Algoritmos e Programação de Computadores - APC

Manipulação de Strings

Prof. Daniel Sundfeld Lima
daniel.sundfeld@unb.br

Strings

- A manipulação de strings é diferente de tipos básicos.
- Vimos que não se pode fazer atribuições de strings:
  - char str[10] = “Minha string”
  - char str2[10];
  - str2 = str1;
- error: assignment to expression with array type

Strings

- Para manipular as strings, assim como vetores, precisamos adicionar alguns loops ou lógica mais complexa
- Para a manipulação de strings, o C inclui uma biblioteca <string.h> que permite a manipulação
- Essa biblioteca, recebe vetores de caracter como parâmetros, muitas vezes assume-se que terminam com '\0'

Strings

- Lembre-se que para evitar problemas de buffer overflow, o usuário precisa saber o final da string
- Isto é, saber onde o '\0' se encontra
- A função strlen pode ser usada para saber quantos caracteres existem no começo da string até o caractere final

Strings

- size_t strlen(const char *s);
- #include <string.h>
- char str[10] = “Olá!”;
  printf(“%lu\n”, strlen(str)); //Imprime 4

Cópia

- Copiando uma string
- char *strcpy(char *dest, char *src)
- A função strcpy copia os dados da string src para a string dest, incluindo o ‘\0’
- char str1[10], str2[10];
- strcpy(str2, str1);

Cópia

- Ao usar a strcpy, o programador precisa garantir que a string destino possui espaço suficiente para armazenar a string (use strlen antes se necessário)
- As strings src e dst não podem se sobrepor (overlap): um efeito possível com ponteiros ou matrizes de strings

Sobreposição de Strings com Matrizes

- char str[2][3] = {“12”, “34” };

1	2	\0
3	4	\0

1	2	\0	3	4	\0

- Str[0][2] = 2;

1	2	2
3	4	\0

1	2	2	3	4	\0

Exercício

- No caso anterior, o que imprime:
- printf(“%s\n”, str[0]);

Cópia

- Caso o usuário deseje também evitar qualquer chance de buffer overflow, pode-se utilizar a função strncpy:
- char *strncpy(char *dest, char *src, size_t n);
- A função irá copiar os dados até o tamanho n

Cópia

- Se não existir, ‘\0’ ele não será copiado
- Se encontrar ‘\0’, os caracteres entre ‘\0’ até o final do vetor ‘n’ também serão zerados
  - Isso pode causar um impacto indesejado na performance

Exercício

- Leia uma string do teclado, imprima o tamanho dela e faça a cópia para outra string

Concatenando

- A função strcat concatena uma string src na outra string dest (append)
- char *strcat(char *dest, char *src);
- char str1[20] = “Ola,”;
- char str2[20] = “mundo!”;
- strcat(str1, str2);
- A string str1 contém Ola,mundo!

Exercício

- Leia duas strings do teclado e faça a concatenação dela (use strcat. Não vale imprimir as duas)

Comparação

- A string.h também provê uma forma de comparar duas strings
- int strcmp(const char *s1, const char *s2);
- A função retorna:
  - 0 se forem iguais
  - Um número positivo se s1 > s2
  - Um número negativo se s1 < s2

Exercício

- Leia duas strings do teclado e imprima uma mensagem dizendo se as duas strings lidas são iguais ou diferentes

Sprintf

- É possível formatar um texto, em formato muito parecido com printf, mas armazenar a saída em uma string, e não na saída padrão
- Muito útil para formatar mensagens de erro e criar logs de execução
- sprintf(str, “Valor a: %d e Valor b: %d”, a, b);
- printf(str);

Sprintf

- Como sempre, devemos ter a preocupação de não exceder o tamanho da string
- snprintf pode ser usado nesse caso, ele recebe um número adicional indicando o tamanho
  - char str[20];
  - snprintf(str, 20, “Valor a: %d e valor b: %d”, a, b);

Exercício

- Faça um programa que leia duas strings (até 20 chars) do teclado. Para cada caractere da primeira string, imprima em **uma linha** o caracter e o valor ASCII dele
- Depois imprima em uma terceira string (100 chars) a primeira string, o tamanho dela entre parenteses, a segunda string e o tamanho dela entre parentes.
- Por último, imprime a terceira string na tela