INSTITUTO DE INFORMÁTICA
Universidade Federal de Goiás

Introdução à Computação
Estrutura de Dados Composta - Unidimensional: Vetor de Caracteres

Prof. Ernesto Veiga
ernestoveiga@ufg.br

INSTITUTO DE INFORMÁTICA UFG

UFG UNIVERSIDADE FEDERAL DE GOIÁS

String - Vetor de Caracteres

Na prática, as **strings** são usadas para representar **textos**.

Em linguagem C, ao contrário de outras linguagens (ex.: Java, Python, etc.), **não existe um tipo de dados string nativo**.

Para representar uma string em C, devemos criar um **vetor de caracteres**, ou seja um **vetor de valores do tipo char**.

O byte nulo **‘\0’** é interpretado como uma sentinela que marca o fim da parte relevante do vetor

ASCII e as strings

Cadeias de caracteres são representadas por strings.

É preciso criar uma correspondência entre caracteres e bytes.

A correspondência mais antiga e mais básica é a tabela ASCII, que associa a cada caractere do alfabeto ASCII um byte entre 00000000 e 01111111

Por exemplo, a cadeia de caracteres ABC é representada por 65 66 67.

```
char a;
a = 65;
printf(“ %c”, a);
saída: A
```

Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
32	20	040		Space	64	40	100	@	@	96	60	140	`	`
33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Recapitulando os usos de vetor...

Declaração de vetor:

```
float vetor[100];
```

Atribuindo valores ao vetor:

```
vetor[0] = 20;
idade[5] = 22;
```

Preenchendo um vetor:

```
for(i=0; i<100; i++)
    scanf("%f", &vetor[i]);
```

Mostrando os elementos do vetor:

```
for(i=0; i<100; i++)
    printf("%f ", vetor[i]);
```

Como declarar uma string

Declaração de vetor:

```
float vetor[100];
int idade[20];
```

Declaração de string:

```
char nome[30];
```

Inicialização de uma string

Atribuindo valores ao vetor:

```
int vetor[0] = 20;
int idade[5] = {10, 20, 30, 40, 50};
```

Atribuindo valor à string:

```
char nome[30] = “Ernesto Fonseca Veiga”;
char nome[30] = {97,195,167,195,163,111,0};
```

Saída: Ernesto Fonseca Veiga

Saída: ação

Obs.: o par de bits 195 167 representa o ç e o par de bits 195 163 representa o ã

Preenchendo uma string

Preenchendo um vetor:

```
for(i=0; i<100; i++)
        scanf(“%f”, &vetor[i]);
```

Preenchendo uma string:

```
scanf ("%s", nome) ; // sem &
scanf("%[^\n]", nome);  // lê string com espaços (até achar '\n’)
fgets(nome, 29, stdin); //identificador, tamanho-1, stdin
```

Mostrando uma string

Mostrando os elementos do vetor:

```
for(i=0; i<100; i++)
        printf(“%f ”, vetor[i]);
```

Mostrando uma string:

```
for(count=0 ; count < 20 ; count++)
        printf("%c",curso[count]);

for (int i = 0; i < strlen(nome); i++)  // imprime o nome, um char por vez
        putchar (nome[i]) ;

printf(“ %s”, nome);
puts(nome);
```

Manipulação de Strings

Em C temos uma biblioteca <string.h> para manipular essa estrutura

int strlen(src): retorna o número de caracteres da string

char * strcpy (dest, orig): copia a string da orig para o dest

char * strncpy (dest, orig, n): copia n caracteres da orig para o destino

int strcmp (st1, st2): compara duas strings e retorna 0 se forem iguais

int strncmp (str1, str2 ,n): idem, mas compara apenas n caracteres

char * strcat (dest, src): concatena a string src ao final de dest

char * strncat (strto, strfrom, n): idem, mas concatena apenas n caracteres

INSTITUTO DE INFORMÁTICA
Universidade Federal de Goiás

Introdução à Computação
Estrutura de Dados Composta - Unidimensional: Vetor de Caracteres

Prof. Ernesto Veiga
ernestoveiga@ufg.br

INSTITUTO DE INFORMÁTICA UFG

UFG UNIVERSIDADE FEDERAL DE GOIÁS