

Strings

Introdução à Ciência da Computação I

Prof. Denis F. Wolf

Strings

- Em C as strings são entendidas como vetores de caracteres.
- Porém como a quantidade de caracteres em uma string pode ser menor do que seu tamanho declarado, elas foram implementadas em C seguindo o modelo “*null terminated*”, isto é, o término de uma string é dado pelo caractere de controle `'\0'`.
- Atenção:
 - Constantes caractere são delimitadas por apóstrofos
 - Constantes string são delimitadas por aspas

Strings

- Por exemplo, para declarar uma string nome capaz de armazenar até 5 caracteres:

```
char nome[6];
```

 - Lembre-se que o tamanho da string deve ser suficiente para conter também o `'\0'` no final.
 - Assim, temos tamanho 6 = 5 caracteres + `'\0'`.
- Do mesmo modo que os arrays, também é possível inicializar uma string no momento de sua declaração.
 - Ex.:

```
char nome[8] = "Leandro";
```

```
char sobrenome[] = "Fernandes";
```

Strings

- Problema:

```
char x[10], y[10];
x = "Ola";          /* Não faça isso! */
x = y;             /* Não faça isso! */
x = x + y;        /* Não faça isso! */
if (x == y) ...   /* Não faça isso! */
```
- Strings não podem ser atribuídas ou comparadas diretamente
- Afim de tornar mais fácil sua manipulação, a linguagem provê recursos específicos através da biblioteca `string.h`

string.h

- **strlen**: Retorna o número de caracteres da cadeia de caracteres, sem contar o `'\0'`.

```
strlen("casa") ==> 4
```
- **strcmp**: Compara, caractere a caractere, duas cadeias e retorna o resultado dessa comparação.

```
strcmp("casa", "carro") == 1
strcmp("casa", "casa") == 0
strcmp("carro", "casa") == -1
```

string.h

- Como a linguagem C entende como diferentes letras minúsculas de maiúsculas, no tratamento de strings podemos utilizar:
- **stricmp**: Compara, caractere a caractere, duas cadeias e retorna o resultado dessa comparação ignorando a diferença entre maiúsculas e minúsculas.

```
stricmp("Casa", "casa") == 1
stricmp("Casa", "Casa") == 0
```

string.h

- **strcpy**: Copia uma string para dentro de outra.
`strcpy(vetor, "carro");`
`strcpy(vetor2, vetor1);` */* copia 1 em 2 */*
- **strcat**: Concatena duas strings, a primeira string recebe o seu conteúdo seguido do conteúdo da segunda string.
`strcpy(vetor, "uva + ");`
`strcat(vetor, "banana + ");`
`strcat(vetor, "pera");`
/ vetor = "uva + banana + pera" */*

string.h

- **strchr**: Retorna um ponteiro para a primeira ocorrência de determinado caractere .
- **strrchr**: Retorna um ponteiro para a última ocorrência de determinado caractere .
- **strstr**: Retorna um ponteiro para uma substring dentro de uma string.

Strings

- Assim, retomando nosso exemplo:
`char x[10], y[10];`
`x = "Ola";`
/ ao invés de */* */* utilize */*
`x = y;` `strcpy(x,y);`
`x = x + y;` `strcat(x,y);`
`if (x == y) ...` `if (strcmp(x,y) == 0) ...`

Conversão de tipos

- `atof(string)` string para float
- `atoi(string)` string para int
- `atol(string)` string para long int
- `strtod(string)` string para double
- `strtol(string)` string para long

Exercícios

- 1) Crie uma função que recebe uma string e retorna o número de palavras dessa string (assumindo que as palavras são separadas por 1 ou mais espaços).
- 2) Faça uma função que receba uma string e um caractere qualquer. A função retorna a mesma string retirando todas as ocorrências do caractere. Exemplo: Seja a string "arvore" e o caractere 'r', então a string ficará "avoe".

Exercícios

- 3) Faça uma função que receba 2 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 'QSueagru tnada'.
- 4) Crie uma função que recebe 2 strings (A e B) e retorna 1 se B é parte de A, sem usar funções da biblioteca string.h. A função retorna 0 caso contrário.
- 5) Faça função que receba uma string e a retorne escrita de trás pra frente.