Fundamentos de Programação

Strings

Prof. Luiz Fernando Carvalho

luizfcarvalho@utfpr.edu.br

Caractere

- Caracteres nada mais são que inteiros com um significado especial, codificados através da tabela ASCII;
- Funções para leitura de UM caractere:

```
getc(stdin)
getchar()
```

- Funções padrão
- Mostram o caractere na tela

- Não são funções padrão
- São da biblioteca `conio.h` (windows)
- `getch()` não mostra o caractere na tela
- Com `getch()` não é necessário pressionar a tecla ENTER

```
getch()
getche()
```

Caractere

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Strings

- String é uma cadeia ou sequência de caracteres (*char*);
- Em resumo: Em C, uma *string* é simplesmente um **vetor** de caracteres, com uma convenção especial: como o **valor zero** não é utilizado por nenhum caractere, ele é utilizado para **marcar o final de uma *string*,** ou seja, ele é o terminador da *string*.

Strings

- O “caractere” de código zero é comumente chamado de caractere nulo, e é representado pela sequência especial ‘\0’
- A declaração de uma *string* é igual a de um vetor convencional:

```
char palavra[4] = {‘O’, ‘L’, ‘A’, ‘\0’};
```

```
char palavra[] = {‘O’, ‘L’, ‘A’, ‘\0’};
```

- E pode ser abreviada para:

```
char palavra[] = “OLA”;
```

No último caso, o compilador se encarrega de atribuir o código de final da String

Aspas simples são usadas para caracteres, aspas duplas para Strings

Alterando Strings

- Na definição da *string* deve-se reservar uma posição do vetor para o símbolo ‘\0’
- Por exemplo, se a *string* for conter 10 caracteres, a *string* deverá ser formada por 11 posições

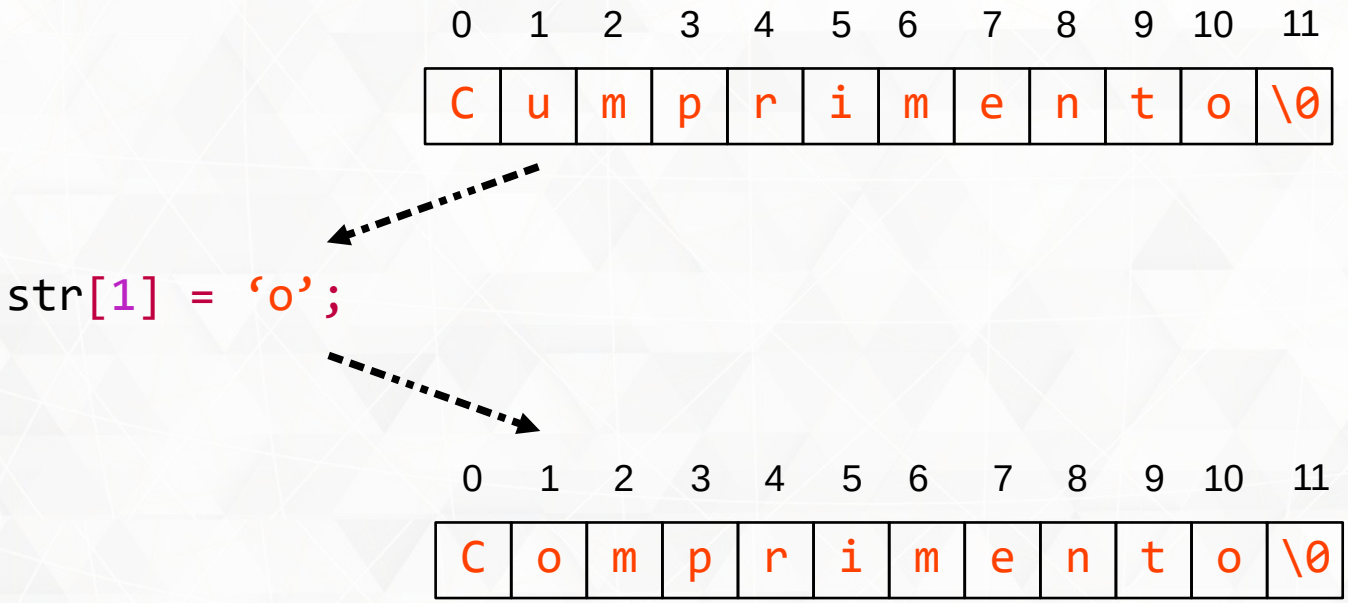
```
char palavra[11];
```

```
palavra[0] = ‘A’;
palavra[1] = ‘r’;
palavra[2] = ‘t’;
palavra[3] = ‘i’;
palavra[4] = ‘f’;
palavra[5] = ‘i’;
palavra[6] = ‘c’;
palavra[7] = ‘i’;
palavra[8] = ‘a’;
palavra[9] = ‘l’;
palavra[10] = ‘\0’;
```

Alterando Strings

- Como em um vetor convencional, se for preciso trocar o valor de um caractere em uma string, deve-se fornecer o seu índice.

```
char str[12] = “Cumprimento”;
```



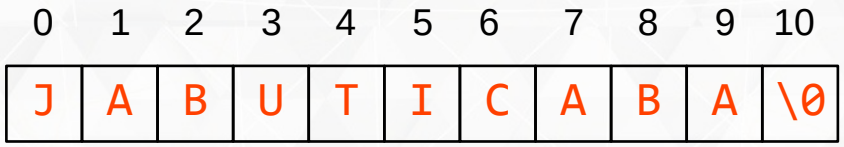
Alterando Strings

- Declarando a *string* como a seguir, ela sempre terá o tamanho fixo máximo de 11 caracteres, mais o símbolo ‘\0’:

```
char str[] = “Cumprimento”;
```

- A dica é sempre atribuir um tamanho de *String* maior do que se pretende utilizar

```
char str[11] = “ABACAXI”;
```



Tamanho da String

- A função `strlen(string)` – String Length
  - retorna um inteiro que indica a quantidade de caracteres;

```
#include<string.h>

int main(){
    char cor[] = “azul”;

    printf(“%d”, strlen(cor));

}
```

Obs.: O caractere terminador ‘\0’ não é contado

Copiando Strings

- O enfado de atribuir elemento por elemento pode ser eliminado graças à função *strcpy* (**STR**ing **C**o**PY**).
  - Copia o conteúdo de uma *string* para outra;
  - O símbolo de término de *string* também é copiado;
  - A string de destino deve poder guardar todos os caracteres da *string* original, mais o caractere terminador.

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

Biblioteca especial destinada à manipulação de strings

```
char mensagem[10];
strcpy(mensagem, “Bom dia!”);
```

```
char destino[10];
strcpy(destino, mensagem);
```

Comparando Strings

- Outra tarefa muito comum é descobrir se duas *strings* são iguais
  - Uma comparação do tipo s1 == s2 compara os endereços em que estão guardadas as Strings, **MAS NÃO O CONTEÚDO DELAS**;
  - A maneira correta é comparar as Strings elemento a elemento;
  - Usa-se a função *strcmp* (**STR**ing **C**o**MP**are).

```
#include<string.h>

int igual;
igual = strcmp(s1, s2);
```

Comparando Strings

```c
#include<string.h>

int igual;
igual = strcmp(s1, s2);
```

- Esta função começa a comparar o primeiro caractere de cada *string*
  - Se eles forem iguais entre si, continuará com os pares a seguir até que os caracteres sejam diferentes ou até que um caractere nulo de terminação seja atingido.
- A função `strcmp` retorna um valor inteiro:
  - **valor zero:** caso as Strings sejam iguais;
  - **valor < 0:** o primeiro caractere diferente entre as strings tem um valor maior na primeira *string* do que na segunda;
  - **valor > 0:** o primeiro caractere diferente entre as strings tem um valor maior na segunda *string* do que na primeira;

Comparando Strings

```c
#include<string.h>

int main(){
    char s1[] = “computacao”;
    char s2[] = “computacao”;
    printf(“%d”, strcmp(s1, s2));
    ...
}
```

0

```c
#include<string.h>

int main(){
    char s1[] = “comparacao”;
    char s2[] = “computacao”;
    printf(“%d”, strcmp(s1, s2));
    ...
}
```

-1

```c
#include<string.h>

int main(){
    char s1[] = “sistema”;
    char s2[] = “computacao”;
    printf(“%d”, strcmp(s1, s2));
    ...
}
```

1

Concatenando Strings

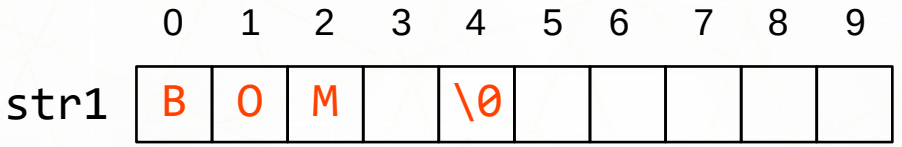
- A concatenação “junta” duas strings em uma única

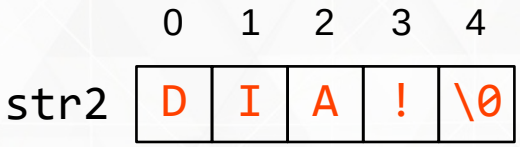
```
#include<string.h>

char str1[10];
char str2[5];

strcpy(str1, “BOM ”);
strcpy(str2, “DIA!”);

strcat(str1, str2);
```





`strcat(`*string_destino*`,` *string*`);`

A *string* destino deve ser grande o suficiente para armazenar ambas *strings* concatenadas

Saída: Imprimindo Strings

- Diferentemente dos vetores que contém números, as *strings* podem ser impressas de forma direta:

Caractere por caractere

```
char cor[20] = “vermelho”;
int i = 0;

while(cor[i] != ‘\0’)
{
      printf(“%c”, cor[i]);
      i++;
}
```

String toda de uma vez só

```
char cor[20] = “vermelho”;
printf(“%s \n”, cor);
```

Equivale à:

```
puts(cor);
```

Função puts (put string) imprime a String e pula uma linha automaticamente

Entrada: recebendo String

- Função `scanf()`:
  - Lê uma sequência de caracteres até um espaço, tabulação, etc.
- Função `gets()`:
  - Lê uma sequência de caracteres até a tecla ENTER ser pressionada

Usando a função scanf

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

```
char cor[30];
scanf("%s", cor);
```

Usando a função gets

```
gets(destino);
```

```
char cor[30];
gets(cor);
```

Evitar o uso

Entrada: recebendo String

- O recebimento de Strings é algo que requer CUIDADO!
- O programador PODE/DEVE fornecer um total de caracteres que deverá ser lido

Usando a função scanf

```
scanf("%ns", destino);
```

```
char cor[30];
scanf("%29s", cor);
```

Quantidade de caracteres para ler

Usando a função gets

```
fgets(destino, n, stdin);
```

```
char cor[30];
fgets(cor, 29, stdin);
```

Quantidade de caracteres para ler

Os caracteres serão lidos do *standard input*

Evita *overflow*

Problema com o fgets

- O `fgets` garante o tamanho correto, mas salva o ‘\n’ final;
- Solução: usar a função `strcspn`
  - Localizar a posição do ‘\n’ e escrever ‘\0’ no local

```
char nome[30];

printf(“Forneca o seu nome: ”);
fgets(nome, 29, stdin);
nome[strcspn(nome, “\n”)] = ‘\0’;
setbuf(stdin, NULL);

printf(“O nome informado e’: %s”, nome);
```

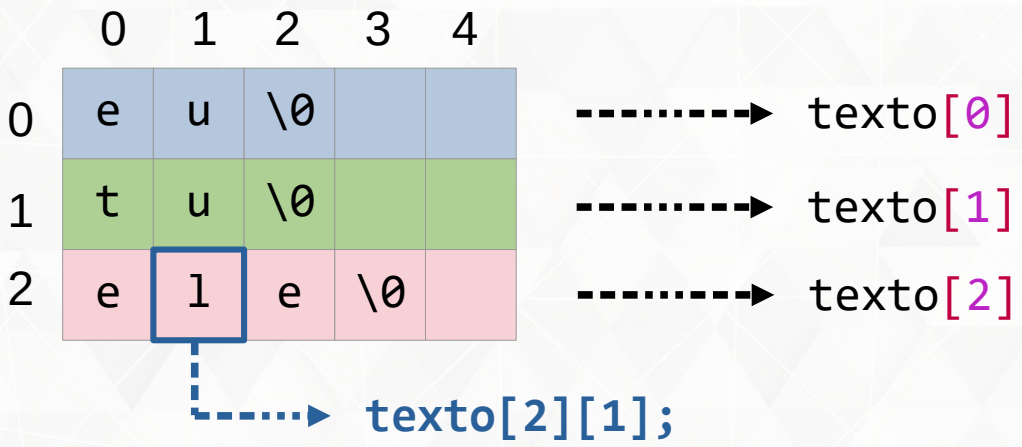
Exercícios

1. Construa um programa que leia através da entrada padrão uma string e retorne na saída padrão o número de caracteres que a mesma possui;
2. Faça um programa que leia uma string e conte a quantidade de vogais;
3. Faça um programa que receba uma palavra e imprima uma nova palavra sendo cada letra a seguinte a da palavra original. Por exemplo: Banana → Cbobob
4. Faça um programa que receba uma palavra com todas as letras minúsculas e transforme-as em maiúscula. Exemplo: banana → BANANA.

Vetor de Strings

- Cada linha da matriz é um índice que representa uma *string*
- O número de colunas indica a quantidade máxima de caracteres que cada *string* possui

```
char texto[3][5];
```




```
printf("%s", texto[0]);
```

Vetor de Strings

	0	1	2	3	4
0	e	u	\0		
1	t	u	\0		
2	e	l	e	\0	

```
1  int i;
2  char nomes[3][5];
3
4  //Preenchendo os nomes
5  for(i=0; i < 3; i++){
6     printf("Nome[%i]:", i);
7     fgets(nomes[i], 5, stdin);
8     nomes[i][strcspn(nomes[i], "\n")] = '\0';
9     setbuf(stdin, NULL);
10 }//for
11
12 //Exibindo os nomes
13 for(i=0; i < 3; i++){
14    printf("Nome[%i]: %s\n", i, nomes[i]);
15 }//for
```

Exercícios

5. Escreva um programa que leia uma senha alfanumérica. Utilize a função strcmp() para compará-la com uma senha definida internamente no programa e retorne ao usuário a validade ou não da senha fornecida por ele, em função do resultado da comparação.

6. Faça um programa que receba 3 variáveis do tipo int. A primeira corresponde ao dia, a segunda ao mês e a terceira ao ano. Faça a validação para que o usuário não possa entrar com o valor de dia maior que 31 e mês maior que 12. Crie 3 strings para receber o valor do dia, o nome do mês e o ano, respectivamente. Converta as variáveis int dia e ano para strings e armazene na em suas respectivas variáveis. Verifique qual o nome do mês equivale ao valor de “mês” fornecido pelo usuário e armazene esse nome na string destinada ao nome do mês. Crie uma quarta string denominada data com tamanho suficiente para armazenar o seguinte conteúdo:

dd/nome_mes/aaaa

Por exemplo: dia = 20, mes = 2, ano = 2016 ...
A string final, a qual deverá estar armazenada na string data é: “20/fevereiro/2016”

Exercícios

7. Leia uma string de tamanho qualquer e indique qual é o caractere que mais aparece e quantas vezes ele ocorreu nesta string. Por exemplo:

   Entrada: Vamos estudar strings

   Saída: O caractere que mais aparece é s. Apareceu 4 vezes.

   Obs.: Se existirem 2 ou mais caracteres de maior ocorrência, todos eles deverão ser mostrados.

8. Escreva um programa que receba uma string e coloque em maiúsculo a primeira letra de cada palavra dessa string. Exemplo:

   Entrada: abobrinha com feijao, muito bom, ou nao.

   Saida: Abobrinha Com Feijao, Muito Bom, Ou Nao.

9. Escreva um programa que receba duas strings como parâmetro e retorne um valor inteiro. As duas strings são informadas pelo usuário. A primeira corresponde a um texto. A segunda é uma string qualquer. Verifique se a segunda string está no texto fornecido pelo usuário. Em caso afirmativo, indique em qual posição do texto esta string começa. Caso contrário, retorne o valor -1, indicando que a string não está no texto.

Exercícios

10. Crie uma matriz com letras maiúsculas (A – Z) aleatoriamente. A matriz deve ter tamanho 8 x 8. Imprima a matriz. Em seguida, procure na matriz a maior sequência alfabética encontrada verticalmente ou horizontalmente. Entenda como sequência alfabética, por exemplo, “ABC”, “XYZ”, “RSTUV”.
11. Faça um programa que receba uma frase do usuário e, a cada ocorrência da palavra TECLADO, insira o texto OU MOUSE na string. Por fim, imprima a string resultante. Por exemplo:

    Entrada: PODE-SE UTILIZAR O TECLADO PARA A ENTRADA DE DADOS.

    Saída: PODE-SE UTILIZAR O TECLADO OU MOUSE PARA A ENTRADA DE DADOS.

12. Faça um programa que receba uma palavra e ordene seus caracteres em ordem alfabética. Por exemplo:

    Entrada: carro

    Saída: acorr

Exercícios

13. Faça um programa que receba um número em formato romano (Max. 5 mil) e imprima seu valor em algarismos arábicos. Por exemplo:

    Entrada: IV

    Saída: 4

14. Faça um programa que receba uma *string*, a qual corresponda ao CPF de uma pessoa. Valide o CPF. Procure na Internet pelas regras de validação.

15. Faça um jogo da forca.