UNESPAR
Universidade Estadual do Paraná

Algoritmos e Técnicas de Programação

Aula 3 - Cadeia de Caracteres

Prof. Guilherme Guerino

Introdução

- A linguagem C utiliza **vetores** para armazenar cadeias de caracteres. Cada **posição do vetor armazena um caractere**.

- Cadeias de caracteres também são conhecidas como **strings**.

Introdução

- Uma variável simples quando é declarada, pode ocupar **qualquer posição** disponível na memória.

- A situação é **diferente** com cadeias de caracteres. Assim como os vetores, uma cadeia ocupa posições **sequenciais de memória**, com cada caractere ocupando **1 byte**.

Introdução

- Para manipular as cadeias de caracteres na linguagem C, deve-se utilizar funções específicas.

- Essas funções estão declaradas na biblioteca <string.h>.

Introdução

- Mais informações sobre a biblioteca podem ser encontradas no seguinte endereço:

```
http://www.csse.uwa.edu.au/programming/ansic-library.html
```

Introdução

- Os compiladores da linguagem C identificam o fim de uma cadeia por meio do **caractere nulo**, representado por **‘\0’**.

- Por conta do caractere nulo, é importante sempre declarar o vetor com uma **posição a mais**.

Introdução

- As variáveis que armazenam cadeias de caracteres podem ser **inicializadas pelo programa** ou podem **receber um valor por meio do teclado**.

Inicializando cadeia de caracteres - Declaração

```
char palavra[ ]={'T', 'e', 's', 't', 'e', '\0'};
```

- Nesse exemplo, a variável *palavra* recebeu as letras **separadamente**, além do caractere nulo.

- Por isso, cada letra está entre **apóstrofos** (‘ ‘), que é a maneira de identificar um caractere específico.

Inicializando cadeia de caracteres - Declaração

```
char descricao[ ]="Teste";
```

- Nesse exemplo, a variável descricao foi inicializada com a palavra teste. **O caractere nulo foi considerado automaticamente.**

- Por isso a palavra é apresentada entre **aspas** (“). Essa é a maneira de identificar uma cadeia de caracteres.

Inicializando cadeia de caracteres - Declaração

```
char descricao[ ]="Teste";
```

- Nesse exemplo, a variável descricao foi inicializada com a palavra teste. **O caractere nulo foi considerado automaticamente.**

- Por isso a palavra é apresentada entre **aspas** (“). Essa é a maneira de identificar uma cadeia de caracteres.

Inicializando cadeia de caracteres - Atribuição após declaração

```
char vetor1[10], vetor2[10];
strcpy(vetor1,"Programa");
strcpy(vetor2, vetor1);
```

Inicializando cadeia de caracteres - Atribuição pelo teclado

```
char nome[30];
printf("Digite o seu nome");
scanf("%s", &nome);
```

Inicializando cadeia de caracteres - Atribuição pelo teclado

- O **scanf( )** é utilizado para **receber** dados de entrada, oriundos de diferentes fontes, tal como o teclado.

- No caso de uma cadeia de caracteres, o scanf( ) armazena todos os símbolos digitados **até a ocorrência do primeiro espaço em branco**.

Inicializando cadeia de caracteres - Atribuição pelo teclado

- Caso um nome composto seja digitado, por exemplo, **Francisvaldo da Silva**, a variável nome armazenará somente Francisvaldo.

- O que vem depois do espaço em branco **não é considerado**.

Inicializando cadeia de caracteres - Atribuição pelo teclado

- Para resolver esse problema, tem-se a função **gets( )**.

```
char nome[30];
printf("Digite o seu nome");
gets(nome);
```

Inicializando cadeia de caracteres - Atribuição pelo teclado

- A função gets( ) armazena na variável nome todos os símbolos digitados até a **ocorrência do ENTER**.

- Esta função exige a utilização da biblioteca **<stdio.h>**.

Outras operações

- Diversas operações podem ser realizadas com cadeias de caracteres.

- As principais operações envolvem: **cópia, concatenação, comparação e verificação do tamanho da cadeia...**

Outras operações

- A função **strcpy** copia os caracteres da cadeia string2 para a cadeia string1.

```
strcpy(string1, string2);
```

- A função **strncpy** copia os n primeiros caracteres da cadeia string2 para a cadeia string1.

```
strncpy(string1, string2, n);
```

Outras operações

- A função **strcat** concatena a cadeia string2 na string1.

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

- A função **strncat** concatena os n primeiros caracteres da cadeia string2 na string1.

```
strncat(string1, string2, n);
```

Outras operações

- A função **strcmp** compara duas cadeias de caracteres e retorna um número inteiro, que poderá ser:

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

Outras operações

- A função **strcmp** compara duas cadeias de caracteres e retorna um número inteiro, que poderá ser:
  - **zero**, se as duas cadeias forem iguais.
  - **um número menor que zero**, se a string1 for alfabeticamente menor que a string2.
  - **um número maior que zero**, se a string1 for alfabeticamente maior que a string2.

Outras operações

- A função **strcmp** considera que letras **maiúsculas e minúsculas são diferentes**.

Outras operações

- A função **strncmp** compara duas cadeias de caracteres, **da primeira posição até a posição n, ou até encontrar alguma diferença**.

- Essa função considera que letras **maiúsculas e minúsculas são diferentes**.

```
Resultado = strncmp(string1, string2, n);
```

Outras operações

- A função **strncmp** retorna um número inteiro que poderá ser:
  - **zero**, se as duas cadeias forem iguais.
  - **um número menor que zero**, se a string1 for alfabeticamente menor que a string2.
  - **um número maior que zero**, se a string1 for alfabeticamente maior que a string2.

Outras operações

- A função **strlen** retorna a quantidade de caracteres da cadeia.

- O valor de retorno pode ser armazenado em uma variável.

```
tamanho = strlen(cadeia);
```

Outras operações

- A função **stricmp** compara duas cadeias de caracteres e retorna um número inteiro.

- Essa função considera que **letras maiúsculas e minúsculas são iguais**.

```
Resultado = stricmp(string1, string2);
```

Outras operações

- A função **stricmp** compara duas cadeias de caracteres e retorna um número inteiro. Possíveis resultados:
  - **zero**, se as duas cadeias forem iguais.
  - **um número menor que zero**, se a string1 for alfabeticamente menor que a string2.
  - **um número maior que zero**, se a string1 for alfabeticamente maior que a string2.

Outras operações

- A função **strset** substitui todos os caracteres da cadeia str pelo caractere *h*.

```
strset(str, 'h');
strnset(str, 'h', n);
```

- A função **strnset** substitui os n primeiros caracteres da cadeia str pelo caractere *h*.

Outras operações

- A função **strrev** inverte todos os caracteres da cadeia str.

```
strrev(str);
```

Outras operações

- É possível realizar diferentes conversões de caracteres. Exemplos:
  - de maiúsculo para minúsculo
  - de minúsculo para maiúsculo
  - de valor numérico para caractere
  - de caractere para valor numérico

Outras operações

- A função **toupper** converte o caractere da cadeia, da posição especificada, para maiúsculo.

```
cadeia[posicao] = toupper (cadeia[posicao]);
```

- Esta função exige a biblioteca **<ctype.h>**.

Outras operações

- A função **strupr** converte todos os caracteres da cadeia para maiúsculo.

```
strupr(cadeia);
```

- Esta função exige a biblioteca **<string.h>**.

Outras operações

- A função **tolower** converte o caractere da cadeia, da posição especificada, para minúsculo.

```
cadeia[posicao] = tolower (cadeia[posicao]);
```

- Esta função exige a biblioteca **<ctype.h>**.

Outras operações

- A função **strlwr** converte todos os caracteres da cadeia para minúsculo.

```
strlwr(cadeia);
```

- Esta função exige a biblioteca **<string.h>**.

Dúvidas?