Algoritmos e Estruturas de Dados I

Prof. Lúcio Mauro Pereira

21/10/2024

Desafios postos

Lista 26: Questão 5

Implemente uma função que receba duas matrizes de reais. A função deverá copiar a primeira matriz na segunda.

Argumentos: os endereços para duas matrizes de reais

Valor gerado: nenhum

Lista 26: Questão 6

Implemente uma função que receba duas matrizes de reais. A função deverá calcular a matriz transposta da primeira, armazenando-a na segunda matriz.

* Planeje, com cuidado, as dimensões de ambas as matrizes parametrizadas
* Planeje, com cuidado, os parâmetros e os valores gerados

Lista 25: Questão 5

Construa uma função que calcule o número de vogais presentes em uma *string.*

a) Abordagem iterativa
b) Abordagem recursiva

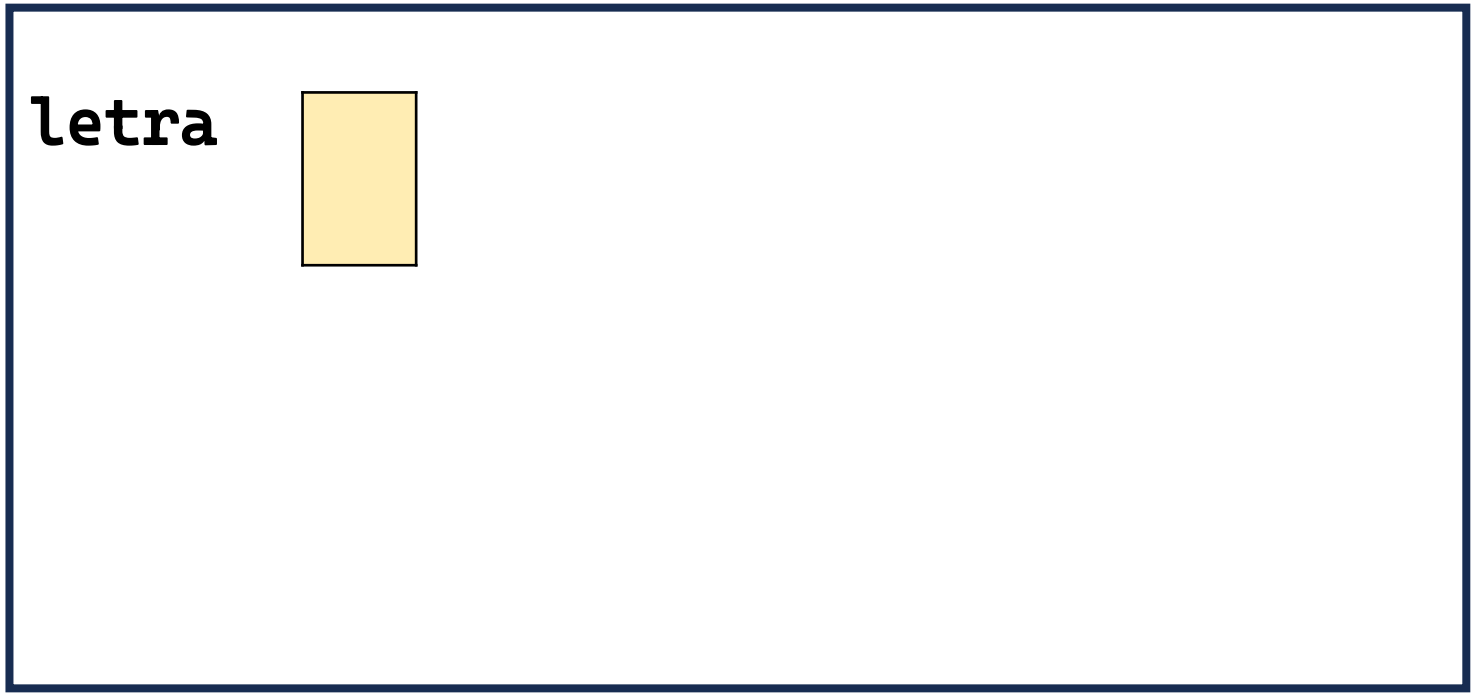
Tente:

Para uma matriz quadrada de reais, construa uma função que calcule a diferença entre o maior valor presente acima de sua diagonal principal e o maior valor presente abaixo da diagonal principal.

Em discussão:
String enquanto vetor de caracteres em C

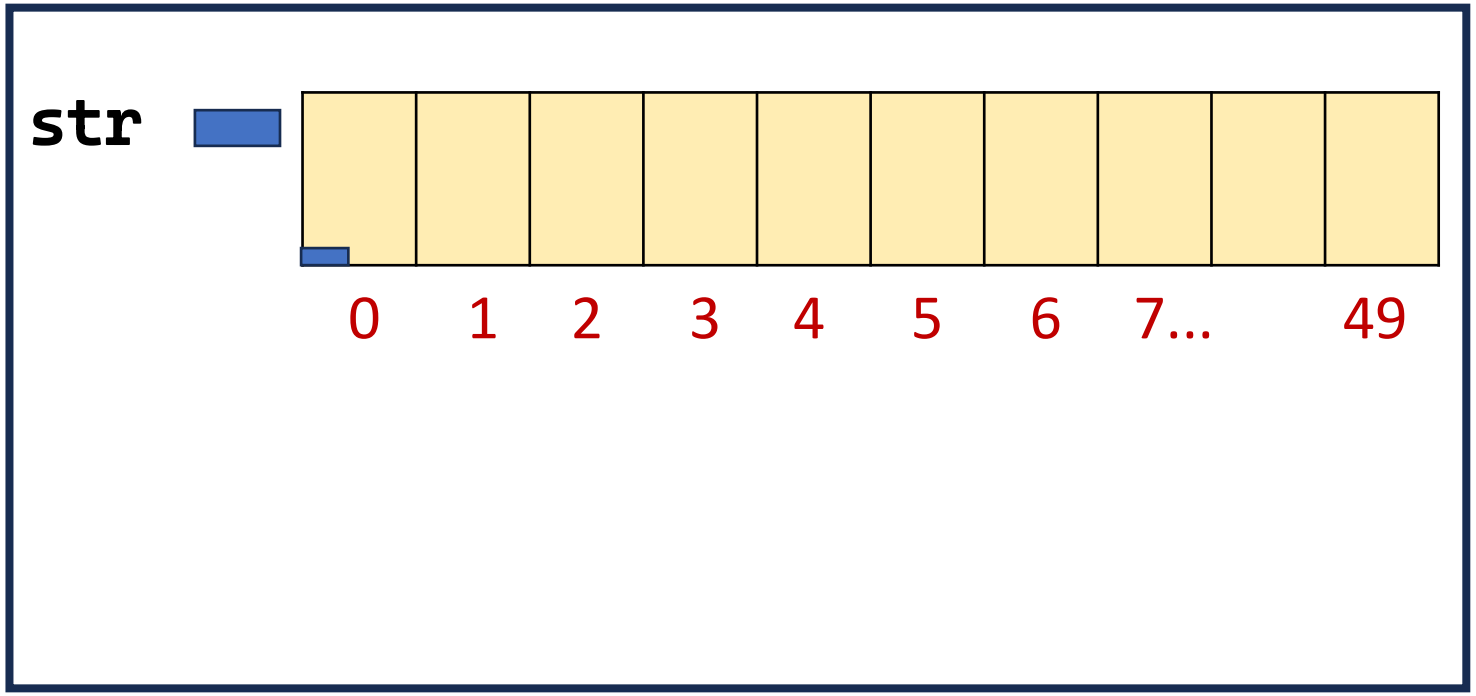
O que já sabemos sobre um caractere

```
char letra;
```



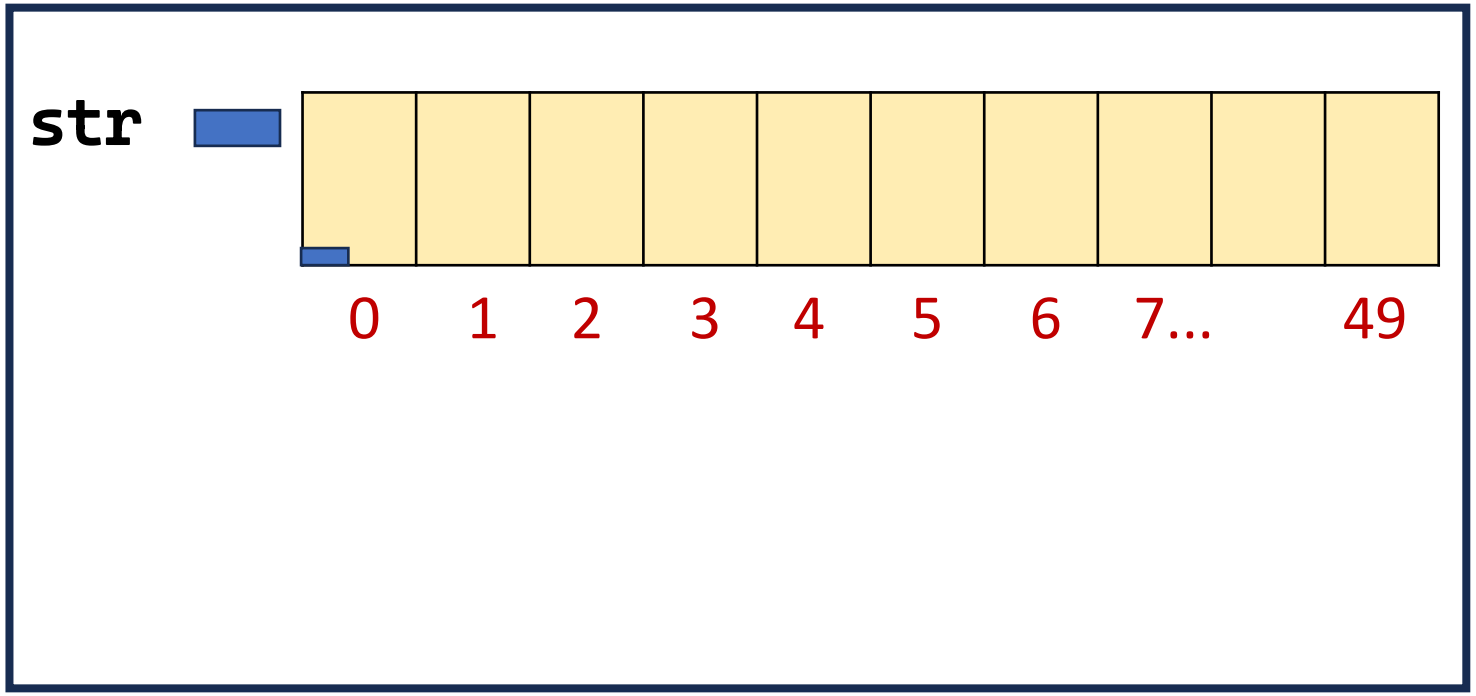
String como um vetor de caracteres

```
char str[50];
```



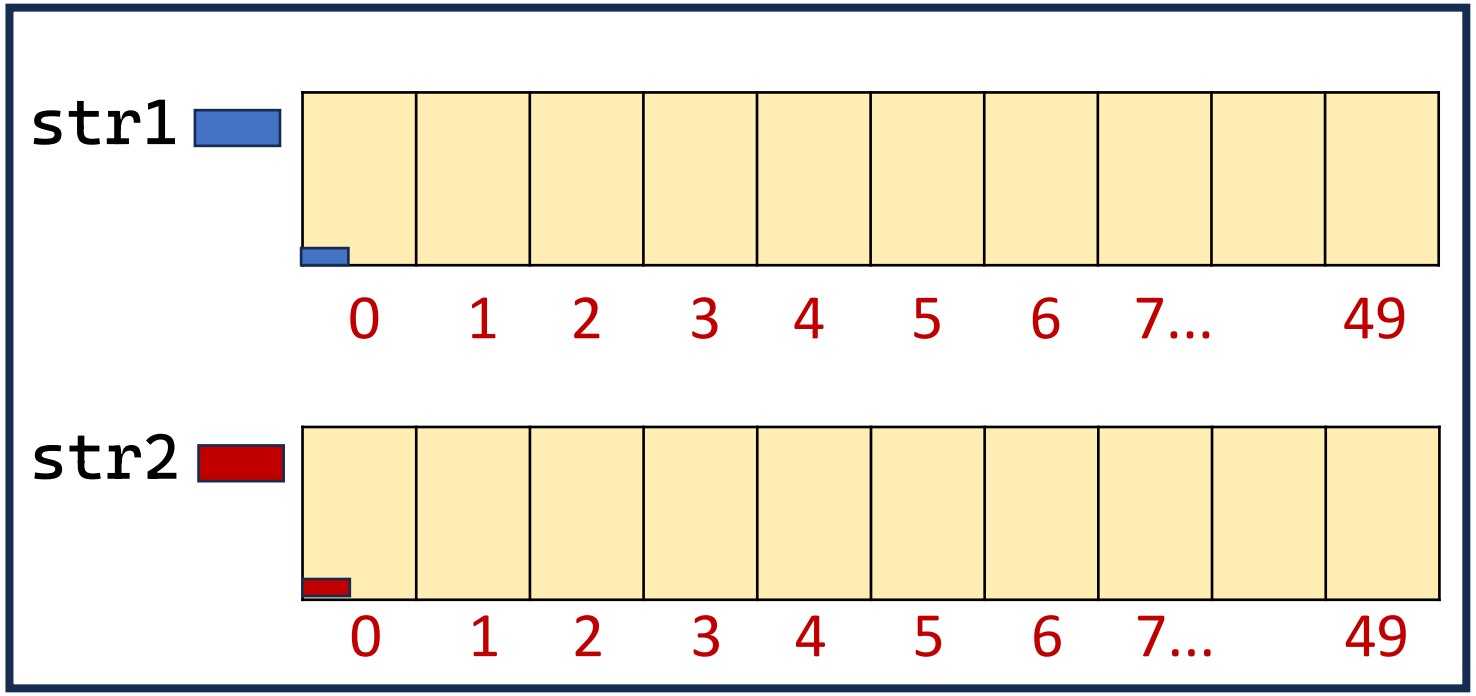
String como um vetor de caracteres

```
const int maxStr = 50;
…
char str[maxStr];
```



Manipulação de *strings*

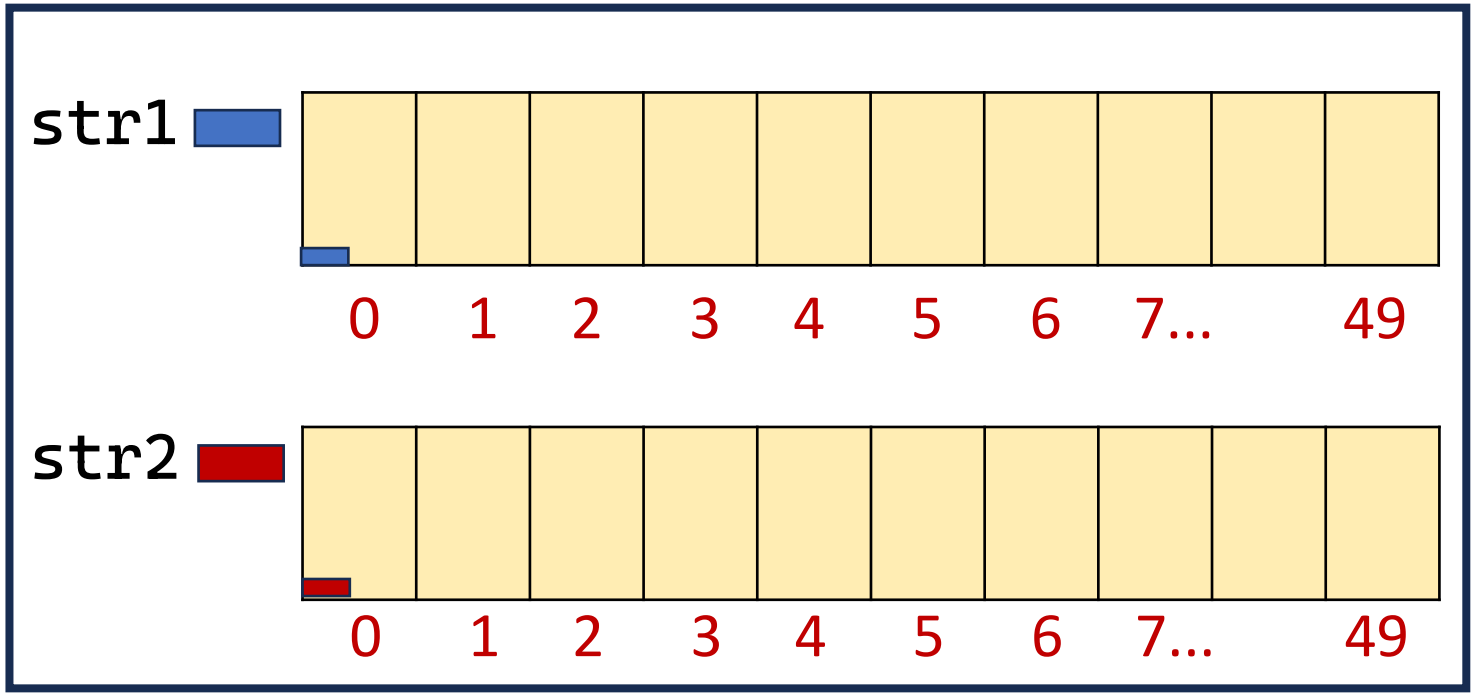
```
const int maxStr = 50;
…

char str1[maxStr];
char str2[maxStr];
```



Uma função que verifique se duas *strings* são iguais

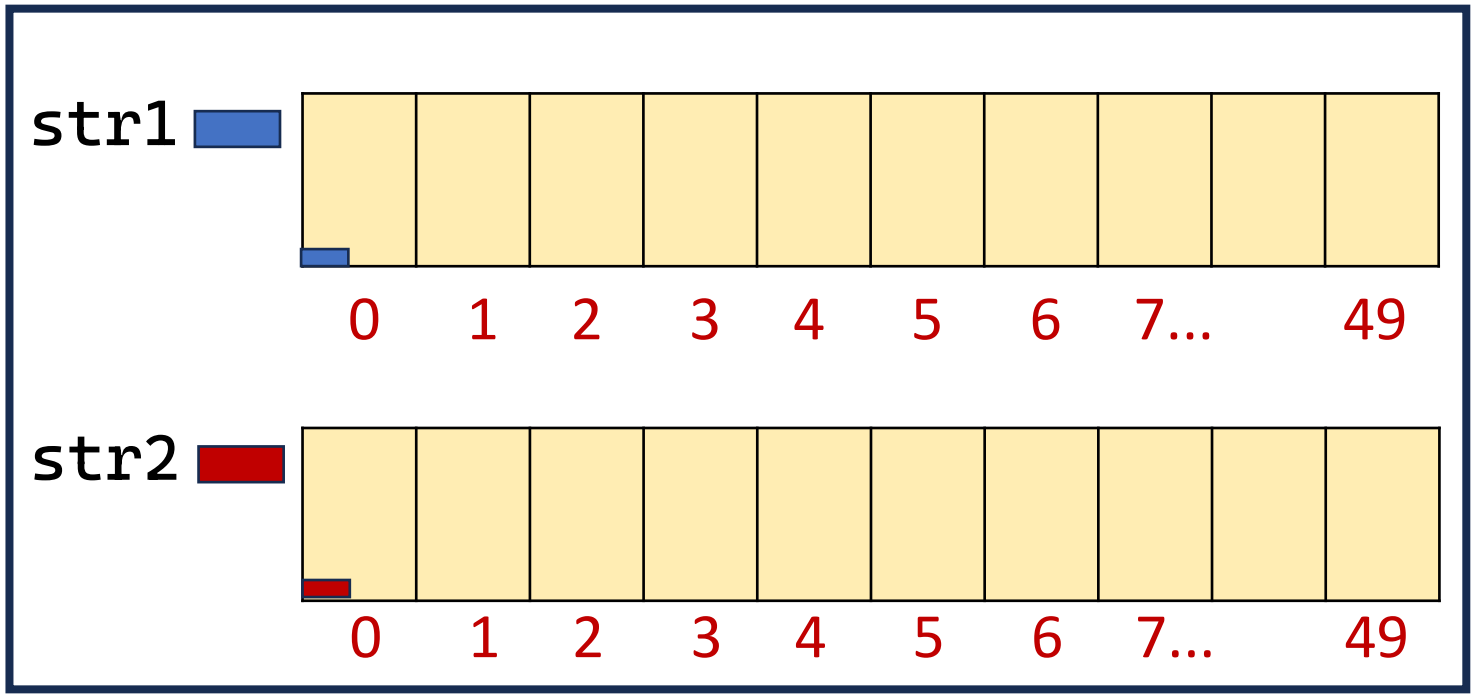
```
const int maxStr = 50;
```

```
bool strIguais(char str1[], char str2[])
{
```



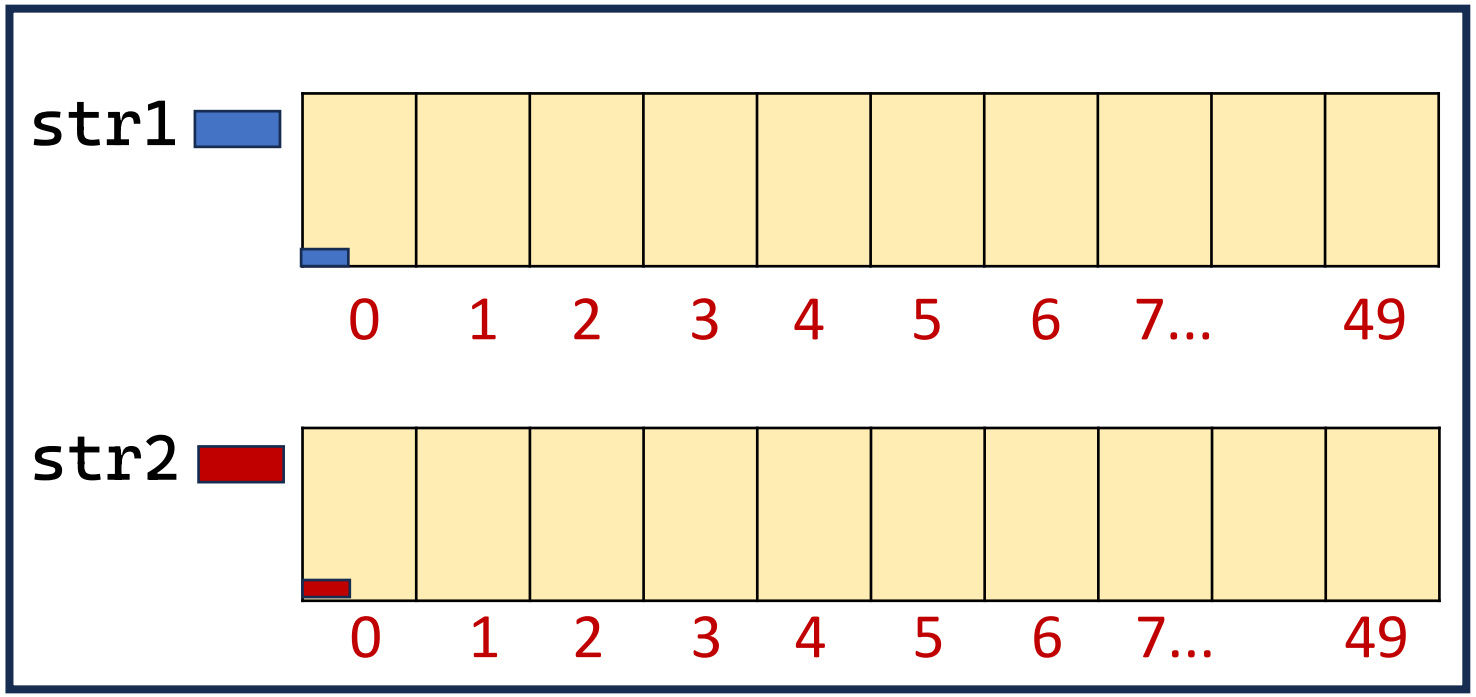

```
} //fim strIguais()
```

```
const int maxStr = 50;

bool strlguais(char str1[], char str2[])
{




} //fim strlguais()
```

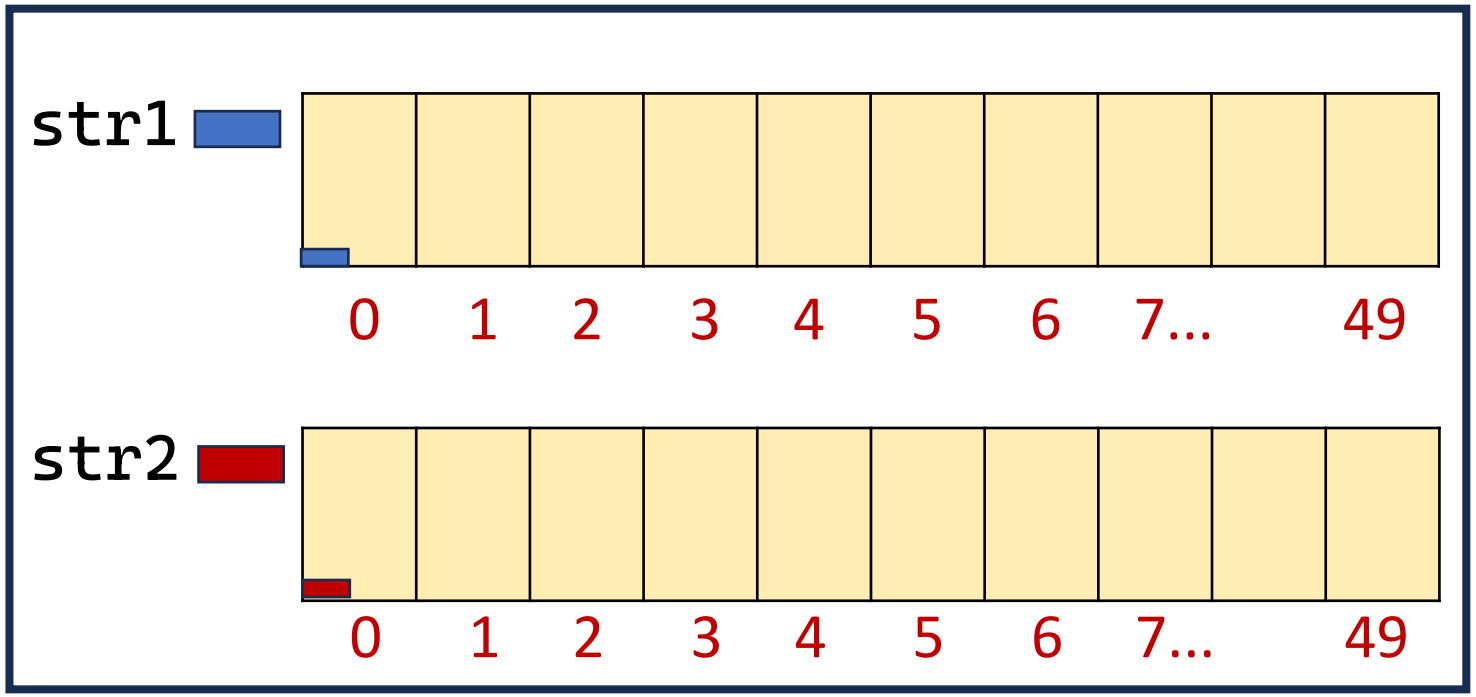


Uma aposta inicial: *strings* diferentes

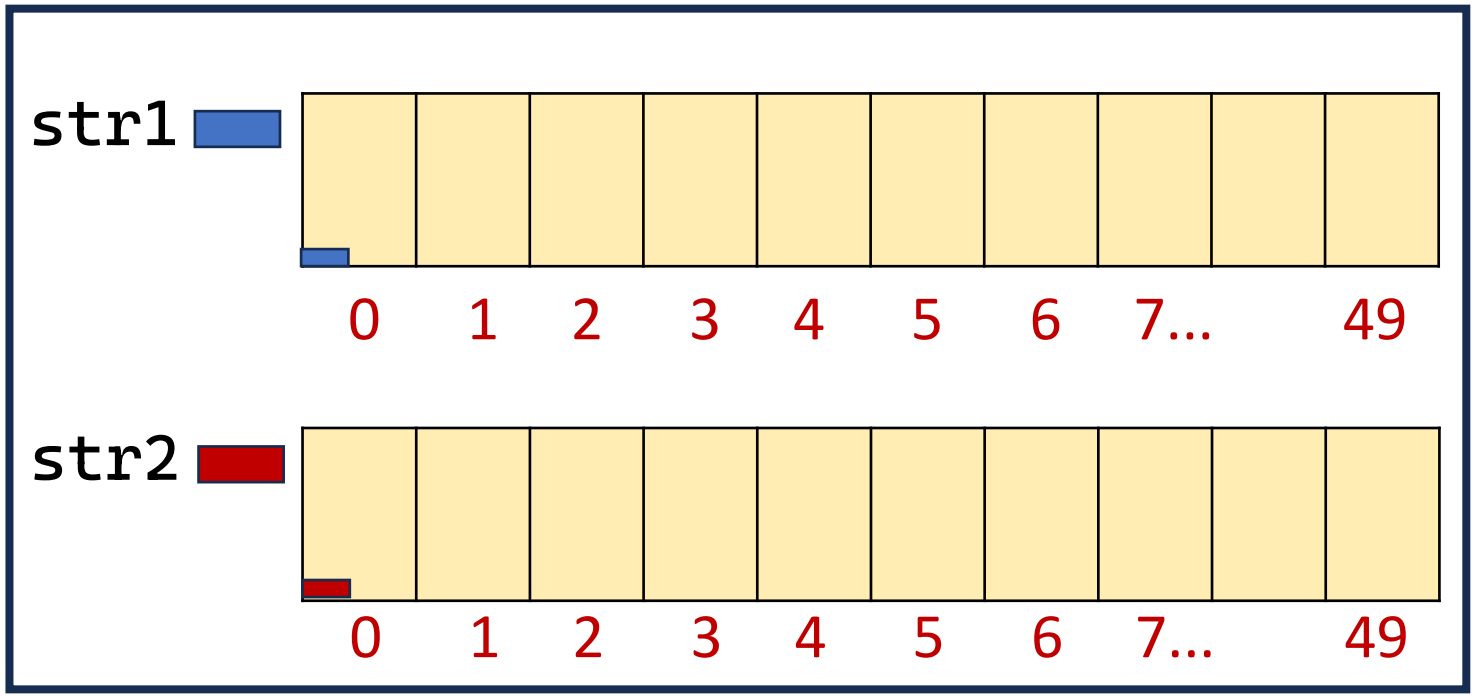
```
const int maxStr = 50;

bool strIguais(char str1[], char str2[]){
    bool iguais=false;




    return iguais;
} //fim strIguais()
```



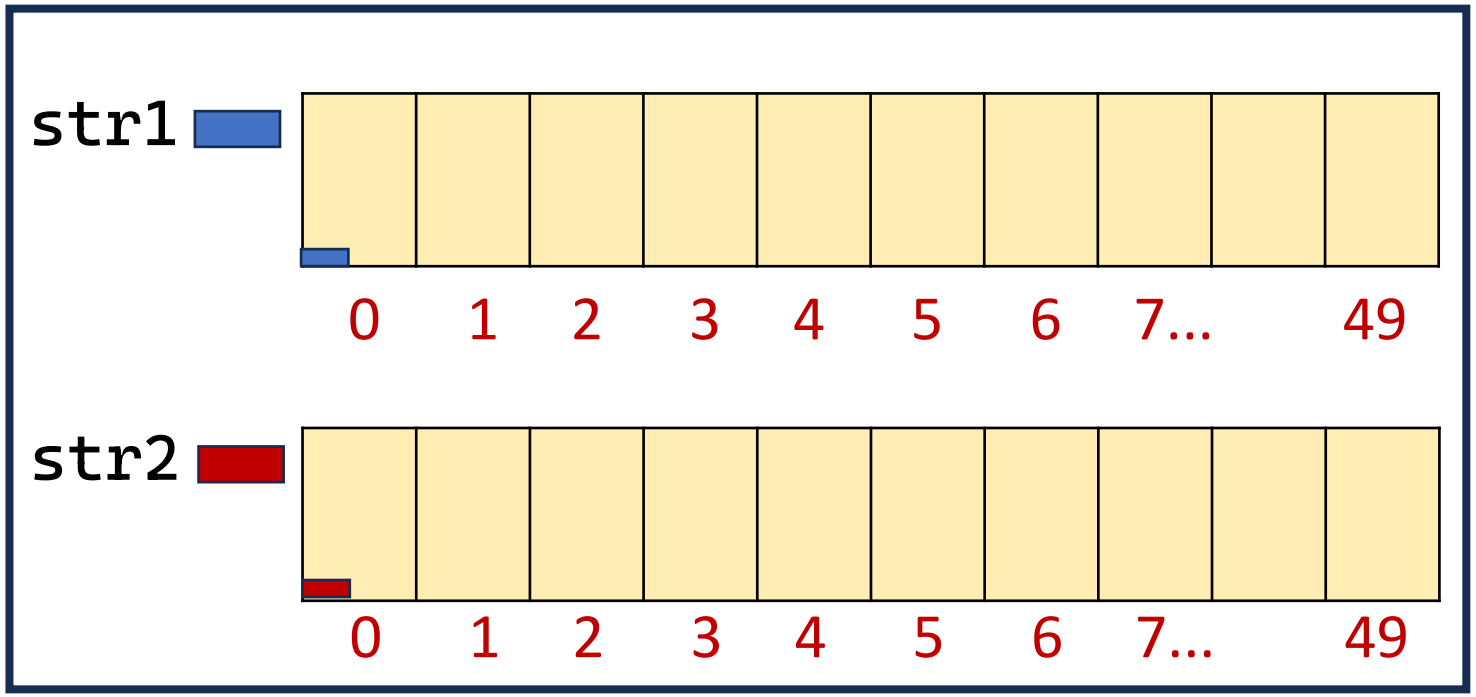
```
const int maxStr = 50;

bool strIguais(char str1[], char str2[]){
    bool iguais=false;
    int i=0;
    while(i<maxStr &&  ){
        i++;
    }
    if(                    ) iguais=true;
    return iguais;
} //fim strIguais()
```



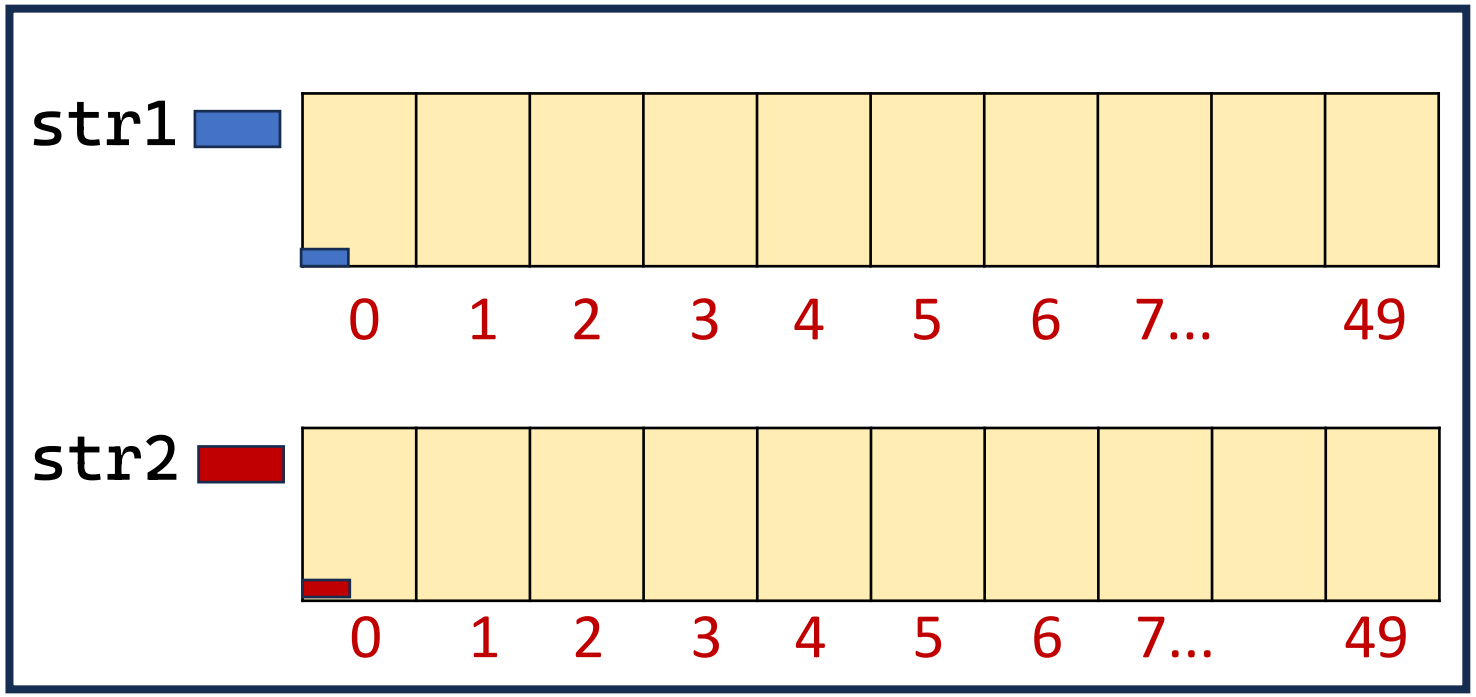
```
const int maxStr = 50;

bool strIguais(char str1[], char str2[]){
    bool iguais=false;
    int i=0;
    while(i<maxStr &&          str1[i] ==          str2[i]    &&       ){
        i++;
    }
    if(                    ) iguais=true;
    return iguais;
} //fim strIguais()
```



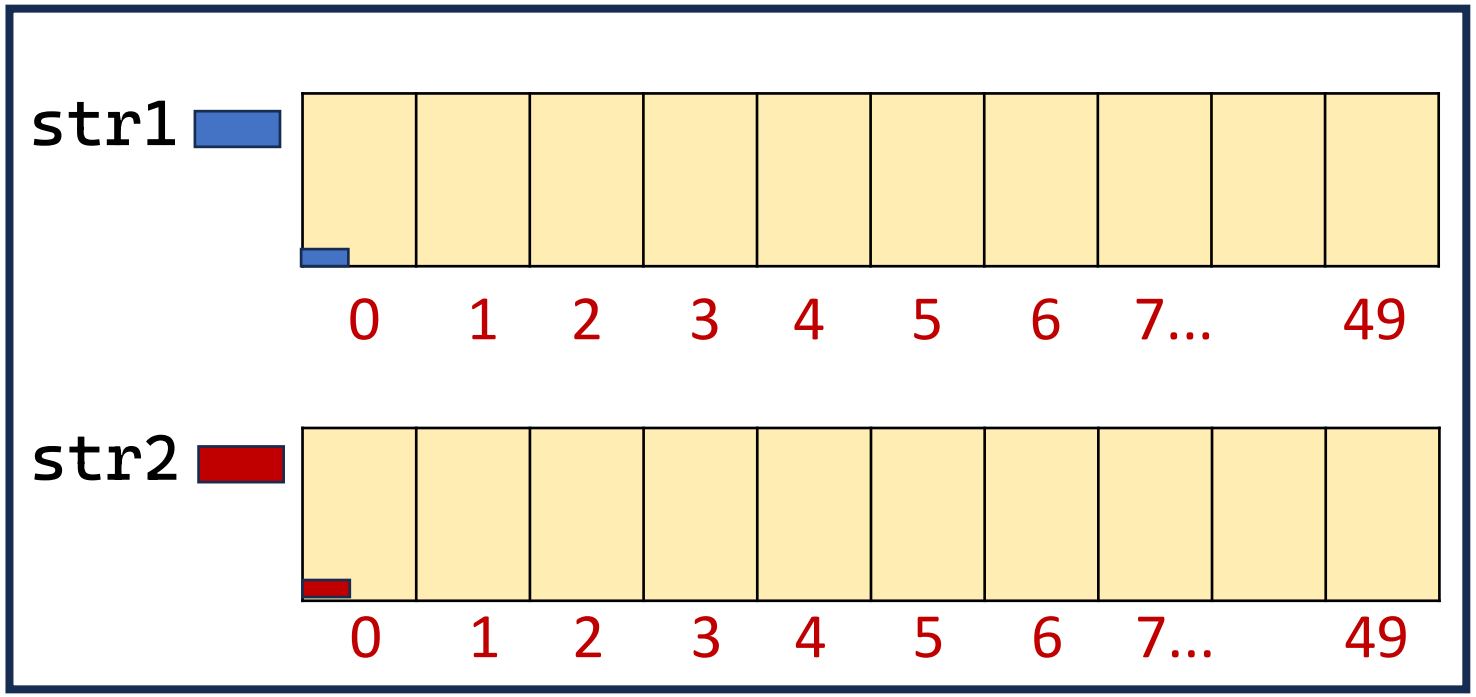
```
const int maxStr = 50;

bool strIguais(char str1[], char str2[]){
    bool iguais=false;
    int i=0;
    while(i<maxStr && toupper(str1[i])==toupper(str2[i]) &&       ){
        i++;
    }
    if(                         ) iguais=true;
    return iguais;
} //fim strIguais()
```



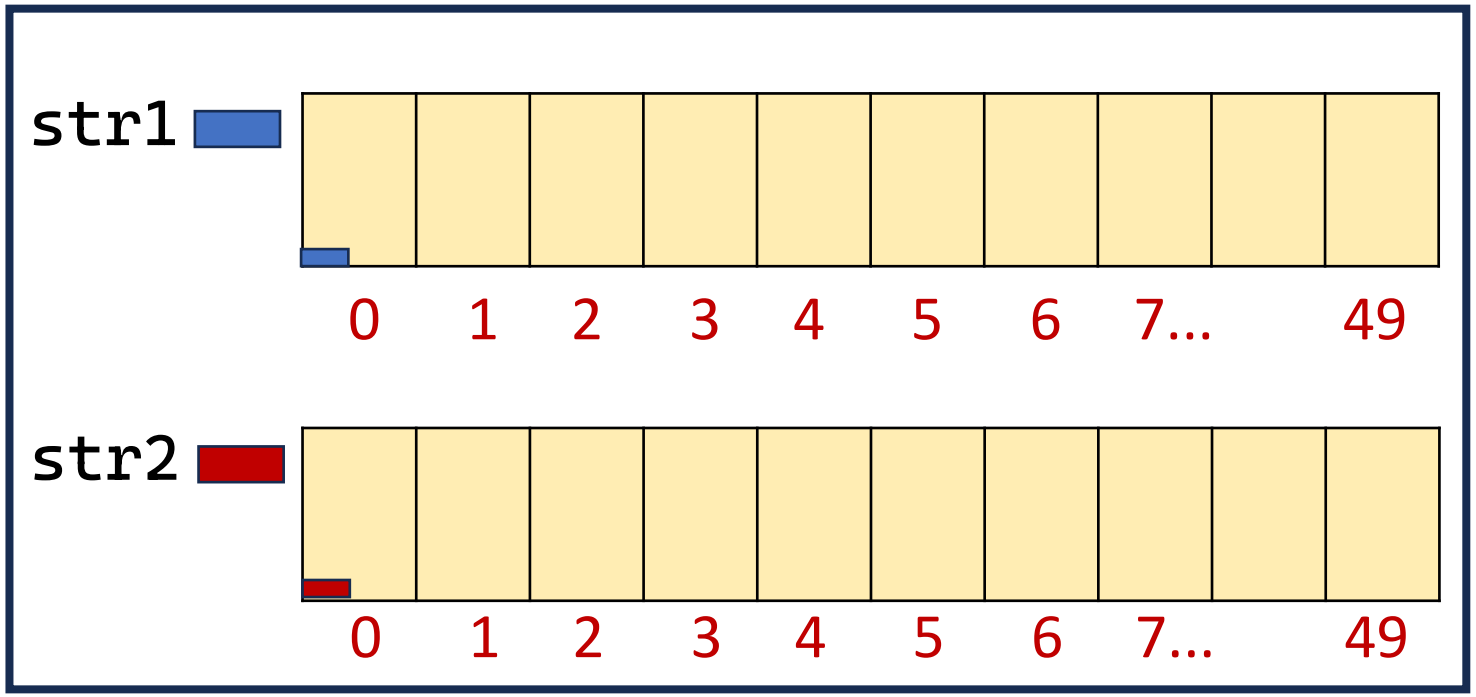
```
const int maxStr = 50;

bool strIguais(char str1[], char str2[]){
    bool iguais=false;
    int i=0;
    while(i<maxStr && toupper(str1[i])==toupper(str2[i]) && str1[i]!='\0' && str2[i]!='\0'){
        i++;
    }
    if(                         ) iguais=true;
    return iguais;
} //fim strIguais()
```



```
const int maxStr = 50;

bool strIguais(char str1[], char str2[]){
    bool iguais=false;
    int i=0;
    while(i<maxStr && toupper(str1[i])==toupper(str2[i]) && str1[i]!='\0' && str2[i]!='\0'){
        i++;
    }
    if( i==maxStr ||        ) iguais=true;
    return iguais;
} //fim strIguais()
```

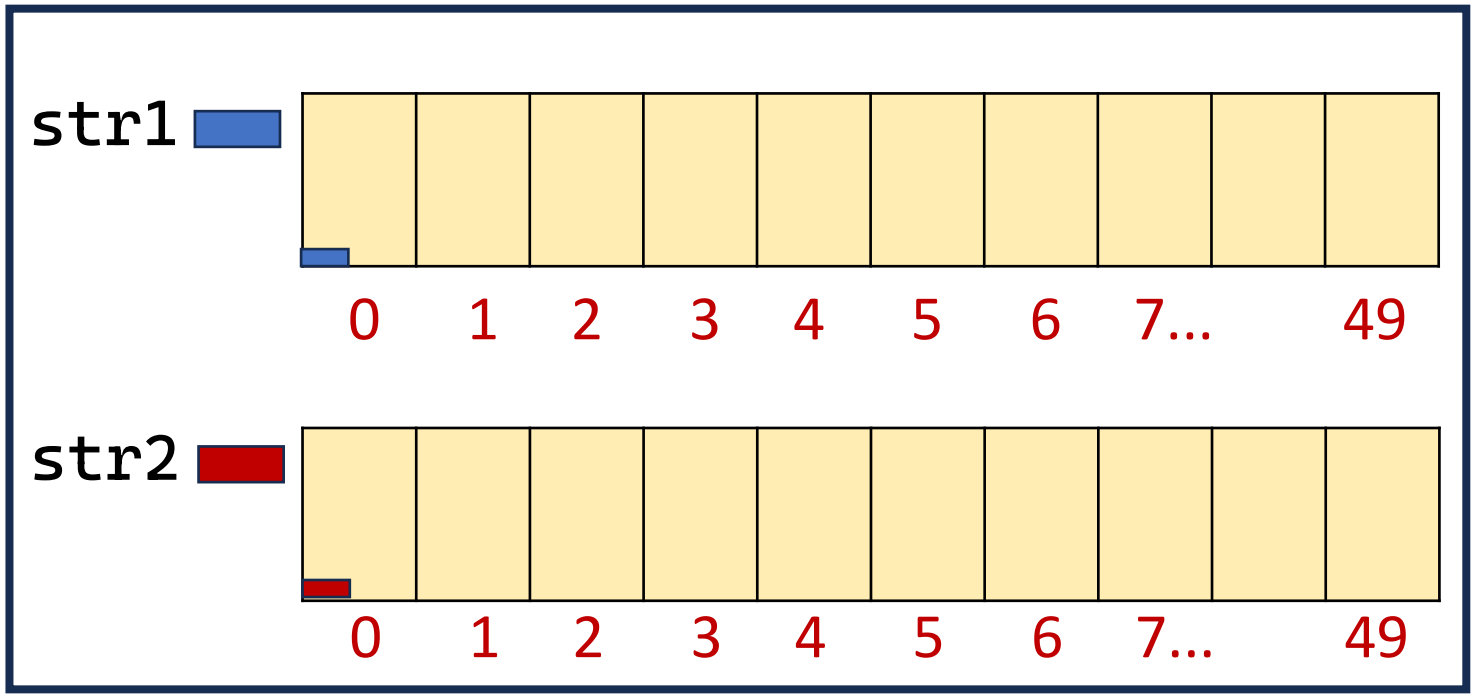


```
const int maxStr = 50;

bool strIguais(char str1[], char str2[]){
    bool iguais=false;
    int i=0;
    while(i<maxStr && toupper(str1[i])==toupper(str2[i]) && str1[i]!='\0' && str2[i]!='\0'){
        i++;
    }
    if( i==maxStr || str1[i]=='\0' || str2[i]=='\0') iguais=true;
    return iguais;
} //fim strIguais()
```

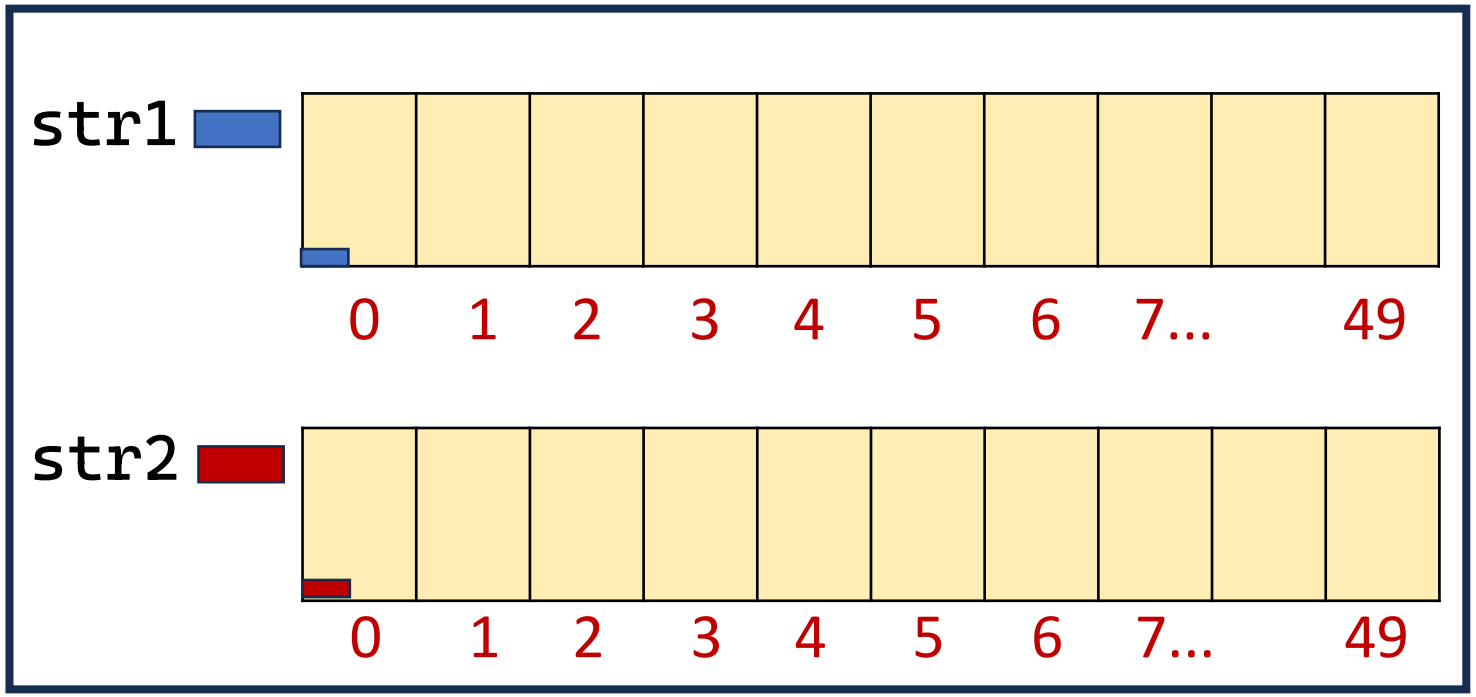


Outra aposta: *strings* iguais

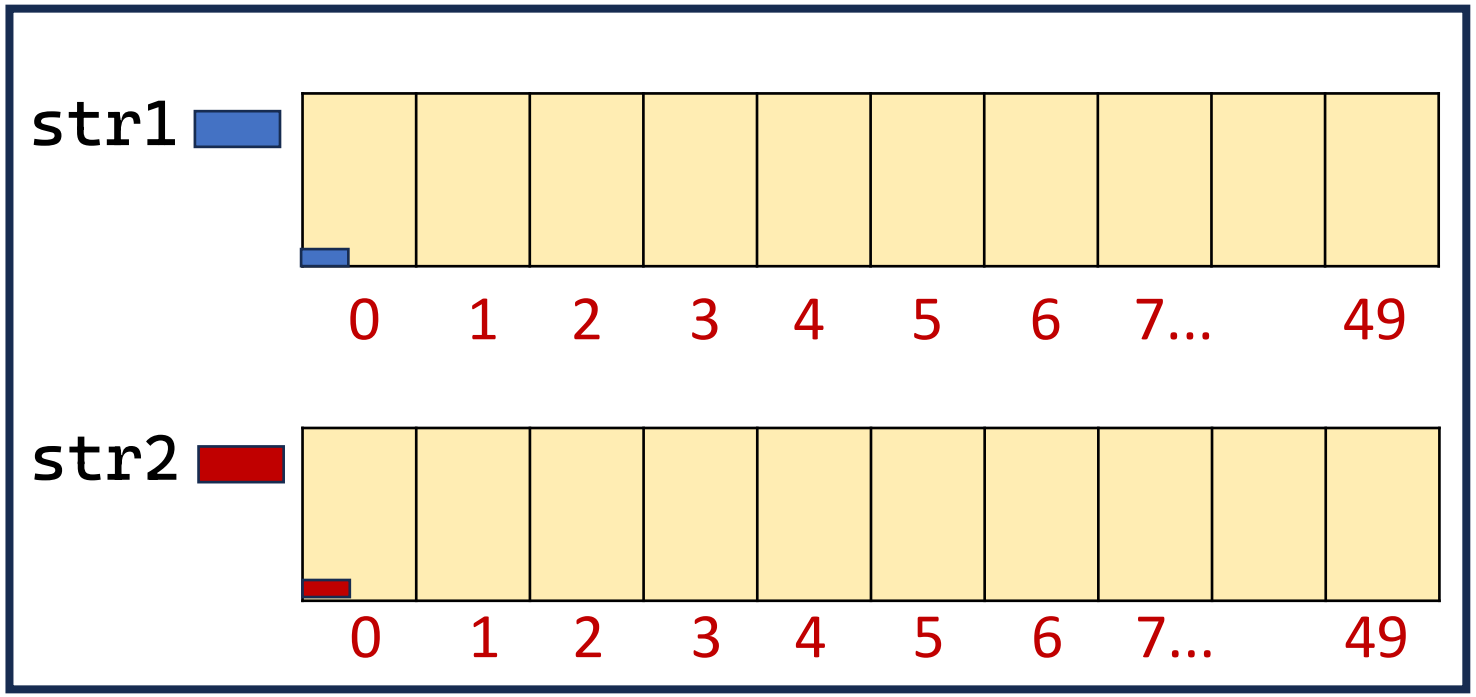
```
const int maxStr = 50;

bool strIguais(char str1[], char str2[]){
    bool iguais=true;




    return iguais;
} //fim strIguais()
```



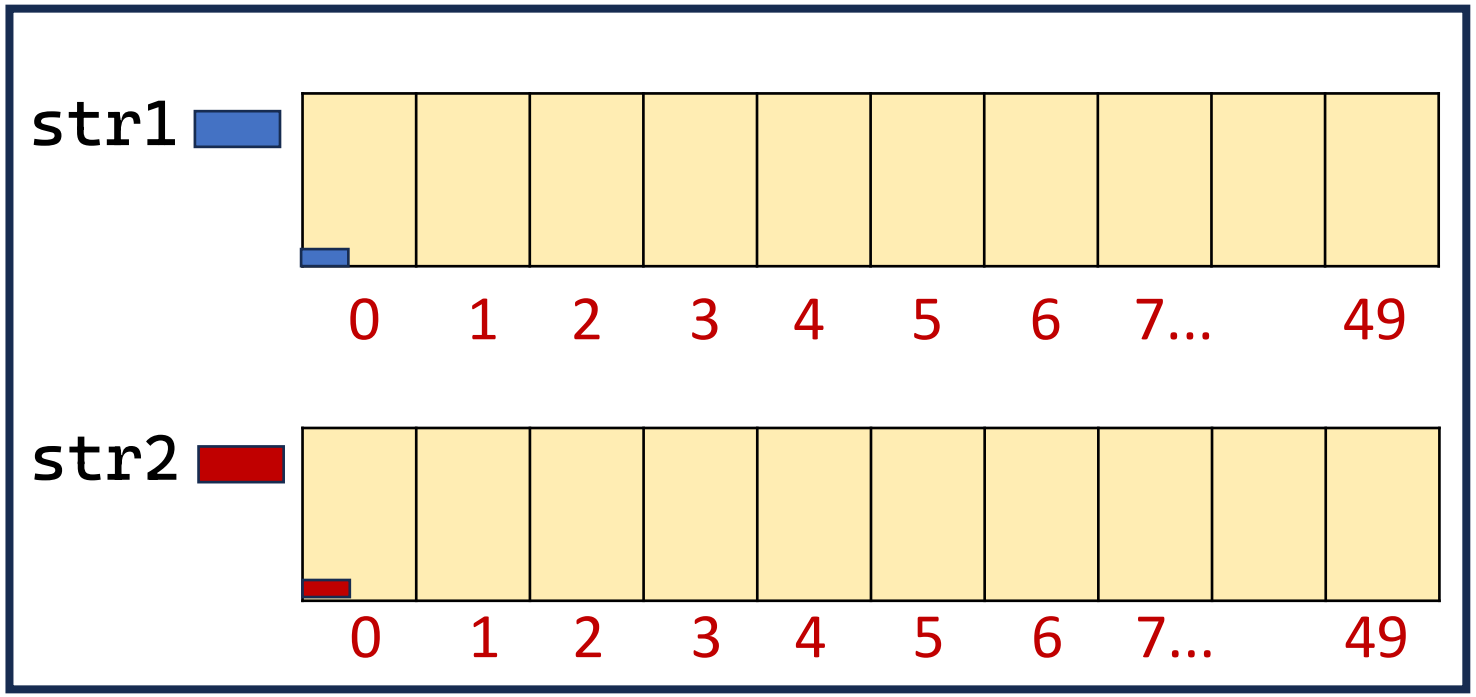
```
const int maxStr = 50;

bool strIguais(char str1[], char str2[]){
    bool iguais=true;
    int i=0;
    while( iguais &&){
        if(              ) iguais=false;
        i++;
    }
    return iguais;
} //fim strIguais()
```



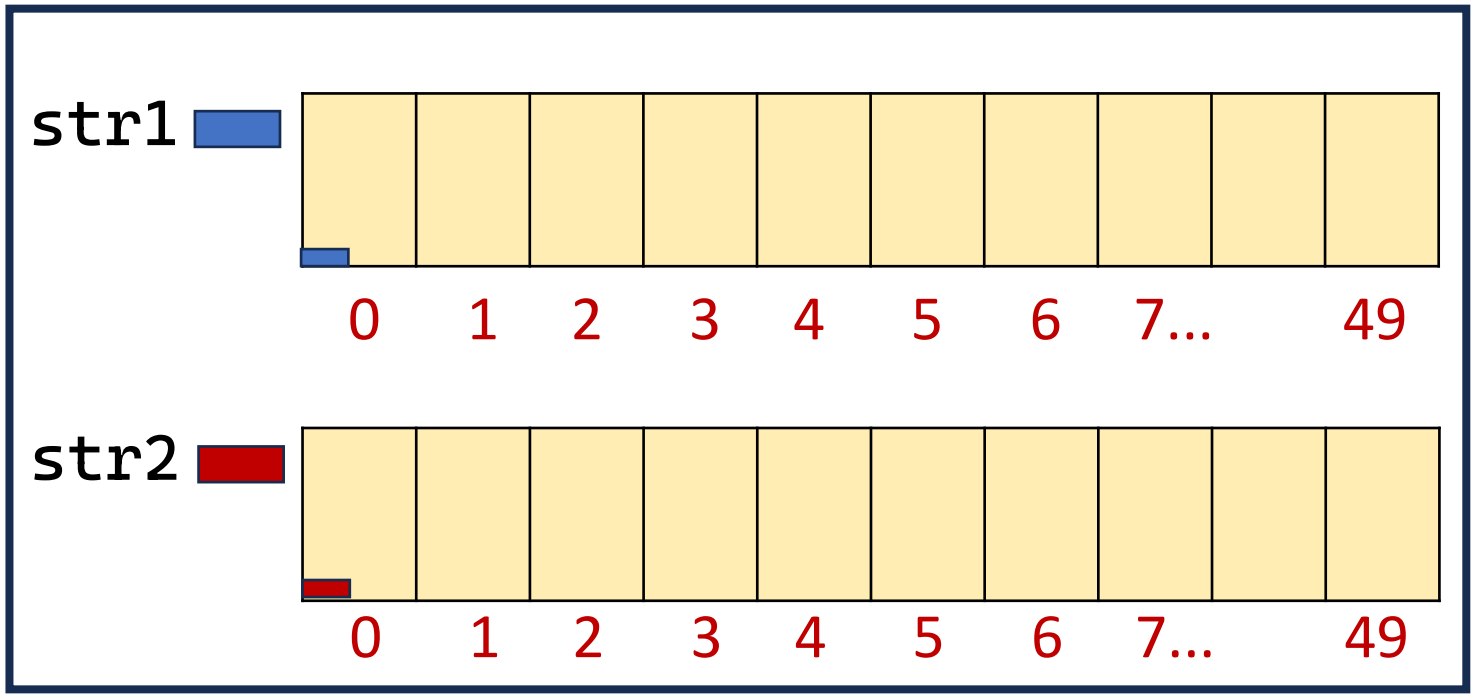
```
const int maxStr = 50;

bool strIguais(char str1[], char str2[]){
    bool iguais=true;
    int i=0;
    while( iguais &&                    && i<maxStr ){
        if(                 ) iguais=false;
        i++;
    }
    return iguais;
} //fim strIguais()
```



```
const int maxStr = 50;

bool strIguais(char str1[], char str2[]){
    bool iguais=true;
    int i=0;
    while( iguais &&                                i<maxStr ){
        if( toupper(str1[i]) != toupper(str2[i]))  iguais=false;
        i++;
    }
    return iguais;
} //fim strIguais()
```



```
const int maxStr = 50;

bool strIguais(char str1[], char str2[]){
    bool iguais=true;
    int i=0;
    while( iguais && str1[i]!='\0' && str2[i]!='\0' && i<maxStr ){
        if( toupper(str1[i]) != toupper(str2[i]))  iguais=false;
        i++;
    }
    return iguais;
} //fim strIguais()
```



Desafios propostos

Lista 26: Questão 8

Problema: escrever uma *string* de forma invertida. Por exemplo, a palavra ROMA deverá ser escrita como AMOR.

Argumento: a *string* a ser escrita

Valor gerado: nenhum

Tente

Construa uma versão recursiva para o problema.

Problema: escrever uma *string* de forma invertida. Por exemplo, a palavra ROMA deverá ser escrita como AMOR.

Argumento: a *string* a ser escrita

Valor gerado: nenhum

Em discussão:
O que já sabemos sobre *strings*

Declarando e escrevendo *strings*

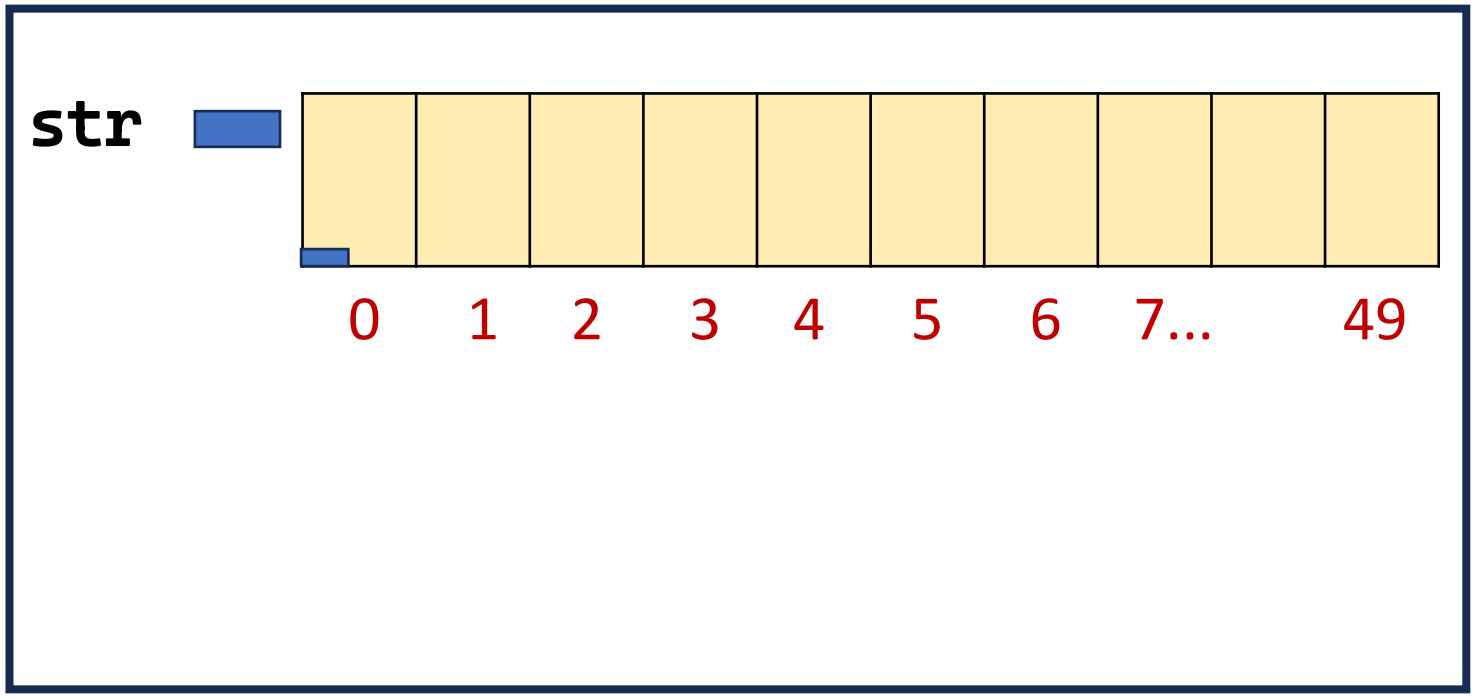
```
char letra = ‘P’;
printf( “%c”, letra );

char* str = “PUC Minas”;
printf( “%s”, str );

char str[] = “PUC Minas”;
printf( “%s”, str );

char palavra[10];
```

Lendo *strings*

```
char palavra[10];

scanf( “%s”, palavra );

printf( “%s”, palavra );
```

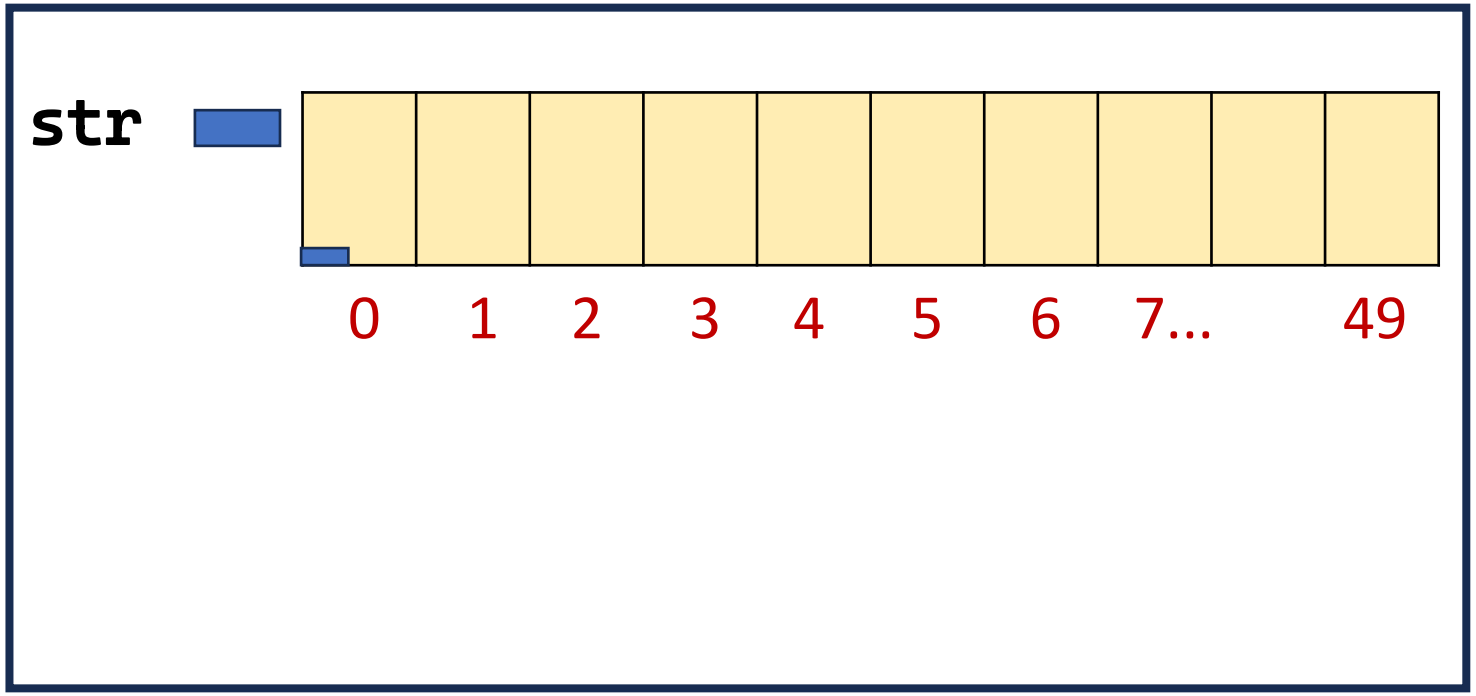
Problemas com *scanf(): espaços e limites*

```
char str[50];

scanf( “%s”, str );
```



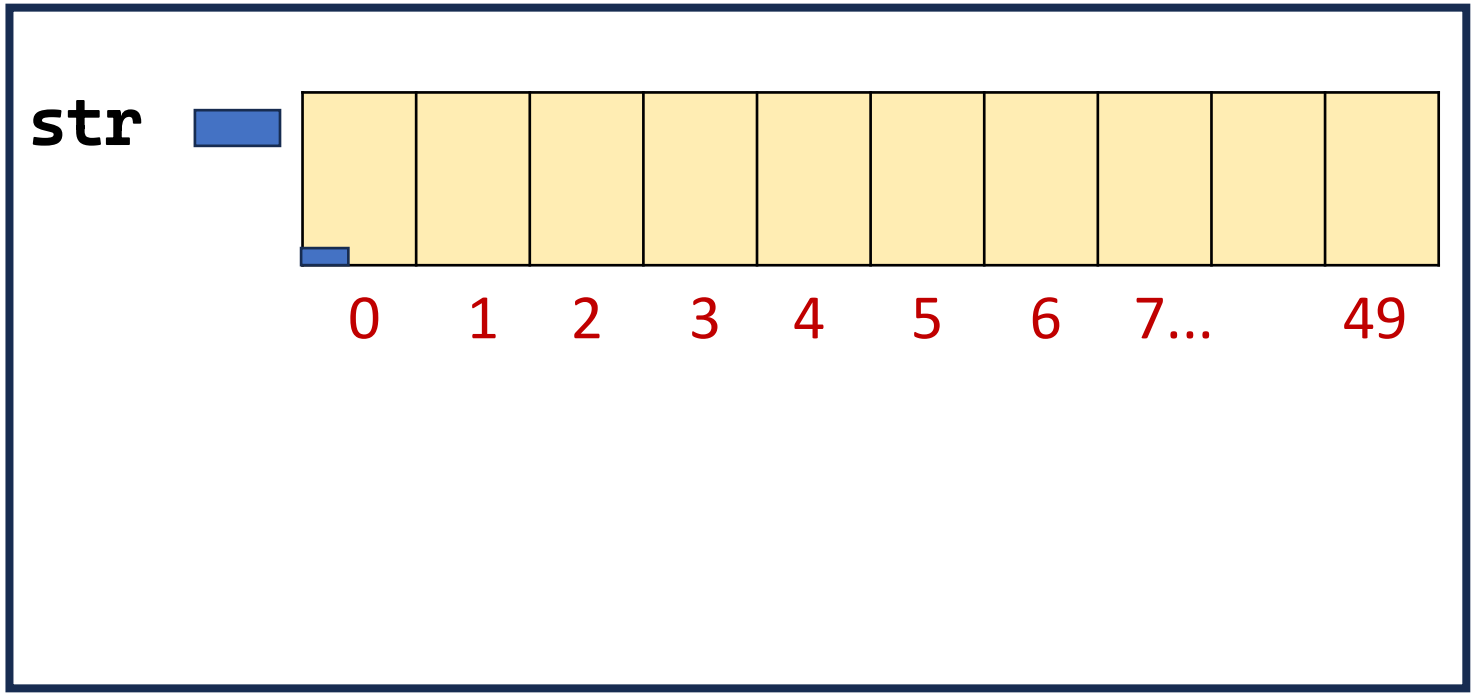
Problemas com *scanf(): espaços e limites*

```
char str[50];

scanf( “%49s”, str );
```



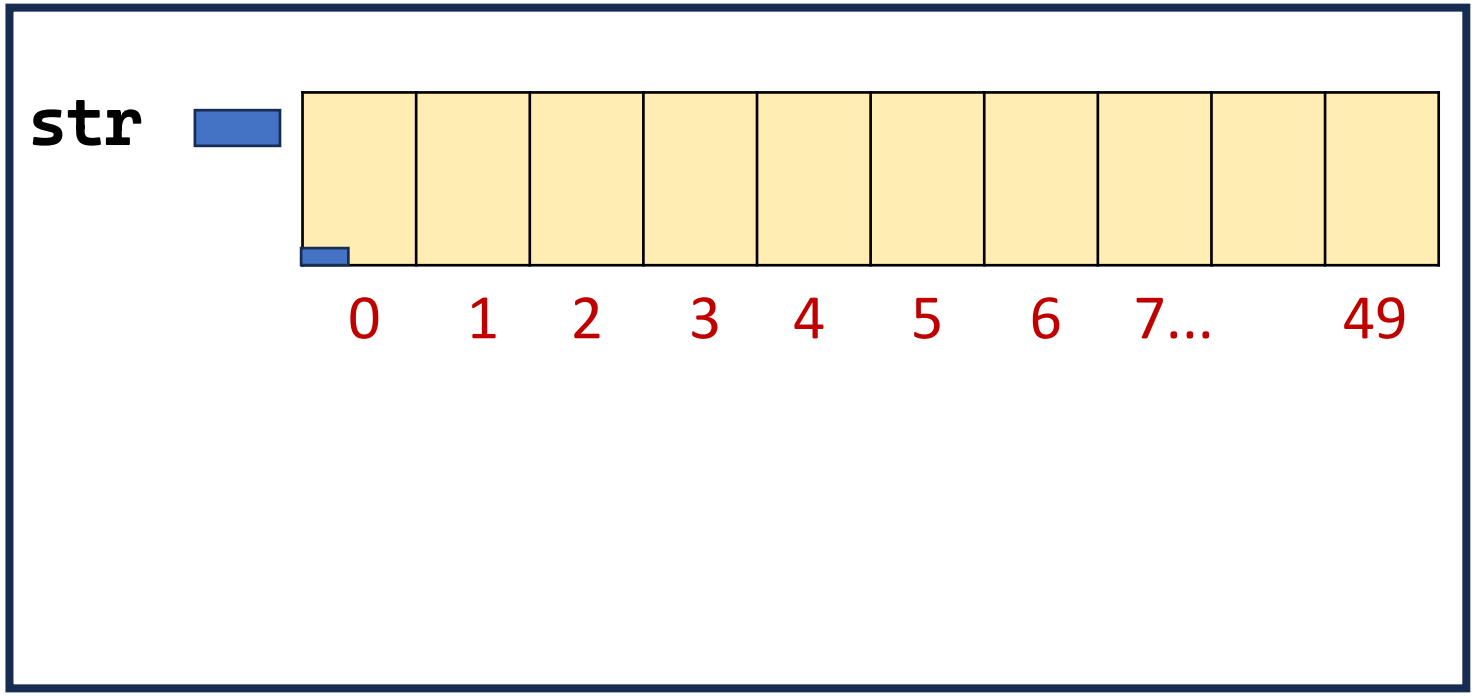
Lendo com *gets()*

```
char str[50];

gets( str );
```



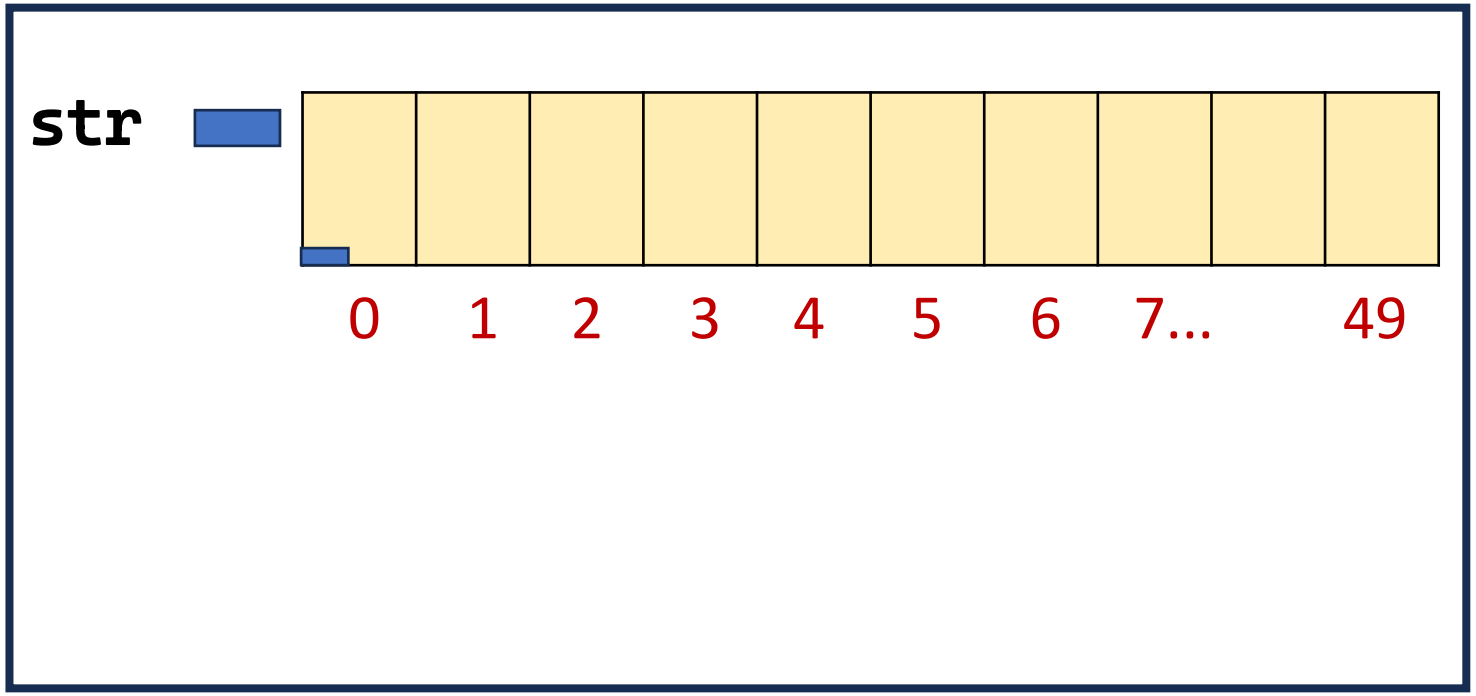
Problema com *gets(): limite*

```
char str[50];

gets( str );
```



Lendo com *fgets()*

```
char str[50];

fgets( str , 50, stdin );
```



Desafios postos

Tente

Construa uma função que receba uma *string* e um arranjo de inteiros de tamanho igual a cinco. A função deverá preencher o vetor de inteiros com o número da vogal correspondente, considerando a seguinte sequência: ‘a’ na posição 0, ‘e’ na posição 1 e assim sucessivamente.

Tente

Construa uma função que receba três *strings.* A função deverá concatenar as duas primeiras *strings* na terceira.

Tente

Construa uma função que receba duas *strings.* A função deverá retornar se são elas iguais ou não.

*Abordagem recursiva