5950257 - Programação de Computadores

Aula 09 - Strings

Prof. Dr. Denis M. L. Martins
DCM | FFCLRP | USP

Objetivos de Aprendizagem

- Compreender o conceito de strings em C
- Declarar e inicializar strings corretamente
- Utilizar funções básicas para manipulação de strings
- Percorrer strings utilizando estruturas de repetição
- Desenvolver algoritmos simples utilizando strings

> Para os exercícios desta aula, você pode utilizar IDEs online como:
>
> https://www.onlinegdb.com/online_c_compiler
>
> https://www.programiz.com/c-programming/online-compiler/

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

O que é uma String?

Uma string é:

- Uma sequência de caracteres
- Utilizada para armazenar textos
- Implementada em C como um vetor de caracteres

Em C, uma string é simplesmente um **array de caracteres terminados por `\0`**.

- O caractere nulo (`\0`, valor ASCII 0) marca o fim da string. Sem ele, a função **`printf`** ou **`strlen`** não sabe onde parar.

```c
char nome[] = "Joao";
// Na memória: 'J', 'o', 'a', 'o', '\0' (5 bytes)
```


Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Declaração e Inicialização

Formas de inicializar:

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

```
char nome[10] = "Carlos";
```

```
char nome[] = {'C','a','r','l','o','s','\0'};
```


Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Exibindo Strings

```c
#include <stdio.h>

int main() {
    char nome[] = "Ana";
    printf("%s", nome);

    return 0;
}
```

> O especificador `%s` é utilizado para imprimir strings.

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Leitura de Strings

Utilizando **scanf**:

```c
#include <stdio.h>

int main() {
    char nome[20];
    scanf("%s", nome);
    printf("%s", nome);

    return 0;
}
```

- **Problema**: a função **scanf** não lê espaços.
- Exemplo de entrada: **Maria Silva**. Resultado armazenado: **Maria**.

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Leitura segura de Strings

```c
#include <stdio.h>

int main() {
    char nome[20];
    fgets(nome, 20, stdin);
    printf("%s", nome);
    return 0;
}
```

fgets lê até o limite ou newline (**\n**).

- Mantém o **\n** na string (precisa remover se quiser).
- Lê espaços
- Evita overflow

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Percorrendo Strings

Podemos utilizar um loop **for** para percorrer strings:

```c
#include <stdio.h>

int main() {

    char texto[] = "Programação";

    for(int i = 0; texto[i] != '\0'; i++) {
        printf("%c\n", texto[i]);
    }

    return 0;
}
```

> **%c** é utilizado para imprimir caracteres individuais

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Percorrendo Strings (cont.)

Para percorrer strings utilizando um loop **while**, podemos utilizar como condição de parada o **\0** e, assim, encerrar o loop:

```c
#include <stdio.h>

int main() {
    char texto[] = "Radiacao";
    int contador = 0;

    while(texto[contador] != '\0') {
        contador++;
    }

    printf("Quantidade: %d", contador);

    return 0;
}
```


Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Exercício

Escreva um programa em C que:

- Leia uma frase qualquer do teclado
- Conte o número de letras "a" nesta frase
- Imprima o número de letras "a"

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

A Tabela ASCII

ASCII (American Standard Code for Information Interchange): Padrão para mapear caracteres em valores inteiros decimais.

Exemplos:

- O caractere 'A' é sempre representada pelo número 65 decimal (ou **01000001** binário).
- O caractere 'a' é 97.

ASCII Table

Dec	Hex	Char	Name	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	NUL	null	32	20	Space	64	40	@	96	60	`
1	1	SOH	start of heading	33	21	!	65	41	A	97	61	a
2	2	STX	start of text	34	22	"	66	42	B	98	62	b
3	3	ETX	end of text	35	23	#	67	43	C	99	63	c
4	4	EOT	end of transmission	36	24	$	68	44	D	100	64	d
5	5	ENQ	enquiry	37	25	%	69	45	E	101	65	e
6	6	ACK	acknowledge	38	26	&	70	46	F	102	66	f
7	7	BEL	bell	39	27	'	71	47	G	103	67	g
8	8	BS	backspace	40	28	(	72	48	H	104	68	h
9	9	HT	horizontal tab	41	29	)	73	49	I	105	69	i
10	A	LF	line feed	42	2A	*	74	4A	J	106	6A	j
11	B	VT	vertical tab	43	2B	+	75	4B	K	107	6B	k
12	C	FF	form feed	44	2C	,	76	4C	L	108	6C	l
13	D	CR	carriage return	45	2D	-	77	4D	M	109	6D	m
14	E	SO	shift out	46	2E	.	78	4E	N	110	6E	n
15	F	SI	shift in	47	2F	/	79	4F	O	111	6F	o
16	10	DLE	data link escape	48	30	0	80	50	P	112	70	p
17	11	DC1	device control 1	49	31	1	81	51	Q	113	71	q
18	12	DC2	device control 2	50	32	2	82	52	R	114	72	r
19	13	DC3	device control 3	51	33	3	83	53	S	115	73	s
20	14	DC4	device control 4	52	34	4	84	54	T	116	74	t
21	15	NAK	negative acknowledge	53	35	5	85	55	U	117	75	u
22	16	SYN	synchronous idle	54	36	6	86	56	V	118	76	v
23	17	ETB	end of transmission block	55	37	7	87	57	W	119	77	w
24	18	CAN	cancel	56	38	8	88	58	X	120	78	x
25	19	EM	end of medium	57	39	9	89	59	Y	121	79	y
26	1A	SUB	substitute	58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	escape	59	3B	;	91	5B	[	123	7B	{
28	1C	FS	file separator	60	3C	<	92	5C	\	124	7C	|
29	1D	GS	group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	RS	record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	US	unit separator	63	3F	?	95	5F	_	127	7F	DEL

From https://ajsmith.org/tools/ascii-table/

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Caracteres como `int`

Em C, um **`char`** *é* um tipo inteiro pequeno (**`int`**). Quando você faz aritmética com caracteres, o compilador trata eles como seus valores ASCII subjacentes.

```c
char letra = 'B';
printf("%d", letra); // Imprime: 66
char proximaLetra = letra + 3;
printf("\nA próxima é %c e tem valor %d)", proximaLetra, proximaLetra);
// Saída esperada: E (Valor: 69)
```


Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Manipulação de strings com string.h

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

Principais Funções:

- **strlen(s)**: Retorna o tamanho (sem contar **\0**).
- **strcpy(dest, src)**: Copia string. **Perigoso** se dest for menor que src.
- **strcat(dest, src)**: Concatena strings. Dest deve ter espaço suficiente.
- **strcmp(a, b)**: Compara duas strings (retorna 0 se iguais).

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Função strlen

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

int main() {

    char texto[] = "Tomografia";

    printf("%lu", strlen(texto));

    return 0;
}
```

Resultado:

```
10
```


Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Função strcpy

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

int main() {
    char origem[] = "MRI";
    char destino[20];

    strcpy(destino, origem);
    printf("%s", destino);

    return 0;
}
```

Resultado:

```
destino = "MRI"
```


Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Função strcat

```c
char a[30] = "Linguagem ";
char b[] = "C";

strcat(a, b);
```

Resultado:

```
"Fisica Medica"
```


Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Função `strcmp`

```
strcmp(a, b)
```

Resultado:

Valor	Significado
0	iguais
diferente de 0	diferentes

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Erros Comuns

Overflow de String: A string excede o tamanho do vetor.

```c
char nome[5];

scanf("%s", nome);
```

Entrada:

```
Linguagem
```

Esquecer o `'\0'`:

```c
char texto[3] = {'O', 'K'};
```


Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Vetores de Strings

Armazena os caracteres diretamente em um bloco contíguo na memória.

```c
#include <stdio.h>

int main() {
    char nomes[3][20] = {
        "Ana",
        "Bob",
        "Charlie"
    };
    printf("Tamanho total: %zu\n", sizeof(nomes));
    printf("%s\n", nomes[0]);

    return 0;
}
```

> Note que o tamanho das strings é fixo por linha. Isso desperdiça memória se strings forem curtas.

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Percorrendo uma matriz de strings

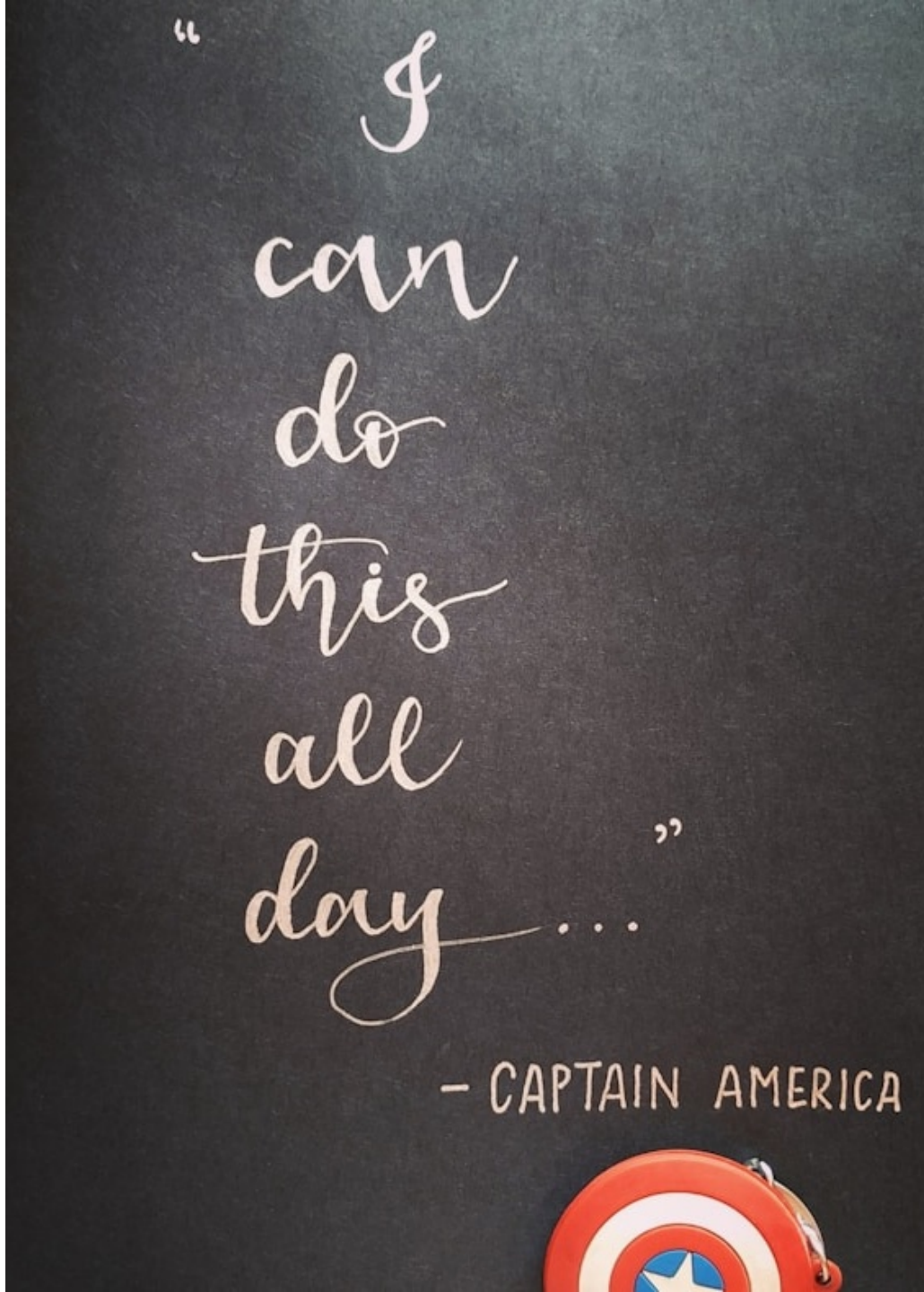
```c
#include <stdio.h>

int main() {
    char nomes[3][20] = {"Ana", "Bob", "Charlie"};
    for(int i = 0; i < 3; i++) {
        for(int j = 0; j < 20; j++) {
            printf("%c", nomes[i][j]); // Acessa caractere específico
            if(nomes[i][j] == '\0') {
                break; // Para no fim da string
            };
        }
        printf("\n");
    }
    return 0;
}
```

> **Dica:** Use `strlen` ou verifique `\0` para não imprimir lixo.

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Exercícios

1. **Crie um programa que:**
   - leia um texto
   - percorra todos os caracteres
   - exiba uma letra por linha
2. **Crie um programa que:**
   - leia nome e sobrenome
   - concatene os textos
   - exiba o nome completo



> Fonte da Imagem: Asmi Pai @Unsplash

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Conclusão

String: Array de `char` + `\0`.

Strings são fundamentais porque:

- permitem manipulação textual
- aparecem em praticamente todos os sistemas

Material adicional:

- Material de Aula do Prof. Carlos Maziero (UFPR)
- Video aula: Trabalhando Com Strings do Prof. Thiago Jabur

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Dúvidas e Discussão

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Exercício e Questões

Questões teóricas

1. Explique o papel do caractere `\0` em uma string.
2. Qual a diferença entre: `scanf` e `fgets()`?
3. Ao chamar `strcmp("B", "A")`, qual valor será retornado? Será positivo, negativo ou zero?

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Exercício 1

Crie um programa que leia uma string e exiba:

- quantidade de caracteres
- primeira letra
- última letra

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Exercício 2

Crie um programa que leia dois nomes de pacientes e informe:

- se os nomes são iguais
- qual possui maior quantidade de caracteres

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Exercício 3

Crie um programa que leia uma string e verifique se ela é um palíndromo.

Palíndromo é toda palavra ou frase que pode ser lida de trás para frente e que, independente da direção, mantém o seu sentido. Exemplos:

- ada
- ana
- missa é assim.
- o lobo ama o bolo.

> Note que na tabela ASCII letras maiúsculas e minúsculas possuem códigos diferentes.

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br