STRINGS PROGRAMAÇÃO

PROJETO AMPERE

STRINGS

O QUE SÃO? O QUE COMEM? ONDE VIVEM?

Uma string é um conjunto de caracteres armazenados num vetor. É um caso especial de vetor unidimensional de *char*.

STRINGS X CHARS

QUAIS AS DIFERENÇAS?

Enquanto um *char* é um único caractere, uma string é um conjunto (vetor) de pelo menos dois caracteres.

Em C, as strings são representadas utilizando aspas duplas, enquanto os caracteres são representados usando aspas simples.

STRINGS X CHARS

EXEMPLOS

- Strings:
  - “Luís”
  - “Zé Manuel Saraiva de Carvalho”
  - “Bolo de Chocolate com 1,2 kg de peso”
  - “A”

STRINGS X CHARS

EXEMPLOS

- Chars:
  - ‘L’
  - ‘>’
  - ‘[‘
  - ‘e’
  - ‘A’

STRINGS X VETORES

QUAIS AS DIFERENÇAS?

Toda string é um vetor de caracteres, mas nem todo vetor de caracteres é uma string.

STRINGS X VETORES

SITUAÇÃO-PROBLEMA

Suponhamos, se declararmos um vetor chamado *v* com 100 posições para o nome e lá colocarmos "Zé". Como poderemos saber quais e quantos dos 100 caracteres estamos efetivamente utilizando?

Z	é	...	...	...	...
v[0]	v[1]	v[2]	...	v[98]	v[99]

DELIMITADOR DE STRING

A.K.A. "STRING-TERMINATOR"

A solução adotada foi colocar um marcador (um *char*) que indique o fim da string dentro do vetor.

Mas a questão é, qual caractere vamos usar? Bom, imagine que escolhêssemos o asterisco (*) como delimitador.

DELIMITADOR DE STRING

COMO FUNCIONA?

Nesse caso, a parte válida da string estaria à esquerda do delimitador (*), e tudo que estivesse à direita seria "lixo" – caracteres quaisquer, sem significado.

Z	é	*	<lixo>	<lixo>	<lixo>
v[0]	v[1]	v[2]	...	v[98]	v[99]

DELIMITADOR DE STRING

EXEMPLOS

String “Zé”:

Z	é	*	a	*	b

String “Lui”:

L	u	i	*	s	a

DELIMITADOR DE STRING

PROBLEMA COM O ASTERISCO

Mas o problema com essa escolha é que, dessa forma, não poderíamos usar o * dentro das strings.

Por exemplo, não poderíamos escrever "***ERRO***". A string a seguir é a string vazia, "", pois o primeiro caractere que possui é o delimitador.

*	*	*	E	R	R	O	*	*	*

DELIMITADOR DE STRING

DELIMITADOR UNIVERSAL

Portanto, é necessário escolher um caractere que utilizemos usualmente.

Para cumprir essa função, o escolhido foi o caractere cujo código ASCII é igual a 0, representado por '\0' (que não possui nenhuma representação gráfica).

DELIMITADOR DE STRING

OBSERVAÇÃO IMPORTANTE

O caractere '\0' (código ASCII 0) não tem nada a ver com o caractere '0' (código ASCII 48).

Como sabemos, pode-se converter um *int* para um *char* que possua aquele inteiro como código ASCII. Assim, *char(0)* é igual a '\0' e *char(48)* é igual a '0'.

DELIMITADOR DE STRING

REPRESENTAÇÃO DEFINITIVA

String “Zé”:

Z	é	\0	<lixo>	<lixo>	<lixo>

String “**ERRO**”:

*	*	E	R	R	O	*	*	\0	<lixo>	<lixo>

STRINGS X VETORES

QUAIS AS DIFERENÇAS?

Uma string é toda a parte de um vetor de caracteres até a posição em que se encontre o delimitador '\0'.

Um vetor de caracteres que não possua '\0' não é nem contém uma string.

TAMANHO DE STRINGS

OBSERVAÇÃO IMPORTANTE

O caractere ‘\0’ deve ser levado em consideração na declaração do vetor. Uma string declarada como “char v[21]” é um vetor de caracteres com espaço para no máximo 20 caracteres normais e o ‘\0’ no final.

Assim, a string “Zé” deveria ser um vetor com 3 elementos: ‘Z’, ‘é’ e ‘\0’, respectivamente.

CARGA INICIAL AUTOMÁTICA

COMO FUNCIONA PARA STRINGS?

É feita da mesma forma que de vetores, porém com a possibilidade de uso da representação de strings em aspas duplas. Por exemplo:

- char nome[20] = “André”;
- char nome[20] = { ‘A’, ‘n’, ‘d’, ‘r’, ‘é’ };
- char nome [ ] = “André”;

CARGA INICIAL AUTOMÁTICA

char *nome[20]* ou *nome[ ]* = “André”;

Sempre que se inicia um vetor de caracteres utilizando a representação de strings em aspas duplas, o delimitador ‘\0’ é automaticamente inserido pelo compilador no final da string.

CARGA INICIAL AUTOMÁTICA

```
char nome[20] = { 'A', 'n', 'd', 'r', 'é' };
```

Quando se inicia um vetor de *char* utilizando a carga inicial entre chaves, o compilador completa os elementos não inicializados com o valor inteiro 0.

Como os elementos são todos *char*, quando o *int* 0 é inserido, ele é convertido para o char '\0' (afinal, *char(0)* é igual a '\0') e age como delimitador.

CARGA INICIAL AUTOMÁTICA

QUANDO NÃO É UMA STRING?

Perceba que na inicialização abaixo, o vetor vogais não é uma string. Isso acontece pois, como o número de elementos não foi especificado, o compilador vai criar o vetor com apenas os elementos que foram inicializados.

```
char vogais[ ] = { 'a', 'e', 'i', 'o', 'u' };
```

ESCRITA DE STRINGS

USO DE PRINTF

O argumento da função *printf* já é uma string por si só. Mas, além disso, podemos utilizar o especificador de formato *%s* para inserir outras strings dentro dele, como qualquer outra variável. Por exemplo:

```
char nome[ ] = "Lucas";
printf("Nome: %s\n", nome);
```

LEITURA DE STRINGS

USO DE SCANF

Funciona da mesma forma que qualquer outra variável, utilizando o especificador de formato *%s*, com o detalhe de que a variável que vai receber a string (o vetor de *char*) não deve ser precedida por um &.

A razão para isso será explicada daqui a algumas aulas, quando falarmos sobre Ponteiros.

LEITURA DE STRINGS

USO DE SCANF

A função scanf lê apenas uma palavra por vez. Ela lê todos os caracteres até encontrar um espaço, um tab ou enter. Em seguida, salva os caracteres lidos na variável, seguidos do caractere delimitador.

EXERCÍCIO 01

COMPARAÇÃO DE STRINGS

Escreva um programa que leia duas strings fornecidas pelo usuário e imprima “IGUAIS” caso as strings sejam idênticas, ou “DIFERENTES”, caso contrário.

EXERCÍCIO 02

TAMANHO DE STRINGS

Escreva um programa capaz de contar a quantidade de caracteres em uma string fornecida pelo usuário.

EXERCÍCIO 03

CONCATENAÇÃO DE STRINGS

Escreva um programa que concatene duas strings diferentes fornecidas pelo usuário.

EXERCÍCIO 04

CÓPIA DE STRINGS

Escreva uma função que permita copiar uma string de uma variável para outra.

EXERCÍCIO 05

MANIPULAÇÃO DE STRING

Escreva uma função que receba uma frase do usuário e a imprima em caps lock.