

Linguagem C: strings

Prof. Críston
Algoritmos e Programação

String

- Uma string é um vetor de caracteres (texto)
`char nome [tamanho];`
- O texto que vamos gravar em uma string não precisa ocupar todos os caracteres do vetor
 - Por isso, utiliza-se o `'\0'` para indicar onde o texto termina

- Exemplo:

```
char nome[10];  
nome[0] = 'J';  
nome[1] = 'o';  
nome[2] = 'ã';  
nome[3] = 'o';  
nome[4] = '\0';
```

Inicialização de strings na declaração

```
char nome[ ] = { 'J', 'o', 'ã', 'o', '\\0' };  
char nome[7] = "João";  
char nome[ ] = "João";
```

- Automaticamente coloca o '\\0' na posição 4
- Esta atribuição pode ser feita **APENAS** na declaração!

```
nome = "Maria"; // NÃO PERMITIDO
```

Lendo strings pela entrada padrão

- Utilizamos %s no scanf

- Exemplo:

```
char nome[10];  
scanf("%s", &nome[0]);
```

- Podemos também utilizar apenas o nome do vetor, pois ele representa um ponteiro para o primeiro elemento

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

Escrevendo strings na saída padrão

- Utilizamos %s no printf
- Neste caso o printf espera o endereço de memória do vetor de caracteres
- Exemplo:

```
char nome[] = "Joao";  
printf("%s\n", nome);  
printf("%s\n", &nome[0]);
```

Exercício

- Leia um texto pela entrada padrão com no máximo 99 caracteres. Em seguida imprima o número de caracteres digitados.
 - Dica: percorra o vetor até encontrar o caracter terminador '\0'

Exercício

```
main()  
{  
    char texto[100], i;  
    scanf("%s", texto);  
    for (i=0; texto[i] != '\0'; i++);  
    printf("%d\n", i);  
}
```

Exercício

- Declare duas strings com capacidade para 20 caracteres. Leia pela entrada padrão a primeira string. Em seguida, copie o texto da primeira string para a segunda. Imprima no final o conteúdo das duas strings.

Exercício

```
main()  
{  
    int i;  
    char s1[20], s2[20];  
    scanf("%s", s1);  
    for (i=0; s1[i] != '\0'; i++)  
        s2[i] = s1[i];  
    s2[i] = '\0';  
    printf("s1 = %s\ns2 = %s\n", s1, s2);  
}
```

- Experimente remover a linha `s2[i] = '\0';`

Vetor de strings

- Como cada string é um vetor de caracteres, um vetor de strings é uma matriz de caracteres

```
char nome [num_strings] [tam_strings];
```

- Desta forma, para acessar a i-ésima string usamos

```
nome [i];
```

Exemplo

- Leia 5 strings e depois imprima cada uma delas.

```
main()  
{  
    char strings [5][100];  
    int i;  
    for (i=0; i < 5; i++)  
    {  
        printf ("Digite uma string: ");  
        scanf ("%s", strings[i]);  
    }  
    for (i=0; i < 5; i++)  
        printf ("%s\n", strings[i]);  
}
```

Exercício

- Depois de ler as 5 strings, leia mais uma string e verifique se ela está no vetor. Caso esteja, indique o índice no vetor.

Exercício

```
main()
{
    char strings[5][100], s[100];
    int i, j;
    for (i=0; i < 5; i++)
    {
        printf ("Digite uma string: ");
        scanf ("%s", strings[i]);
    }
    printf("String para localizar: ");
    scanf ("%s", s);
    for (i=0; i < 5; i++)
    {
        for (j=0; (strings[i][j] != '\0') && (strings[i][j] == s[j]); j++);
        if ((strings[i][j] == '\0') && (s[j] == '\0'))
            printf("string encontrada na posicao %d\n", i);
    }
}
```

Linguagem C: funções da biblioteca padrão para manipulação de strings

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

Lendo um string da entrada padrão, com limite de caracteres

```
fgets(string, tamanho, stdin);
```

- Lê da entrada padrão até tamanho-1 caracteres e copia para a string s
- Interrompe a leitura se encontrar '\n' ou fim de arquivo
 - Não interrompe com espaço (ao contrário do scanf)
- Mais seguro que o scanf, pois evita que os caracteres lidos sejam gravados depois do final do vetor

```
main()  
{  
    char s[10];  
    fgets(s, 10, stdin);  
    printf("%s\n", s);  
}
```

Copiando uma string para outra

`strcpy(string_destino, string_origem);`

- Copia `string_origem` para `string_destino`

```
main()
{
    char s1[20], s2[20];
    scanf("%s", s1);

    strcpy(s2, s1); // copia s1 para s2

    printf("s1 = %s\ns2 = %s\n", s1, s2);
}
```


Tamanho de uma string

`strlen(string);`

- Retorna o número de caracteres até o terminador
- O terminador não é contabilizado

```
main()  
{  
    char s[20];  
    scanf("%s", s);  
    printf("Voce digitou %d caracteres.\n", strlen(s));  
}
```

Concatenação de duas strings

```
strcat(string1, string2);
```

- Copia o conteúdo de string2 para o final de string1
- String2 permanece inalterada

```
main()
```

```
{
```

```
    char s1[20], s2[20];
```

```
    printf("s1 = "); scanf("%s", s1);
```

```
    printf("s2 = "); scanf("%s", s2);
```

```
    strcat(s1, s2);
```

```
    printf("s1 = %s\ns2 = %s\n", s1, s2);
```

```
}
```

Comparação de duas strings

```
strcmp(string1, string2);
```

- Retorna 0 se as duas strings são iguais
- Retorna -1 se a posição de string1 no dicionário é MENOR que a posição de string2
- Retorna +1 se a posição de string1 no dicionário é MAIOR que a posição de string2

```
main()
```

```
{
```

```
    char s1[20], s2[20];
```

```
    printf("s1 = "); scanf("%s", s1);
```

```
    printf("s2 = "); scanf("%s", s2);
```

```
    printf("comparacao de s1 com s2 = %d\n", strcmp(s1,s2));
```

```
}
```

Exercício

- Leia duas strings. Se as strings forem iguais escreva “strings iguais”. Caso contrário, concatene as duas strings e imprima a string resultante.

Exercício

```
main()
{
    char s1[20], s2[20];
    printf("s1 = "); scanf("%s", s1);
    printf("s2 = "); scanf("%s", s2);
    if (strcmp(s1, s2) == 0)
        printf("strings iguais!\n");
    else
    {
        strcat(s1, s2);
        printf("%s\n", s1);
    }
}
```

Entrada/saída formatada sobre strings

```
sprintf(string, formatação, argumentos);
```

```
sscanf (string, formatação, argumentos);
```

- Ao invés de imprimir/ler na saída/entrada padrão, este comando imprime/lê em/de uma string
- Muito utilizadas para converter de string para um formato numérico e vice-versa

Exemplo

```
main()  
{  
    int n;  
    char s[20];  
    printf("Entre valor inteiro: ");  
    scanf("%d", &n);  
  
    sprintf(s, "%d", n); // converte para string  
  
    printf("string: %s\n", s);  
}
```

Exemplo

```
main()  
{  
    int n; double x;  
    char s[] = "10 5.8";  
  
    // converte a string para um inteiro e um double  
    sscanf(s, "%d %lf", &n, &x);  
  
    printf("n = %d\nx = %f\n", n, x);  
}
```