

MC-102 — Aula 14

Cadeias de Caracteres

Instituto de Computação – Unicamp

Segundo Semestre de 2009

Cadeias de caracteres

- Uma cadeia de caracteres, mais conhecida como *string*, é uma sequência de letras e símbolos, onde os símbolos podem ser espaços em branco, dígitos e vários outros como pontos de exclamação e interrogação, símbolos matemáticos, etc.
- Em C, uma cadeia de caracteres é representada por um vetor de variáveis do tipo `char` e é terminada com o marcador `'\0'`.

Exemplo de declaração

```
char texto [TAMANHO + 1];
```

- Devemos utilizar uma posição além do tamanho máximo desejado para que possa ser colocado o marcador '\0' no final da maior cadeia armazenável nesta variável.

Lendo uma cadeia do teclado

- Podemos ler uma cadeia caracter a caracter, como faríamos com qualquer outro vetor, mas é mais simples ler a cadeia inteira, utilizando o formato %s.

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

- Note que não utilizamos o e comercial (&) para cadeias. Isso ocorre pois o nome de um vetor já é um endereço de memória (o endereço de memória do começo do vetor).

Veja o exemplo em `scanf.c`.

Lendo uma cadeia do teclado

- Infelizmente, a leitura a partir do teclado utilizando o `scanf` lê somente até o primeiro espaço, ou seja, lê somente uma palavra, o que torna o seu uso desta forma um pouco restrito.
- Podemos explorar as possibilidades fornecidas pela função `scanf`. Por exemplo, a opção abaixo lê uma cadeia de caracteres até encontrar um `enter`.

```
scanf ("%[^\\n]");
```

Veja um exemplo em `scanf-alternativo.c`. Veja mais opções consultando a página de manual com o comando `man scanf`.

Lendo uma cadeia do teclado

- Podemos, também, utilizar a função `gets`, que faz a leitura até encontrar o caracter de fim de linha (*enter*).

```
gets(texto);
```

Veja o exemplo em `gets.c`.

Lendo uma cadeia do teclado

- Aqui, deparamos com outro problema: tanto o comando `scanf` quanto o `gets` podem ler mais caracteres que os existentes na cadeia, provocando erros.
- A solução aqui, é utilizar uma função que tenha o mesmo comportamento do `gets`, mas que permita limitar a leitura a um tamanho máximo. As opções são:

```
fgets (texto, 50, stdin);  
scanf ("%50[^\n]", texto);
```

Veja os exemplos em `fgets.c` e `scanf-alternativoN.c`.

Escrevendo uma cadeia na tela

- Podemos escrever uma cadeia na tela caracter a caracter, mas é mais simples escrever utilizando o comando `printf`, com o mesmo formato utilizado para lê-la (`%s`)

```
printf ("%s", texto);
```

Escrevendo uma cadeia na tela

- De forma análoga ao `gets` e `fgets`, temos o `puts` e `fputs`, que escrevem a cadeia na tela.

```
puts (texto);  
fputs (texto, stdout);
```

Veja o exemplo em `puts.c`.

Manipulando cadeias de caracteres

- Retorna o tamanho da cadeia **string** em número de caracteres. (meuStrlen.c)
- Copia a cadeia **fonte** para a cadeia **destino**. (meuStrcpy.c)
- Concatena a cadeia **fonte** no fim da cadeia **destino**. (meuStrcat.c)
- Compara a cadeia **string1** com a cadeia **string2**. (meuStrcmp.c)

Manipulando cadeias de caracteres

As cadeias de caracteres são tão importantes que existe uma biblioteca de funções só com comandos para ela, a biblioteca `string.h`. Entre as diversas funcionalidades oferecidas por esta biblioteca, podemos destacar:

- `strlen(texto)` — Retorna o tamanho da cadeia texto em número de caracteres.
- `strcpy(destino, fonte)` — Copia a cadeia fonte para a cadeia destino.
- `strcat(destino, fonte)` — Concatena a cadeia fonte no fim da cadeia destino.

Veja o exemplo de uso em `funcoes.c`.

Mais uma função útil da biblioteca `string.h`:

- `strcmp(str1, str2)` — Compara duas cadeias de caracteres e retorna um valor:

= 0: se `str1` e `str2` forem iguais;

< 0: se `str1` for menor que `str2`;

> 0: se `str1` for maior que `str2`.

Veja um exemplo de uso em `comparaStrings.c`.

Manipulando cadeias de caracteres

- Apesar de ser mais prático usar as funções da biblioteca `string.h`, é importante sabermos como manipular cadeias diretamente.
- Em especial, cadeias não possuem um valor indicando explicitamente o seu tamanho, pois são terminadas pelo caracter `'\0'`.