

# Recursão – Lista de exercícios

# Solução lista de exercícios

- Vamos tentar, antes de mais nada, estabelecer como seria a definição recursiva do problema
- Depois usaremos na implementação

# Exercício 3

- 3) Implemente uma função recursiva que, dados dois números inteiros  $x$  e  $n$ , calcula o valor de  $x^n$ .
- Caso base?

# Exercício 3

- 3) Implemente uma função recursiva que, dados dois números inteiros  $x$  e  $n$ , calcula o valor de  $x^n$ .
- Caso base?  $x^0 = 1$
- Passo da recursão:

# Exercício 3

- 3) Implemente uma função recursiva que, dados dois números inteiros  $x$  e  $n$ , calcula o valor de  $x^n$ .
- Caso base?  $x^0 = 1$
- Passo da recursão:  $x^n = x * x^{n-1}$

## Exercício 3

```
int pot(int b, int p)
{
    if ( p == 0 ) return 1;
    return ( b * pot(b, p-1) );
}
```

# Exercício 7

- 7) Usando recursividade, calcule a soma de todos os valores de um array de reais.
- Caso base?

# Exercício 7

- 7) Usando recursividade, calcule a soma de todos os valores de um array de reais.
- Caso base? **Tamanho do array = 0. Soma é 0.**

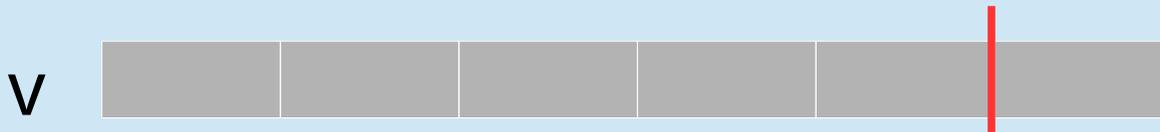
# Exercício 7

- 7) Usando recursividade, calcule a soma de todos os valores de um array de reais.
- Caso base? **Tamanho do array = 0. Soma é 0.**
- Passo da recursão:



# Exercício 7

- 7) Usando recursividade, calcule a soma de todos os valores de um array de reais.
- Caso base? Tamanho do array = 0. Soma é 0.
- Passo da recursão:



$v[n-1]$  + soma do restante do array.

# Exercício 7

```
int soma_a(int v[], int n)
{
    if ( n == 0 ) return 0;
    return v[n-1] + soma(v, n-1);
}
```

# Exercício 1

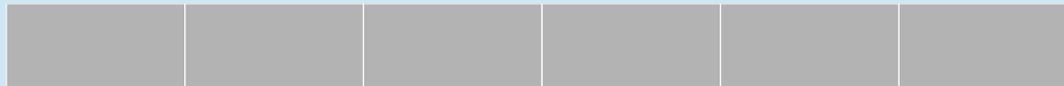
- 1) Dado um array de inteiros e o seu número de elementos, inverta a posição dos seus elementos.
  
- Caso base?

# Exercício 1

- 1) Dado um array de inteiros e o seu número de elementos, inverta a posição dos seus elementos.
- Caso base? **Tamanho do array menor ou igual a 1**

# Exercício 1

- 1) Dado um array de inteiros e o seu número de elementos, inverta a posição dos seus elementos.
- Caso base? **Tamanho do array menor ou igual a 1**
- Passo da recursão:

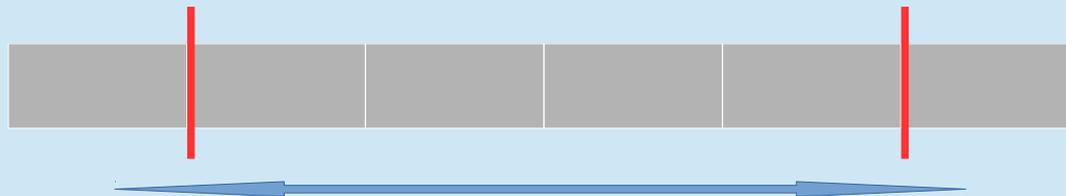


# Exercício 1

- 1) Dado um array de inteiros e o seu número de elementos, inverta a posição dos seus elementos.

- Caso base? **Tamanho do array menor ou igual a 1**

- Passo da recursão:



- Troca 1o. e último elementos e inverte resto do array.

# Exercício 1

```
void invertre(int v[], int esq, int dir)
{
    int t;
    if (esq >= dir) return;
    t = v[esq];
    v[esq] = v[dir];
    v[dir] = t;
    invertre(v, esq+1, dir - 1);
}
```

# Exercício 9

- 9) Escreva uma função recursiva que determine quantas vezes um dígito  $K$  ocorre em um número natural  $N$ . Por exemplo, o dígito 2 ocorre 3 vezes em 762021192.
- Caso base?

# Exercício 9

- 9) Escreva uma função recursiva que determine quantas vezes um dígito  $K$  ocorre em um número natural  $N$ . Por exemplo, o dígito 2 ocorre 3 vezes em 762021192.
- Caso base? Quando todos os dígitos já foram examinados, ou seja,  $N = 0$

# Exercício 9

- 9) Escreva uma função recursiva que determine quantas vezes um dígito  $K$  ocorre em um número natural  $N$ . Por exemplo, o dígito 2 ocorre 3 vezes em 762021192.
- Caso base? Quando todos os dígitos já foram examinados, ou seja,  $N = 0$
- Passo da recursão:  $n_4n_3n_2n_1n_0$

# Exercício 9

- 9) Escreva uma função recursiva que determine quantas vezes um dígito  $K$  ocorre em um número natural  $N$ . Por exemplo, o dígito 2 ocorre 3 vezes em 762021192.
- Caso base? Quando todos os dígitos já foram examinados, ou seja,  $N = 0$
- Passo da recursão:  $n_4n_3n_2n_1n_0$   
(0 ou 1) + número de ocorrências em  $N / 10$  ( $n_4n_3n_2n_1$ )

# Exercício 9

```
int conta_dig(int N, int K)
{
    if ( N == 0 ) return 0;
    return conta_dig( N / 10 , K) + ( N % 10 == K );
}
```

# Exercício 4

- 4) Um problema típico em ciência da computação consiste em converter um número da sua forma decimal para a forma binária.
- Caso base?

# Exercício 4

- 4) Um problema típico em ciência da computação consiste em converter um número da sua forma decimal para a forma binária.
- Caso base? Quando o número já foi todo transformado em binário. Ou seja:  $x = 0$
- Passo da recursão:

# Exercício 4

- 4) Um problema típico em ciência da computação consiste em converter um número da sua forma decimal para a forma binária.
- Caso base? Quando o número já foi todo transformado em binário. Ou seja:  $x = 0$
- Passo da recursão: Saber como  $x/2$  é convertido. Depois, adicionar um dígito (0 ou 1) relativo a  $x$ .

# Exercício 4

```
void print_bin(int x)
{
    if ( x == 0 ) return;
    print_bin(x / 2);
    printf("%d", x % 2);
}
```

# Exercício 4

```
void print_bin(int x)
{
    if ( x == 0 ) return;
    print_bin(x / 2);
    printf("%d", x % 2);
}
```

```
void print_bin(int x)
{
    if ( x == 0 )
    {
        printf("0");
        return;
    }
    print_bin(x / 2);
    printf("%d", x % 2);
}
```

# Exercício 5

- 5) O máximo divisor comum (MDC) de dois números inteiros  $x$  e  $y$  pode ser calculado usando-se uma definição recursiva:
  - $\text{MDC}(x, y) = \text{MDC}(x - y, y)$ , se  $x > y$
  - $\text{MDC}(x, y) = \text{MDC}(y, x)$
  - $\text{MDC}(x, x) = x$
- Caso base?

# Exercício 5

- 5) O máximo divisor comum (MDC) de dois números inteiros  $x$  e  $y$  pode ser calculado usando-se uma definição recursiva:
  - $\text{MDC}(x, y) = \text{MDC}(x - y, y)$ , se  $x > y$
  - $\text{MDC}(x, y) = \text{MDC}(y, x)$
  - $\text{MDC}(x, x) = x$
- Caso base?  $x == y$

# Exercício 5

- 5) O máximo divisor comum (MDC) de dois números inteiros  $x$  e  $y$  pode ser calculado usando-se uma definição recursiva:
  - $\text{MDC}(x, y) = \text{MDC}(x - y, y)$ , se  $x > y$
  - $\text{MDC}(x, y) = \text{MDC}(y, x)$
  - $\text{MDC}(x, x) = x$
- Caso base?  $x == y$
- Passo da recursão?

# Exercício 5

- 5) O máximo divisor comum (MDC) de dois números inteiros  $x$  e  $y$  pode ser calculado usando-se uma definição recursiva:
  - $\text{MDC}(x, y) = \text{MDC}(x - y, y)$ , se  $x > y$
  - $\text{MDC}(x, y) = \text{MDC}(y, x)$
  - $\text{MDC}(x, x) = x$
- Caso base?  $x == y$
- Passo da recursão? Os dois outros casos acima. O problema é definido recursivamente.

# Exercício 5

```
int mdc(int p, int q)
{
    if ( p == q ) return p;
    if ( p < q ) return mdc(q, p);
    return mdc(p - q, q);
}
```

# Exercício 6

- 6) Pode-se calcular o resto da divisão, MOD, de  $x$  por  $y$ , dois números inteiros positivos, usando-se a seguinte definição:
  - $\text{MOD}(x,y) = \text{MOD}(x - y, y)$  se  $x > y$
  - $\text{MOD}(x,y) = x$  se  $x < y$
  - $\text{MOD}(x,y) = 0$  se  $x = y$
- Caso base?

# Exercício 6

- 6) Pode-se calcular o resto da divisão, MOD, de  $x$  por  $y$ , dois números inteiros positivos, usando-se a seguinte definição:
  - $\text{MOD}(x,y) = \text{MOD}(x - y, y)$  se  $x > y$
  - $\text{MOD}(x,y) = x$  se  $x < y$
  - $\text{MOD}(x,y) = 0$  se  $x = y$
- Caso base? São dois:  $x < y$  ou  $x = y$

# Exercício 6

- 6) Pode-se calcular o resto da divisão, MOD, de  $x$  por  $y$ , dois números inteiros positivos, usando-se a seguinte definição:
  - $\text{MOD}(x,y) = \text{MOD}(x - y, y)$  se  $x > y$
  - $\text{MOD}(x,y) = x$  se  $x < y$
  - $\text{MOD}(x,y) = 0$  se  $x = y$
- Caso base? São dois:  $x < y$  ou  $x = y$
- Passo da recursão:

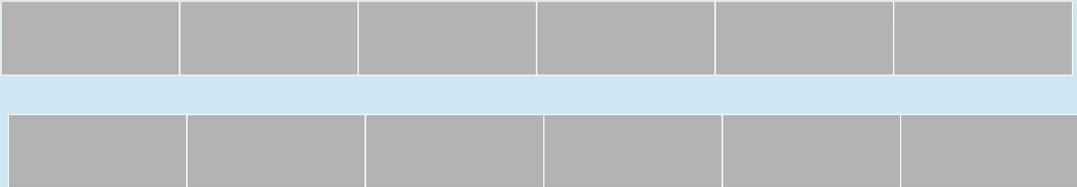
# Exercício 6

- 6) Pode-se calcular o resto da divisão, MOD, de  $x$  por  $y$ , dois números inteiros positivos, usando-se a seguinte definição:
  - $\text{MOD}(x,y) = \text{MOD}(x - y, y)$  se  $x > y$
  - $\text{MOD}(x,y) = x$  se  $x < y$
  - $\text{MOD}(x,y) = 0$  se  $x = y$
- Caso base? São dois:  $x < y$  ou  $x = y$
- Passo da recursão:  $\text{MOD}(x - y, y)$  se  $x > y$

# Exercício 2a

- 2) Escreva as funções recursivas que unem dois (arrays), sem elementos repetidos, classificadas considerando que as duas listas não têm elementos em comum
  
- Caso base?

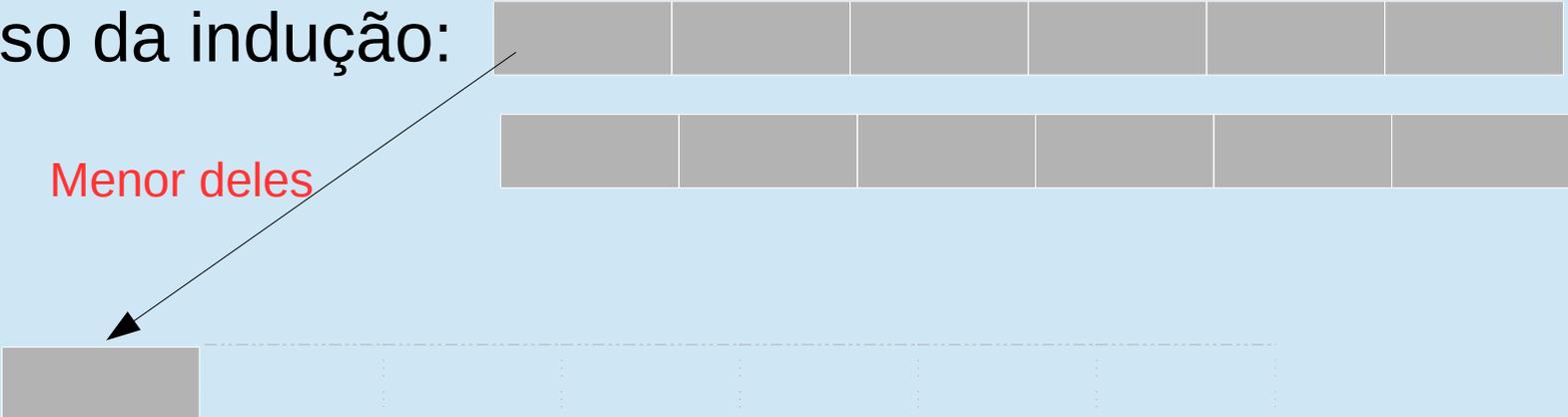
# Exercício 2a

- 2) Escreva as funções recursivas que unem dois (arrays), sem elementos repetidos, classificadas considerando que as duas listas não têm elementos em comum
- Caso base? Quando ambos os arrays têm tamanho 0
- Passo da indução: 

# Exercício 2a

- 2) Escreva as funções recursivas que unem dois (arrays), sem elementos repetidos, classificadas considerando que as duas listas não têm elementos em comum

- Caso base? Quando ambos os arrays têm tamanho 0

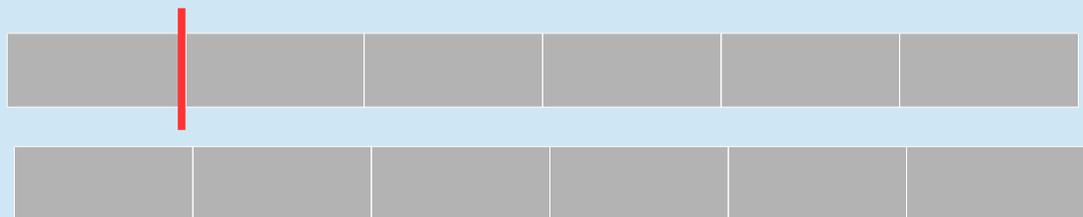
- Passo da indução:
 

# Exercício 2a

- 2) Escreva as funções recursivas que unem dois (arrays), sem elementos repetidos, classificadas considerando que as duas listas não têm elementos em comum

- Caso base? Quando ambos os arrays têm tamanho 0

- Passo da indução:



# Exercício 2a

```

void merge(int v1[], int n1,
           int v2[], int n2, int v3[])
{
    if ( n1 == 0 && n2 == 0 ) return;

    if (n1 == 0) {
        v3[0] = v2[0];
        merge(v1, n1, ++v2, --n2, ++v3);
    }
    else
    if (n2 == 0) {
        v3[0] = v1[0];
        merge(++v1, --n1, v2, n2, ++v3);
    }
    else

```

```

    if (v1[0] <= v2[0]) {
        v3[0] = v1[0];
        merge(++v1, --n1, v2, n2, ++v3);
    }
    else {
        v3[0] = v2[0];
        merge(v1, n1, ++v2, --n2, ++v3);
    }
}

```

# Exercício 2c

```

void merge(int v1[], int n1,
           int v2[], int n2, int v3[])
{
    if ( n1 == 0 && n2 == 0 ) return;

    if (n1 == 0) {
        v3[0] = v2[0];
        merge(v1, n1, ++v2, --n2, ++v3);
    }
    else
    if (n2 == 0) {
        v3[0] = v1[0];
        merge(++v1, --n1, v2, n2, ++v3);
    }
    else

```

```

    if (v1[0] < v2[0]) {
        v3[0] = v1[0];
        merge(++v1, --n1, v2, n2, ++v3);
    }
    else
    if (v1[0] > v2[0]){
        v3[0] = v2[0];
        merge(v1, n1, ++v2, --n2, ++v3);
    }
    else {
        v3[0] = v2[0];
        merge(++v1, --n1, ++v2, --n2, ++v3);
    }
}

```

# Exercício 8

- 8) Escreva um algoritmo recursivo capaz de gerar todos os elementos do conjunto potência dado um conjunto formado por letras.
- Caso base?

# Exercício 8

- 8) Escreva um algoritmo recursivo capaz de gerar todos os elementos do conjunto potência dado um conjunto formado por letras.
- Caso base?  $2^{\{\}} = \{\}$
- Passo da recursão:

# Exercício 8

- 8) Escreva um algoritmo recursivo capaz de gerar todos os elementos do conjunto potência dado um conjunto formado por letras.
- Caso base?  $2^{\{\}} = \{\}$
- Passo da recursão:  
 $2^{\{a,b,c\}} = 2^{\{b,c\}} \cup \{a\} \times 2^{\{b,c\}}$

# Exercício 8

```

void potencia(char pref[],
char s[], char *p) {
int l; char aux[100];
    l = strlen(s);
    if ( l == 0 )    {
        if (strlen(pref) ==
0 )
            strcpy(p, "{}");
        else {
            strcat(p, " ");
            strcat(p, pref);
        }
        return;
    }
    potencia(pref, s+1, p);
    strcpy(aux, pref);
    l = strlen(aux);
    aux[l] = s[0];
    aux[l+1] = '\0';
    potencia(aux, s+1, p);
    return;
}

```