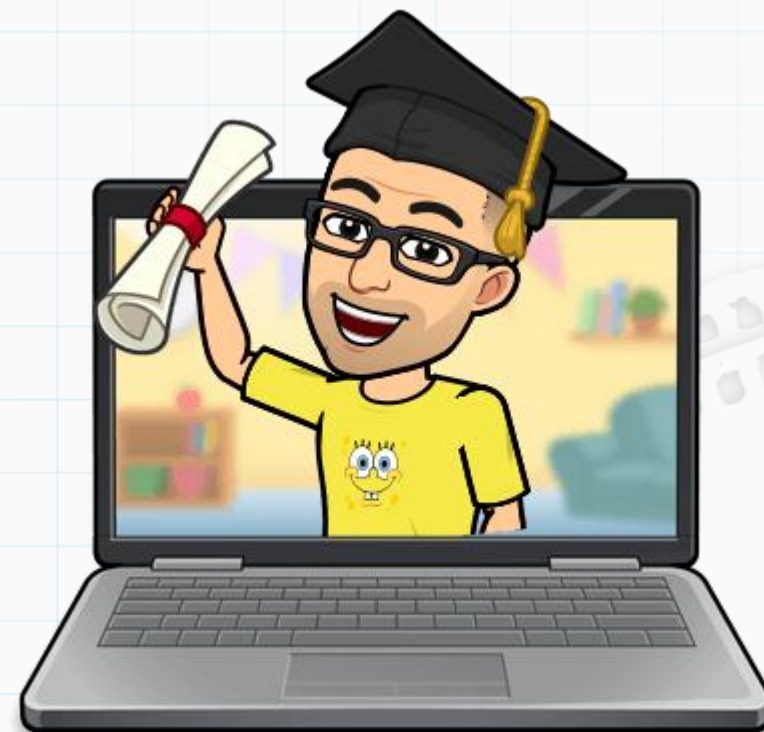
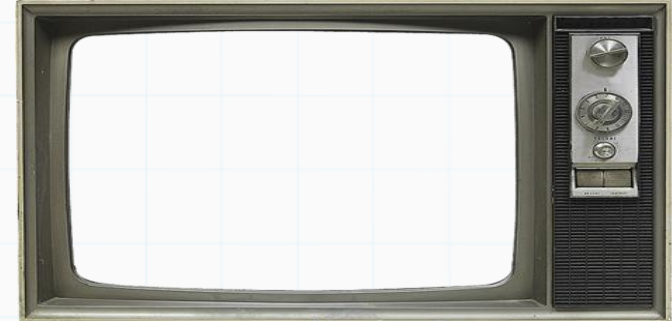


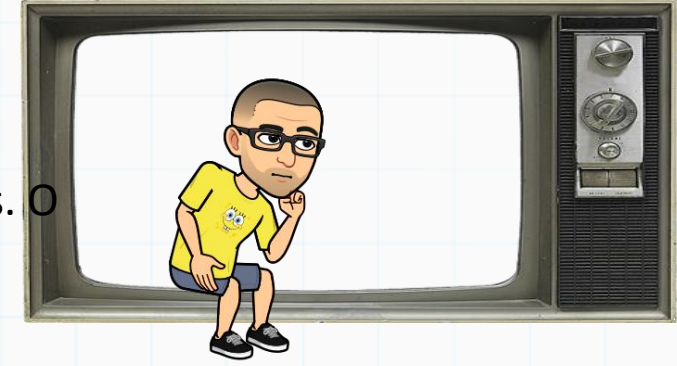
Programação Estruturada

Professor : Yuri Frota

yuri@ic.uff.br



Recursão - LAB



1) Conta Dígitos: faça um programa que receba número $n \geq 0$ e imprima a soma de seus dígitos. O programa deve usar uma função recursiva para fazer a soma.

Obs: O programa deve ser feito apenas com variáveis numéricas unidimensionais (ex: int, float, etc)

$$\begin{array}{r} 183 \\ 1 + 8 + 3 = 12 \\ 48 \\ 4 + 8 = 12 \end{array}$$

Use só o que aprendemos até hoje

Lembre-se:

```
int x;
```

```
x%10; // ultimo dígito de x
```

```
x/10; // x sem o último dígito
```

DICA: Na função recursiva deve-se:

- receber um número n
- retornar a soma entre o último dígito de n **E** chamada recursiva de n sem o último dígito

Ex: $n=183$

$\text{conta_dig}(183) = \text{retorna } 3 + \text{conta_dig}(18)$



Recursão - LAB

2) Granizo: A sequencia de granizo é definida para um determinado número $n \geq 0$ da seguinte maneira:

$$f(n) = \left\{ \begin{array}{l} \frac{n}{2}, \quad \text{se } n \text{ for par} \\ 3n + 1. \text{ se } n \text{ for impar} \end{array} \right\}$$

A sequencia termina quando alcançado o número 1. Exemplo:

n: 20
20, 10, 5, 16, 8, 4, 2, 1,

n: 7
7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1,

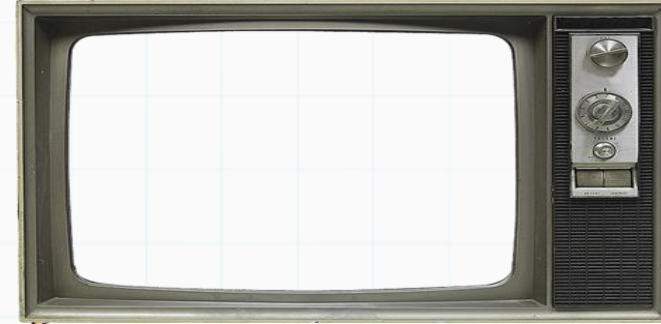
Faça um programa que dado $n \geq 0$ imprima a sequencia de granizo com uma função recursiva

```
void granizo(int n)
```

Depois faça outra função recursiva para imprimir a quantidade de elementos da sequencia

```
int conta_granizo(int n)
```

continua



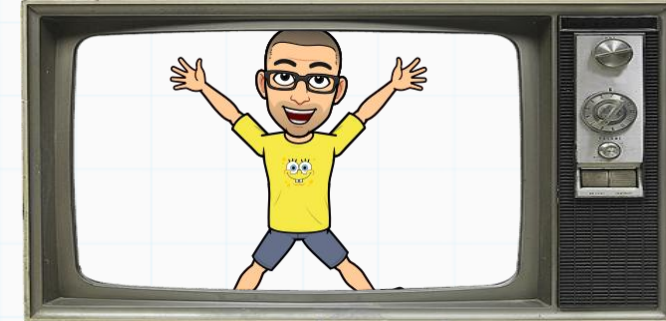
Use só o que aprendemos até hoje

Recursão - LAB

2) Granizo: Exemplo

n: 20
20, 10, 5, 16, 8, 4, 2, 1,
tamanho = 8

n: 7
7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1,
tamanho = 17



DICA: Na função recursiva de granizo deve-se:

- receber um número n
- imprimir n
- chamar função recursiva de acordo com a equação de recorrência.

DICA2: Na função recursiva de conta_granizo:

- mesma coisa, mas sem impressão e retornar $(1 + \text{conta_granizo}())$



Quantos elementos tem a sequência de Granizo começando de $n=27$?

Recursão - LAB

3) Inverte: Dado um vetor de tamanho $n > 0$, faça um programa para inverter a ordem dos elementos do vetor. O programa deve usar uma função recursiva para inverter o vetor:

```
void inverte(int n, int vetor[n], int inicio, int fim)
```

e outra para imprimir o vetor

```
void imprime(int n, int vetor[n], int ind)
```

Exemplo:

```
n: 6
```

```
v[0] = 4
```

```
v[1] = 2
```

```
v[2] = 7
```

```
v[3] = 9
```

```
v[4] = 1
```

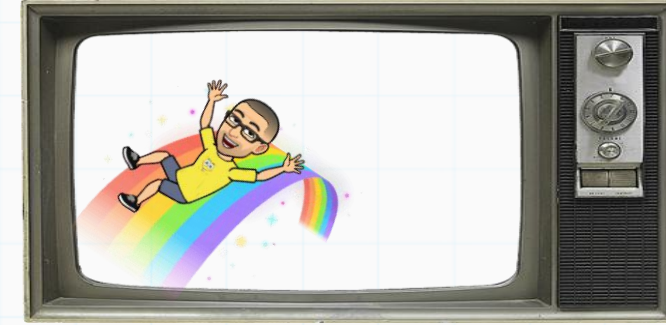
```
v[5] = 2
```

```
vetor
```

```
v = { 4, 2, 7, 9, 1, 2 }
```

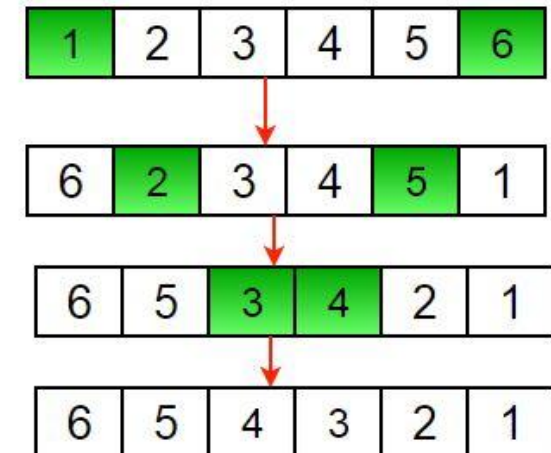
```
vetor invertido
```

```
v = { 2, 1, 9, 7, 2, 4 }
```



Use só o que aprendemos até hoje

DICA



Recursão - LAB

4) Voldemort : Faça um programa que calcule os N primeiros termos da sequencia de Voldemort.

1, 1, 1, 0, -2, -5, -8, -9, -5, 7, 28, ...

A sequencia começa com os 3 números 1, 1 e 1, e o próximo número é dado por duas vezes o ultimo menos o penúltimo menos o antepenúltimo.

Quarto número = $2 * 1 - 1 - 1 = 0$

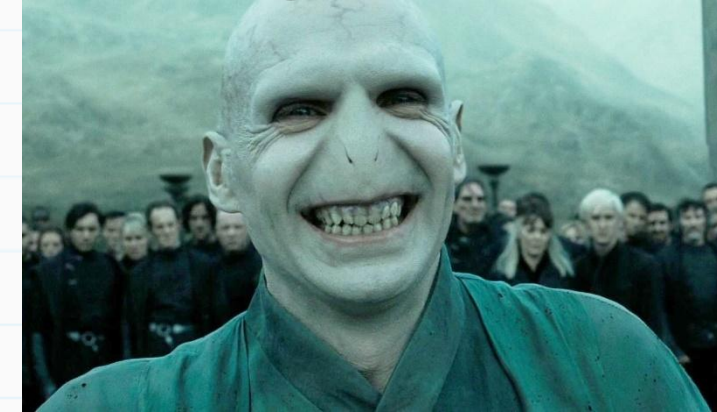
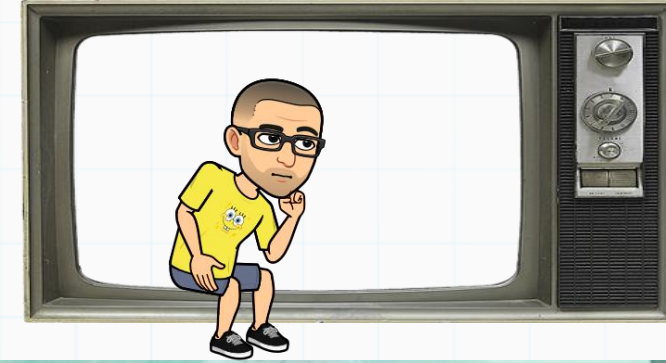
Exemplo:

```
n:10  
1 1 1 0 -2 -5 -8 -9 -5 7
```

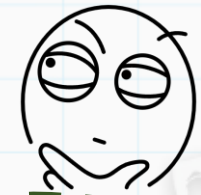
O programa deve calcular a séria através de uma função recursiva:

```
int Vold(int n)
```

Que retorna o n-ésimo termo da sequencia de Voldermort.



Use só o que aprendemos até hoje



Qual é o trigésimo termo da sequencia de Voldemort ?

Recursão - LAB



5) Vamos relembrar a Ordenação da Bolha: seja um vetor de tamanho n a ser ordenado de forma crescente. Vamos iterativamente **varrer** as posições adjacentes (de 1 a $n-1$) e toda vez que o par de elementos não estiverem ordenados vamos trocá-los

4	3	5	1
---	---	---	---

4	3	5	1
---	---	---	---

i $i+1$

3	4	5	1
---	---	---	---

i $i+1$

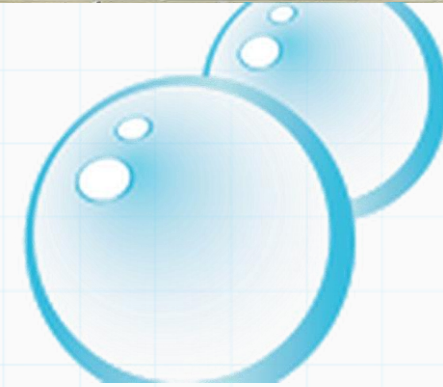
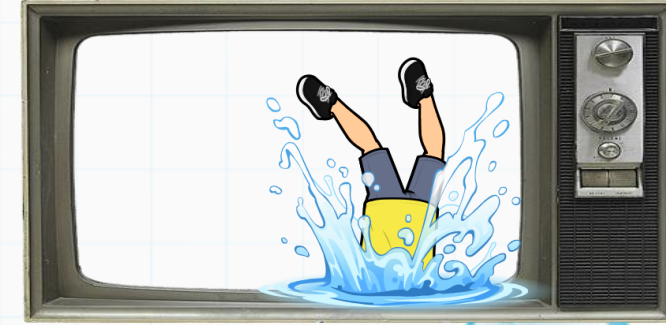
3	4	5	1
---	---	---	---

i $i+1$

3	4	1	5
---	---	---	---

Varredura 1

Recursão - LAB



5) Vamos relembrar a Ordenação da Bolha: seja um vetor de tamanho n a ser ordenado de forma crescente. Vamos iterativamente **varrer** as posições adjacentes (de 1 a $n-1$) e toda vez que o par de elementos não estiverem ordenados vamos trocá-los

4	3	5	1
---	---	---	---

3	4	1	5
---	---	---	---

4	3	5	1
---	---	---	---

3	4	1	5
---	---	---	---

i $i+1$

i $i+1$

3	4	5	1
---	---	---	---

3	4	1	5
---	---	---	---

i $i+1$

i $i+1$

3	4	5	1
---	---	---	---

3	1	4	5
---	---	---	---

i $i+1$

i $i+1$

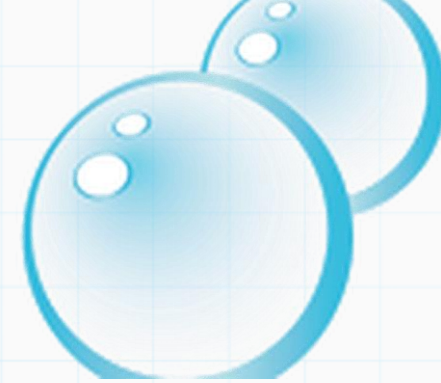
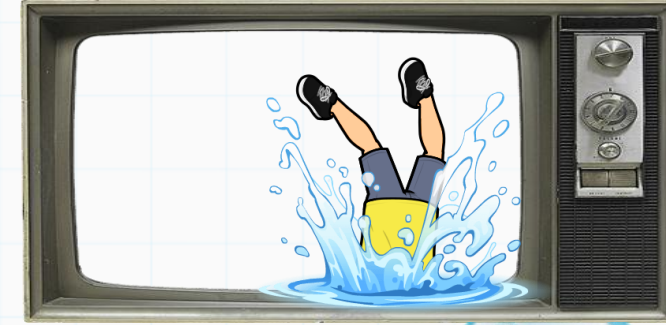
3	4	1	5
---	---	---	---

3	1	4	5
---	---	---	---

Varredura 1

Varredura 2

Recursão - LAB



5) Vamos relembrar a Ordenação da Bolha: seja um vetor de tamanho n a ser ordenado de forma crescente. Vamos iterativamente **varrer** as posições adjacentes (de 1 a $n-1$) e toda vez que o par de elementos não estiverem ordenados vamos trocá-los

4	3	5	1
---	---	---	---

3	4	1	5
---	---	---	---

3	1	4	5
---	---	---	---

4	3	5	1
---	---	---	---

3	4	1	5
---	---	---	---

3	1	4	5
---	---	---	---

i $i+1$

i $i+1$

i $i+1$

3	4	5	1
---	---	---	---

3	4	1	5
---	---	---	---

1	3	4	5
---	---	---	---

i $i+1$

i $i+1$

i $i+1$

3	4	5	1
---	---	---	---

3	1	4	5
---	---	---	---

1	3	4	5
---	---	---	---

i $i+1$

i $i+1$

i $i+1$

3	4	1	5
---	---	---	---

3	1	4	5
---	---	---	---

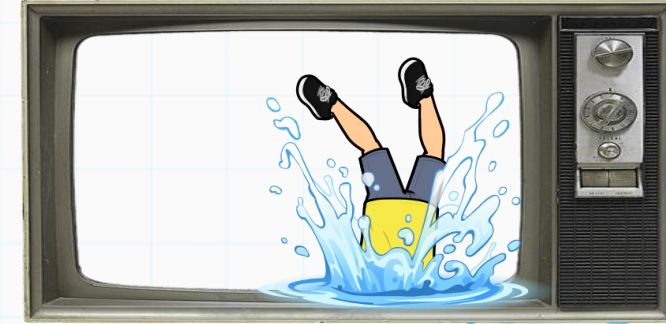
1	3	4	5
---	---	---	---

Varredura 1

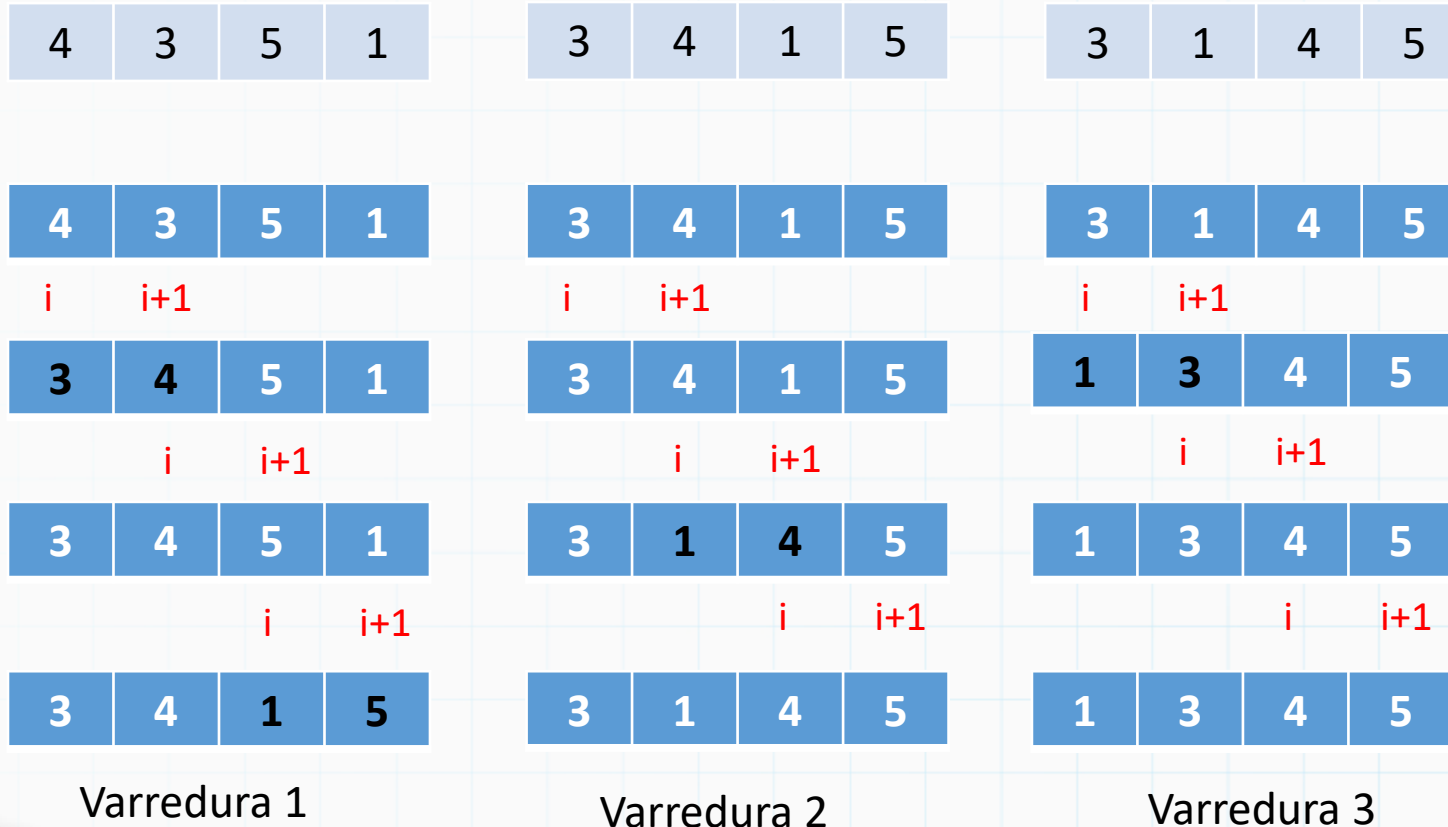
Varredura 2

Varredura 3

Recursão - LAB



5) Vamos relembrar a Ordenação da Bolha: seja um vetor de tamanho n a ser ordenado de forma crescente. Vamos iterativamente **varrer** as posições adjacentes (de 1 a $n-1$) e toda vez que o par de elementos não estiverem ordenados vamos trocá-los



Algoritmo

I percorre o vetor ($0 \dots n-1$)

J percorre o vetor vetor ($0 \dots n-2$)

Varredura: troca as posições J e $J+1$ que não estiverem de acordo com a ordenação

Teorema: É preciso no máximo n (tamanho do vetor) varreduras para ordenar um vetor

O nome do método vem devido aos elementos tenderem (flutuarem) a se mover para a posição correta como bolhas indo em direção da superfície.

Recursão - LAB

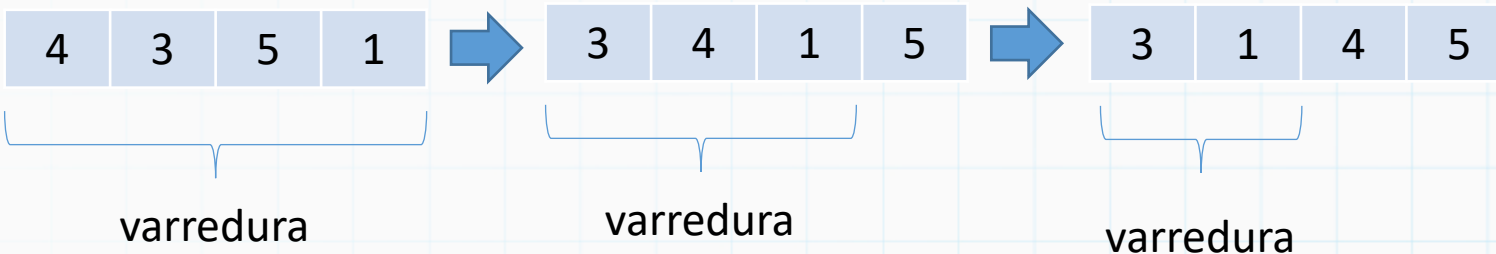


5) Bolhas: Faça um programa que receba um vetor de inteiros de tamanho n ordene de forma crescente usando a função recursiva

```
void bolha_rec(int n, int vetor[n])
```

Para fazer a função recursiva repare que ao fim da primeira varredura, o maior elemento do vetor é deslocado para a última posição, no final da segunda varredura, o segundo maior elemento é deslocado para a penúltima posição, e assim por diante.

Então a cada chamada recursiva você pode fazer a varredura até posição n e chamar recursivamente para tamanho $n-1$.



- O laço vermelho será substituído pela recursão

- O laço amarelo permanece

Algoritmo

I percorre o vetor (0... $n-1$)

J percorre o vetor vetor (0... $n-2$)

Varredura: troca as posições J e J+1 que não estiverem de acordo com a ordenação

Recursão - LAB



5) Bolhas: Exemplo

```
n:6
```

```
v[0]= 1
```

```
v[1]= 0
```

```
v[2]= 2
```

```
v[3]= 9
```

```
v[4]= 3
```

```
v[5]= 8
```

```
vetor
```

```
v = { 1, 0, 2, 9, 3, 8 }
```

```
bolha recursiva
```

```
v = { 0, 1, 2, 3, 8, 9 }
```



Recursão - LAB



6) Multiplicação recursiva: Dado 3 inteiros $n, m, p > 0$ e duas matrizes de inteiros A $n \times m$ e B $m \times p$, faça um programa que calcule e imprima a multiplicação $C = A \times B$.

O programa deve saber calcular de forma recursiva o produto escalar de uma linha de A por uma coluna de B através da função:

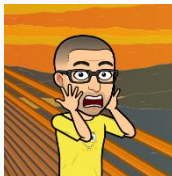
```
int produto_escalar_rec(int n, int m, int p, int A[n][m], int B[m][p], int linha, int coluna, int posicao)
```

Além disso, uma outra função também recursiva

```
void multiplica_rec(int n, int m, int p, int A[n][m], int B[m][p], int C[n][p], int i, int j)
```

irá calcular a multiplicação, chamando a função `produto_escalar_rec` e recursivamente iterando os índices i e j .

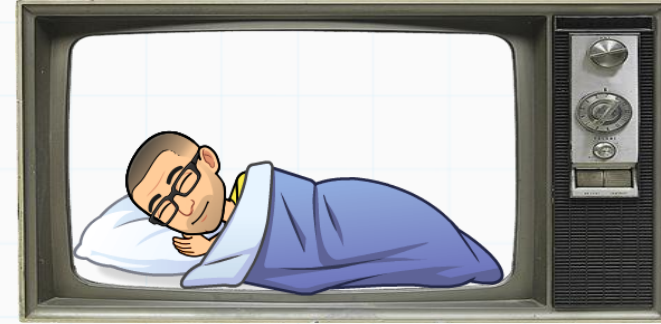
***OBS* Não utiliza nenhum tipo de laço para calcular a multiplicação, apenas funções recursivas**



Veja o exemplo a seguir:

$$\begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{bmatrix} \begin{bmatrix} b_1 & b_2 & b_3 \\ b_4 & b_5 & b_6 \\ b_7 & b_8 & b_9 \end{bmatrix} = \begin{bmatrix} c_1 & c_2 & c_3 \\ c_4 & c_5 & c_6 \\ c_7 & c_8 & c_9 \end{bmatrix}$$

Recursão - LAB



Exemplo:

n: 2
m: 3
p: 3

Matriz A:

(0,0)=1
(0,1)=2
(0,2)=3
(1,0)=4
(1,1)=5
(1,2)=6

Matriz A:

1	2	3
4	5	6

Matriz B:

(0,0)=0
(0,1)=1
(0,2)=2
(1,0)=2
(1,1)=1
(1,2)=0
(2,0)=1
(2,1)=2
(2,2)=3

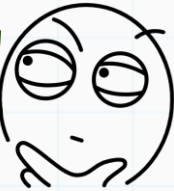
Matriz B:

0	1	2
2	1	0
1	2	3

Matriz C:

7	9	11
16	21	26

Mostre o código que você fez (sem nenhum laço)



$$\begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{bmatrix} \begin{bmatrix} b_1 & b_2 & b_3 \\ b_4 & b_5 & b_6 \\ b_7 & b_8 & b_9 \end{bmatrix} = \begin{bmatrix} c_1 & c_2 & c_3 \\ c_4 & c_5 & c_6 \\ c_7 & c_8 & c_9 \end{bmatrix}$$

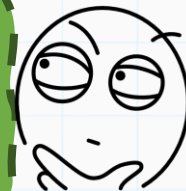
Recursão - LAB



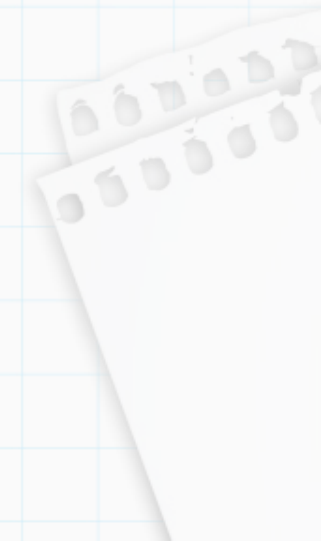
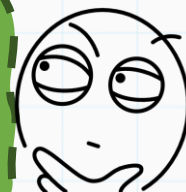
Quantos elementos tem a sequência de Granizo começando de $n = 27$?



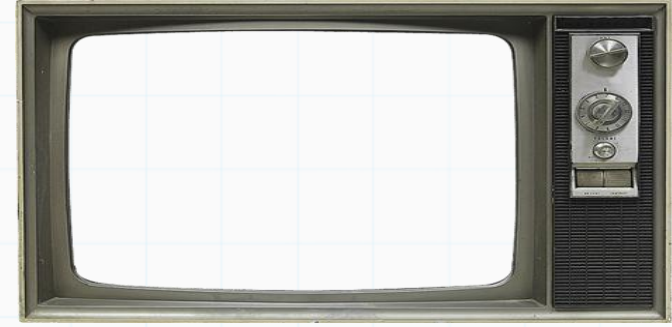
Qual é o trigésimo termo da sequência de Voldemort?



Mostre o código que você fez (sem nenhum laço)



Até a próxima



Slides baseados no curso de Aline Nascimento