http://www.nickgentry.com/

Algoritmos e Programação de Computadores

Disciplina 113476

Prof. Alexandre Zaghetto
http://alexandre.zaghetto.com
zaghetto@unb.com

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Módulo 09
Strings
(Vetores de Caracteres)

1. Strings

• *String* é usada para armazenar e manipular textos como palavras, nomes e sentenças.

• Em C, não há tipo de dado ***string*** como em outras linguagens.

• Para contornar este problema, o tipo string foi definido como um conjunto de dados do mesmo tipo.

• Neste caso, estamos falando de vetores.

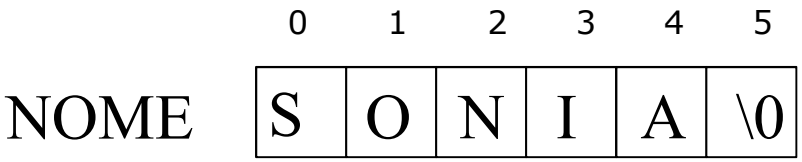
• Logo, uma string em C é definida por um vetor de elementos do tipo ***char.***

• Cada caractere de uma string pode ser acessado como um elemento de um vetor, proporcionando flexibilidade.

1. Strings

• Após o último elemento válido do vetor de caracteres (ou da string) tem-se o caractere NULL ('\0').

• Quando o usuário digita um valor a ser inserido em uma string, ao pressionar <enter>, o NULL ('\0') é inserido automaticamente após o último caractere válido.



2. Declaração de Strings

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

int main()
{
    char nome[50];


    return 0;
}
```

3. Inicialização de Strings

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

int main()
{
    char nome[50] = "Ana";


    return 0;
}
```

3. Inicialização de Strings

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

int main()
{
    char nome[] = "Ana";


    return 0;
}
```

3. Inicialização de Strings

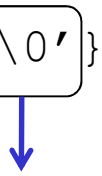
```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    char nome[50];

    nome = "Ana";


    return 0;
}
```

3. Inicialização de Strings

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

int main()
{
    char nome[50];

    nome = "Ana";     // Não está correto.


    return 0;
}
```

3. Inicialização de Strings

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

int main()
{
    char nome[50] = {'A','n','a','\0'};


    return 0;
}
```



Tem que colocar!

3. Inicialização de Strings

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

int main()
{
    char nome[] = {'A','n','a','\0'};

    return 0;
}
```

Tem que colocar!

3. Inicialização de Strings

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

int main()
{
   char nome[50];

   nome = {'A','n','a','\0'};


   return 0;
}
```

3. Inicialização de Strings

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

int main()
{
   char nome[50];

   nome = {'A','n','a','\0'}; // Não está correto.


   return 0;
}
```

4. Leitura e Impressão de Strings

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

int main()
{
    char nome[50];


    printf("Digite seu nome:");
    scanf("%s", &nome[0]);

    printf("%s\n\n", nome);


    return 0;
}
```

4. Leitura e Impressão de Strings

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

int main()
{
    char nome[50];


    printf("Digite seu nome:");
    scanf("%s", &nome[0]);

    printf("%s\n\n", nome);


    return 0;
}
```

O problema com a leitura realizada dessa forma é que o caractere NULL é inserido no primeiro "espaço".

4. Leitura e Impressão de Strings

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

int main()
{
    char nome[50];


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

    printf("%s\n\n", nome);


    return 0;
}
```

4. Leitura e Impressão de Strings

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

int main()
{
    char nome[50];

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

    printf("%s\n\n", nome);


    return 0;
}
```

O problema com a leitura é o mesmo que o anterior.

4. Leitura e Impressão de Strings

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

int main()
{
    char nome[50];

    printf("Digite seu nome:");
    scanf("%[^\n]", nome);

    printf("%s\n\n", nome);


    return 0;
}
```

Alternativa para superar o problema com a leitura.

4. Leitura e Impressão de Strings

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

int main()
{
    char nome[50];

    printf("Digite seu nome: ");
    gets(nome);
    printf("O nome digitado foi: ");
    puts(nome);


    return 0;
}
```

Alternativa para superar o problema com a leitura.

5. Vetores de Strings

• Da mesma forma que uma *string* é um vetor de caracteres, podemos ter um vetor de de strings, ou seja, uma matriz de caracteres.

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

int main()
{
    char nome[3][5]={"ana", "beto", "eva"};
    printf("%s\n",&nome[0][0]);
    printf("%s\n",&nome[1][0]);
    printf("%s\n",&nome[2][0]);


    return 0;
}
```

6. Funções de Manipulação de Strings

- strlen()
- strcmp()
- strcpy()
- strcat()
- Requerem a inclusão de <string.h>.

6. Funções de Manipulação de Strings

- strlen(): retorna o tamanho da string.

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

int main()
{
    char nome[20];
    int tamanho;

    printf("Digite nome:");
    gets(nome);
    tamanho = strlen(nome);
    printf("O tamanho eh: %d \n\n",tamanho);


    return 0;
}
```

6. Funções de Manipulação de Strings

• strcmp(): compara duas strings e retorna 0 se forem iguais e outro valor se forem diferentes.

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

int main()
{
    char nome1[20] = "zagh";
    char nome2[20] = "zagh";
    int compara;

    compara = strcmp(nome1, nome2);
    printf("Comparacao: %d \n\n",compara);


    return 0;
}
```

6. Funções de Manipulação de Strings

- strcpy(): copia uma string para outra.

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

int main()
{
    char nome1[20] = "Zaghetto";
    char nome2[20];

    strcpy(nome2, nome1);
    printf("nome2: %s \n\n",nome2);

    system("PAUSE");
    return 0;
}
```

6. Funções de Manipulação de Strings

- strcat(): concatena duas strings.

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

int main()
{

     char nome1[20] = "Alexandre", nome2[20] = "Zaghetto";
     char nomecompleto[50];

     strcpy(nomecompleto, nome1);
     strcat(nomecompleto, " ");
     strcat(nomecompleto, nome2);
     printf("Nome: %s \n\n", nomecompleto);

     system("PAUSE");
     return 0;
}
```

7. Outras Funções Interessantes

- tolower()
- toupper()
- Requerem a inclusão de <ctype.h>.

7. Outras Funções Interessantes

- tolower(): converte um caractere para caixa baixa.

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

int main()
{

char nome_alta[20] = "ALEXANDRE", nome_baixa[20];
int i;

for (i=0; i<strlen(nome_alta);i++)
     nome_baixa[i] = tolower(nome_alta[i]);

     nome_baixa[strlen(nome_alta)] = '\0';
```

7. Outras Funções Interessantes

- tolower(): converte um caractere para caixa baixa.

```
printf("Nome em caixa alta: %s \n", nome_alta);
printf("Nome em caixa baixa: %s \n", nome_baixa);


return 0;

}
```

7. Outras Funções Interessantes

- toupper(): converte um caractere para caixa alta.

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

int main()
{

char nome_baixa[20] = "alexandre", nome_alta[20];
int i;

for (i=0; i<strlen(nome_baixa);i++)
     nome_alta[i] = toupper(nome_baixa[i]);

     nome_alta[strlen(nome_baixa)] = '\0';
```

7. Outras Funções Interessantes

- toupper(): converte um caractere para caixa alta.

```
printf("Nome em caixa alta: %s \n", nome_alta);
printf("Nome em caixa baixa: %s \n", nome_baixa);


return 0;
}
```

"A humilhação cosmológica: Nosso mundo é uma das incontáveis esferas girando em um espaço infinito sobre a qual se desenvolveu 'um revestimento embolorado que é capaz de viver e perceber.' A humilhação biológica: O ser humano é um animal cuja inteligência serve exclusivamente para compensar perante o mundo sua falta de instintos desenvolvidos e seu ajustamento orgânico defeituoso. A humilhação psicológica: Nosso Eu consciente não é o senhor de sua própria casa."

Rüdiger Safransk,
em Schopenhauer
e os anos mais selvagens da filosofia.