

“STRINGS”



STRINGS

- ✖ Computador só pode guardar números.
- ✖ Como armazenar então os caracteres?
- ✖ Solução: códigos numéricos associados aos caracteres.
- ✖ Padronização através da tabela ASCII (American Standard Code for Information Interchange)

STRINGS

Exemplos:

<i>Caracter</i>	<i>Código ASCII</i>
A	65
B	66
a	97
b	98

TABELA ASCII (AMERICAN STANDARD CODE FOR INFORMATION INTERCHANGE)

Dec	Hex	Sinal	Caract.	Dec	Hex	Caract.	Dec	Hex	Caract.	Dec	Hex	Caract.
0	0	NULL		32	20	SPACE	64	40	@	96	60	`
1	1	SOH	␣	33	21	!	65	41	A	97	61	a
2	2	STX	␣	34	22	"	66	42	B	98	62	b
3	3	ETX	♥	35	23	#	67	43	C	99	63	c
4	4	EOT	♦	36	24	\$	68	44	D	100	64	d
5	5	ENQ	♣	37	25	%	69	45	E	101	65	e
6	6	ACK	♠	38	26	&	70	46	F	102	66	f
7	7	BEL		39	27	'	71	47	G	103	67	g
8	8	BS		40	28	<	72	48	H	104	68	h
9	9	HT		41	29	>	73	49	I	105	69	i
10	A	LF		42	2A	*	74	4A	J	106	6A	j
11	B	VT	␣	43	2B	+	75	4B	K	107	6B	k
12	C	FF	♀	44	2C	,	76	4C	L	108	6C	l
13	D	CR	␣	45	2D	-	77	4D	M	109	6D	m
14	E	SO	␣	46	2E	.	78	4E	N	110	6E	n
15	F	SI	✱	47	2F	/	79	4F	O	111	6F	o
16	10	DLE	▶	48	30	0	80	50	P	112	70	p
17	11	DC1	◀	49	31	1	81	51	Q	113	71	q
18	12	DC2	‡	50	32	2	82	52	R	114	72	r
19	13	DC3	!!	51	33	3	83	53	S	115	73	s
20	14	DC4	¶	52	34	4	84	54	T	116	74	t
21	15	NAK	§	53	35	5	85	55	U	117	75	u
22	16	SYN	≡	54	36	6	86	56	V	118	76	v
23	17	ETB	±	55	37	7	87	57	W	119	77	w
24	18	CAN	↑	56	38	8	88	58	X	120	78	x
25	19	EM	↓	57	39	9	89	59	Y	121	79	y
26	1A	SUB		58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	←	59	3B	;	91	5B	[123	7B	<
28	1C	FS	└	60	3C	<	92	5C	\	124	7C	!
29	1D	GS	↔	61	3D	=	93	5D]	125	7D	>
30	1E	RS	▲	62	3E	>	94	5E	^	126	7E	~
31	1F	US	▼	63	3F	?	95	5F	_	127	7F	Δ

Até o código 31 o uso é especial (comunicação por linha telefônica, rede e impressoras).

STRINGS

- ✖ A tabela ASCII padrão possui 128 itens (de 0 a 127).
- ✖ Computadores entendem somente 0's e 1's
- ✖ Conversão decimal para binário ($255_{10} = 11111111_2$)
- ✖ 1 character = 8 bits = 1 byte

STRINGS

- ✗ Lendo e imprimindo caracteres
 - + Quando uma tecla é digitada (lida pelo scanf) o código correspondente à tecla é traduzido para o número binário correspondente, e armazenado na variável utilizada no scanf.
 - + Quando uma variável caractere é utilizada no printf, esse número é utilizado para imprimir o caractere correspondente na tabela ASCII.

STRINGS – EXERCÍCIO 1

Faça uma **função** que mostre todos os caracteres e símbolos da tabela ASCII e seus respectivos códigos. Exemplo de saída:

```
Codigo: 36 → char: $
```


STRINGS – EXERCÍCIO 1

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

```
int main () {
```

```
    char c;
```

```
    printf("Usando codigo ASCII:\n");
```

```
    for (c = 0; c < 127; c++)
```

```
        printf("Codigo: %d --> char: %c \n", c, c);
```

```
}
```


STRINGS – EXERCÍCIO 1

```
#include<stdio.h>
#include<stdlib.h>
int main () {
    char c;
    printf("Usando codigo ASCII:\n");
    for (c = 0; c < 127; c++)
        printf("Codigo: %d --> char: %c \n", c, c);
    system("pause");
}
```

TABELA ASCII ESTENDIDA

128	Ç	144	É	161	í	177	⌠	193	⌞	209	⌞	225	β	241	±
129	û	145	æ	162	ó	178	⌡	194	⌟	210	⌟	226	Γ	242	≥
130	é	146	Æ	163	ú	179		195	⌠	211	⌠	227	π	243	≤
131	â	147	ô	164	ñ	180	⌡	196	—	212	⌡	228	Σ	244	∫
132	ä	148	ö	165	Ñ	181	⌢	197	⌢	213	⌢	229	σ	245	∫
133	à	149	ò	166	²	182	⌢	198	⌢	214	⌢	230	μ	246	÷
134	â	150	û	167	°	183	⌢	199	⌢	215	⌢	231	τ	247	≈
135	ç	151	ù	168	¿	184	⌢	200	⌢	216	⌢	232	Φ	248	°
136	ê	152	—	169	—	185	⌢	201	⌢	217	⌢	233	⊙	249	·
137	ë	153	Ö	170	¬	186	⌢	202	⌢	218	⌢	234	Ω	250	·
138	è	154	Û	171	½	187	⌢	203	⌢	219	■	235	δ	251	√
139	í	156	£	172	¼	188	⌢	204	⌢	220	■	236	∞	252	—
140	î	157	¥	173	¡	189	⌢	205	=	221	■	237	φ	253	²
141	ï	158	—	174	«	190	⌢	206	⌢	222	■	238	ε	254	■
142	Ä	159	f	175	»	191	⌢	207	⌢	223	■	239	∧	255	
143	Å	160	á	176	⌠	192	⌢	208	⌢	224	α	240	≡		

Source: www.LookupTables.com

O TIPO **STRING**

- + É diferente comparando-se a int e float.
- + Está sempre delimitado por aspas (“eu sou uma string”).
- + É composto de partes menores, **os caracteres**.
- + Por este motivo é considerado um **tipo de dado composto**.
- + Pode ser tratado como um **único componente** ou **acessar caracteres individualmente**.
- + Uma String pode ser formada a partir da **concatenação** de outras partes de strings.

STRINGS

Conteúdo de var	o	l	á	\0
Índice	0	1	2	3



```
var[0] = 'o'
```

```
var[1] = 'l'
```

```
var[2] = 'á'
```

'\0' indica o final da string.

Uma string definida com um tamanho de 50 caracteres, apenas 49 estarão disponíveis para armazenar o texto digitado pelo usuário.

O valor entre [] indica qual letra você quer obter. Pode ser uma variável ou expressão.

STRINGS

Caracteres podem ser alterados individualmente.

```
char msg[20] = "Hello, world!";  
msg[0] = 'J';
```

LENDO STRINGS

```
char str[20];  
scanf("%s", str);
```

Quando usamos a função **scanf()** para ler uma string, o símbolo & antes do nome da variável não é utilizado.

```
char str[20];  
gets(str);
```

Os colchetes também não são utilizados pois queremos ler a string toda e não apenas uma letra.

Função específica para **ler uma string**.

Ela armazena todos os caracteres lidos até que o enter seja digitado.

COPIANDO STRINGS

- ✗ A linguagem C não suporta a atribuição de um *array* para outro. Para atribuir o conteúdo de uma string a outra, o correto é copiar a string elemento por elemento para a outra string.

```
int main () { //copia string
    int cont;
    char str1[20] = "Ola Mundo.", str2[20];
    for(cont=0; str1[cont] != '\0'; cont++)
        str2[cont] = str1[cont];
    str2[cont] = '\0';
    printf("\nStr2: "); puts(str2);
}
```

FUNÇÕES DE **STRINGS**

Comprimento de uma String.

Função **strlen()**

```
char str1[20] = "Ola Mundo.";
int tamanho = strlen(str2);
printf("\nTamanho da str2: %d\n", tamanho);
```


STRINGS – EXERCÍCIO 2

Escreva uma **função** que mostre cada caracter de uma String, seus respectivos índices e códigos ASCII.

Exemplo de saída:

Character: O,	Indice: 0,	ASCII: 79
Character: L,	Indice: 1,	ASCII: 76
Character: A,	Indice: 2,	ASCII: 65

STRINGS – EXERCÍCIO 2

Escreva uma **função** que mostre cada caracter de uma String, seus respectivos índices e códigos ASCII.

```
void mostraChars(char str[]){  
    int i, tamanho = strlen(str);  
    for (i = 0; i < tamanho; i++){  
        printf("\nCaracter: %c, \tIndice: %d", str[i], i);  
        printf("\tCodigo ASCII: %d \n", str[i]);  
    }  
}
```

STRINGS – EXERCÍCIO 2

Escreva uma **função** que mostre cada caracter de uma String, seus respectivos índices e códigos ASCII.

```
void mostraChars(char str[]){  
    int i, tamanho = strlen(str);  
    for (i = 0; i < tamanho; i++){  
        printf("\nCaracter: %c, \tIndice: %d", str[i], i);  
        printf("\tCodigo ASCII: %d \n", str[i]);  
    }  
}
```

```
int main(){  
    char s[10] = "OLA";  
    mostraChars(s);  
}
```

STRINGS – EXERCÍCIO 3

Faça um algoritmo que receba uma string e um caracter, percorra esta String a procura do caracter e retorne o seu índice ou a mensagem “caracter não encontrado”.

STRINGS – EXERCÍCIO 3

Faça uma **função** que receba uma string e um character, percorra esta String a procura do character e informe o seu índice ou a mensagem “character não encontrado”.

```
int main(){
    char s[20], c;
    int i=0;
    printf("Digite uma string: ");   gets(s);
    printf("\nDigite o character para procurar: ");
    scanf("%c", &c);
    buscaChar(s,c);
}
```

STRINGS – EXERCÍCIO 3

```
void buscaChar(char str[], char ch){
    int i=0;
    while(i < strlen(str) && str[i]!=ch)
        i++;
    if (str[i]!=ch)
        printf("\n\tCaracter nao encontrado.");
    else
        printf("\nA primeira ocorrencia eh na posicao: %d\n", i);
}
```

```
int main(){
    char s[20], c;
    int i=0;
    printf("Digite uma string: ");   gets(s);
    printf("\nDigite o caracter para procurar: ");
    scanf("%c", &c);
    buscaChar(s,c);
}
```

FUNÇÕES DE STRINGS

- ✖ A operação de **concatenação** consiste em copiar uma string para o final de outra string.
- ✖ Em C, para concatenar duas strings usa-se a função:
strcat(char *destino, char *origem)

```
char str1 [15] = "bom";  
char str2 [15] = "dia";  
strcat (str1, str2);  
printf( "%s", str1);
```

ANTES

Conteúdo de str1	b	o	m	\0
Índice	0	1	2	3

Conteúdo de str2	d	i	a	\0
Índice	0	1	2	3

DEPOIS

Conteúdo de str1	b	o	m	d	i	a	\0
Índice	0	1	2	3	4	5	6

Conteúdo de str2	d	i	a	\0
Índice	0	1	2	3

FUNÇÕES DE STRINGS

- ✗ O operador relacional '==' não funciona para comparar duas strings.
- ✗ Para isso pode-se utilizar a função **strcmp()** que compara posição a posição as duas strings e retorna:
 - + um valor inteiro igual a zero no caso das duas strings serem iguais, ou
 - + um valor diferente de zero quando as strings são diferentes.

`strcmp(str1, str2);`

`strcmp()` é case-sensitive.

Isso significa que letras maiúsculas e minúsculas tornam as strings diferentes.

STRINGS – EXERCÍCIO 4

Faça um algoritmo que receba duas strings e verifique se as duas são iguais apresentando como saída se elas são iguais ou diferentes.

STRINGS – EXERCÍCIO 4

```
int main (void) {  
    char str1[100], str2[100];  
    printf("Entre com uma string: ");      gets(str1);  
    printf("\nEntre com outra string: ");  gets(str2);  
    if(strcmp(str1,str2) == 0)  
        printf("\nStrings iguais");  
    else printf("\nStrings diferentes");  
}
```

STRINGS – EXERCÍCIO 5

Escreva uma **função** que receba uma string que contenha uma frase terminada por um ponto final. Mostre o número de palavras que existem na frase.

STRINGS – EXERCÍCIO 5

```
int main (void) {  
    char str1[100];  
    printf("Entre com uma frase terminada com ponto final: ");  
    gets(str1);  
    printf("\nNumero de palavras da frase: %d\n", contapalavras(str1));  
}
```

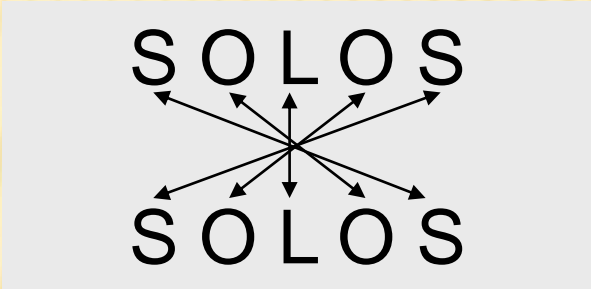

STRINGS – EXERCÍCIO 5

```
int contapalavras(char str[]){
    int i, tamanho, cont;
    tamanho=strlen(str);
    if(tamanho==0) cont=0;
    else cont = 1;
    for(i=0;i<tamanho;i++)
        if (str[i]==' ')
            cont++;
    return cont;
}

int main (void) {
    char str1[100];
    printf("Entre com uma frase: ");
    gets(str1);
    printf("\nNumero de palavras da frase: %d\n", contapalavras(str1));
}
```

STRINGS – EXERCÍCIO 6

Escreva uma **função** que leia strings e escreva se é uma palavra palíndroma. Uma string palíndroma é aquela que tem o mesmo resultado se lida da esquerda para a direita ou da direita para a esquerda. Exemplo: 'RIR', 'ASA', 'SOLOS', 'RALAR', 'AMA'.



The diagram shows the word "SOLOS" written twice, once above and once below a central point. Arrows connect the letters of the top word to the corresponding letters of the bottom word, demonstrating that the word reads the same forwards and backwards. The arrows are: a vertical arrow from the first 'S' to the first 'S', a diagonal arrow from the 'O' to the second 'S', a diagonal arrow from the 'L' to the 'O', a vertical arrow from the second 'O' to the 'L', a diagonal arrow from the 'S' to the 'O', and a diagonal arrow from the final 'S' to the first 'S'.

S O L O S
S O L O S

STRINGS – EXERCÍCIO 6

```
int main (void) {  
    char str1[100];  
    printf("Entre com uma palavra: ");    gets(str1);  
    palindroma(str1);  
}
```

STRINGS – EXERCÍCIO 6

```
void palindroma(char str[100]){ //verifica se str é palindroma
    int i, tam, palindroma=1;
    tam = strlen(str);
    for(i=0;i<tam/2;i++){
        if (str[i]!= str[tam-1-i]){
            palindroma=0;
            break;
        }
        tam--;
    }
    if (palindroma)
        printf("\nPalavra palindroma!");
    else printf("\nNao eh palindroma.");
}
```