Caracteres e String

Prof. Lilian Berton

São José dos Campos, 2019

Caracteres

- Um caractere pode ser uma letra (maiúscula ou minúscula), e símbolos que normalmente encontramos num teclado de um computador.

- Em C, caracteres são armazenados como números inteiros usando uma **tabela de conversão chamada ASCII** (*American Standard Code for Information Interchange*). Existem extensões dessa tabela.

032	! 033	" 034	# 035	$ 036	% 037	& 038	' 039
(040	) 041	* 042	+ 043	, 044	- 045	. 046	/ 047
0 048	1 049	2 050	3 051	4 052	5 053	6 054	7 055
8 056	9 057	: 058	; 059	< 060	= 061	> 062	? 063
@ 064	A 065	B 066	C 067	D 068	E 069	F 070	G 071
H 072	I 073	J 074	K 075	L 076	M 077	N 078	O 079
P 080	Q 081	R 082	S 083	T 084	U 085	V 086	W 087
X 088	Y 089	Z 090	[091	\ 092	] 093	^ 094	_ 095
` 096	a 097	b 098	c 099	d 100	e 101	f 102	g 103
h 104	i 105	j 106	k 107	l 108	m 109	n 110	o 111
p 112	q 113	r 114	s 115	t 116	u 117	v 118	w 119
x 120	y 121	z 122	{ 123	| 124	} 125	~ 126	127

Char em C

- Um caractere em C é representado pelo tipo char.
- Para ler e imprimir um caractere:

```
char a;
printf (“Entre com um caractere: “);
scanf(“%c”, &a);
printf (“Caractere digitado: %c\n”,a);
```

Char em C

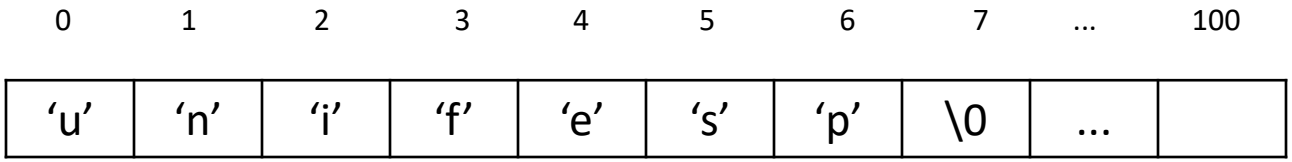
- Para evitar a necessidade de decorar a tabela ASCII, na linguagem C, escrever **um caractere entre apóstrofes equivale a escrever o seu código ASCII**. Assim, escrever 65 e ‘A’ são equivalentes.

```
char i = 65;
char j = ‘A’;
if (i == j) return true;
```

String em C

- **Strings são vetores de caracteres** com um código para marcar sua terminação. O caractere especial designado para indicar fim da sequência é o caractere ‘\0’ (barra zero). O que vier depois do \0 é considerado lixo de memória.

```
char palavra [101]; //declara um vetor de char com capacidade de armazenar 100 caracteres
palavra = “unifesp”;
palavra = {‘u’,’n’,’i’,’f’,’e’,’s’,’p’};
```



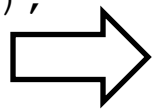
Como foram declaradas 101 posições e uma delas é ocupada pelo terminador ‘\0’, as outras 100 posições estão livres para armazenamento de caracteres.

Ler String em C

- Para ler uma string o usuário vai digitar uma sequência de caracteres e depois, para terminar, digita a tecla “enter” do teclado, que produz o caractere ‘\n’.

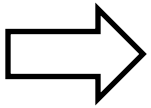
- O comando scanf lê caractere a caractere e coloca um a um em cada casa do vetor palavra; no final, o caractere ‘\n’ é ignorado e coloca-se o caractere ‘ \0’ .

```
char palavra[100];
printf("Entre com uma palavra: ");
scanf ("%[^\n]", palavra);
```



scanf(“%[^\n]”) armazena várias palavras.

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



scanf(“%s”) armazena apenas uma palavra.

Imprimir String em C

- Uma forma de imprimir um string em C é utilizar o scanf com %s e em seguida coloca-se o nome do vetor.

```
printf ("A palavra digitada foi: %s\n", palavra);
```

```c
#include<stdio.h>

int main(void)
{
    char nome[61];

    printf("Digite seu nome: ");
    scanf("%s",nome);

    printf("O nome armazenado foi: %s", nome);

    return 0;
}
```

Exemplo 1

Faça um programa que leia uma frase e imprima esta frase usando apenas letras maiúsculas.

```
#include <stdio.h>
int main () {
    char frase [80];
    int i;
    printf ("Entre com uma frase: ");
    scanf ("%[^\n]", frase);
    for (i=0; frase[i] !='\0'; i++) {
        if (frase[i] >= 'a' && frase[i] <= 'z')
            frase[i] = frase[i] - ('d' - 'D');
    }
    printf ("Frase Digitada em Maiusculas: %s\n", frase);
    return 0;
}
```

Para cada casa do vetor, verifico se é uma letra minúscula. Em caso afirmativo, transforma para maiúscula subtraindo da diferença entre uma letra minúscula e sua correspondente maiúscula (note que esta diferença é a mesma para qualquer letra - neste caso, foi escolhida a letra 'd').

Exemplo 1

Faça um programa que leia uma frase e imprima esta frase usando apenas letras maiúsculas.

```
# include <stdio.h>
int main ( ) {
    char frase [80];
    int i;
    printf ("Entre com uma frase: ");
    scanf ("%[^\n]", frase);
    for (i=0; frase[i] !='\0'; i++) {
        if (frase[i] >= 97 && frase[i] <= 122)
            frase[i] = frase[i] - (100 - 68);
    }
    printf ("Frase Digitada em Maiusculas: %s\n", frase);
    return 0;
}
```

Exemplo 2

Faça um programa que encontre o tamanho de um vetor de char, isto é, quantos caracteres ele contém.

```
#include <stdio.h>
int main () {
    char frase [80];
    int i, cont = 0;
    printf ("Entre com uma frase: ");
    scanf ("%[^\n]", frase);
    for (i=0; frase[i] !='\0'; i++) {
            cont++;
    }
    printf ("Frase Digitada contem: %d caracteres\n", cont);
    return 0;
}
```

Exemplo 3

Faça um programa que copie o conteúdo de uma string em outro vetor.

```
#include <stdio.h>
int main ( ) {
    char frase [80], frase2 [80]; ;
    int i, cont = 0;
    printf ("Entre com uma frase: ");
    scanf ("%[^\n]", frase);
    for (i=0; frase[i] !='\0'; i++) {
            frase2 [i] = frase [i]; ;
    }
    printf (" Frase : %s\n Frase 2: %s\n", frase, frase2);
    return 0;
}
```

Biblioteca String

- Deve-se incluir a biblioteca **<string.h>**
- Pode usar funções prontas para:
- ler uma string:

```
char palavra[100];
fgets (palavra, 100, stdin);
```

- Saber o tamanho do vetor: `strlen(palavra);`
- Copiar o conteúdo de um vetor2 para outro:

```
strcpy(palavra,"unifesp");
```

- Etc.

Biblioteca String

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

int main() {
    char nome[10];
    int i;

    for(i = 0; i < 5; i++) {
        printf("Insira um nome\n");
        fgets(nome,10,stdin);
         printf("O nome %s possui %d caracteres\n",nome,strlen(nome));
    }
}
```

Limpar o buffer do teclado

- Toda informação digitada é armazenada em um buffer e o scanf( ) recupera a informação do buffer. Porém, pode ficar “lixo” no buffer.
- Em muitos casos, ao ler um char/String seu programa pode pular a leitura da string seguinte.

```
#include <stdio.h>

int main() {
        char caractere;
        int i;
        for(i = 0; i < 5; i++) {
                printf("Insira o %d caractere\n",i+1);
                scanf("%c",&caractere);
        }
}
```

Limpar o buffer do teclado

- Nesses casos é necessário utilizar o SETBUF para limpar o buffer do teclado.

```
#include <stdio.h>

int main() {
        char caractere;
        int i;
        for(i = 0; i < 5; i++) {
                printf("Insira o %d caractere\n",i+1);
                scanf("%c",&caractere);
                setbuf(stdin,NULL);
                getchar();
        }
}
```

Exercícios

1. Dada uma sequência de caracteres representando um texto, determinar a frequência relativa de vogais no texto (por exemplo, no texto “Em terra de cego quem tem um olho é caolho”, essa frequência é 16 vogais/42 caracteres).

2. Fazer um programa de “criptografia” (codificação de dados visando a privacidade de acesso as informações), onde dada uma string (vetor de caracteres) este programa codifique os dados através de um processo de substituição de letras.

   Implemente uma função que recebe o vetor de palavras e criptografe a frase substituindo cada caractere por 3 caracteres adjacentes a ele.

   Ex: “bom dia” -> mensagem criptografada = erp#gld


My Code
Me the
Next Day
My Code