



# STRINGS

Uma **string** é um **vector de char**  
terminado por `'\0'`.

Uma **string** é um **vector de char**  
terminado por `'\0'`.

"Ada Lovelace"

'A'	'd'	'a'	' '	'L'	'o'	'v'	'e'	'l'	'a'	'c'	'e'	'\0'
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	------



Hex	Value														
00	NUL	10	DLE	20	SP	30	0	40	@	50	P	60	`	70	p
01	SOH	11	DC1	21	!	31	1	41	A	51	Q	61	a	71	q
02	STX	12	DC2	22	"	32	2	42	B	52	R	62	b	72	r
03	ETX	13	DC3	23	#	33	3	43	C	53	S	63	c	73	s
04	EOT	14	DC4	24	\$	34	4	44	D	54	T	64	d	74	t
05	ENQ	15	NAK	25	%	35	5	45	E	55	U	65	e	75	u

string -> char vector\_de\_chars[tamanho]

literal de string

"uma string"

literal de char

'c'

não confundir

string -> char vector\_de\_chars[tamanho]

literal de string

"uma string"

literal de char

'c'

não confundir

```
char nome[20] = "Andre";
```

```
char nome[20] = { 'A', 'n', 'd', 'r', 'e', '\0' };
```

```
char nome[] = "Andre";
```

```
char *nome = "Andre"; //vão perceber quando  
//aprenderem apontadores
```

strings são obrigatoriamente terminadas em `'\0'`;  
vetores de `char` não

```
int i;  
char nome[] = "Tesla";  
  
for (i=0; i<6; i++)  
    printf("%c - %d\n", nome[i], nome[i]);
```

ASCII control characters			
DEC	HEX	Simbolo ASCII	
00	00h	NULL	(carácter nulo)
01	01h	SOH	(inicio encabezado)
02	02h	STX	(inicio texto)
03	03h	ETX	(fin de texto)
04	04h	EOT	(fin transmisión)

O que irá aparecer na  
consola?

strings são obrigatoriamente terminadas em `'\0'`;  
vetores de char não

ASCII control characters			
DEC	HEX	Símbolo ASCII	
00	00h	NULL	(carácter nulo)
01	01h	SOH	(início encabeçado)
02	02h	STX	(início texto)
03	03h	ETX	(fin de texto)
04	04h	EOT	(fin transmisión)

```
int i;  
char nome[] = "Tesla";  
  
for (i=0; i<6; i++)  
    printf("%c - %d\n", nome[i], nome[i]);
```

```
T - 84  
e - 101  
s - 115  
l - 108  
a - 97  
- 0
```

<code>'T'</code>	<code>'e'</code>	<code>'s'</code>	<code>'l'</code>	<code>'a'</code>	<code>'\0'</code>
------------------	------------------	------------------	------------------	------------------	-------------------

## Leitura e escrita de strings

```
char nome[100] = "Thomas";  
//escrita  
printf ( "Hello World\n" )  
printf ( "O meu nome e : %s .\n" , nome)  
puts ( "Hello World" )  
//leitura  
scanf ("%s", nome) //Variável NÃO é precedida de um &.  
fgets(nome, 100, stdin)
```

**scanf** realiza apenas a leitura de uma única palavra.

**%s** é o descritor de string no **printf** e **scanf**.

strings são obrigatoriamente terminadas em `'\0'`;  
vetores de char não

```
char nome_completo[20] = "Nikola Tesla";  
  
printf("nome_completo = %s\n", nome_completo);  
nome_completo[6] = '\0';  
printf("nome_completo = %s\n", nome_completo);
```

'N'	'i'	'k'	'o'	'l'	'a'	' '	'T'	'e'	's'	'l'	'a'	'\0'
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	------

O que irá aparecer na  
consola?

strings são obrigatoriamente terminadas em `'\0'`;  
vetores de char não

```
char nome_completo[20] = "Nikola Tesla";  
  
printf("nome_completo = %s\n", nome_completo);  
nome_completo[6] = '\0';  
printf("nome_completo = %s\n", nome_completo);  
printf("nome_completo = %s\n", &nome_completo[7]);
```

'N'	'i'	'k'	'o'	'l'	'a'	' '	'T'	'e'	's'	'l'	'a'	'\0'
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	------

'N'	'i'	'k'	'o'	'l'	'a'	'\0'	'T'	'e'	's'	'l'	'a'	'\0'
-----	-----	-----	-----	-----	-----	------	-----	-----	-----	-----	-----	------

```
nome_completo = Nikola Tesla  
nome_completo = Nikola  
nome_completo = Tesla
```

## Funções que recebem strings

```
int funcaoABC ( char s[] ) { }  
int funcaoABC ( char *s ) { }
```

Igual a funções que recebem vectores.

Conseguimos saber quando a string acaba ao localizar o  
caracter `'\0'`.

Logo, não precisamos de passar outro argumento a indicar o  
tamanho do vector.

## Principais funções de manipulação de strings

```
int strlen ( char *s)
```

Devolve o tamanho da string.

```
char *strcpy(char *dest, const char *src)
```

Copia a string apontada por **src**, incluindo '\0' para a apontada por **dest**.

```
char *strcat ( char *dest, char *src)
```

Coloca a string **src** imediatamente a seguir ao final da string **dest**.

```
int strcmp ( char *s1, char *s2)
```

Compara duas strings.

```
#include <strings.h>
```

```
int strcasecmp(const char *s1, const char *s2);  
int strncasecmp(const char *s1, const char *s2, size_t n);  
char *index(const char *s, int c);  
char *rindex(const char *s, int c);
```

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

```
char *strcpy(char *dest, const char *src);  
char *strcat(char *dest, const char *src);  
char *strchr(const char *s, int c);  
int strcmp(const char *s1, const char *s2);  
int strcoll(const char *s1, const char *s2);  
char *strncpy(char *dest, const char *src);  
size_t strcspn(const char *s, const char *reject);  
char *strdup(const char *s);  
char *strfry(char *string);  
size_t strlen(const char *s);  
char *strncat(char *dest, const char *src, size_t n);  
int strncmp(const char *s1, const char *s2, size_t n);  
char *strncpy(char *dest, const char *src, size_t n);  
char *strpbrk(const char *s, const char *accept);  
char *strrchr(const char *s, int c);  
char *strsep(char **stringp, const char *delim);  
size_t strspn(const char *s, const char *accept);  
char *strstr(const char *haystack, const char *needle);  
char *strtok(char *s, const char *delim);  
size_t strxfrm(char *dest, const char *src, size_t n);
```

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

Biblioteca para manipulação de strings.

[man7.org/linux/man-pages/man3/string.3.html](http://man7.org/linux/man-pages/man3/string.3.html)

Implemente as seguintes funções (ver descrição nas man pages):

`strlen` - tamanho da string

`strcpy` – copiar string

`strcat` – concatenar 2 strings

`strcmp` – comparar 2 strings

Não é da biblioteca `string.h`:

`strcount` → Receber 2 strings e devolver quantas vezes a primeira está dentro da segunda.

Implemente as seguintes funções (ver descrição nas man pages):

- strchr - localizar char em string (1ª ocorrência)
- strrchr - localizar char em string (última ocorrência)
- strspn
- strcspn

Exercício da cifra de César, pedindo a mensagem ao utilizador.

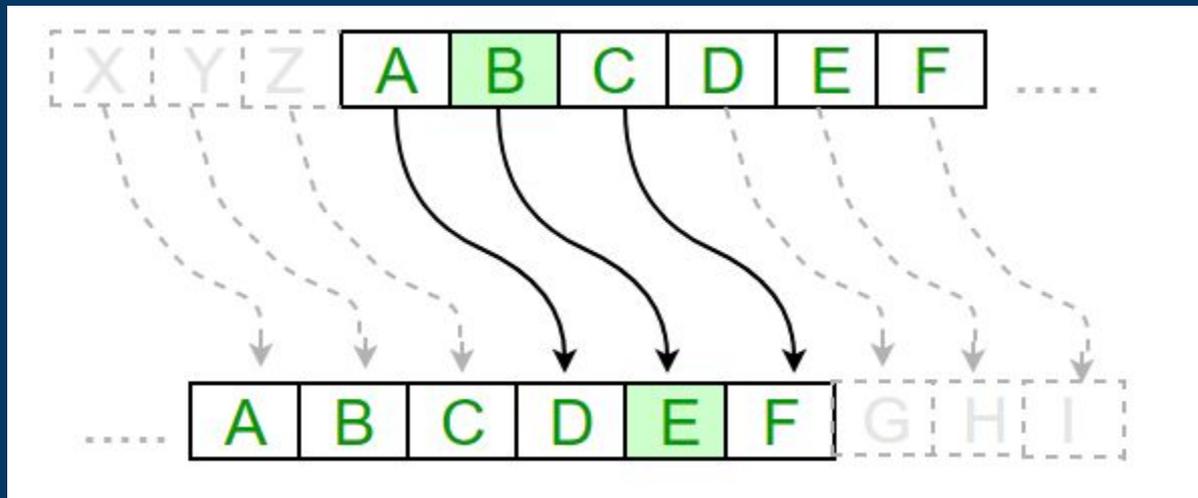
O programa pede uma string e uma cifra ao utilizador e depois permite cifrar ou decifrar.

Use `fgets` para receber a string. Assuma mensagem com tamanho máximo de 1000 caracteres.

Escreva uma função que recebe um char (letra a cifrar) e um inteiro (cifra) e faz a deslocação da letra no abecedário, apenas se os dados forem letras (números, espaços, etc. não alteram).

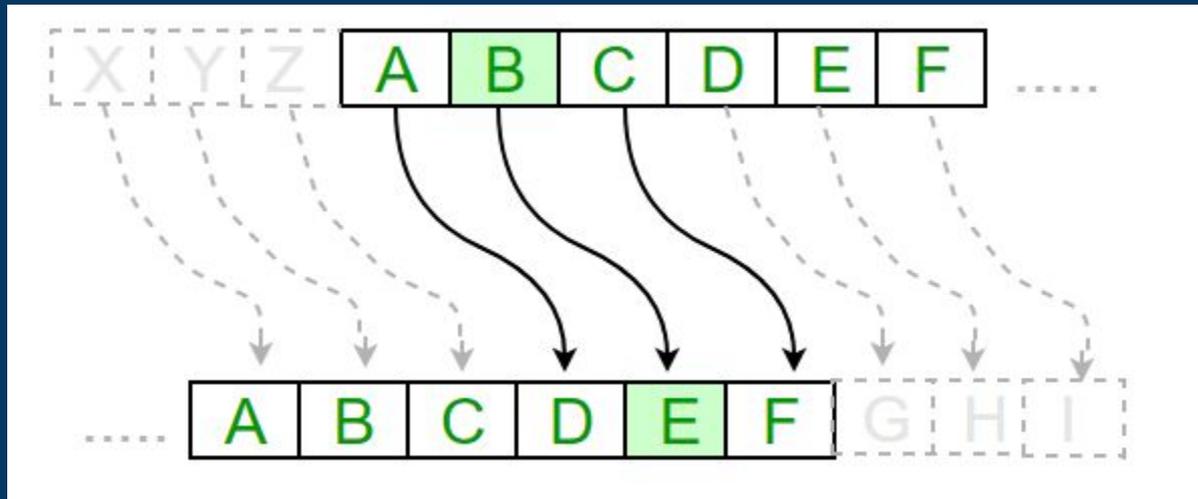
## Cifra de César

Desloca o abecedário uma unidade para a direita, ou para a esquerda se a cifra for negativa.



## Cifra de César

Desloca o abecedário uma unidade para a direita, ou para a esquerda se a cifra for negativa.



Atenção à passagem do Z->A e A->Z

Escreva uma função que recebe uma string (a cifrar), um inteiro (cifra) e um vetor de char onde guardar o resultado do processo de cifrar.

A função percorre a string, cifra cada um dos caracteres chamando a função descrita no slide anterior e guarda o resultado no vetor.

A *main* apenas recebe os dados do utilizador, chama a função para cifrar a mensagem e imprime na consola o resultado.

Crie um programa que implementa a cifra de Vigènere.

<https://goto.pachanka.org/crypto/vigenere-cipher>