Strings

Rafael Beserra Gomes

UFRN

Material compilado em 19 de março de 2018. Licença desta apresentação:



http://creativecommons.org/licenses/

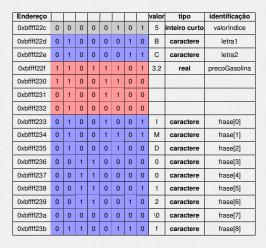
Strings em C

Strings em C

- Strings é o conceito abstrato de uma sequência de caracteres.
- Representação em C: vetor de caracteres ou literais (entre aspas duplas)
- O caractere '\0' indica o final da string

Representação da String na memória

Exemplo de dados na memória:



Strings em C

Lembre-se de que os caracteres são codificados como números de acordo com a tabela ascii

```
#include <stdio.h>
2
  int main() {
4
5
       char nome[] = "Eu tenho uma casa";
6
       int i;
 7
8
       printf("%s\n", nome);
9
10
       nome[15] = 'm';
11
12
       printf("%s\n", nome);
13
14
       return 0;
15
```

Escrita de strings na tela



Escrita de strings na tela

printf: use %s no especificador de formato, a escrita é feita até o caractere nulo ('\0')

```
#include <stdio.h>
 2
   int main() {
 4
 5
       char frase[300] = "Ouviram do Ipiranga as margens placidas";
 6
       printf("%s\n", frase);
 8
 9
       frase[19] = ' \setminus 0';
10
11
       printf("%s\n", frase);
12
13
       return 0;
14
```



Leitura de strings:

```
char palavra[100];
2
3 scanf("%s", palavra);
4 printf("%s\n", palavra);
```

- ¿ cuidado: lembre-se do espaço para o \0, nesse exemplo palavra deve ter no máximo 99 caracteres
- **cuidado:** não há verificação de buffer overflow (exceder o tamanho da área de armazenamento)

buffer	h	i	n	0	∖n	
próximo					1	

Leitura de strings

Cuidado com o buffer de entrada! Quando apertamos enter em um scanf, o \n continua lá

```
1 char palavra[100];
2 char t:
 scanf("%s", palavra);
4 printf("%s\n", palavra);
5 scanf("%c", &t);
6 printf("(%c)\n", t);
```

getchar() retorna o próximo caractere do buffer de entrada

No scanf há um matching entre o buffer de entrada e a primeira string

```
1 #include <stdio.h>
2
3 int main() {
4
5     scanf("Ouviram");
6     printf("%c\n", getchar());
7     printf("%c\n", getchar());
8     printf("%c\n", getchar());
9
10     return 0;
11 }
```

string do scanf	0	u	٧	i	r	а	m
buffer	0	u	٧	Х	у	Z	\n
próximo				1			

Cada especificador de formato espera algo diferente, podendo consumir vários caracteres do buffer

```
#include <stdio.h>
2
   int main() {
4
5
       int h, m, s;
6
 7
       scanf("%d:%d:%d", &h, &m, &s);
8
       printf("%c\n", getchar());
9
       printf("%c\n", getchar());
10
       printf("%c\n", getchar());
11
       printf("%d:%d:%d\n", h, m, s);
12
13
       return 0:
14
```

string do scanf	%d	:	%d	:	%d
buffer					

Opções para ler uma string com espaços:

```
#include <stdio.h>
2
3
  int main() {
4
5
       char frase[100];
6
       scanf("%[^\n]", frase);
       printf("%s\n", frase);
8
9
       return 0:
10
```

- usar [^\n] significa consumir qualquer coisa exceto \n
- o \n continua no buffer
- **La cuidado:** não há verificação de buffer overflow

Opções para ler uma string com espaços:

```
#include <stdio.h>
2
  int main() {
4
5
       char frase[100];
6
       gets(frase);
       printf("%s\n", frase);
8
9
       return 0;
10
```

Leitura de strings

- consome o \n, substituindo-o pelo \0
- **La cuidado:** não há verificação de buffer overflow

Opções para ler uma string com espaços:

```
#include <stdio.h>
2
3
  int main() {
4
5
       char frase[5];
6
       fgets(frase, 5, stdin);
       printf("%s\n", frase);
8
9
       return 0:
10
```

Leitura de strings

- lê no máximo o segundo argumento 1 caracteres da entrada (evita buffer overflow)
- o que ultrapassar esse máximo, continua no buffer de entrada
- cuidado: fgets pode armazenar o \n na string!
- **stdin** é uma macro definida em stdio.h, veremos mais na frente no tópico de arquivos

Percorrendo uma string



Percorrendo uma string

- A função strlen (de string.h) retorna a quantidade de caracteres
- Ex: ler uma string e trocar todos os espaços por -

```
1 #include <stdio.h>
  #include <string.h>
 3
   int main() {
 5
 6
       char frase[100];
       int i;
 8
 9
       fgets(frase, 100, stdin);
10
11
       for(i = 0; i < strlen(frase); i++) {</pre>
12
            if(frase[i] == ' ') {
13
                frase[i] = '-';
14
15
16
       printf("%s\n", frase);
17
18
       return 0;
19
```

Percorrendo uma string

Percorrendo uma string

Resolvendo o problema do possível \n no fgets

```
#include <stdio.h>
 2
   int main() {
 4
 5
       char frase[5];
 6
       fgets(frase, 5, stdin);
       if(frase[strlen(frase)-1] == '\n') {
 8
            frase[strlen(frase)-1] = ' \setminus 0';
 9
10
       printf("%s\n", frase);
11
12
       return 0;
13
```

Exercício em sala

Uma das regras mais conhecidas da ortografia no português é que 'n' não deve preceder 'p' e 'b'. Escreva um programa que leia uma string de até 20 caracteres e realize o conserto de acordo com essa regra. Por exemplo, a string "inpedido no canto do canpo" é transformada em "impedido no canto do campo".



Escreva um programa para ler uma string qualquer e, em seguida, escrever a mesma string em caixa alta.