

Strings

DCC 119 – Algoritmos



Sequências de Caracteres



- Sequências de caracteres justapostos são fundamentais no desenvolvimento de programas computacionais.
- Exemplos de sequências de caracteres (representadas internamente num programa):
 - Mensagem de e-mail;
 - Texto de um programa;
 - Nome e endereço em cadastro de clientes, alunos;
 - Sequencia genética. Um gene (ou o DNA de algum organismo) é composto de sequencias dos caracteres A, T, G e C (nucleotídeos);
 - E etc...

Caracteres em C

- Os caracteres em C são representados internamente por códigos numéricos (ASCII);

Alguns caracteres visíveis (podem ser impressos)

	0	1	2	3	4	5	6	7	8	9
30			sp	!	"	#	\$	%	&	'
40	()	*	+	,	-	.	/	0	1
50	2	3	4	5	6	7	8	9	:	;
60	<	=	>	?	@	A	B	C	D	E
70	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y
90	Z	[\]	^	_	`	a	b	c
100	d	e	f	g	h	i	j	k	l	m
110	n	o	p	q	r	s	t	u	v	w
120	x	y	z	{		}				

sp: espaço em branco

Caracteres em C

- A diferença entre caracteres e inteiros está apenas na maneira como são tratados, por exemplo, o código

```
char c = 97;  
printf("int : %d char : %c\n", c, c);
```

- Imprimirá

```
int : 97 char : a
```

- O conteúdo da variável *c* é impressa em dois formatos distintos: %d (int) e %c (char).

Caracteres em C

- Por questão de portabilidade, não se deve usar diretamente, no programa, o código de caracteres;
- Ao invés, usa-se uma constante caractere: caractere entre apóstrofe. Reescrevendo o fragmento de código anterior:

```
char c = 'a';  
rintf("int : %d char: %c\n", c, c);
```

- Imprimirá

```
int : 97 char : a
```

- Pode-se trocar **char c** por **int c**: o resultado será o mesmo.

Caracteres em C



Entrada/Saída de caracteres em C:

- **ch = getchar() ;**

armazena um caractere digitado em **ch** até que ENTER seja pressionado;

- **putchar(ch) ;**

Imprime o caractere **ch** na tela do computador;
semelhante a **printf("%c",ch) ;**

Exemplo:

```
int main() {  
    char ch;  
    ch = getchar();  
    putchar(ch); //ou printf("%c",ch);  
    return 0;  
}
```

Caracteres em C - Exemplo



- Programa para verificar se um dado caractere **c** é um dígito (aproveita o fato de os dígitos estarem em sequência na tabela ASCII):

```
int main()
{
    char c;
    printf("Digite uma letra ou numero: ");
    c = getchar();
    if((c >= '0') && (c <= '9'))
        printf("Eh um digito");
    else
        printf("Nao eh um digito");
    return 0;
}
```

Caracteres em C - Exemplo



- Programa para converter uma letra (caractere) minúscula para maiúsculo (aproveita o fato de os caracteres estarem em sequência na tabela ASCII):

```
int main()
{
    char c;
    printf("Digite uma letra: ");
    c = getchar();

    // verifica se é letra minuscula
    // e imprime maiuscula
    if(c >= 'a' && c <= 'z')
        putchar(c - 'a' + 'A');

    return 0;
}
```


Sequências de Caracteres

- Uma variável usada para armazenar um caractere é representada da seguinte maneira:

```
char letra; // variavel letra do tipo character  
letra = 'a'; // atribuida a letra "a" para a variavel
```

- Se em uma variável do tipo **char** podemos armazenar somente um caractere, então para armazenar vários caracteres (ex: “jose”, “carro”) é necessário **utilizar as sequências de caracteres**, representadas por vetores do tipo **char**.

- Em Programação, sequências de caracteres são usualmente chamadas de **strings**.
- Na linguagem de programação C, uma string é representada por um vetor de caracteres cujo final é marcado pelo caractere **nulo** (`'\0'`).
- Em C, as strings devem **obrigatoriamente** terminar com o caractere nulo.

- Exemplo de declaração:

```
char cidade [15];
```

- A variável **cidade** é um vetor de caracteres (cadeia de caracteres).
- A variável **cidade** pode armazenar qualquer cadeia de até 15 caracteres, incluindo o caractere nulo.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
cidade	J	u	i	z		d	e		F	o	r	a	\0		

Outros exemplos de *strings* em C:

```
char cidade[4] = {'R', 'i', 'o', '\0'};  
char disc[40] = {'A', 'l', 'g', 'o', 'r', 'i', 't', 'm', 'o', '\0'};
```

Outras formas de inicializarmos *strings* em C:

```
char cidade[4] = "Rio";  
char disc[40] = "Algoritmo";
```

Lembre-se sempre de reservar espaço suficiente para a *string* conter o caractere nulo!

Para ilustrar a declaração e a inicialização de *strings*, consideremos as seguintes declarações:

```
char s1[] = ""; //2 aspas sem espaços
char s2[81];
char s3[81] = "Rio";
```

- s1 armazena uma string vazia. Tem um único elemento: '\0';
- s2 representa uma cadeia de caracteres com até 80 caracteres e não é inicializada;
- s3 também é dimensionada para conter até 80 caracteres e é inicializada com a cadeia “Rio”.

Strings: Manipulação



- Como uma string é um **vetor** de caracteres, podemos manipular os seus caracteres.
- A impressão de uma string usa o formato %s.
- Exemplo:

```
char nome[10] = "rio";  
  
char inicial = nome[0];  
printf("Primeira letra: %c\n", inicial);  
  
nome[0] = 'R';  
printf("String alterada: %s\n", nome);
```

Strings: Entrada e Saída

scanf

```
int main()
{
    char s[8];
    printf("Digite uma string: ");
    scanf("%7s", s);
    printf("String digitada: %s", s);
    return 0;
}
```

- A leitura será feita até encontrar um caractere branco - espaço (' '), tabulação ('\t') ou nova linha ('\n') – ou chegar ao tamanho máximo indicado no %s.
- Se digitarmos "Rio de Janeiro", s conterá apenas "Rio";

Strings: Entrada e Saída



scanf

```
int main()
{
    char s[8];
    printf("Digite uma string: ");
    scanf("%7s", s);
    printf("String digitada: %s", s);
    return 0;
}
```

- É necessário indicar no %s qual é o número máximo de caracteres que podem ser lidos, para evitar acesso inválido de memória.
- Assim, se o usuário digitar uma string maior do que o tamanho indicado (neste caso, 7), os caracteres excedentes são descartados e o '\0' será posicionado após os 7 caracteres lidos.

Strings: Entrada e Saída

scanf

```
int main()  
{  
    char s[8];  
    printf("Digite uma string: ");  
    scanf("%7s", s);  
    printf("String digitada: %s", s);  
    return 0;  
}
```

Exemplos de execução do código acima:

```
Digite uma string: 012 345 678  
String digitada: 012
```

```
Digite uma string: 012345678910  
String digitada: 0123456
```

Strings: Entrada e Saída

scanf

```
int main()
{
    char s[8];
    printf("Digite uma string: ");
    scanf("%7s", s);
    printf("String digitada: %s", s);
    return 0;
}
```

Observação importante:

Não é necessário o **& antes da variável **s** no **scanf**.**

Strings: Entrada e Saída



gets , puts

```
int main ()
{
    char s[20];
    printf("Digite uma string: ");
    gets(s);
    printf("String digitada: ");
    puts(s);
    return 0;
}
```

gets(s): lê a string s a partir do teclado;

puts(s): imprime uma string na tela seguida de nova linha.

Strings: Entrada e Saída



gets , puts

```
int main ()
{
    char s[20];
    printf("Digite uma string: ");
    gets(s);
    printf("String digitada: ");
    puts(s);
    return 0;
}
```

Se digitarmos "Rio de Janeiro", s conterà "Rio de Janeiro".

Strings - Exemplo 1



O programa a seguir imprime uma *string*, caractere por caractere:

```
int main()
{
    char str[30];
    int i;
    printf("Digite uma string: ");
    gets(str);

    //imprime cada caractere da string lida
    for(i=0; str[i]!='\0'; i++)
        printf("%c",str[i]);

    return 0;
}
```

Note que, o **for** acima equivale a `printf("%s",str);`

Strings - Exemplo 2



Esse programa calcula e imprime o comprimento (número de caracteres) de uma string:

```
int main()
{
    char str[30];
    int i, n = 0;
    printf("Digite uma string: ");
    gets(str);

    for(i=0; str[i]!='\0'; i++)
        n++;

    printf("\nO tamanho de \"%s\" é: %d", str, n);
    return 0;
}
```

Observe que para imprimir as aspas duplas é necessário utilizar uma barra invertida antes: `printf(" \"%s\" ", s);`

Strings - Exemplo 3



O programa abaixo faz uma cópia de uma string fornecida pelo usuário para outra:

```
int main()
{
    char dest[50], //string destino
          orig[50]; //string origem
    int i;
    printf("Digite uma string: ");
    gets(orig);

    //copia cada caractere de orig para dest
    for(i=0; orig[i]!='\0'; i++)
        dest[i] = orig[i];

    //coloca o caractere nulo para marcar o fim da string
    dest[i] = '\0';

    puts(dest);
    return 0;
}
```

Exercício resolvido 1



Criar uma função que receba como parâmetro uma cadeia de caracteres (*cadeia*) e um caractere adicional (*procurado*). A função deverá retornar a quantidade do caractere *procurado* que foi encontrada na *cadeia*.

Solução proposta:

Precisamos “varrer” a cadeia de caracteres (estrutura de repetição) e contar quantos são iguais ao caractere procurado, caractere a caractere.

Exercício 1 – Solução proposta



```
int conta( char str[], char procurado )
{
    int cont, i;
    i = 0;
    cont = 0;
    while ( str[i] != '\0' )
    {
        if ( str[i] == procurado )
        {
            cont++;
        }
        i++;
    }
    return cont;
}
```

Exercício 1 – Teste de Mesa



```
1  int conta( char str[],
2          char procurado )
3  {
4      int cont, i;
5      i = 0;
6      cont = 0;
7
8      while ( str[i] != '\0' )
9      {
10         if ( str[i] == procurado )
11         {
12             cont++;
13         }
14         i++;
15     }
16     return cont;
17 }
```

Entrada:

str = "teste"
procurado = 't'

Variáveis:

i =
cont =

i	0	1	2	3	4	5
str	t	e	s	t	e	\0

Ordem de execução:

1

Exercício 1 – Teste de Mesa



```
1  int conta( char str[],
2           char procurado )
3  {
4      int cont, i;
5      i = 0;
6      cont = 0;
7
8      while ( str[i] != '\0' )
9      {
10         if ( str[i] == procurado )
11         {
12             cont++;
13         }
14         i++;
15     }
16     return cont;
17 }
```

Entrada:

str = "teste"
procurado = 't'

Variáveis:

i = 0
cont = 0

i	0	1	2	3	4	5
str	t	e	s	t	e	\0

Ordem de execução:

1 5 6

Exercício 1 – Teste de Mesa



```
1  int conta( char str[],
2           char procurado )
3  {
4      int cont, i;
5      i = 0;
6      cont = 0;
7
8      while ( str[i] != '\0' )
9      {
10         if ( str[i] == procurado )
11         {
12             cont++;
13         }
14         i++;
15     }
16     return cont;
17 }
```

Entrada:

str = "teste"
procurado = 't'

Variáveis:

i = 0
cont = 0

i	0	1	2	3	4	5
str	t	e	s	t	e	\0

Ordem de execução:

1 5 6 8

Exercício 1 – Teste de Mesa



```
1  int conta( char str[],
2           char procurado )
3  {
4      int cont, i;
5      i = 0;
6      cont = 0;
7
8      while ( str[i] != '\0' )
9      {
10         if ( str[i] == procurado )
11         {
12             cont++;
13         }
14         i++;
15     }
16     return cont;
17 }
```

Entrada:

str = "teste"

procurado = 't'

Variáveis:

i = 0

cont = 0

i	0	1	2	3	4	5
str	t	e	s	t	e	\0

Ordem de execução:

1 5 6 8 10

Exercício 1 – Teste de Mesa



```
1  int conta( char str[],
2           char procurado )
3  {
4      int cont, i;
5      i = 0;
6      cont = 0;
7
8      while ( str[i] != '\0' )
9      {
10         if ( str[i] == procurado )
11         {
12             cont++;
13         }
14         i++;
15     }
16     return cont;
17 }
```

Entrada:

str = "teste"
procurado = 't'

Variáveis:

i = 0
cont = 1

i	0	1	2	3	4	5
str	t	e	s	t	e	\0

Ordem de execução:

1 5 6 8 10 12

Exercício 1 – Teste de Mesa



```
1  int conta( char str[],
2           char procurado )
3  {
4      int cont, i;
5      i = 0;
6      cont = 0;
7
8      while ( str[i] != '\0' )
9      {
10         if ( str[i] == procurado )
11         {
12             cont++;
13         }
14         i++;
15     }
16     return cont;
17 }
```

Entrada:

str = "teste"
procurado = 't'

Variáveis:

i = 1
cont = 1

i	0	1	2	3	4	5
str	t	e	s	t	e	\0

Ordem de execução:

1 5 6 8 10 12 14

Exercício 1 – Teste de Mesa



```
1  int conta( char str[],
2           char procurado )
3  {
4      int cont, i;
5      i = 0;
6      cont = 0;
7
8      while ( str[i] != '\0' )
9      {
10         if ( str[i] == procurado )
11         {
12             cont++;
13         }
14         i++;
15     }
16     return cont;
17 }
```

Entrada:

str = "teste"
procurado = 't'

Variáveis:

i = 1
cont = 1

i	0	1	2	3	4	5
str	t	e	s	t	e	\0

Ordem de execução:

1 5 6 8 10 12 14 8

Exercício 1 – Teste de Mesa



```
1  int conta( char str[],
2           char procurado )
3  {
4      int cont, i;
5      i = 0;
6      cont = 0;
7
8      while ( str[i] != '\0' )
9      {
10         if ( str[i] == procurado )
11         {
12             cont++;
13         }
14         i++;
15     }
16     return cont;
17 }
```

Entrada:

str = "teste"

procurado = 't'

Variáveis:

i = 1

cont = 1

i	0	1	2	3	4	5
str	t	e	s	t	e	\0

Ordem de execução:

1 5 6 8 10 12 14 8
10

Exercício 1 – Teste de Mesa



```
1  int conta( char str[],
2           char procurado )
3  {
4      int cont, i;
5      i = 0;
6      cont = 0;
7
8      while ( str[i] != '\0' )
9      {
10         if ( str[i] == procurado )
11         {
12             cont++;
13         }
14         i++;
15     }
16     return cont;
17 }
```

Entrada:

str = "teste"
procurado = 't'

Variáveis:

i = 2
cont = 1

i	0	1	2	3	4	5
str	t	e	s	t	e	\0

Ordem de execução:

1 5 6 8 10 12 14 8
10 14

Exercício 1 – Teste de Mesa



```
1  int conta( char str[],
2           char procurado )
3  {
4      int cont, i;
5      i = 0;
6      cont = 0;
7
8      while ( str[i] != '\0' )
9      {
10         if ( str[i] == procurado )
11         {
12             cont++;
13         }
14         i++;
15     }
16     return cont;
17 }
```

Entrada:

str = "teste"
procurado = 't'

Variáveis:

i = 2
cont = 1

i	0	1	2	3	4	5
str	t	e	s	t	e	\0

Ordem de execução:

1 5 6 8 10 12 14 8
10 14 8

Exercício 1 – Teste de Mesa



```
1  int conta( char str[],
2           char procurado )
3  {
4      int cont, i;
5      i = 0;
6      cont = 0;
7
8      while ( str[i] != '\0' )
9      {
10         if ( str[i] == procurado )
11         {
12             cont++;
13         }
14         i++;
15     }
16     return cont;
17 }
```

Entrada:

str = "teste"

procurado = 't'

Variáveis:

i = 2

cont = 1

i	0	1	2	3	4	5
str	t	e	s	t	e	\0

Ordem de execução:

1 5 6 8 10 12 14 8
10 14 8 10

Exercício 1 – Teste de Mesa



```
1  int conta( char str[],
2           char procurado )
3  {
4      int cont, i;
5      i = 0;
6      cont = 0;
7
8      while ( str[i] != '\0' )
9      {
10         if ( str[i] == procurado )
11         {
12             cont++;
13         }
14         i++;
15     }
16     return cont;
17 }
```

Entrada:

str = "teste"
procurado = 't'

Variáveis:

i = 3
cont = 1

i	0	1	2	3	4	5
str	t	e	s	t	e	\0

Ordem de execução:

1 5 6 8 10 12 14 8
10 14 8 10 14

Exercício 1 – Teste de Mesa



```
1  int conta( char str[],
2           char procurado )
3  {
4      int cont, i;
5      i = 0;
6      cont = 0;
7
8      while ( str[i] != '\0' )
9      {
10         if ( str[i] == procurado )
11         {
12             cont++;
13         }
14         i++;
15     }
16     return cont;
17 }
```

Entrada:

str = "teste"
procurado = 't'

Variáveis:

i = 3
cont = 1

i	0	1	2	3	4	5
str	t	e	s	t	e	\0

Ordem de execução:

1 5 6 8 10 12 14 8
10 14 8 10 14 8

Exercício 1 – Teste de Mesa



```
1  int conta( char str[],
2           char procurado )
3  {
4      int cont, i;
5      i = 0;
6      cont = 0;
7
8      while ( str[i] != '\0' )
9      {
10         if ( str[i] == procurado )
11         {
12             cont++;
13         }
14         i++;
15     }
16     return cont;
17 }
```

Entrada:

str = "teste"

procurado = 't'

Variáveis:

i = 3

cont = 1

i	0	1	2	3	4	5
str	t	e	s	t	e	\0

Ordem de execução:

1 5 6 8 10 12 14 8
10 14 8 10 14 8 10

Exercício 1 – Teste de Mesa



```
1  int conta( char str[],
2           char procurado )
3  {
4      int cont, i;
5      i = 0;
6      cont = 0;
7
8      while ( str[i] != '\0' )
9      {
10         if ( str[i] == procurado )
11         {
12             cont++;
13         }
14         i++;
15     }
16     return cont;
17 }
```

Entrada:

str = "teste"
procurado = 't'

Variáveis:

i = 3
cont = 2

i	0	1	2	3	4	5
str	t	e	s	t	e	\0

Ordem de execução:

1 5 6 8 10 12 14 8
10 14 8 10 14 8 10
12

Exercício 1 – Teste de Mesa



```
1  int conta( char str[],
2           char procurado )
3  {
4      int cont, i;
5      i = 0;
6      cont = 0;
7
8      while ( str[i] != '\0' )
9      {
10         if ( str[i] == procurado )
11         {
12             cont++;
13         }
14         i++;
15     }
16     return cont;
17 }
```

Entrada:

str = "teste"
procurado = 't'

Variáveis:

i = 4
cont = 2

i	0	1	2	3	4	5
str	t	e	s	t	e	\0

Ordem de execução:

1 5 6 8 10 12 14 8
10 14 8 10 14 8 10
12 14

Exercício 1 – Teste de Mesa



```
1  int conta( char str[],
2           char procurado )
3  {
4      int cont, i;
5      i = 0;
6      cont = 0;
7
8      while ( str[i] != '\0' )
9      {
10         if ( str[i] == procurado )
11         {
12             cont++;
13         }
14         i++;
15     }
16     return cont;
17 }
```

Entrada:

str = "teste"
procurado = 't'

Variáveis:

i = 4
cont = 2

i	0	1	2	3	4	5
str	t	e	s	t	e	\0

Ordem de execução:

1 5 6 8 10 12 14 8
10 14 8 10 14 8 10
12 14 8

Exercício 1 – Teste de Mesa



```
1  int conta( char str[],
2           char procurado )
3  {
4      int cont, i;
5      i = 0;
6      cont = 0;
7
8      while ( str[i] != '\0' )
9      {
10         if ( str[i] == procurado )
11         {
12             cont++;
13         }
14         i++;
15     }
16     return cont;
17 }
```

Entrada:

str = "teste"

procurado = 't'

Variáveis:

i = 4

cont = 2

i	0	1	2	3	4	5
str	t	e	s	t	e	\0

Ordem de execução:

```
1 5 6 8 10 12 14 8
10 14 8 10 14 8 10
12 14 8 10
```

Exercício 1 – Teste de Mesa



```
1  int conta( char str[],
2           char procurado )
3  {
4      int cont, i;
5      i = 0;
6      cont = 0;
7
8      while ( str[i] != '\0' )
9      {
10         if ( str[i] == procurado )
11         {
12             cont++;
13         }
14         i++;
15     }
16     return cont;
17 }
```

Entrada:

str = "teste"
procurado = 't'

Variáveis:

i = 5
cont = 2

i	0	1	2	3	4	5
str	t	e	s	t	e	\0

Ordem de execução:

1 5 6 8 10 12 14 8
10 14 8 10 14 8 10
12 14 8 10 14

Exercício 1 – Teste de Mesa



```
1  int conta( char str[],
2           char procurado )
3  {
4      int cont, i;
5      i = 0;
6      cont = 0;
7
8      while ( str[i] != '\0' )
9      {
10         if ( str[i] == procurado )
11         {
12             cont++;
13         }
14         i++;
15     }
16     return cont;
17 }
```

Entrada:

str = "teste"
procurado = 't'

Variáveis:

i = 5
cont = 2

i	0	1	2	3	4	5
str	t	e	s	t	e	\0

Ordem de execução:

1	5	6	8	10	12	14	8
10	14	8	10	14	8	10	
12	14	8	10	14	8		

Exercício 1 – Teste de Mesa



```
1  int conta( char str[],
2           char procurado )
3  {
4      int cont, i;
5      i = 0;
6      cont = 0;
7
8      while ( str[i] != '\0' )
9      {
10         if ( str[i] == procurado )
11         {
12             cont++;
13         }
14         i++;
15     }
16     return cont;
17 }
```

Entrada:

str = "teste"
procurado = 't'

Variáveis:

i = 5
cont = 2

i	0	1	2	3	4	5
str	t	e	s	t	e	\0

Ordem de execução:

1	5	6	8	10	12	14	8
10	14	8	10	14	8	10	
12	14	8	10	14	8	16	

Strings e Funções



A passagem de strings como parâmetros é por referência assim como vetores numéricos:

- O tamanho da string não precisa ser especificado na definição da função e, com isso, strings de qualquer tamanho podem ser passadas na chamada da função;

```
void novaFuncao(char nomeDaString[]);
```

- Se uma string é passada por parâmetro, qualquer alteração feita no interior da função é realizada diretamente na string original.

```
str = "abc";
```

```
passaParaMaiuscula(str); //funcao modifica str
```

```
printf("%s",str); //vai imprimir "ABC"
```

Funções para Strings



- Existem várias funções em C para manipulação de *strings*.
- Essas funções estão no arquivo ***string.h***.
- Entre elas pode-se destacar:
 - ***strcpy(char destino[], char origem[])***
copia a string origem na string destino
 - ***strlen(char str[])***
retorna o tamanho da string “str”
 - ***strcat(char destino[], char origem[])***
faz concatenação (junção) da string origem com a string destino. O resultado é armazenado na string destino

Exercício resolvido 2

Criar uma função para verificar se a string *s2* está contida na string *s1*. A função deverá retornar 1 se encontrar a string ou 0, caso contrário.

Exemplo:

- Se *s1* fosse “*Ana Maria Silva*” e *s2* fosse “*Maria*”, a função retornaria 1, pois *s2* está contido em *s1*.

Exercício 2 – Solução proposta

```
int buscaString(char s1[], char s2[]) {  
    int i,j;  
    for(i=0; s1[i]!='\0'; i++) {  
  
        //compara caracteres de s1 e s2 começando  
        // da posição i de s1 e da posição 0 de s2  
        for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)  
            //se os caracteres forem distintos, para de comparar  
            if (s2[j] != s1[i+j])  
                break;  
  
        //se as comparações chegaram no final de s2, então  
        //s1 tem copia de s2 começando na posição i  
        if (s2[j]=='\0')  
            return 1;  
  
        //se não chegaram no final, s1 não tem cópia de  
        //s2 começando em i, mas pode ter em outra posição.  
    }  
    //se não foi encontrada s2 em cada posição i de s1,  
    //então s1 não tem cópia de s2  
    return 0;  
}
```

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {  
2     int i,j;  
3     for(i=0; s1[i]!='\0'; i++) {  
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)  
5             if (s2[j] != s1[i+j])  
6                 break;  
7         if (s2[j]=='\0')  
8             return 1;  
9     }  
10    return 0;  
11 }
```

Ordem de execução:

1

Entrada:

s1 = "Este é um teste"

s2 = "este"

Variáveis adicionais:

i = ?

j = ?

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

1 3

Entrada:

s1 = "Este é um teste"

s2 = "este"

Variáveis adicionais:

i = 0

j = ?

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

1 3 4

Entrada:

s1 = "Este é um teste"

s2 = "este"

Variáveis adicionais:

i = 0

j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

1 3 4 **5**

Entrada:

s1 = "Este é um teste"

s2 = "este"

Variáveis adicionais:

i = 0

j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

1 3 4 5 6

Entrada:

s1 = "Este é um teste"

s2 = "este"

Variáveis adicionais:

i = 0

j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

1 3 4 5 6 7

Entrada:

s1 = "Este é um teste"

s2 = "este"

Variáveis adicionais:

i = 0

j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

1 3 4 5 6 7 **3**

Entrada:

s1 = "Este é um teste"

s2 = "este"

Variáveis adicionais:

i = 1

j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

1 3 4 5 6 7 3 4

Entrada:

s1 = "Este é um teste"

s2 = "este"

Variáveis adicionais:

i = 1

j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

1 3 4 5 6 7 3 4 5

Entrada:

s1 = "Este é um teste"

s2 = "este"

Variáveis adicionais:

i = 1

j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa



```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

1 3 4 5 6 7 3 4 5
6

Entrada:

s1 = "Este é um teste"

s2 = "este"

Variáveis adicionais:

i = 1

j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

1 3 4 5 6 7 3 4 5
6 7

Entrada:

s1 = "Este é um teste"

s2 = "este"

Variáveis adicionais:

i = 1

j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa



```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

1 3 4 5 6 7 3 4 5
6 7 3

Entrada:

s1 = "Este é um teste"

s2 = "este"

Variáveis adicionais:

i = 2

j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

1 3 4 5 6 7 3 4 5
6 7 3 4

Entrada:

s1 = "Este é um teste"

s2 = "este"

Variáveis adicionais:

i = 2

j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

1 3 4 5 6 7 3 4 5
6 7 3 4 5

Entrada:

s1 = "Este é um teste"

s2 = "este"

Variáveis adicionais:

i = 2

j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa



```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

1 3 4 5 6 7 3 4 5
6 7 3 4 5 6

Entrada:

s1 = "Este é um teste"

s2 = "este"

Variáveis adicionais:

i = 2

j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa



```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7

Entrada:

s1 = "Este é um teste"

s2 = "este"

Variáveis adicionais:

i = 2

j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3

Entrada:

s1 = "Este é um teste"

s2 = "este"

Variáveis adicionais:

i = 3

j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4

Entrada:

s1 = "Este é um teste"

s2 = "este"

Variáveis adicionais:

i = 3

j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5

Entrada:

s1 = "Este é um teste"

s2 = "este"

Variáveis adicionais:

i = 3

j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 4

Entrada:

s1 = "Este é um teste"

s2 = "este"

Variáveis adicionais:

i = 3

j = 1

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 4 5

Entrada:

s1 = "Este é um teste"
s2 = "este"

Variáveis adicionais:

i = 3
j = 1

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa



```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 4 5 6

Entrada:

s1 = "Este é um teste"
s2 = "este"

Variáveis adicionais:

i = 3
j = 1

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

```
1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 4 5 6 7
```

Entrada:

s1 = "Este é um teste"
s2 = "este"

Variáveis adicionais:

i = 3
j = 1

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {  
2     int i,j;  
3     for(i=0; s1[i]!='\0'; i++) {  
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)  
5             if (s2[j] != s1[i+j])  
6                 break;  
7         if (s2[j]=='\0')  
8             return 1;  
9     }  
10    return 0;  
11 }
```

Ordem de execução:

1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 4 5 6 7 3

Entrada:

s1 = "Este é um teste"
s2 = "este"

Variáveis adicionais:

i = 4
j = 1

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 4 5 6 7 3 4

Entrada:

s1 = "Este é um teste"
s2 = "este"

Variáveis adicionais:

i = 4
j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 4 5 6 7 3 4 5

Entrada:

s1 = "Este é um teste"
s2 = "este"

Variáveis adicionais:

i = 4
j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 4 5 6 7 3 4 5 6

Entrada:

s1 = "Este é um teste"
s2 = "este"

Variáveis adicionais:

i = 4
j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 4 5 6 7 3 4 5 6
7

Entrada:

s1 = "Este é um teste"

s2 = "este"

Variáveis adicionais:

i = 4

j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {  
2     int i,j;  
3     for(i=0; s1[i]!='\0'; i++) {  
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)  
5             if (s2[j] != s1[i+j])  
6                 break;  
7         if (s2[j]=='\0')  
8             return 1;  
9     }  
10    return 0;  
11 }
```

Ordem de execução:

1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 4 5 6 7 3 4 5 6
7 3

Entrada:

s1 = "Este é um teste"
s2 = "este"

Variáveis adicionais:

i = 5
j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 4 5 6 7 3 4 5 6
7 3 4

Entrada:

s1 = "Este é um teste"

s2 = "este"

Variáveis adicionais:

i = 5

j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 4 5 6 7 3 4 5 6
7 3 4 5

Entrada:

s1 = "Este é um teste"
s2 = "este"

Variáveis adicionais:

i = 5
j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa



```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

```
1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 4 5 6 7 3 4 5 6
7 3 4 5 6
```

Entrada:

s1 = "Este é um teste"
s2 = "este"

Variáveis adicionais:

i = 5
j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa



```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 4 5 6 7 3 4 5 6
7 3 4 5 6 7

Entrada:

s1 = "Este é um teste"

s2 = "este"

Variáveis adicionais:

i = 5

j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

```
1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 4 5 6 7 3 4 5 6
7 3 4 5 6 7 3
```

Entrada:

s1 = "Este é um teste"
s2 = "este"

Variáveis adicionais:

i = 6
j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

```
1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 4 5 6 7 3 4 5 6
7 3 4 5 6 7 3 4
```

Entrada:

s1 = "Este é um teste"

s2 = "este"

Variáveis adicionais:

i = 6

j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

```
1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 4 5 6 7 3 4 5 6
7 3 4 5 6 7 3 4 5
```

Entrada:

s1 = "Este é um teste"

s2 = "este"

Variáveis adicionais:

i = 6

j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 4 5 6 7 3 4 5 6
7 3 4 5 6 7 3 4 5
6

Entrada:

s1 = "Este é um teste"
s2 = "este"

Variáveis adicionais:

i = 6
j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 4 5 6 7 3 4 5 6
7 3 4 5 6 7 3 4 5
6 7

Entrada:

s1 = "Este é um teste"
s2 = "este"

Variáveis adicionais:

i = 6
j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 4 5 6 7 3 4 5 6
7 3 4 5 6 7 3 4 5
6 7 3

Entrada:

s1 = "Este é um teste"
s2 = "este"

Variáveis adicionais:

i = 7
j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 4 5 6 7 3 4 5 6
7 3 4 5 6 7 3 4 5
6 7 3 4

Entrada:

s1 = "Este é um teste"

s2 = "este"

Variáveis adicionais:

i = 7

j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 4 5 6 7 3 4 5 6
7 3 4 5 6 7 3 4 5
6 7 3 4 5

Entrada:

s1 = "Este é um teste"
s2 = "este"

Variáveis adicionais:

i = 7
j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 4 5 6 7 3 4 5 6
7 3 4 5 6 7 3 4 5
6 7 3 4 5 6

Entrada:

s1 = "Este é um teste"

s2 = "este"

Variáveis adicionais:

i = 7

j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

```
1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 4 5 6 7 3 4 5 6
7 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7
```

Entrada:

s1 = "Este é um teste"
s2 = "este"

Variáveis adicionais:

i = 7
j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 4 5 6 7 3 4 5 6
7 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3

Entrada:

s1 = "Este é um teste"
s2 = "este"

Variáveis adicionais:

i = 8
j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 4 5 6 7 3 4 5 6
7 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4

Entrada:

s1 = "Este é um teste"
s2 = "este"

Variáveis adicionais:

i = 8
j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 4 5 6 7 3 4 5 6
7 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5

Entrada:

s1 = "Este é um teste"

s2 = "este"

Variáveis adicionais:

i = 8

j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 4 5 6 7 3 4 5 6
7 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 6

Entrada:

s1 = "Este é um teste"
s2 = "este"

Variáveis adicionais:

i = 8
j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 4 5 6 7 3 4 5 6
7 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 6 7

Entrada:

s1 = "Este é um teste"
s2 = "este"

Variáveis adicionais:

i = 8
j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

```
1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 4 5 6 7 3 4 5 6
7 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 6 7 3
```

Entrada:

s1 = "Este é um teste"
s2 = "este"

Variáveis adicionais:

i = 9
j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 4 5 6 7 3 4 5 6
7 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 6 7 3 4

Entrada:

s1 = "Este é um teste"
s2 = "este"

Variáveis adicionais:

i = 9
j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

```
1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 4 5 6 7 3 4 5 6
7 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 6 7 3 4 5
```

Entrada:

s1 = "Este é um teste"
s2 = "este"

Variáveis adicionais:

i = 9
j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa



```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

```
1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 4 5 6 7 3 4 5 6
7 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 6 7 3 4 5 6
```

Entrada:

s1 = "Este é um teste"
s2 = "este"

Variáveis adicionais:

i = 9
j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

```
1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 4 5 6 7 3 4 5 6
7 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 6 7 3 4 5 6 7
```

Entrada:

s1 = "Este é um teste"
s2 = "este"

Variáveis adicionais:

i = 9
j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 4 5 6 7 3 4 5 6
7 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 6 7 3 4 5 6 7 3

Entrada:

s1 = "Este é um teste"
s2 = "este"

Variáveis adicionais:

i = 10
j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Entrada:

s1 = "Este é um teste"
s2 = "este"

Variáveis adicionais:

i = 10
j = 0

Ordem de execução:

1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 4 5 6 7 3 4 5 6
7 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 6 7 3 4 5 6 7 3
4

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

```
1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 4 5 6 7 3 4 5 6
7 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 6 7 3 4 5 6 7 3
4 5
```

Entrada:

s1 = "Este é um teste"
s2 = "este"

Variáveis adicionais:

i = 10
j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa



```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

```
1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 4 5 6 7 3 4 5 6
7 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 6 7 3 4 5 6 7 3
4 5 6
```

Entrada:

s1 = "Este é um teste"
s2 = "este"

Variáveis adicionais:

i = 10
j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

```
1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 4 5 6 7 3 4 5 6
7 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 6 7 3 4 5 6 7 3
4 5 6 7
```

Entrada:

s1 = "Este é um teste"
s2 = "este"

Variáveis adicionais:

i = 10
j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

```
1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 4 5 6 7 3 4 5 6
7 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 6 7 3 4 5 6 7 3
4 5 6 7 3
```

Entrada:

s1 = "Este é um teste"
s2 = "este"

Variáveis adicionais:

i = 11
j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa



```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 4 5 6 7 3 4 5 6
7 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 6 7 3 4 5 6 7 3
4 5 6 7 3 4

Entrada:

s1 = "Este é um teste"
s2 = "este"

Variáveis adicionais:

i = 11
j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

```
1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 4 5 6 7 3 4 5 6
7 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 6 7 3 4 5 6 7 3
4 5 6 7 3 4 5
```

Entrada:

s1 = "Este é um teste"
s2 = "este"

Variáveis adicionais:

i = 11
j = 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 4 5 6 7 3 4 5 6
7 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 6 7 3 4 5 6 7 3
4 5 6 7 3 4 5 4

Entrada:

s1 = "Este é um teste"
s2 = "este"

Variáveis adicionais:

i = 11
j = 1

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Entrada:

s1 = "Este é um teste"

s2 = "este"

Variáveis adicionais:

i = 11

j = 1

Ordem de execução:

1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 4 5 6 7 3 4 5 6
7 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 6 7 3 4 5 6 7 3
4 5 6 7 3 4 5 4 5

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa



```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Entrada:

s1 = "Este é um teste"
s2 = "este"

Variáveis adicionais:

i = 11
j = 2

Ordem de execução:

1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 4 5 6 7 3 4 5 6
7 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 6 7 3 4 5 6 7 3
4 5 6 7 3 4 5 4 5
4

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Entrada:

s1 = "Este é um teste"

s2 = "este"

Variáveis adicionais:

i = 11

j = 2

Ordem de execução:

1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 4 5 6 7 3 4 5 6
7 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 6 7 3 4 5 6 7 3
4 5 6 7 3 4 5 4 5
4 5

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

```
1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 4 5 6 7 3 4 5 6
7 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 6 7 3 4 5 6 7 3
4 5 6 7 3 4 5 4 5
4 5 4
```

Entrada:

s1 = "Este é um teste"
s2 = "este"

Variáveis adicionais:

i = 11
j = 3

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa



```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

```
1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 4 5 6 7 3 4 5 6
7 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 6 7 3 4 5 6 7 3
4 5 6 7 3 4 5 4 5
4 5 4 5
```

Entrada:

s1 = "Este é um teste"
s2 = "este"

Variáveis adicionais:

i = 11
j = 3

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

```
1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 4 5 6 7 3 4 5 6
7 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 6 7 3 4 5 6 7 3
4 5 6 7 3 4 5 4 5
4 5 4 5 4
```

Entrada:

s1 = "Este é um teste"
s2 = "este"

Variáveis adicionais:

i = 11
j = 4

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 4 5 6 7 3 4 5 6
7 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 6 7 3 4 5 6 7 3
4 5 6 7 3 4 5 4 5
4 5 4 5 4 7

Entrada:

s1 = "Este é um teste"
s2 = "este"

Variáveis adicionais:

i = 11
j = 4

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Teste de Mesa

```
1 int buscaString(char s1[], char s2[]) {
2     int i,j;
3     for(i=0; s1[i]!='\0'; i++) {
4         for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
5             if (s2[j] != s1[i+j])
6                 break;
7         if (s2[j]=='\0')
8             return 1;
9     }
10    return 0;
11 }
```

Ordem de execução:

```
1 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 4 5 6 7 3 4 5 6
7 3 4 5 6 7 3 4 5
6 7 3 4 5 6 7 3 4
5 6 7 3 4 5 6 7 3
4 5 6 7 3 4 5 4 5
4 5 4 5 4 7 8
```

Entrada:

s1 = "Este é um teste"
s2 = "este"

Variáveis adicionais:

i = 11
j = 4

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s1	E	s	t	e		é		u	m		t	e	s	t	e	\0

	0	1	2	3	4	5
s2	e	s	t	e	\0	

Exercício 2 – Completo



```
#include <stdio.h>
#include <string.h>

int buscaString(char s1[], char s2[]) {
    int i, j;
    for(i=0; s1[i]!='\0'; i++) {
        for(j=0; s2[j]!='\0' && s1[i+j]!='\0'; j++)
            if (s2[j] != s1[i+j])
                break;
        if (s2[j]=='\0')
            return 1;
    }
    return 0;
}
```

```
int main()
{
    char s1[100], s2[100];
    gets(s1);
    gets(s2);
    int res = buscaString(s1, s2);
    if(res)
        printf("Encontrou");
    else
        printf("Nao encontrou");
    return 0;
}
```

- 1) Fazer um programa para contar o número de espaços em brancos de uma *string*.
- 2) Refaça o programa anterior criando uma função que receberá como parâmetro a *string* e retornará o número de espaços em branco que a *string* contém.
- 3) Fazer um programa para contar o número de vogais numa cadeia de caractere.
- 4) Refaça o programa anterior criando uma função que receberá como parâmetro a *string* e retornará o número de vogais que a *string* contem.

- 5) Escrever um programa que leia uma *string* (com mais de uma palavra) e faça com que a primeira letra de cada palavra fique em maiúscula. Para isso, basta modificar cada letra através da expressão:

$$\text{chrNome}[0] = \text{chrNome}[0] - 'a' + 'A';$$

Exemplo:

Entrada: lab. de linguagem de programacao

Saída: Lab. De Linguagem De Programacao

- 6) Escreva uma função que receba uma *string*, conte quantos caracteres desta *string* são iguais a 'a' e substitua os que forem iguais a 'a' por 'b'. A função deverá retornar o número de caracteres modificados.

- 7) Crie uma função que receba uma frase e a exiba na tela de forma soletrada, ou seja, cada letra deve ser exibida na tela separada por hífen.
- 8) Crie uma função que receba uma *string* de no máximo 50 caracteres e inverta a ordem da *string* digitada;
Exemplo:
Entrada: Teste
Saída: etseT
- 9) Fazer um programa para criar e imprimir uma *string* que será a concatenação de duas outras *strings* lidas.

Vetores de caracteres

DCC120



- Em C, **strings** são cadeias de caracter terminadas, **obrigatoriamente**, pelo caractere nulo: `'\0'` (\zero). Portanto, deve-se reservar uma posição para este caractere que marca o fim da *string*.
- Exemplos:

```
char str[10] = {'a', 'b', 'c', '\0'};  
char uni[5];  
uni[0] = 'U';  
uni[1] = 'F';  
uni[2] = 'J';  
uni[3] = 'F';  
uni[4] = '\0';  
char disc[] = "Programacao";
```

- Leitura e impressão
 - **scanf** e **printf** com especificador **%s**
 - Ou com as funções de strings **gets** e **puts**

```
int main()  
{  
    char s[20];  
    printf("Digite uma string: ");  
    gets(s);  
    printf("String digitada: ");  
    puts(s);  
    return 0;  
}
```

- **gets(s)**: lê a string **s** a partir do teclado;
- **puts(s)**: imprime uma string na tela seguida de nova linha.

- Uso de *string* como parâmetro de função

```
int conta( char str[], int tam,
           char procurado )
{
    int cont, i;
    i = 0;
    cont = 0;
    while ( i < tam )
    {
        if ( str[i] == procurado )
        {
            cont++;
        }
        i++;
    }
    return cont;
}
```

```
#include <stdio.h>
#include <string.h>
int main()
{
    char nome[] = "UFJF";
    int total;
    int tam;
    tam = strlen(nome);
    total = conta(nome, tam, 'F');
    return 0;
}
```


- 1) Fazer uma função para imprimir uma string recebida como parâmetro sem os espaços em branco. Para isso, a string não deve ser modificada.
- 2) Fazer uma função que leia uma string do teclado (máx. 50 caracteres) e imprima uma “estatística” dos caracteres digitados. Isto é, imprima a quantidade de vogais, a quantidade de consoantes e a quantidade de outros caracteres.
- 3) Fazer um programa para ler uma string e transferir as consoantes para um vetor e as vogais para outro. Ao final, imprima cada um dos vetores.

- 4) Faça uma função que receba uma string do usuário (máx. 20 caracteres) e um caractere qualquer. A função deverá remover todas as ocorrências do caractere da string e retornar o número de remoções.
- 5) Escreva uma função que receba uma cadeia de caracteres de tamanho máximo 100, e retornar 1 se esta cadeia é *palíndrome* e zero caso contrário. Uma palavra é dita ser palíndrome se a seqüência de seus caracteres da esquerda para a direita é igual a seqüência de seus caracteres da direita para a esquerda. Ex.: **arara, asa, ovo...**

- 6) Um dos sistemas de encriptação mais antigos é atribuído a Júlio César: se uma letra a ser criptografada é a letra de número N do alfabeto, substitua-a com a letra $(N+K)$, onde K é um número inteiro constante (César utilizava $K = 3$).

Dessa forma, para $K = 1$ a mensagem

`"Adoro programar em C"`

se torna

`"Bepsp! qsphsbnbs! fn! D".`

Faça um programa que receba como entrada uma mensagem e um valor de K e altere a mensagem criptografando-a pelo código de César.

Exercícios – Biblioteca String



DESAFIO) Montar uma biblioteca com funções para manipular strings.

Ideia: Uma biblioteca é um conjunto de funções e tipos de dados que você pode “incluir” no seu programa sem a necessidade de digitar novamente o código.

Exemplo:

```
#include <stdio.h> // biblioteca do sistema que contém  
                  // a implementação de printf, scanf,  
                  // entre outras.
```

Podemos criar nossas próprias bibliotecas!

Exercícios – Biblioteca String



Como fazer:

- 1) No seu Projeto, crie e inclua um novo arquivo chamado “**biblio.h**”, onde iremos colocar somente os protótipos das subrotinas da nossa biblioteca.
- 2) Crie e inclua no mesmo Projeto o arquivo “**biblio.c**”, onde iremos colocar a implementação das subrotinas prototipadas no arquivo biblio.h
- 3) No arquivo main.c do seu projeto, basta fazer referência à biblioteca criada (**#include “biblio.h”**). Observe que você não utilizou os sinais de maior e menor para indicar a biblioteca, mas sim aspas. Isso indica que a biblioteca está no mesmo diretório do projeto.

Exercícios – Biblioteca String



Agora, construa sua biblioteca que forneça os principais recursos para a manipulação de strings. Sendo as funcionalidades propostas:

- **Copia():** copia uma string para outra
- **Comprimento():** retorna o comprimento da string
- **Iguais():** verifica se duas strings são iguais
- **Minusculo():** transforma os caracteres em minúsculo
- **Maiusculo():** transforma os caracteres em maiúsculo
- **Convertelnteiro():** se a string é composta apenas por dígitos, retorna o seu valor. Se não for, retorna -1.
- **InicialMaiuscula():** passa o caractere inicial de cada palavra para maiúsculo
- **Concatena():** acrescenta uma string no final da outra, se houver espaço
- **ProcuraTrecho():** procura uma string em outra
- **ProcuraCaracter():** procura a ocorrência de caracteres na string