Estrutura de Dados

Recursividade

Uma função recursiva é uma função que se refere a si própria. A ideia consiste em utilizar a própria função que estamos definindo, na sua definição.

Em todas as funções recursivas existe:

- Um passo básico (ou mais) cujo resultado é imediatamente conhecido.
- Um passo recursivo em que se tenta resolver um sub-problema do problema inicial.

- **Recursividade** é uma idéia inteligente que desempenha um papel central na programação funcional e na ciência da computação em geral.
- **Recursividade** é o mecanismo de programação no qual uma definição de função ou de outro objeto refere-se ao próprio objeto sendo definido.
- Assim **função recursiva** é uma função que é definida em termos de si mesma.
- Recursividade é o mecanismo básico para repetições nas linguagens funcionais.
- São sinônimos: recursividade, recursão, recorrência.

- Estratégia para a definição recursiva de uma função:
  1. dividir o problema em problemas menores do mesmo tipo
  2. resolver os problemas menores (dividindo-os em problemas ainda menores, se necessário)
  3. combinar as soluções dos problemas menores para formar a solução final
- Ao dividir o problema sucessivamente em problemas menores eventualmente os casos simples são alcançados:
  - não podem ser mais divididos
  - suas soluções são definidas explicitamente

De modo geral, uma **definição de função recursiva** é dividida em duas partes:

- Há um ou mais **casos base** que dizem o que fazer em situações simples, onde não é necessária nenhuma recursão.
  Nestes casos a resposta pode ser dada de imediato, sem chamar recursivamente a função sendo definida.
  Isso garante que a recursão eventualmente possa parar.
- Há um ou mais **casos recursivos** que são mais gerais, e definem a função em termos de uma chamada *mais simples* a si mesma.

Exercícios

- Forneça soluções recursivas para os problemas abaixo
  - cálculo do fatorial de um número.
  - cálculo do elemento *n* da série de Fibonacci.
    - $f_0 = 0$, $f_1 = 1$,
    - $f_n = f_{n-1} + f_{n-2}$ para $n >= 2$.
  - busca sequencial.
  - busca binária.
  - torre de Hanói.

- Solução para o algoritmo recursivo para cálculo do fatorial de um número

```
int fatr(int m) {
  if(m == 0) {
    return 1;
  }
  else {
    return (m * fatr(m-1));
  }
}
```

- Solução para o cálculo do elemento *n* da série de Fibonacci utilizando um algoritmo recursivo

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

- Solução para o algoritmo recursivo de busca sequencial

```
int sequencial(int valor, int[] vetor, int n) {
  if(n == 1) {
    if(vetor[0] == valor) {
      return 0;
    }
    else {
      return -1;
    }
  } else {
    int index = sequencial(valor, vetor, n-1);
    if(index < 0) {
      if(vetor[n-1] == valor) {
        index = n-1;
      }
    }
    return index;
  }
}
```

```
//Função recursiva que transforma de decimal a binário
#include <stdio.h>
int bin(int k)
{
    if (k < 2) // Momento de parada (CASO BASE)
        return k;

    return (10 * bin( k / 2 )) + k % 2;
     // Iniciando a multiplicação pelo valor que
     // não pode mais ser dividido por 2 e somando com a sobra da divisão.
}
main(){
  int x;
  printf("Digite o valor em decimal: \n\n");
  scanf("%d",&x);
  printf("\n\nBase 10 = %d  - Base 2 = %d",x,bin(x));
}
```

Exercícios:

1. Implemente uma função recursiva que, dados dois números inteiros x e n, calcula o valor de x^n

2. Considere a função abaixo:

```
int X(int a)
{
  if ( a <= 0 )
    return 0;
  else
    return a + X(a-1);
}
```

a. O que essa função faz?

b) Calcule e faça a árvore de recorrência de X(7)

3. O máximo divisor comum (***MDC***) de dois números inteiros ***x*** e ***y*** pode ser calculado usando-se uma definição recursiva:

$$MDC(x, y) = MDC(x - y, y), \text{ se } x > y .$$

Além disso, sabe-se que:

$$MDC(x, y) = MDC(y, x)$$

$$MDC(x, x) = x$$

Exemplo:

$$MDC(10,6) = MDC(4,6) = MDC(6,4) = MDC(2,4) = MDC(4,2) = MDC(2,2) = 2$$

Então, pede-se que seja criada uma função recursiva para descrever tal definição. Crie, também, um algoritmo que leia os dois valores inteiros e utilize a função criada para calcular o ***MDC*** de ***x*** e ***y***, e imprima o valor computado.

Calcule e faça a árvore de recursão de MDC(15,36)

O mínimo múltiplo comum (***M.M.C.***) entre dois números inteiros e positivos ***X*** e ***Y*** é definido como sendo o menor inteiro positivo, que seja múltiplo comum a ***X*** e ***Y***. Pede-se que seja criada uma função recursiva (não serão aceitas funções não recursivas) para o cálculo do ***M.M.C.***, onde a função deverá retornar 0 caso não seja possível computar o ***M.M.C.*** e o valor do ***M.M.C.*** entre ***X*** e ***Y*** em caso contrário. Então, apresenta-se a seguinte definição recursiva que deve ser implementada:

$$M.M.C.(X,Y) = \begin{cases} Z * M.M.C.(X/Z, Y/Z), & \text{se } X \bmod Z = 0 \text{ e } Y \bmod Z = 0 \text{ para } 1 < Z \le X, Y \\ Z * M.M.C.(X/Z, Y) & \text{se } X \bmod Z = 0 \text{ e } Y \bmod Z \neq 0 \text{ para } 1 < Z \le X, Y \\ Z * M.M.C.(X, Y/Z) & \text{se } X \bmod Z \neq 0 \text{ e } Y \bmod Z = 0 \text{ para } 1 < Z \le X, Y \end{cases}$$

$$M.M.C.(1,1) = 1$$

Escreva também um algoritmo para testar a função criada.

Crie a árvore de recursão para calcular: MMC(15,40)

Qual o valor de X (4) se X é dada pelo seguinte código?

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