SI100

Algoritmos e Programação de Computadores I

1º Semestre - 2018

Tópico 7 – *Strings*

Unidade 11

Prof. Guilherme Palermo Coelho

guilherme@ft.unicamp.br

Roteiro

- Introdução;
- Constantes *String*;
- Variáveis *String*;
- Leitura de *strings*;
- Impressão de *strings*;
- Inicialização de *strings*;
- Funções de manipulação de *strings*;
- Exercícios;
- Referências.

Introdução e Definições

Introdução

- ***Strings*** são usadas para armazenar e manipular textos em C;
  - Tais textos podem ser nomes, palavras e até mesmo frases;
  - *Strings* são utilizadas na maioria dos programas.
- Em C, ***strings*** são:
  - Vetores do tipo **char**;
  - Terminadas pelo caractere NULL (‘\0’)
    - NULL tem código ASCII igual a 0 em decimal.
- É possível acessar cada caractere de uma ***string*** através do acesso à posição correspondente do vetor.

Constantes *String*

- O compilador da linguagem C sempre considera tudo que está entre aspas duplas uma **constante** do tipo ***string***;
  - **Ex.:** printf(“%s”, “Saudacoes!”);
    - “Saudacoes!” é uma ***string*** constante;
    - %s – é o código de impressão associado a uma ***string***.
  - Ao detectar uma ***string*** constante, o compilador aloca posições de memória para armazená-la:
    - 01 byte por caractere;
    - A última posição **sempre** terá o caractere **null** (‘\0’):
      - Permite reconhecer o final da ***string***.

Constantes *String*

- Supondo que a constante “Saudacoes!” seja armazenada a partir do endereço de memória 1450, teríamos:

1449	…
1450	S
1451	a
1452	u
1453	d
1454	a
1455	c
1456	o
1457	e
1458	s
1459	!
1460	\0

Variáveis *String*

- Variáveis do tipo ***string*** nada mais são do que ***vetores*** com elementos do tipo ***char***;
  - **Ex.:** char nome[15];

O comando ***scanf*** lê automaticamente o texto do teclado até encontrar um caractere branco, salva os dados na memória e coloca um '\0' no final.

```
#include <stdio.h>

int main() {
        char nome[15];

        printf("Digite o seu nome: ");
        scanf("%s", nome);
        printf("Saudacoes, %s.\n", nome);

        return 0;
}
```

Atenção: como em qualquer vetor, **nome** corresponde ao endereço da primeira posição de memória alocada → não se deve usar o & na função **scanf**.

LEITURA E IMPRESSÃO DE *STRINGS*

Lendo *Strings*

- A leitura de uma ***string*** consiste em dois passos:
  - Reservar um espaço de memória para armazená-la (variável);
  - Chamar uma função de leitura que receba os dados do teclado e os armazene na variável (***scanf***, ***gets***, ...);
- Como vimos no exemplo anterior, a função ***scanf()*** faz o seguinte:
  - Lê cada caractere **não branco** da entrada e os armazena sequencialmente no vetor (variável);
  - Ao encontrar o primeiro caractere em branco, o substitui pelo caractere '\0' (null).

Scanf é limitado! Como ler, por exemplo, nomes completos?

Lendo *Strings*

- Exemplo:

```
#include <stdio.h>

int main() {
        char nome[35];

        printf("Digite o seu nome: ");
        scanf("%s", nome);
        printf("Saudacoes, %s.\n", nome);

        return 0;
}
```

```
Digite o seu nome: Guilherme Coelho
Saudacoes, Guilherme.
```

Lendo *Strings*

- Uma função alternativa para ler ***strings*** é a função ***gets*()**:
  - Esta função lê caracteres do teclado até encontrar o de nova linha '\n' (gerado quando pressionamos ENTER);
  - Quando **gets()** encontra o '\n', todos os caracteres anteriores são armazenados na **string** e '\0' é inserido no final;
- **ATENÇÃO**: **gets()** lê caracteres e os armazena em memória até que seja encontrado o '\n'.
  - Não é feita nenhuma verificação do número de caracteres lidos;
  - Há o risco de se estourar o tamanho do vetor declarado para ler o texto.
    - Nesse caso, as posições de memória seguintes serão sobrepostas!

Lendo *Strings*

- Exemplo:

```
#include <stdio.h>

int main() {
        char nome[35];

        printf("Digite o seu nome: ");
        gets(nome);
        printf("Saudacoes, %s.\n", nome);

        return 0;
}
```

```
Digite o seu nome: Guilherme Coelho
Saudacoes, Guilherme Coelho.
```

Imprimindo *Strings*

- Existem duas funções principais para impressão de ***strings*** na tela: ***printf*()** e ***puts*()**:
  - **printf()** utiliza o código de impressão “%s” para identificar a impressão de uma *string*;
    - **Ex.**: printf(“Saudacoes, %s.\n”, nome);
    - A *string* é impressa exatamente como armazenada.
  - **puts()** é o complemento da função *gets*();
    - Recebe diretamente o nome da *string* que se deseja imprimir;
    - Reconhece o ‘\0’ como final da *string*, e imprime automaticamente um ‘\n’ (quebra de linha) no final;
    - **Ex.**: puts(nome);

Imprimindo *Strings*

```
#include <stdio.h>

int main() {
        char nome[81];

        printf("Digite o seu nome: ");
        gets(nome);
        puts("Saudacoes, ");
        puts(nome);

        printf("Sobrenome: ");
        puts(&nome[10]);
        return 0;
}
```

- É possível imprimir **parte** de uma **string** usando **puts()**.
- Basta fornecer o endereço da posição a partir da qual se deseja imprimir.

```
Digite o seu nome: Guilherme Coelho
Saudacoes,
Guilherme Coelho
Sobrenome: Coelho
```

Inicialização de Strings

Inicialização de *Strings*

- Como *strings* nada mais são do que **vetores** de elementos do tipo **char**, é possível inicializá-las como fizemos com vetores de valores numéricos:
  - **Ex.:** char nome[] = {'A', 'n', 'a', '\0'};
  - Não se pode esquecer do caractere '\0' ao fazer este tipo de inicialização!
- No entanto, os compiladores da linguagem C oferecem uma forma alternativa de inicialização, que é equivalente mas muito mais simples:
  - **Ex.**: char nome[] = "Ana";
  - Neste segundo caso, o '\0' é colocado automaticamente.

FUNÇÕES DE MANIPULAÇÃO DE *STRINGS*

Funções de Manipulação de *Strings*

- A manipulação de cadeias de texto é tão comum em programação que existe até uma biblioteca própria em C para reunir estas funções
  - Biblioteca **string.h**;
- Neste tópico, veremos algumas das funções de manipulação de *strings* oferecidas em C:
  - **strlen(texto)**: retorna o tamanho da *string* **texto** em número de caracteres (**não** considera o '\0' na contagem);
  - **strcat(destino, fonte):** concatena a *string* **fonte** no final da *string* **destino**;
  - **strcmp(texto1, texto2):** compara as *strings* **texto1** e **texto2**, usando a *ordem alfabética*.
    - Pode retornar valor negativo (se **texto1 < texto2**), positivo (se **texto1 > texto 2**) ou nulo (se **texto1 == texto2**).

Funções de Manipulação de *Strings*

- **Exemplo 1**: strlen()

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

int main() {
    char nome[81];
    int i;

    printf("Digite nome: ");
    gets(nome);

    puts("Seu nome eh: ");
    for (i=0; i < strlen(nome); i++){
        printf("%c\n", nome[i]);
    }

    return 0;
}
```

```
Digite nome: Maria
Seu nome eh:
M
a
r
i
a
```

Funções de Manipulação de *Strings*

- **Exemplo 2**: strcat()

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

int main() {
        char nome[81], sobrenome[81];
        int i;

        printf("Digite nome: ");
        gets(nome);
        printf("Digite sobrenome: ");
        gets(sobrenome);

        puts(”Nome completo: ");
        strcat(nome, sobrenome);
        printf("%s\n", nome);
        return 0;
}
```

```
Digite nome:
Guilherme
Digite sobrenome:
Coelho
Nome completo:
GuilhermeCoelho
```

Funções de Manipulação de *Strings*

- **Exemplo 3**: strcmp()

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

int main() {
        char resp[] = "branco";
        char r[40];

        puts("Qual a cor do cavalo branco de Napoleão?");
        gets(r);

        while (strcmp(r, resp) != 0) {
                puts("Resposta errada. Tente de novo.");
                gets(r);
        }

        puts("Correto!");
        return 0;}
```

Funções de Manipulação de *Strings*

- Outras funções importantes da biblioteca **string.h**:
  - **strcpy(destino, fonte)**: copia a *string* **fonte** para a *string* **destino**;

    http://www.lix.polytechnique.fr/~liberti/public/computing/prog/c/C/FUNCTIONS/strcpy.html

  - **strstr(texto, busca):** procura a ocorrência da sub*string* **busca** na *string* **texto** e retorna o endereço do início da primeira ocorrência de **busca** (ou **null** caso não encontre);

    http://www.lix.polytechnique.fr/~liberti/public/computing/prog/c/C/FUNCTIONS/strstr.html

  - **strchr(texto, ch):** procura a primeira ocorrência do caractere **ch** na *string* **texto** e retorna a posição deste caractere.

    http://www.lix.polytechnique.fr/~liberti/public/computing/prog/c/C/FUNCTIONS/strchr.html

EXERCÍCIOS

Exercícios

1. Escreva um programa que leia uma *string* qualquer (máximo 80 caracteres) e retorne o número total de caracteres contidos nessa *string*. **Não** use a função *strlen*().

2. Escreva um programa que leia uma *string* qualquer (máximo 80 caracteres) e imprima na tela a sua versão espelhada, ou seja, a mesma *string* mas com a ordem dos caracteres invertida .

3. Faça um programa que teste se uma palavra é um ***palíndromo***, ou seja, se ela pode ser lida da mesma forma tanto na ordem correta quanto na ordem invertida. **Ex.**: arara, ovo, omo.

4. Refaça o exercício 3 de forma que seu programa não seja ***sensível a letras maiúsculas***.

Exercícios

5. Modifique o exercício 4 para que ele passe a verificar se uma ***frase*** é um palíndromo.

- **Observação**:
  - Os seguintes exercícios devem ser entregues via SuSy.
    - Exercício 1;
    - Exercício 2;
    - Exercício 4.
  - Veja os enunciados atualizados no site do sistema:
    - https://susy.ic.unicamp.br:9999/si100a (Turma A);
    - https://susy.ic.unicamp.br:9999/si100b (Turma B).

Referências

Referências

- MIZRAHI, V. V., *Treinamento em Linguagem C – Curso Completo*. 2a Edição, Pearson Makron Books, 2005.
- MIZRAHI, V. V., *Treinamento em Linguagem C – Curso Completo – Módulo 2*. 2a Edição, Pearson Makron Books, 2005.
- ASCENCIO, A. F. G. & DE CAMPOS, E. A. V., *Fundamentos da Programação de Computadores – Algoritmos, Pascal e C/C++*. Pearson Prentice Hall, 2003.
- The GNU C Library – Referência on-line: http://www.gnu.org/software/libc/manual/html_node/String-and-Array-Utilities.html