STRING

LISIEUX ANDRADE

DEFINIÇÕES

  - Strings são seqüências de caracteres diversos. São conhecidos por "literais" na teoria de algoritmos estruturados, sendo representados entre aspas. Alguns exemplos de strings:
  - "Fulano da Silva",
  - "? Interrogação? ",
  - "1,234",
  - "0".
- As strings são tipo de dados não primitivos da linguagem C para guardar uma palavra.
- Lembre-se que os tipos de dados básicos ou primitivos em linguagem C são:
  - int , float
  - , double
  - e char

DEFINIÇÕES

- Strings são vetores de **chars**.
- As strings são o uso mais comum para os vetores. Devemos apenas ficar atentos para o fato de que as strings têm o seu último elemento como um '\0'.
- A declaração geral para uma string é:

```
char nome_da_string [tamanho];
```

- Devemos lembrar que o tamanho da string deve incluir o '\0' final.

ONDE ENCONTRA-LAS?

- A biblioteca padrão do C possui diversas funções que manipulam strings, muitas delas contidas na biblioteca `<string.h>`.
- Estas funções são úteis pois, não se pode, por exemplo, igualar duas strings:

```
string1 = string2; /* NAO faca isto */
```

- As strings devem ser igualadas elemento a elemento.

EXEMPLO

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

int main(){
    char str1[100]="lar",str2[100]="";
    int cont;
   /* Aqui o programa le str1 que sera copiada para str2 */
   for (cont=0;str1[cont];cont++)
         str2[cont]=str1[cont];
   str2[cont]='\0';
   printf("%s", str2);
}
```

TUM COMPUTADOR OPERA COM NÚMEROS. COMO ELE TRATA LETRAS E TEXTOS?

- Codificação para traduzir uma letra em um número
- - ASCII (American Standard Code for Information Interchange)
- - UNICODE
- - UTF-8

Tabela ASCII - American Standard Code Information Interchange

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

EXEMPLO

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

int main(){
    char c;
    printf("Cod\tChar\n");
    printf("---\t----\n");
    for (c=-128;c<127;c++)
    {
        printf("%d\t %c\n",c,c);
    }
}
```

FUNÇÕES

gets

A função **gets()** lê uma string do teclado.

Sua forma geral é:

```
gets (nome_da_string);
```

GETS

Corresponde a leitura de uma string do teclado

UM EXEMPLO

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


int main () {
    char string[100];
    printf ("Digite o seu nome: ");
    gets (string);
    printf ("\n\n Ola %s\n\n",string);
    return(0);
}
```

strcpy

Sua forma geral é:

```
strcpy(string_destino,string_origem);
```

Seu funcionamento é semelhante ao programa visto anteriormente. As funções de manipulação de strings são definindas no arquivo cabeçalho **string.h**.

STRCPY

A função **strcpy()** copia a string-origem para a string- destino.

UM EXEMPLO

```c
#include <stdio.h>
#include <string.h>
int main () {

   char str1[100],str2[100],str3[100];
   printf ("Entre com uma string: ");
   gets (str1);

   strcpy (str2,str1); /* Copia str1 em str2 */
   strcpy (str3,"Voce digitou a string: ");
   printf ("\n\n%s%s",str3,str2);

}
```

strcat

Sua forma geral é:

```
strcat(string_destino,string_origem);
```

STRCAT

A string de origem permanecerá inalterada e será anexada ao fim da string de destino.

UM EXEMPLO

```c
#include <stdio.h>
#include <string.h>
int main () {

    char str1[100],str2[100];
    printf ("Entre com uma string: ");
    gets (str1);

    strcpy (str2,"Voce digitou a string ");
    strcat (str2,str1);
    printf ("\n\n%s",str2);

}
```

STRLEN

A função **strlen()** retorna o comprimento da string fornecida

Sua forma geral é:

```
strlen (string);
```

O terminador nulo não é contado. Isto quer dizer que, de fato, o comprimento do vetor da string deve ser um a mais que o inteiro retornado por **strlen()**.

UM EXEMPLO

```c
#include <stdio.h>
#include <string.h>
int main () {

    int tamanho;
    char palavra[100];
    printf ("Entre com uma string: ");
    gets (palavra);

    tamanho=strlen (palavra);
    printf ("\n\nA string que voce digitou tem tamanho %d",tamanho);
}
```

Sua forma geral é:

```
strcmp
(string1,string2);
```

A função **strcmp()** compara a string 1 com a string 2.

STRCMP

Se as duas forem idênticas a função retorna zero. Se elas forem diferentes a função retorna não-zero.

UM EXEMPLO

```c
#include <stdio.h>
#include <string.h>
#include <locale.h>
int main (){
    setlocale(LC_ALL, "Portuguese");
    char str1[100],str2[100];
    printf ("Entre com uma string: ");
    gets (str1);
    printf ("\n\nEntre com outra string: ");
    gets (str2);

    if (strcmp(str1,str2))
        printf ("\n\nAs duas strings são diferentes.");
    else
        printf ("\n\nAs duas strings são iguais.");

}
```

LOCALE?

#INCLUDE <LOCALE.H> :

A BIBLIOTECA LOCALE.H CONTÉM ESPECIFICAÇÕES REGIONAIS TAIS COMO A REPRESENTAÇÃO DE NÚMEROS FRACIONÁRIOS (‘.’ OU ‘,’), SÍMBOLO DE MOEDAS E FORMATO DE DATAS. NOS PROGRAMAS

```
SETLOCALE(LC_ALL, “PORTUGUESE”);
```

QUANDO USAMOS ESTA FUNÇÃO DEVEMOS ATENTAR QUE A ENTRADA DE VALORES FRACIONÁRIOS DEVE SER FEITA SEPARANDO-SE AS DECIMAIS COM VÍRGULA AO INVÉS DO PONTO DECIMAL.

- São as funções mais simples do cabeçalho stdio.h.
- Ambas enviam (ou "imprimem") à saída padrão os caracteres fornecidos a elas;
- putchar() manda apenas um caractere, e puts() manda uma sequência de caracteres (ou string).

PUTS() E PUTCHAR()

puts significa "put string" (colocar string), utilizado para "colocar" uma string na saída de dados. **putchar** significa "put char" (colocar caractere), utilizado para "colocar" um caractere na saída de dados.

UM EXEMPLO

```c
#include <stdio.h>
#include <string.h>
int main (){
    char nome[50];
    printf("Seu nome? ");
    gets(nome);
    puts(nome);
}
```

OUTROS DETALHES

PROBLEMA!

```
char c;
while (c!=65)
{
    printf("Uma letra:\n");
    scanf("%c", &c);
}
```

```
char c;
while (c!=65)
{
    printf("Uma letra:\n");
    c = getchar();
}
```

FFLUSH(STDIN)

```
char c;
while (c!=65)
{
    printf("Uma letra:\n");
    fflush(stdin);
    scanf("%c", &c);
}
```

```
char c;
while (c!=65)
{
    printf("Uma letra:\n");
    fflush(stdin);
    c = getchar();
}
```

CARACTERES SÃO SEQUENCIAIS

```c
#include <stdio.h>
#include <string.h>
#include <locale.h>
int main (){
    setlocale(LC_ALL, "Portuguese");
    char c = getchar();
    if ((c >= 'A')&&(c <= 'Z'))
    printf("digitou uma letra maiúscula");
    else
    printf("não digitou uma letra maiúscula");
}
```

RECEBENDO UMA STRING COM ESPAÇOS EM BRANCO

```c
#include <stdio.h>
#include <string.h>
#include <locale.h>
int main (){
    char nome[50];
    printf("Seu nome? ");
    scanf("%s", &nome);
    printf("Olá, %s", nome);
}
```

```c
#include <stdio.h>
#include <string.h>
#include <locale.h>
int main (){
    char nome[50];
    printf("Seu nome? ");
    scanf("%[^\n]s", &nome);
    printf("Olá, %s", nome);
}
```

Corrigimos o erro, forçando o *scanf* ler a string até encontrar o [enter]

VAMOS PRATICAR!

QUESTÃO 1

FAÇA UM PROGRAMA QUE LEIA UMA FRASE E EXIBA OS CARACTERES QUE OCUPAM AS POSIÇÕES ÍMPARES.

- **Ex.: Dada a frase:** ”Informática da UFC.”
- Seria exibido: “Os caracteres impares são: I f r á i a d F C”

QUESTÃO 2

ESCREVA UM PROGRAMA PARA LER DEZ PALAVRAS E DIZER QUANTAS SÃO PALÍNDROMOS.

OBS: Palíndromo é uma palavra que representa a mesma escrita tanto da esquerda para a direita como da direita para a esquerda. Ex: OSSO, ASA, ERRE.

QUESTÃO 3

ESCREVA UM PROGRAMA QUE RECEBA DUAS STRINGS (A E B) E RETORNE UMA TERCEIRA STRING (C) FORMADA PELOS CARACTERES DE A E B INTERCALADOS.

Ex.: Se A='Quarta' e B='Segunda', a resposta deve ser 'QSueagrutnada'

QUESTÃO 4

ESCREVA UM PROGRAMA PARA LER UMA STRING E UM CARACTERE. SEMPRE QUE O CARACTERE LIDO APARECER NA FRASE ELE DEVE SER SUBSTITUÍDO POR ASTERISCO.

Frase: o dia esta nublado

Caracter: d

Resultado: o *ia esta nubla*o

QUESTÃO 5

FAÇA UM PROGRAMA PARA LER UMA FRASE E EXIBIR AS FREQUÊNCIAS ABSOLUTA DE CADA UMA DAS VOGAIS E DOS DEMAIS CARACTERES.

Ex.: dada frase: "Os olhos dela pareciam duas estrelas brilhando." Seria exibido:

```
---------
 a  :  6
 e  :  4
 i  :  2
 o  :  3
 u  :  1
 Outro: 31
---------
 Total: 47
```

QUESTÃO 6

FAÇA UM PROGRAMA QUE LEIA UMA FRASE E EXIBA A QUANTIDADE DE PALAVRAS.
EX:
DADA A FRASE:“LADEIRA DA BORBOREMA, TÚ ÉS MAIOR DO QUE EU.”
SERIA EXIBIDO: “A FRASE DADA TEM 9 PALAVRA(S)”

QUESTÃO 7

FAÇA UM PROGRAMA QUE LEIA UMA FRASE E RETIRE TODOS OS CARACTERES REPETIDOS.

EX.: DADA A FRASE: “UUNIIVERSIDDADE -- FEEDERAL DOO CE...”

SERIA EXIBIDO: “UNIVERSIDADE - FEDERAL DO CE.”