

AULA 13: STRINGS

Como sabemos não há um tipo de variável que pode definir palavras ou textos, logo a ideia de vetor para aplicar ao texto lógico.

→ Em C uma string é um vetor de caracteres que é terminado com um caractere nulo (`\0`);

→ No momento declarado, devemos atribuir o espaço reservado ao ~~vetor~~ caracteres;

Ex: `char str[11] = "teste";` (para guardar o `\0`)

T	E	S	T	E	\0					
0	1	2	3	4	5	6	7	8	9	10

→ Inputs:

• `gets (nome);`

↳ Problema: Não seguro, pode passar o tamanho máximo.

↳ Solução: Não usar, ele não garante o tamanho.
→ não há o `\0` no final e é vetor.

• `scanf ("%s", nome);`

↳ Problema: Não dá espaço e não garante tamanho.

• `fgets (nome, 20, stdin);`

↳ Problema: Garante o tamanho certo, mas adiciona `\n`.

16 • 05 • 24

D S T Q Q S S

↳ Solução: `none[strlen(nome, "\n") - 1] '\0'`
ret. null.

↳ char e string / n por \0.

Usa %s Para valores string de '\0'.

→ Vetor bidimensional: armazena dados extra, ex: 2D.

↳ Cada linha representa uma string;

↳ as colunas são os caracteres de cada string;

Parênteses / subscrito para acesso de repetição.

Em caso de erro de acesso, para apagar o buffer usa a função!

→ Biblioteca string:

Permite a manipulação de strings.

↳ `strlen("rea");` // retorna o tamanho da string.

↳ `strcpy("destino", "origem");` // copia a string.

↳ `strcmp("destino", "origem");` // compara a origem no destino.

↳ `strcmp("string1", "string2");` // compara as strings e
retorna 0 → iguais | < → string1 < string2

↳ `string1 > string2;`