


A decorative graphic on the left side of the slide. It consists of a blue parallelogram and a light green parallelogram, both tilted at an angle. The blue shape is in the foreground, and the green shape is partially behind it. The background of the entire slide is dark blue with faint, lighter blue diagonal stripes.

Strings

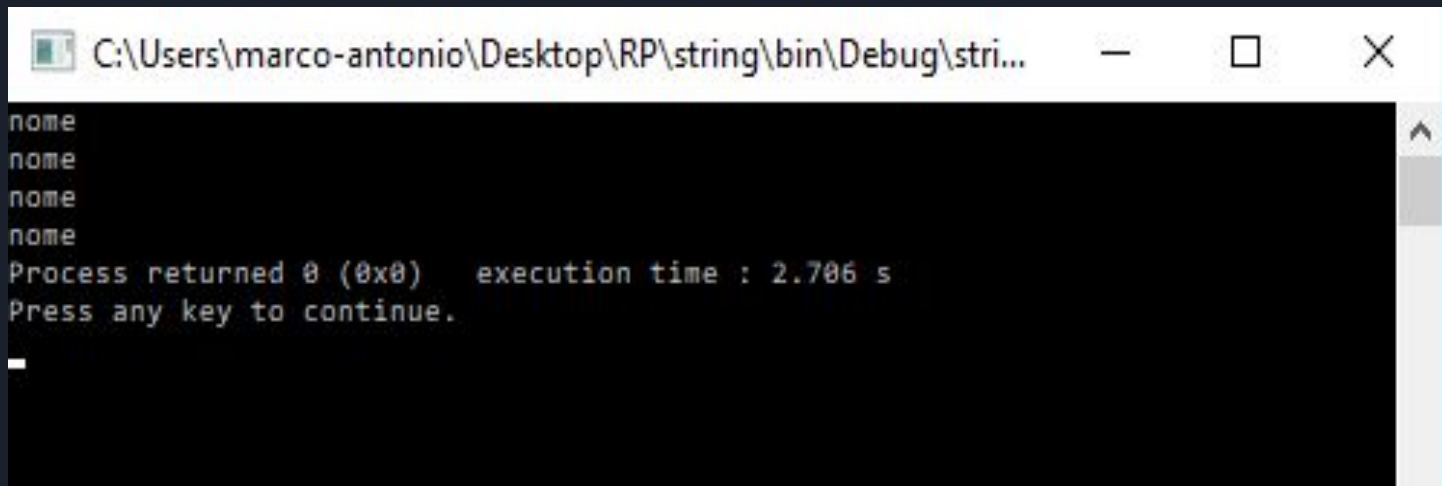


Uma string é uma seqüência de caracteres utilizada para o armazenamento de texto. Em C não existe um tipo string explícito, não existe uma palavra reservada que declare uma variável para armazenar uma string. Na linguagem C strings são representadas por vetores de caracteres que possuem um caracter que indica o término de seu conteúdo, o caracter nulo '\0'. Por serem representadas como vetores de caracteres, a elas se aplicam todas as propriedades e limitações de um vetor. Sendo assim, é possível alterar conteúdo da posições específicas, iterar sobre elas e etc.

Declaração

```
int main()  
{  
  
    char vetor[]="nome";  
    char vetor2[]={ "nome"};  
    char vetor3[5]="nome";  
    char vetor4[5]={'n','o','m','e','\0'};  
  
    printf("%s\n%s\n%s\n%s",vetor,vetor2,vetor3,vetor4);  
    return(0);  
}
```

Declaração - Execução



A screenshot of a Windows command prompt window. The title bar shows the file path: C:\Users\marco-antonio\Desktop\RP\string\bin\Debug\stri... The window has standard minimize, maximize, and close buttons. The command prompt displays the following text:

```
nome  
nome  
nome  
nome  
Process returned 0 (0x0)   execution time : 2.706 s  
Press any key to continue.  
_
```

Funções - Necessidade

Existem operações básicas de strings de outras linguagens que são INVÁLIDAS em C:

- Atribuição (separado da declaração)

```
char[30] curso;  
curso = "Bacharelado em Sistemas de Informacao";
```

- Comparação
string1 > string2

- Concatenação
string3 = string1 + string 2





Funções - Solução

Para a realização destas operações podem ser utilizadas as seguintes funções da biblioteca `string.h`:

Atribuição: `strcpy(destino , origem)`

Comparação: `strcmp(string1, string2)`

Concatenação: `strcat(string1, string2)`

* PARA INCLUSÃO DA BIBLIOTECA `string.h` UTILIZA-SE `include < string.h >` NO CABEÇALHO

- strlen(string)

```
1      #include <stdio.h>
2      #include <stdlib.h>
3
4      int main(){
5          /*A função strlen() descobre
6           o tamanho de uma string.*/
7          char vetor[20];
8          scanf("%s",&vetor);
9          int tamanho=strlen(vetor);
10         printf("%d", tamanho);
11
12         return 0;}
```

C:\Users\marco-antonio\Desktop\RP\teste1\bin\Debug\

UNIVERSIDADE

12

Process returned 0 (0x0) execution time : 4.591 s

Press any key to continue.

- strcpy(destino, origem)

```
1      #include <stdio.h>
2      #include <stdlib.h>
3
4      int main() {
5          /*A função strcpy() copia o conteúdo
6           da string de origem para o destino.*/
7          char origem[13];
8          char destino[13];
9          scanf("%s %s", origem, destino);
10         printf("DESTINO ORIGINAL: %s \n", destino);
11         strcpy(destino, origem);
12         printf("NOVO DESTINO: %s", destino);
13
14         return 0;}
15
```

C:\Users\marco-antonio\Desktop\RP\teste1\bin\Debug\test...

```
UFU
UNIVERSIDADE
DESTINO ORIGINAL: UNIVERSIDADE
NOVO DESTINO: UFU
Process returned 0 (0x0)   execution time : 8.133 s
Press any key to continue.
```


- strncpy(destino, origem, n)

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main() {
5      /*A função strncpy() realiza a cópia do
6      conteúdo de uma variável para outra, porém
7      deve ser especificado o tamanho a ser copiado
8      também deve ser atribuído o '\0' na última
9      posição do vetor que está recebendo a string.*/
10     char origem[23];
11     char destino[10];
12     scanf("%s", &origem);
13     strncpy(destino, origem, 10);
14     destino[10] = '\0';
15     printf("%s", destino);
16
17     return 0; }
18
```

C:\Users\marco-antonio\Desktop\RP\teste1\bin\Debug\test...

RESOLUÇÃO DE PROBLEMAS

RESOLUÇÃO

Process returned 0 (0x0) execution time : 14.659 s

Press any key to continue.

- strcat(string1, string2)

```
int main()  
{  
    /* A função strcat(string1, string2)  
       realiza a concatenação de  
       string2 em string1  
    */  
    char string1[22] = "Resolucao ";  
    char string2[13] = "de Problemas";  
    strcat(string1, string2);  
  
    printf("%s\n", string1);  
}
```

C:\Users\marco-antonio\Desktop\RP\string\bin\Debug\stri...

Resolucao de Problemas

Process returned 0 (0x0) execution time : 0.173 s
Press any key to continue.

- `strncat(string1, string2, tamanho)`

```
{  
    /*A função strncat() realiza a  
    concatenação do conteúdo de uma variável  
    a outra, porém, deve ser especificado  
    o tamanho a ser concatenado.  
    Ambas devem ser strings.  
    */  
    char vetor[22]= "Resolucao ";  
    char vetor2[15]= "de Problemas";  
  
    strncat(vetor, vetor2, 12);  
    printf ("%s\n", vetor);  
}
```

C:\Users\marco-antonio\Desktop\RP\string\bin\Debug\stri...
Resolucao de Problemas
Process returned 0 (0x0) execution time : 0.064 s
Press any key to continue.

● Palíndromo

```
1  #include <string.h>
2  #include <stdio.h>
3  int main(){
4      char vetor[6] = "ARARA";
5      char vetor2[6];
6      strcpy(vetor2, vetor);
7      strrev(vetor2);
8      if(strcmp(vetor, vetor2) == 0){
9          printf("%s EH PALINDROMO!", vetor);
10     } else {
11         printf("%s NAO EH PALINDROMO!", vetor);
12     }
13 }
14
```

"C:\Users\Tadeu Rodrigues\Documents\Curso\OPT-Resolucao\002-palindrome.exe"

ARARA EH PALINDROMO!

Process returned 0 (0x0) execution time : 0.404 s

Press any key to continue.

- strcmp(string1, string2)

```
#include <string.h>
#define TAMANHO 8

int main()
{
    /* A função strcmp(string1, string2) faz a comparação
       alfabética de duas strings;
       retornando: 1 se a primeira for maior, 0 se forem
       iguais, -1 se a segunda for maior.
    */
    char string1[TAMANHO] = "ABACATE";
    char string2[TAMANHO] = "ARARA";

    printf ("%d\n", strcmp(string1, string2));
}
```

C:\Users\marco-antonio\Desktop\RP\string\bin\Debug\stri...

-1

Process returned 0 (0x0) execution time : 0.189 s
Press any key to continue.

- `strncmp(string1, string2, tamanho)`

```
int main()  
{  
    /*A função strncmp() Também faz a  
    comparação do conteúdo de duas  
    strings, porém, deve ser  
    especificado o tamanho a ser comparado;  
    */  
    char vetor[5]= "MARIA";  
    char vetor2[5]= "MARIO";  
    int retorno;  
  
    retorno=strncmp(vetor, vetor2, 4);  
    printf ("RETORNO: %d\n", retorno);  
  
    retorno=strncmp(vetor, vetor2, 5);  
    printf ("RETORNO: %d\n", retorno);  
}
```

C:\Users\marco-antonio\Desktop\RP\string\bin\Debug\stri... —

RETORNO: 0
RETORNO: -14

Process returned 0 (0x0) execution time : 0.052 s
Press any key to continue.

- `strrev(string)`

```
int main()  
{  
    /*A função strrev() Inverte  
    a string s sobre ela mesma;  
    */  
    char vetor[10]= "RESOLUCAO";  
    printf ("%s\n",strrev(vetor));  
}
```

C:\Users\marco-antonio\Desktop\RP\string\bin\Debug\

OACULOSER

Process returned 0 (0x0) execution time : 0.742 s
Press any key to continue.

- strchr(string, char)

```
int main()  
{  
    /*A função strchr() considera  
    a String a partir do caracter informado.  
    Obs.: Valerá o primeiro caracter  
    que encontrar;  
    */  
    char vetor[10]= "RESOLUCAO";  
    printf ("%s\n", strchr(vetor, 'O'));
```

C:\Users\marco-antonio\Desktop\RP\string\bin\Debug\stri...

OLUCAO

Process returned 0 (0x0) execution time : 0.202 s
Press any key to continue.

- strstr(string, substring)

```
int main()
```

```
{
```

```
    /*A função strstr() é muito parecida com  
    a strchr, mas ela considera uma String a  
    partir de outra String informada;  
    */
```

```
    char vetor[10]= "RESOLUCAO";
```

```
    char v[3]="OL";
```

```
    printf ("%s\n", strstr(vetor,v));
```

```
}
```

C:\Users\marco-antonio\Desktop\RP\string\bin\Debug\stri

OLUCAO

Process returned 0 (0x0) execution time : 0.335 s
Press any key to continue.

- strtok(string, delimitador)

```
1  #include <string.h>
2  #include <stdio.h>
3  int main () {
4      /*A função strtok() divide a string, devolvendo a
5      cada chamada a substring até o delimitador*/
6      char vetor[21]="RESOLUCAO-60H-SABADO";
7      char *auxiliar;
8      auxiliar = strtok(vetor, "-");
9      while(auxiliar != NULL) {
10         printf("%s\n", auxiliar);
11         auxiliar = strtok(NULL, "-");
12     }
13 }
14
```

"C:\Users\Tadeu Rodrigues\Documents\Curso\OPT-Resoluc...

RESOLUCAO
60H
SABADO

Process returned 0 (0x0) execution time : 0.517 s
Press any key to continue.



Lidando com scanf

- Em determinados casos, é necessário garantir que a entrada possua somente um conjunto de caracteres ou que seu tamanho seja limitado, para isso utiliza-se expressões regulares:
- Para obtermos o resultado de uma entrada do teclado por meio do scanf precisamos utilizar as conversões: %s, %c, %lc, %ls, entre outras.

A sintaxe do scanf é a seguinte: `int scanf("[conversões de formato]", argumentos);`

- argumentos: ponteiros para o dado, separados por vírgula.
- Uma limitação de utilizar a conversão string %s é que ela lida com os espaços em branco como se fossem '\n'.



Validações com Expressões Regulares

Para limitarmos a quantidade de caracteres de uma `string1` já existente a 20 caracteres sem espaço:

```
scanf("%20s", string1);
```

Caso deseje se permitir caracteres em branco:

```
scanf("%20c", string1);
```

Mas não queremos que '\n' (quebra de linha) seja incluída em nossa `string1`, para isso utilizamos expressões regulares :

```
scanf("%20[^\n]\n", string1);
```



Validações com Expressões Regulares

- Há um tipo de conversão “%[“ que não pula espaços em branco, mas permite limitar a entrada a uma expressão regular.
- Por exemplo, digamos que seja necessário pagar um número de telefone (composto somente por caracteres numéricos):

```
scanf("%[123456789]", string1);
```

Ou somente caracteres alfabéticos:

```
scanf("%[A-z]", string1);
```

ASCII control characters

00	NULL	(Null character)
01	SOH	(Start of Header)
02	STX	(Start of Text)
03	ETX	(End of Text)
04	EOT	(End of Trans.)
05	ENQ	(Enquiry)
06	ACK	(Acknowledgement)
07	BEL	(Bell)
08	BS	(Backspace)
09	HT	(Horizontal Tab)
10	LF	(Line feed)
11	VT	(Vertical Tab)
12	FF	(Form feed)
13	CR	(Carriage return)
14	SO	(Shift Out)
15	SI	(Shift In)
16	DLE	(Data link escape)
17	DC1	(Device control 1)
18	DC2	(Device control 2)
19	DC3	(Device control 3)
20	DC4	(Device control 4)
21	NAK	(Negative acknowl.)
22	SYN	(Synchronous idle)
23	ETB	(End of trans. block)
24	CAN	(Cancel)
25	EM	(End of medium)
26	SUB	(Substitute)
27	ESC	(Escape)
28	FS	(File separator)
29	GS	(Group separator)
30	RS	(Record separator)
31	US	(Unit separator)
127	DEL	(Delete)

ASCII printable characters

32	space	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	\$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(72	H	104	h
41)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[123	{
60	<	92	\	124	
61	=	93]	125	}
62	>	94	^	126	~
63	?	95	_		

Extended ASCII characters

128	Ç	160	à	192	Ł	224	ó
129	ù	161	í	193	ł	225	ô
130	é	162	ó	194	ŧ	226	õ
131	â	163	û	195	ț	227	ö
132	ä	164	ñ	196	—	228	ø
133	à	165	Ñ	197	†	229	õ
134	â	166	ª	198	ā	230	μ
135	ç	167	º	199	Ä	231	þ
136	ê	168	¿	200	Ē	232	þ
137	ë	169	®	201	ƒ	233	ú
138	è	170	¬	202	±	234	û
139	ï	171	½	203	ƒ	235	ü
140	î	172	¾	204	ƒ	236	ý
141	ì	173	¿	205	=	237	ÿ
142	Ä	174	«	206	½	238	—
143	Å	175	»	207	¼	239	˘
144	É	176	¼	208	ð	240	=
145	æ	177	½	209	Ð	241	±
146	Æ	178	¾	210	É	242	¼
147	ø	179	¿	211	Ê	243	¾
148	ö	180	¿	212	Ë	244	¶
149	ö	181	À	213	Ì	245	§
150	ù	182	Á	214	Í	246	÷
151	ú	183	Â	215	Î	247	˘
152	ÿ	184	©	216	Ï	248	˘
153	Ö	185	ƒ	217	ƒ	249	˘
154	Ü	186		218	ƒ	250	˘
155	ø	187	ƒ	219	ƒ	251	˘
156	£	188	ƒ	220	ƒ	252	˘
157	Ø	189	€	221	ƒ	253	˘
158	×	190	¥	222	ƒ	254	■
159	f	191	ƒ	223	■	255	nbsp



Validações com Expressões Regulares

- Seria mais seguro então utilizar duas faixas de caracteres:

```
scanf("%[A-Za-z]", string1);
```



Mais sobre Expressões Regulares

character set	[ABC]
negated set	[^ABC]
range	[A-Z]
dot	.
word	\w
not word	\W
digit	\d
not digit	\D
whitespace	\s
not whitespace	\S

* referência completa em www.regexr.com



Tadeu Rodrigues dos Santos Braga - 11421BSI249

Gustavo Teixeira Perche Mahlow - 11421BSI225

Marco Antonio da Silva Rodrigues - 11421BSI201