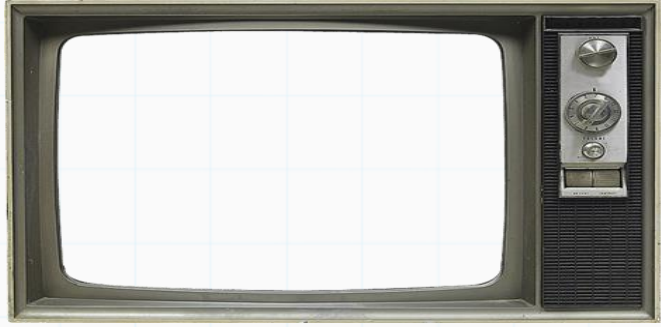
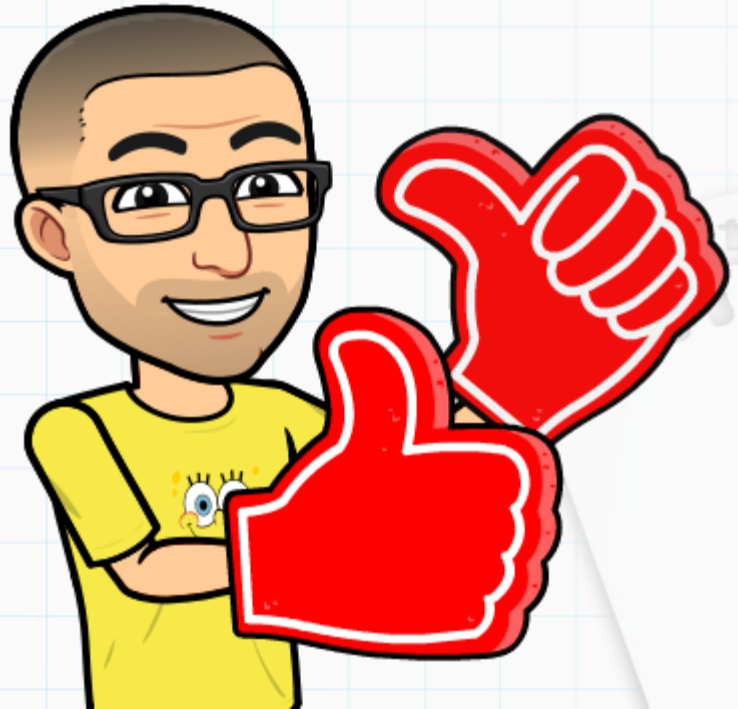
Programação Estruturada



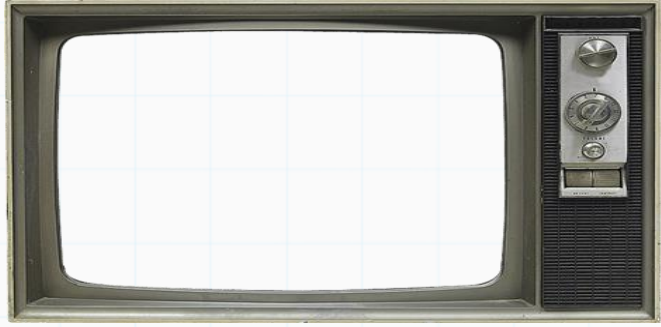
Professor : Yuri Frota

yuri@ic.uff.br

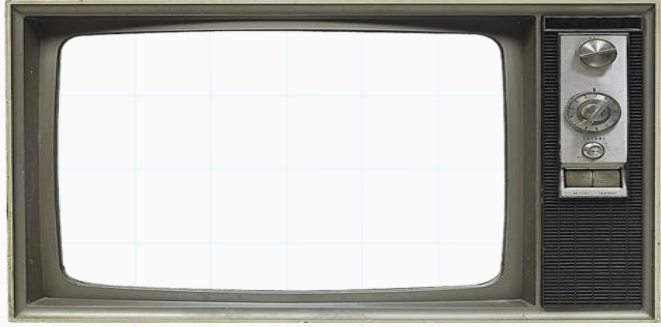


Vetores

- Variável composta **unidimensional,** que armazena dados, em forma de sequência:



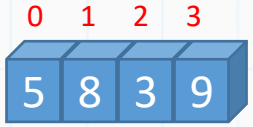
Vetores



- Variável composta **unidimensional,** que armazena dados, em forma de sequência:

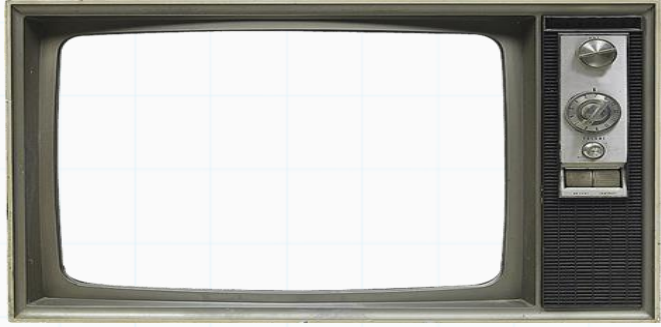
Possuem tamanho fixo 

Os elementos estão armazenados em posições contíguas da memória



Cada elemento do vetor pode ser acessado individualmente, especificando a sua posição.

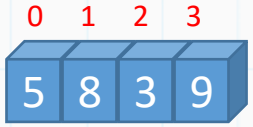
Vetores



- Variável composta **unidimensional,** que armazena dados, em forma de sequência:

  Possuem tamanho fixo 

  Os elementos estão armazenados em posições contíguas da memória



  Cada elemento do vetor pode ser acessado individualmente, especificando a sua posição.

- Formato:

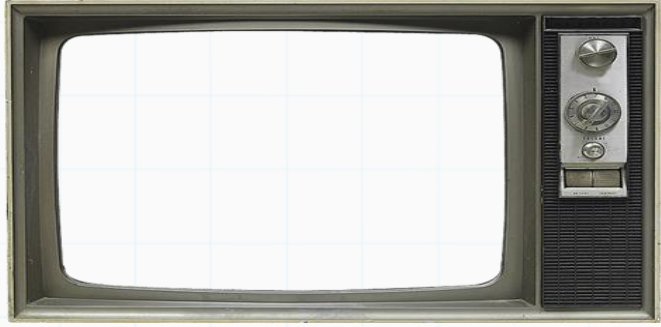
  tipo nome_vetor[tamanho];

- Exemplo:

  int vetNum[10];

  - Vetor de 10 posições que armazena um números inteiros
  - O primeiro elemento do vetor fica na posição 0
  - O último elemento do vetor na posição 9 (tamanho-1)

Vetores



Alocação: Estática (tamanho do vetor não muda)

Fixa:

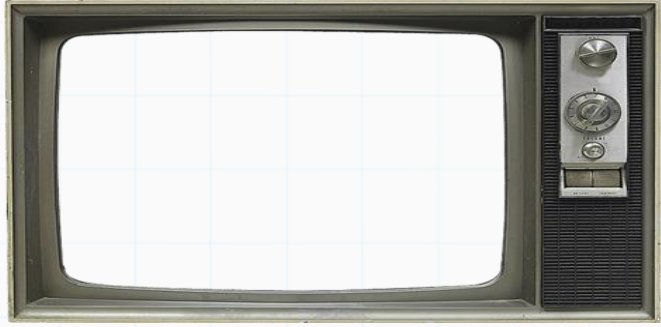
```
int vetNum[10];
```

Fixa Informada:

```
int tam;
printf("tamanho:");
scanf("%d", &tam);

int vetNum[tam];
```

Depois do tamanho estabelecido, não pode ser alterado

Vetores



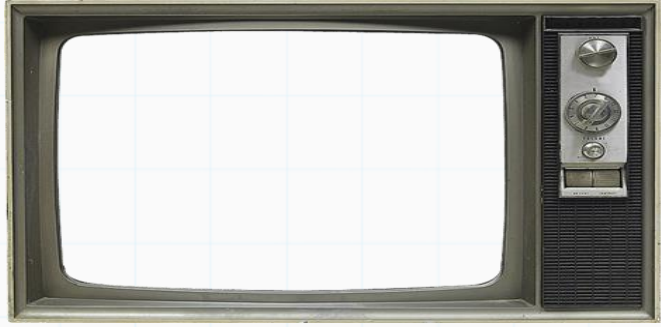
Inicialização:

Padrão:

```
int vetNum[3];
```

Indeterminado, pode ser lixo

Vetores



Inicialização:

Padrão:

```
int vetNum[3];
```

Indeterminado, pode ser lixo

Direta:

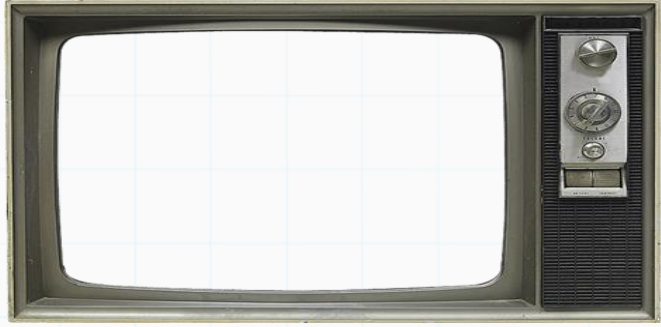
```
int vetNum[3] = {5, 6, 8};
```

Direta incompleta:

```
int vetNum[100] = {5, 6, 8};
```

Apenas os 3 primeiros elementos, o resto indeterminado

Vetores



Inicialização:

Padrão:

```
int vetNum[3];
```

Indeterminado, pode ser lixo

Direta:

```
int vetNum[3] = {5, 6, 8};
```

Direta incompleta:

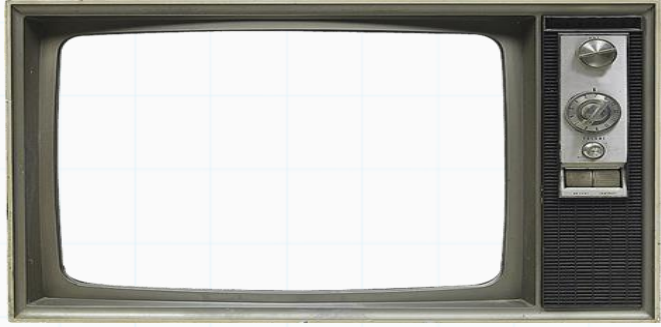
```
int vetNum[100] = {5, 6, 8};
```

Apenas os 3 primeiros elementos, o resto indeterminado

Atribuição:

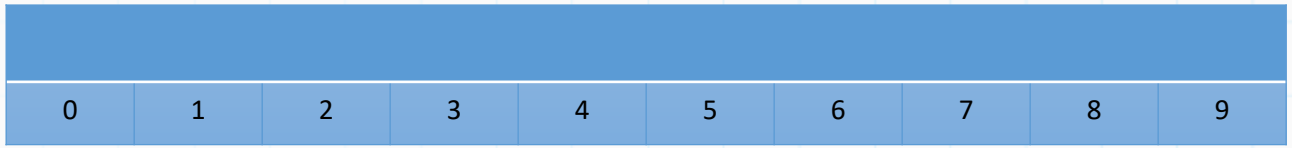
```
int vetNum[10];

for (i=0; i<10; i++)
    vetNum[i] = i;
```

Vetores

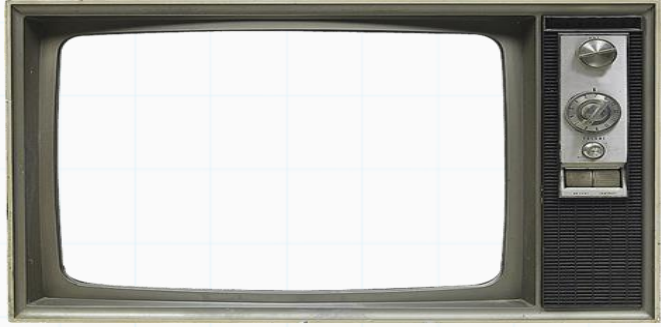


E como fica na memória

```
int vetNum[10];
    vetNum[0] = 11;
    vetNum[5] = -2
    scanf("%d", &vetNum[8]) ⟶ 18
```



Vetores

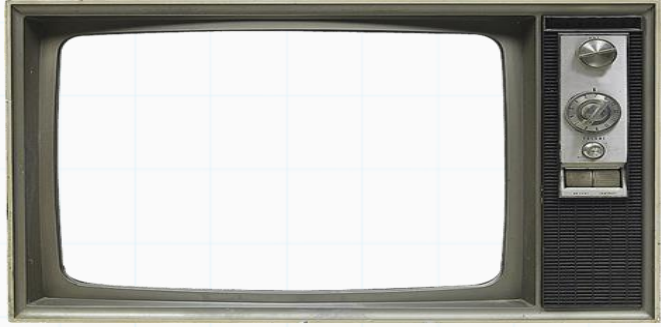


E como fica na memória

```
int vetNum[10];
    vetNum[0] = 11;
    vetNum[5] = -2
    scanf("%d", &vetNum[8]) ⟶ 18
```

11					-2			18	
0	1	2	3	4	5	6	7	8	9

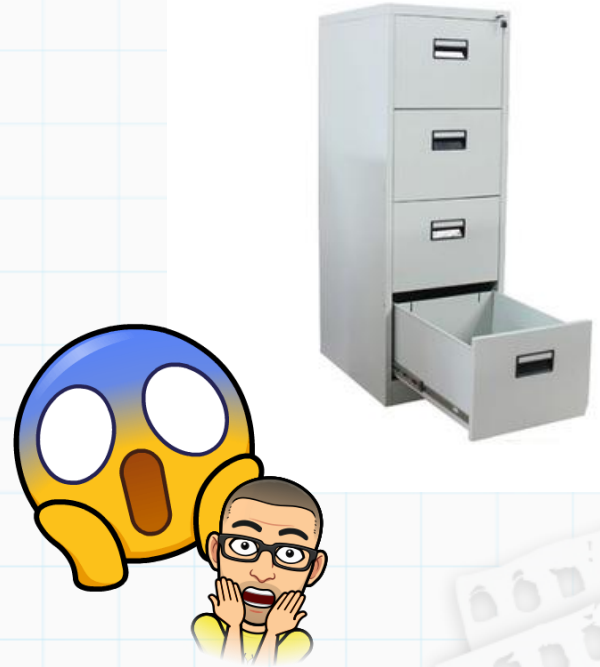
Vetores



E como fica na memória

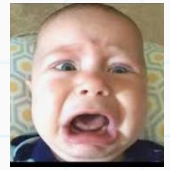
```
int vetNum[10];
    vetNum[0] = 11;
    vetNum[5] = -2
    scanf("%d", &vetNum[8]) ⟶ 18
    vetNum[10] = 20;
```

11					-2			18	
0	1	2	3	4	5	6	7	8	9



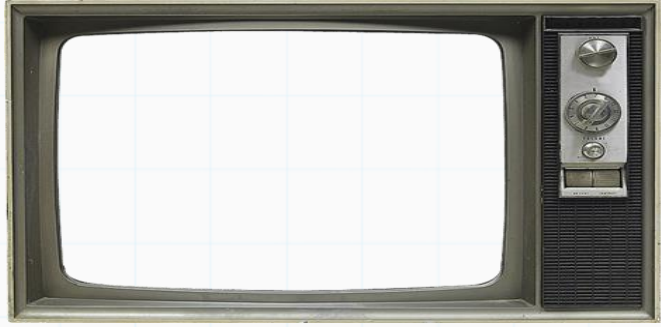
Não da erro, mas você está escrevendo em uma posição de memória não alocada -> TUDO PODE ACONTECER !!!

- Pode dar certo
- Pode dar "crash"
- Pode rodar e armazenar valor diferente (lixo) EM QUALQUER POSIÇÃO !



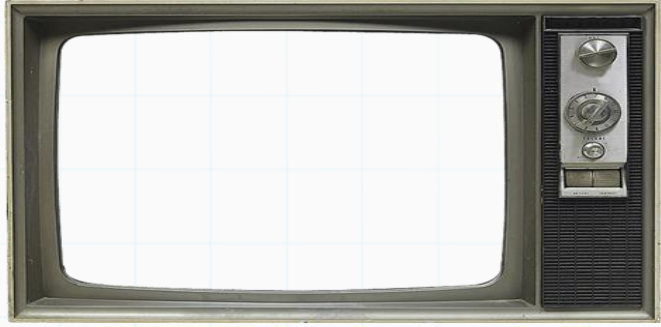
Vetores

Exemplo: Armazenar 10 valores inteiros em um vetor e depois calcular a sua média



```c
#include <stdio.h>
#include <stdlib.h>

int main (void) {
        int i, soma=0, vetor[10];
        float media = 0;

        // lendo e armazenando os valores
        for (i=0; i<10; i++) {
            printf("Digite um número inteiro: ");
            scanf("%d", &vetor[i]);
        }

        // calculando a média dos números do vetor
        for (i=0; i<10; i++)
            soma += vetor[i];
        media = soma/10.0;
        printf("A média é: %.2f", media);
        return 0;
}
```

Matrizes

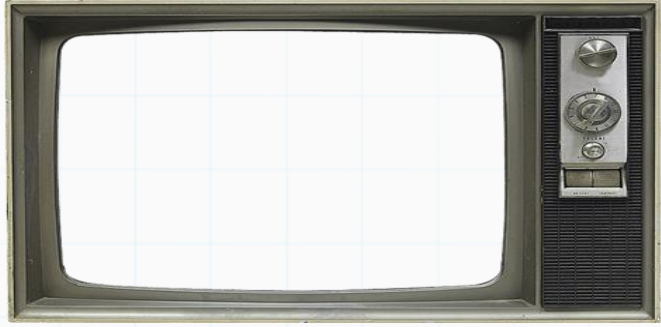


Vetores de 2 ou mais dimensões

- Formato:

tipo Mat[dim1][dim2]...[dimN];

Matrizes



Vetores de 2 ou mais dimensões

- Formato:

  tipo Mat[dim1][dim2]...[dimN];

- Exemplo:

  int matNum[3][4];

  matNum[0][2] = 5;
  scanf("%d", &matNum[2][3]); → 9

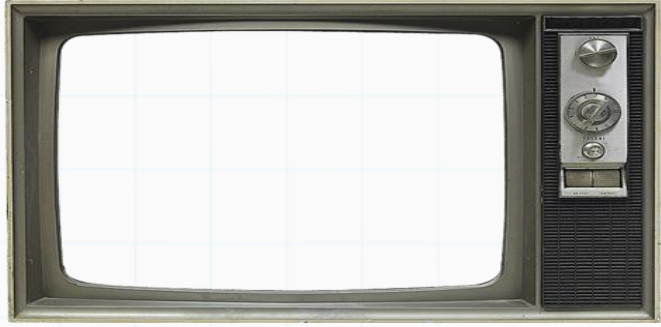
  matNum[0,2] = 5; → errado!

	0	1	2	3
0			5	
1				
2				9

3d

int mat3D[5][10][20];

Matrizes



Alocação: Estática

Fixa:

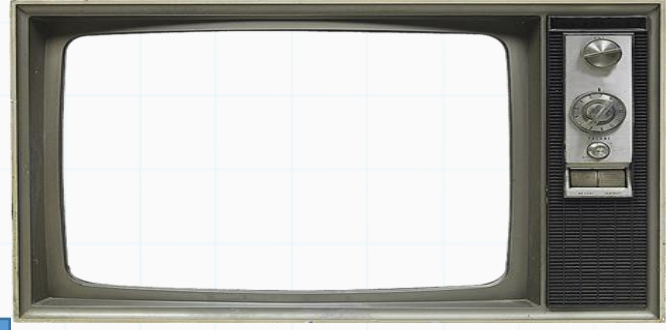
```
int Mat[10][5];
```

Fixa Informada:

```
int tam1, tam2;

printf("tamanho:");
scanf("%d", &tam1);
printf("tamanho2:");
scanf("%d", &tam2);

int Mat[tam1][tam2];
```

Depois do tamanho estabelecido, não pode ser alterado

Matrizes



Inicialização:

Padrão:

```
int Mat[3][4];
```

Indeterminado, pode ter lixo !

Direta:

```
int Mat[2][3] = {{5,10,15}, {20,25,30}};
```

Direta incompleta:

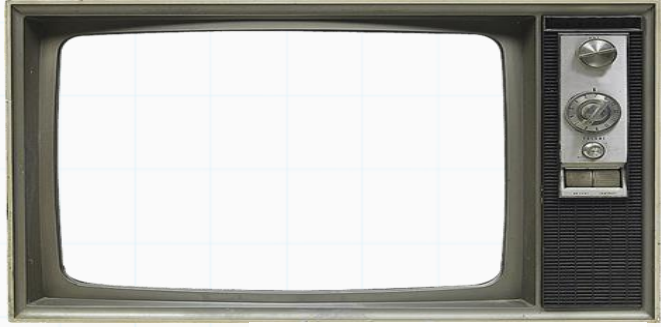
```
int Mat[2][3] = {{5,10}, {20,25}};
```

Apenas os 2 primeiros elementos de cada linha, o resto será indeterminado

Atribuição:

```
int Mat[10][10];

for (i=0; i<10; i++)
    for (j=0; j<10; j++)
        Mat[i][j] = i+j;
```

Vetores de Caracteres

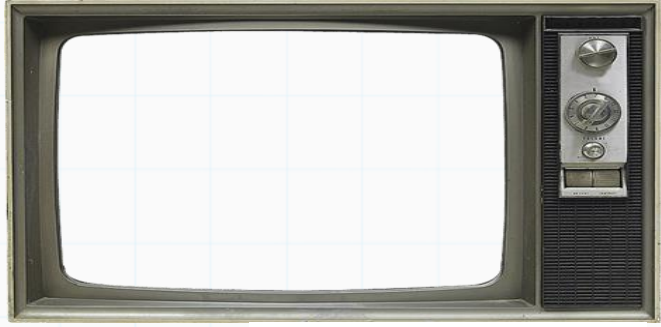


- São usados para armazenar uma cadeia de caracteres (strings)

Exemplo:

```
char palavra[40];
char endereco[50];
```

Vetores de Caracteres



- São usados para armazenar uma cadeia de caracteres (strings)

  Exemplo:

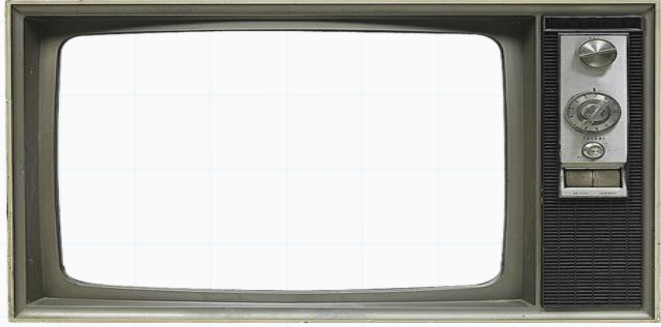
  ```
  char palavra[40];
  char endereco[50];
  ```

- Internamente as cadeias de caracteres terminam com '\0', para que os programas possam encontrar o fim de uma cadeia

"This is a string."

T	h	i	s		i	s		a		s	t	r	i	n	g	.	\0

Vetores de Caracteres



Inicialização:

Padrão:

```
char nome[30];
```

Indeterminado, pode ser lixo

Direta:

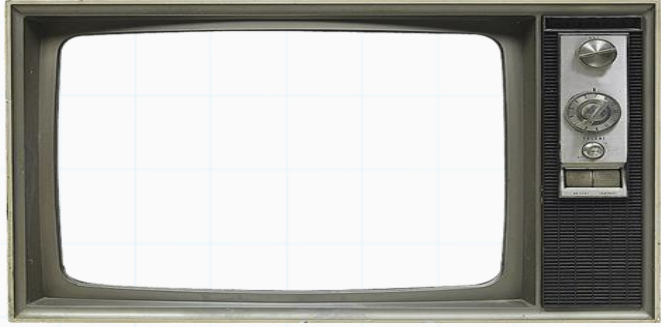
```
char nome[] = {'O','l','a'};
```

```
char nome[] = {"Ola"};
```

- Na ausência do tamanho do vetor, o compilador aloca o número de elementos do vetor.
- Automaticamente coloca o caractere '\0' no fim do vetor

'O'	'l'	'a'	'\0'

Vetores de Caracteres



Inicialização:

Padrão:

```
char nome[30];
```

Indeterminado, pode ser lixo

Direta:

```
char nome[] = {'O','l','a'};
```

```
char nome[] = {"Ola"};
```

- Na ausência do tamanho do vetor, o compilador aloca o número de elementos do vetor.
- Automaticamente coloca o caractere '\0' no fim do vetor

'O'	'l'	'a'	'\0'

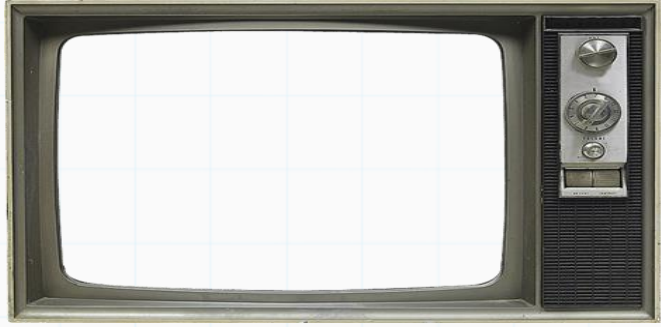
ou:

```
char nome[100] = {'O','l','a'};
```

```
char nome[100] = {"Ola"};
```

'O'	'l'	'a'	'\0'	...

Vetores de Caracteres

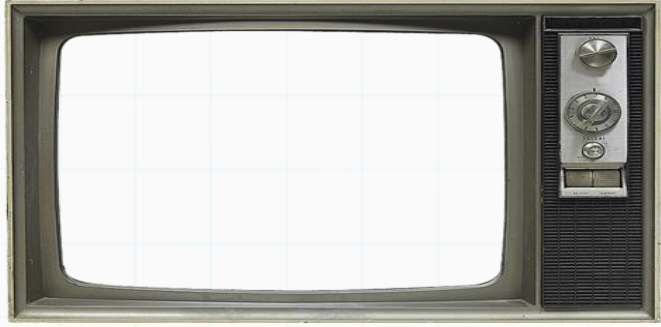


Atribuição:

```
char nome[30];

Nome = "Seya"
```

```
Nome[0] = 'S'
Nome[1] = 'e'
Nome[2] = 'y'
Nome[3] = 'a'
Nome[4] = '\0'
```

Vetores de Caracteres



Atribuição:

```
char nome[30];

Nome = "Seya"
```

✖

```
Nome[0] = 'S'
Nome[1] = 'e'
Nome[2] = 'y'
Nome[3] = 'a'
Nome[4] = '\0'
```

o

Ou usar Pode ser usada a função `strcpy` que copia uma cadeia de caracteres para outra, incluindo o `'\0'`.

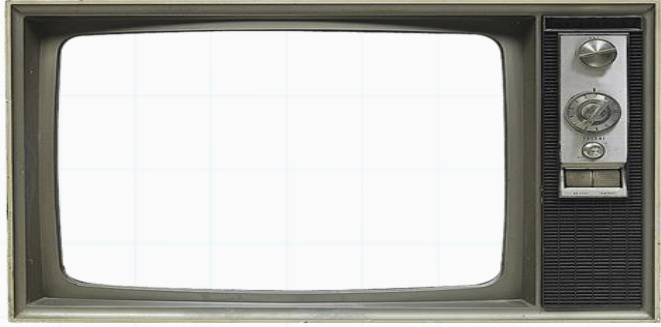
Formato:
strcpy(nome1,nome2);

Copia nome2 para nome1

Função da biblioteca
#include <string.h>

```
strcpy(nome, "Seya");
```

Vetores de Caracteres



Entrada:

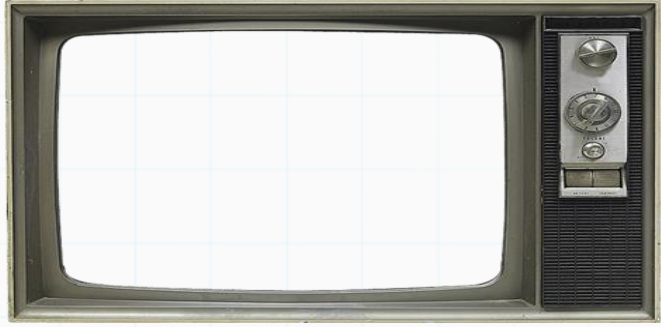
```
char nome[30];
printf("Nome:");
scanf("%s",nome)
```

strings não precisam passar "&" no scanf

Saida:

```
char nome[30] = "Kakashi";
printf("%s",nome);
```

Vetores de Caracteres



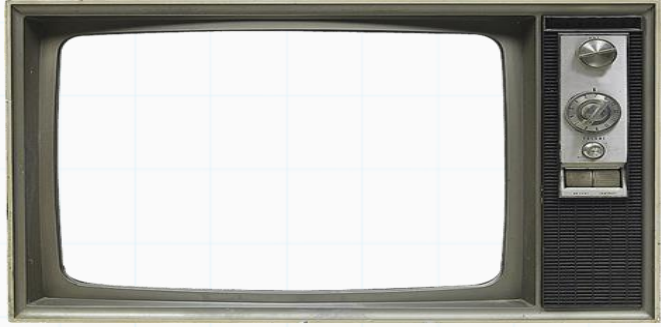
Outras funções úteis da <string.h>:

Tamanho:

```
char nome[30] = "Luffy";
tam=strlen(nome);
printf("%d",tam);
```

5

Vetores de Caracteres



Outras funções úteis da <string.h>:

Tamanho:

```
char nome[30] = “Luffy”;
tam=strlen(nome);
printf(“%d”,tam);
```

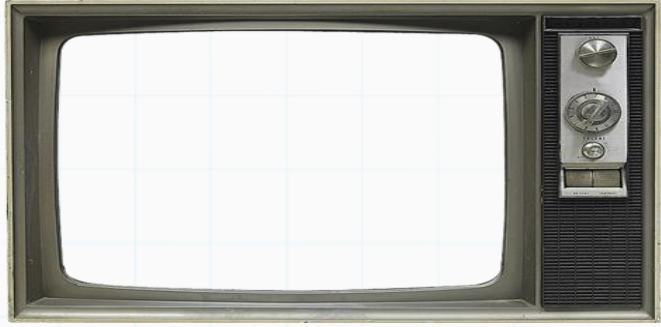
5

Cópia dos primeiros caracteres:

```
char nome[30] = "Luffy eh o cara";
char nome2[30];
strncpy(nome2, nome, 8);
printf("%s",nome2);
```

Luffy eh

Vetores de Caracteres



Outras funções úteis da <string.h>:

Tamanho:

```
char nome[30] = “Luffy”;
tam=strlen(nome);
printf(“%d”,tam);
```

5

Cópia dos primeiros caracteres:

```
char nome[30] = "Luffy eh o cara";
char nome2[30];
strncpy(nome2, nome, 8);
printf("%s",nome2);
```

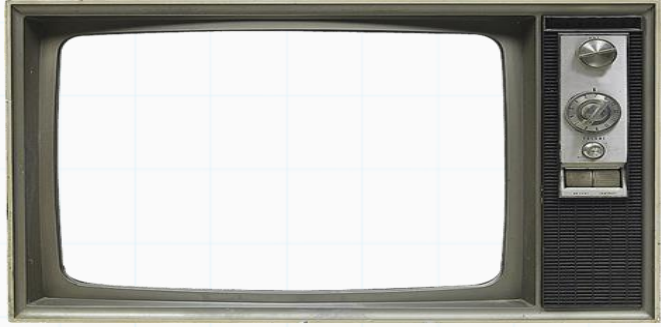
Luffy eh

Concatenação retorna no 1 arg:

```
char nome[30] = "Luffy eh";
char nome2[30] = "o cara";
strcat(nome, nome2);
printf("%s",nome);
```

Luffy eho cara

Vetores de Caracteres



Outras funções úteis da <string.h>:

Comparação:

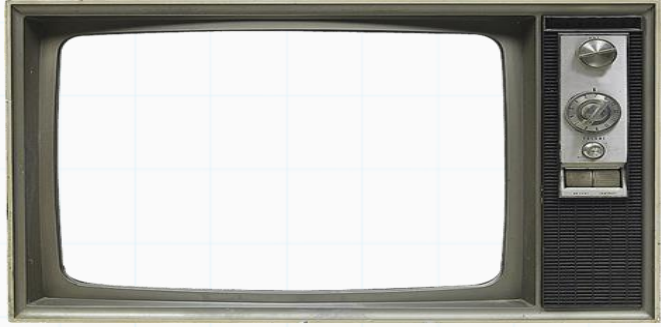
```
char nome[30] = "Luffy";
char nome2[30] = "Luffi";

if (strcmp(nome,nome2)==0)
   printf("iguais");
else
   printf("diferentes");
```

diferentes

strcmp(n1,n2)

Retorna < 0 → se n1 < n2
Retorna = 0 → se n1 = n2
Retorna > 0 → se n2 > n2

Vetores de Caracteres



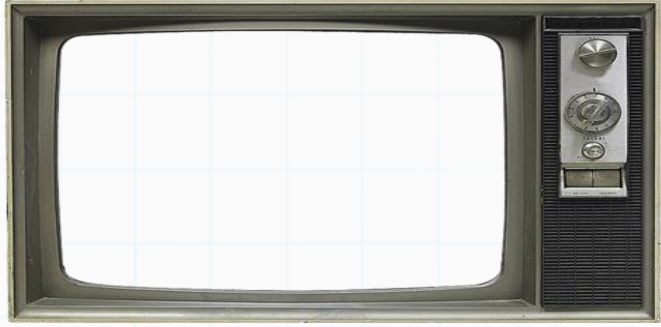
Outras funções úteis da <string.h>:

Conversão:

```
char nome[30]  = "1234";
char nome2[30] = "56.78";

int   n1 = atoi(nome);
float n2 = atof(nome2);

printf("%d %f",n1,n2);
```

1234 56.8

Exercícios

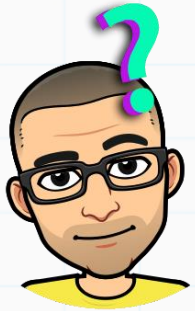


1) Vetores: Receba dois vetores de inteiros distintos A e B dados pelo usuário, de tamanhos n>0 e m>0. Imprima que números aparecem nos dois vetores ao mesmo tempo, e a quantidade destes números.

n=m=5

A	23	47	12	8	7
B	8	101	23	76	82

R:
23, 8,
2 números

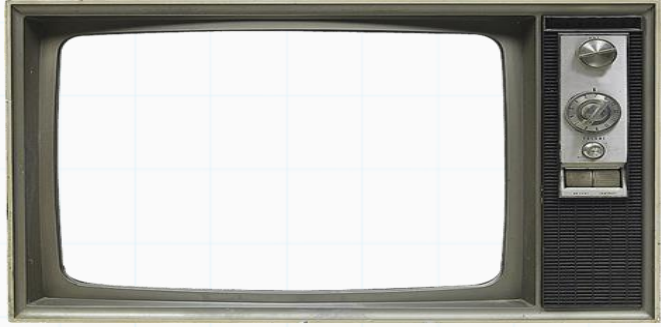


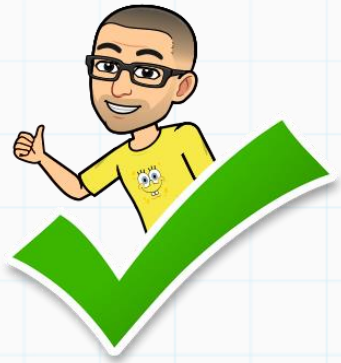
```
int vetNum[10];
```

```
int tam;
printf("tamanho:");
scanf("%d", &tam);
```

```
int vetNum[tam];
```

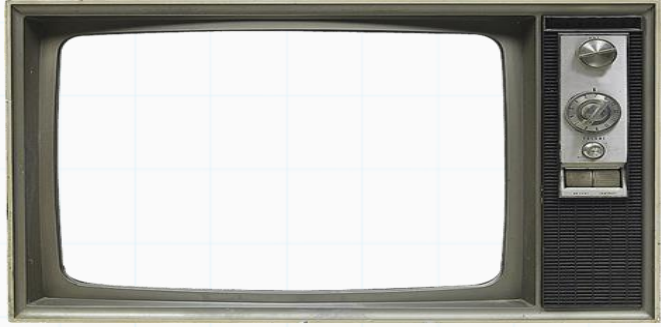
Exercícios





```c
int main (void) {

        int n,m,i,j;
        printf("n:");
        scanf("%d", &n);
        printf("m:");
        scanf("%d", &m);

        int A[n];
        int B[m];

        for (i=0; i<n; i++)
           scanf("%d", &A[i]);
        for (i=0; i<m; i++)
           scanf("%d", &B[i]);


        int cont=0;
        for (i=0; i<n; i++)
            for (j=0; j<m; j++)
                if (A[i]==B[j])
                {
                    printf("%d, ", A[i]);
                    cont++;
                }

        printf("\n%d numeros\n", cont);
        return 0;
}
```

Exercícios

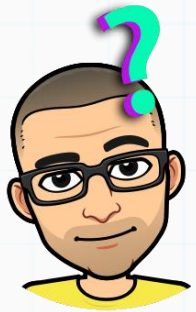


2) Primos: Dado um inteiro n>0, crie e imprima um vetor que contenha todos os números primos até n.

Exemplo

```
n:30
2 3 5 7 11 13 17 19 23 29
```

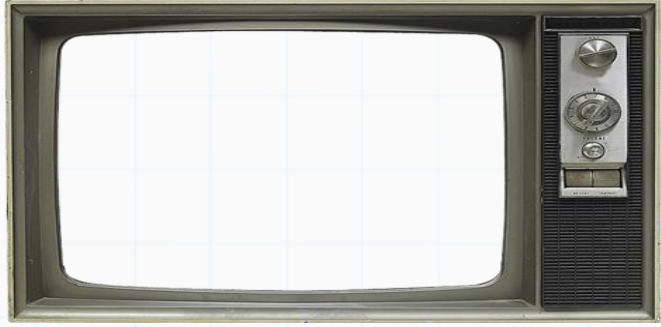
```
n:100
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97
```



```
int vetNum[10];
```

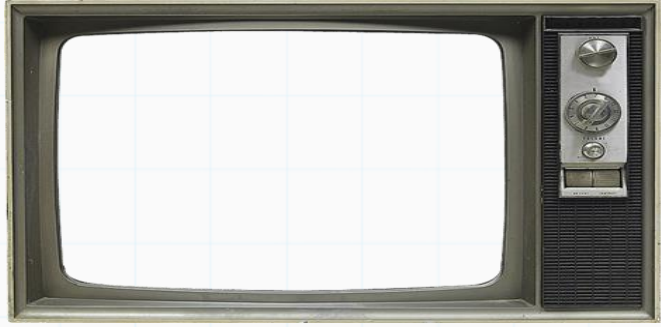
```
int tam;
printf("tamanho:");
scanf("%d", &tam);

int vetNum[tam];
```

Exercícios



```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int n;
    int p = 0;
    int i = 0;
    int primos[50];
    int qtd_primos = 0;
    int eh_primo = 0;

    printf("n:");
    scanf("%d",&n);

    for(p = 2; p <= n; p++)
    {
        eh_primo = 1;
        for (i = 2; i < p; ++i)
            if (p % i == 0)
            {
                eh_primo = 0;
                break;
            }

        if (eh_primo == 1)
        {
            primos[qtd_primos] = p;
            ++qtd_primos;
        }
    }
    for ( i = 0;  i < qtd_primos;  ++i )
        printf ("%i  ", primos[i]);

    printf("\n");
    return 0;
}
```

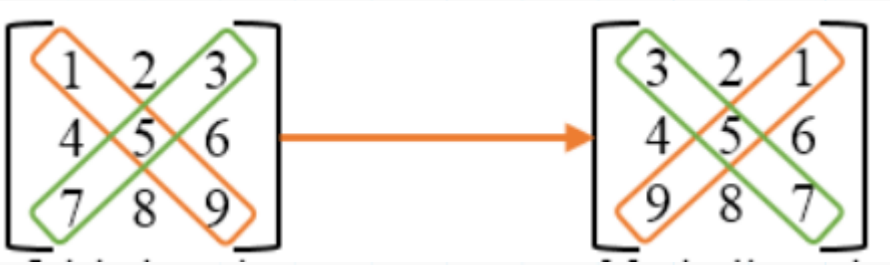
Exercícios



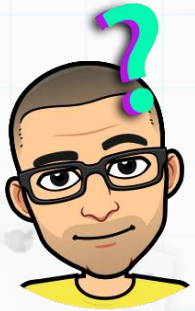
3) Diagonais: Dado um inteiro n>0, receba uma matriz quadrada M de inteiros de cardinalidade n x n, e troca os elementos da diagonal principal e secundária.

Exemplo

N=3



Lembrando:
D.P. j=i
D.S. j=n-i-1

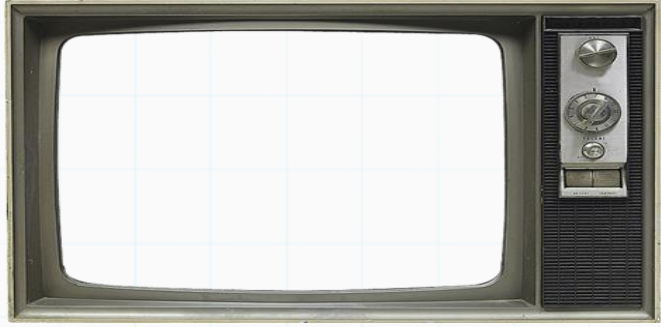


```
int Mat[10][5];
```

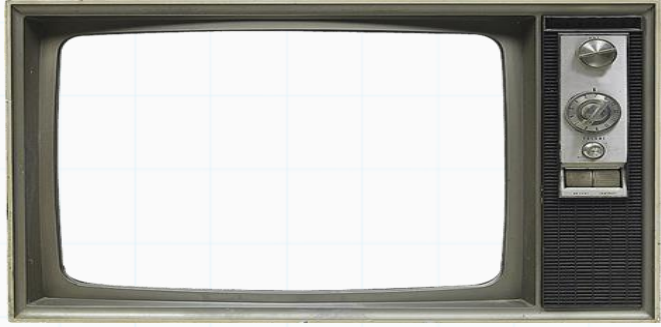
```
int tam1, tam2;

printf("tamanho:");
scanf("%d", &tam1);
printf("tamanho2:");
scanf("%d", &tam2);

int Mat[tam1][tam2];
```

```c
#include <stdio.h>

int main()
{
    int n;

    printf("n:");
    scanf("%d",&n);

    int A[n][n];
    int linha, col, temp;

    for(linha=0; linha<n; linha++)
        for(col=0; col<n; col++)
            scanf("%d", &A[linha][col]);

    for(linha=0; linha<n; linha++)
    {
        temp = A[linha][linha];
        A[linha][linha] = A[linha][(n-linha) - 1];
        A[linha][(n-linha) - 1] = temp;
    }

    for(linha=0; linha<n; linha++)
    {
        for(col=0; col<n; col++)
            printf("%d ", A[linha][col]);
        printf("\n");
    }

    return 0;
}
```

Exercícios



Exercícios



4) Intercalação: Faça um programa recebe dois vetores A e B de inteiros, de tamanhos n>0 e m>0, e gere e imprima um terceiro vetor C que intercala os elementos de ambos.

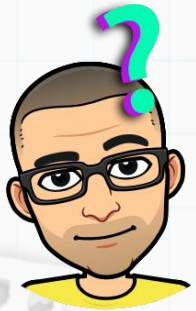
OBS: O usuário irá entrar com os todos os elementos do primeiro vetor A, e quando acabar, irá informar os elementos do segundo vetor B.

Exemplo:

A = {1,2,3,4}
B = {10,20,30,40,50,15}

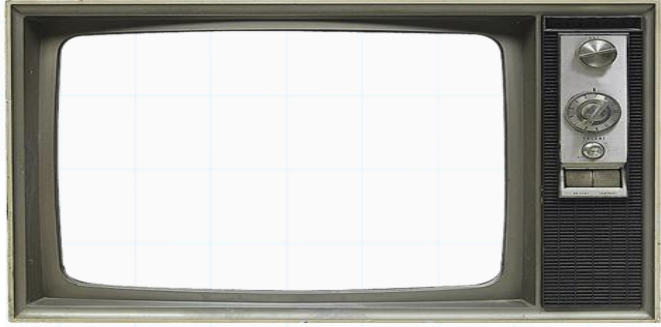
C = {1,10,2,20,3,30,4,40,50,15}

Veja que quando acabar os elementos de um vetor, pode colocar o que sobrou do outro



```
int vetNum[10];
```

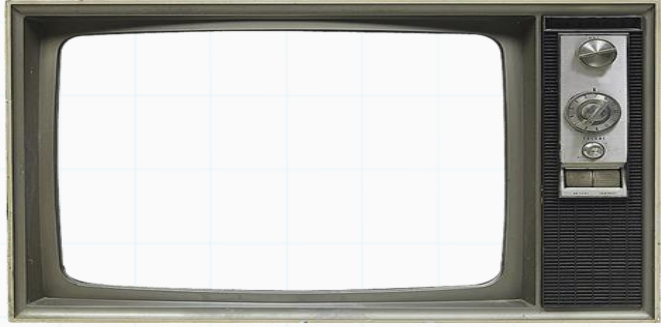
```
int tam;
printf("tamanho:");
scanf("%d", &tam);

int vetNum[tam];
```



Exercícios

```c
#include <stdio.h>

int main()
{
    int n,m;

    printf("n:");
    scanf("%d",&n);
    printf("m:");
    scanf("%d",&m);

    int A[n];
    int B[m];
    int C[n+m];
    int i,ind, menor;

    for(i=0; i<n; i++)
    {
        printf("A[%d]=",i);
        scanf("%d", &A[i]);
    }

    for(i=0; i<m; i++)
    {
        printf("B[%d]=",i);
        scanf("%d", &B[i]);
    }

    if (n<m) menor = n;
    else     menor = m;

    ind=0;
    for(i=0; i<menor; i++)
    {
        C[ind++] = A[i];
        C[ind++] = B[i];
    }

    if (menor==n)
        for(i=menor; i<n+m; i++)
            C[ind++] = B[i];
    else
        for(i=menor; i<n+m; i++)
            C[ind++] = A[i];

    for(i=0; i<n+m; i++)
      printf("%d, ",C[i]);

    return 0;
}
```

Até a próxima



Slides baseados no curso de Aline Nascimento