcula a sequência de fibonacci
    static int fibonacciRecursivo(int numero,int percorre,int resp,int
proxNum,int numAnterior){
        // Condição de parada
        if(percorre == numero){
            return resp;
        }
        // Recursividade
        else{
            // Cálculos fibonacci
            resp = numAnterior + proxNum;
            numAnterior = proxNum;
            proxNum = resp;

            return
fibonacciRecursivo(numero,percorre+1,resp,proxNum,numAnterior);
        }
    }
}
```

Ex 4 – POTÊNCIA

```
using System;
class Program{
    public static void Main(string[] args){
        // Declarando variáveis
        int numero, expoente, resp = 1;
        // Entrada de dados
        numero = int.Parse(Console.ReadLine());
        expoente = int.Parse(Console.ReadLine());
        // Chamando a função Recursiva
        Console.WriteLine(potenciaRecursivo(numero,expoente,resp));
    }

    // Declarando a função recursiva
    static int potenciaRecursivo(int numero, int expoente, int resp){
        // Condição de parada
        if(expoente == 0){
            return resp;
        }
        // Fazendo a operação por Recursividade
        else{
            resp *= numero;
            return potenciaRecursivo(numero,expoente-1,resp);
        }
    }
}
```

Ex 5 – INVERSÃO DE STRING

```
using System;
class Program{
    public static void Main(string[] args){
        // Entrada de dados e variáveis
        string frase, contrario = "";
        frase = Console.ReadLine();
        int percorre = frase.Length-1;
        // Chamando a função recursiva e imprimindo resultado
        Console.WriteLine(inverteRecursivo(frase,contrario,percorre));
    }
    // Função Recursiva
    static string inverteRecursivo(string frase, string contrario, int
percorre){
        // Condição de parada
        if(percorre < 0){
            return contrario;
        }
        // Operação + recursividade
        else{
            contrario += frase[percorre];
            return inverteRecursivo(frase,contrario,percorre-1);
        }
    }
}
```

Ex 6 – SOMA DE ELEMENTOS

```
using System;
class Program{
    public static void Main(string[] args){
        int percorre = 0;
        int resp = 0;
        // Entrada de dados do tamanho da lista(vetor)
        Console.Write("Digite o tamanho da lista: ");
        int tam = int.Parse(Console.ReadLine());
        int[] entradas = new int [tam];
        // Chamando função recursiva para preencher a lista(vetor)
        preencheRecursivo(entradas,percorre,tam);
        // Calculando os números da lista através de uma função recursiva
e imprimindo o resultado
        Console.WriteLine(calculaRecursivo(entradas,percorre,tam,resp));
    }

    // Criando Função Recursiva que preenche o vetor
    static int[] preencheRecursivo(int[] entradas, int percorre, int
tam){
        // Condição de parada
        if(percorre == tam){
            percorre = 0;
            return entradas;
        }
        // Entrada de dados + recursividade
        else{
            entradas[percorre] = int.Parse(Console.ReadLine());
            return preencheRecursivo(entradas,percorre+1,tam);
        }
    }

    // Criando função recursiva que calcula os números
    static int calculaRecursivo(int[] entradas, int percorre, int tam,
int resp){
        // Condição de parada
        if(percorre == tam){
            return resp;
        }
        // Soma + Recursividade
        else{
            resp += entradas[percorre];
            return calculaRecursivo(entradas,percorre+1,tam,resp);
        }
    }
}
```

Ex 7 – CONTAGEM DE ELEMENTOS

```
using System;
class Program{
    public static void Main(string[] args){
        int tam;
        int percorre = 0;
        int cont = 0;
        // Entrada de dados com o tamanho do vetor
        Console.Write("Digite o tamanho da lista: ");
        tam = int.Parse(Console.ReadLine());
        int[] numeros = new int[tam];
        // Chamando função recursiva para preencher a lista(vetor)
        preencheRecursivo(numeros,percorre,tam);
        // Chamando função recursiva que verifica os numeros positivos e
imprime a resposta
        Console.WriteLine(contagemRecursiva(numeros,percorre,tam,cont));
    }

    // Criando Função Recursiva que preenche o vetor
    static int[] preencheRecursivo(int[] numeros, int percorre, int tam){
        // Condição de parada
        if(percorre == tam){
            percorre = 0;
            return numeros;
        }
        // Entrada de dados + recursividade
        else{
            numeros[percorre] = int.Parse(Console.ReadLine());
            return preencheRecursivo(numeros,percorre+1,tam);
        }
    }

    // Criando Função Recursiva que verifica e faz a contagem de
positivos
    static int contagemRecursiva(int[] numeros, int percorre, int tam,
int cont){
        // Condição de parada
        if(percorre == tam){
            return cont;
        }
        // Recursividade
        else{
            // Verificando se é positivo
            if(numeros[percorre] >= 0){
                // Fazendo a contagem
                cont++;
            }
            return contagemRecursiva(numeros,percorre+1,tam,cont);
```

```
        }
    }
}
```