Laboratório de Algoritmos I
Strings

Strings

- Um String é uma cadeia de caracteres. Ou seja, usamos strings para representar uma palavra, uma frase ou um texto qualquer.

- Em C, um string nada mais é que um vetor do tipo char. No entanto, nem todo vetor do tipo char é uma string! Uma string deve sempre ter o caracter NULL (‘\0’) como último caracter.

Strings - Declaração

- A declaração de uma string é feita da mesma forma que a declaração de um vetor de caracteres. Ela segue o seguinte formato:

```
char nome_da_string[tamanho];
```

- Devemos lembrar que o tamanho da string declarada deve incluir também o espaço para o ‘\0’. Se queremos, por exemplo, uma string de 100 caracteres, devemos declarar essa string com tamanho 101 (100 caracteres + ‘\0’).

Strings - Leitura

- Podemos fazer a leitura de strings de duas formas.
- Considerando que tenhamos declarado, por exemplo, a string char nome[80].
  - Podemos usar o scanf():

```
scanf("%s", nome); /*lê até o primeiro
                     espaço*/
```

  - Ou a função gets():

```
gets(nome); /*lê até a tecla ENTER*/
```

- Note que, para strings, **não** utilizamos o & no scanf!
- O gets() é a forma mais simples de pegar strings com espaços em branco.

Strings - Escrita

- Podemos fazer a escrita de strings de duas formas.
- Considerando que tenhamos declarado, por exemplo, a string char nome[80].
  - Podemos usar o printf():

```
printf("%s\n", nome);
```

  - Ou a função puts():

```
puts(nome); /* puts coloca um \n após a
               impressão*/
```

- Note que o puts() coloca automaticamente um '\n' após a impressão, de forma que os dois exemplos acima são equivalentes!

Strings - Inicialização

- A inicialização pode ser feita como em um vetor convencional:

```
char nome[] = {'A', 'n', 'a', '\0'};
char nome[4] = {'A', 'n', 'a', '\0'};
```

- Podemos também usar as aspas duplas:

```
char nome[] = "Ana";
char nome[4] = "Ana";
```

- Usando aspas duplas não é necessário incluir o '\0' diretamente, o compilador faz isso para você!

Strings – Erros comuns

- Strings não são tipos básicos de C!
  - Não podem ser usadas como int, float, char, etc.
- Não podemos fazer comparações a atribuições diretamente.

```
char nome1[30] = "Thiago"; /*Correto*/
char nome2[30] = "Joao"; /*Correto*/
nome1 = nome2; /* ERRADO! */
if (nome1 == nome2) { /* ERRADO! */
   …
}
```

- Como veremos mais tarde, existem outras formas de fazer comparações e atribuições entre strings.

Matrizes de Strings

- São como matrizes de caracteres. Podemos vê-las como:

```
char nome[num_de_strings][comprimento_das_strings]
```

- Exemplo:

```
char diadasemana[7][14] = {“Domingo”, ”Segunda-feira”, ”Terca-feira”, ”Quarta-feira”, ”Quinta-feira”, ”Sexta-feira”, ”Sabado”};
```

- Como acessar somente uma das strings?
  - Usando apenas o primeiro ínidce!
  - Para acessar o primeiro dia da semana da matriz acima:

```
diasdasemana[0];
```

Exemplo

- Exemplo: Vamos criar um programa que receba duas strings e copie a primeira para a segunda.
  - Como veremos posteriormente, existe uma função pronta para realizar esse procedimento. Nesse exemplo não utilizaremos essa função.

Exemplo

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]){
  char c1[101], c2[101];
  int i;
  printf("Entre com 2 palavras: ");
  scanf("%s %s", c1, c2);
  printf("palavra 1: %s\n", c1); // Verificacao de que as palavras
  printf("palavra 2: %s\n", c2); // foram lidas corretamente.
  i = 0;
  while (c1[i] != '\0') {
    c2[i] = c1[i];
    i++;
  }
  c2[i] = '\0';
  printf("palavra 1: %s\n", c1);     // Verificacao de que a palavra
  printf("nova palavra 2: %s\n", c2);// foi copiada.
  return 0;
}
```

Funções para manipulação de strings

- Strings são muito utilizadas, e muito importantes em C.
- Existe uma biblioteca que implementa várias funcionalidades importantes para trabalharmos com strings.
  - Biblioteca <string.h>.
- Sempre que usarmos qualquer uma das funções que serão descritas a seguir, devemos incluir no programa:

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

Função strlen

- Retorna um inteiro com o tamanho do string (sem contar o caracter NULL do final).

- Exemplo:

```
int tamanho;
char nome[] = "Paulo";
tamanho = strlen(nome); /* tamanho receberá
                           o valor 5 */
```

Função strcat

- Recebe dois strings e concatena o segundo no primeiro.

- Exemplo:

```
char nome1[80] = “Paulo”;
char nome2[80] = “ Souza”;
strcat (nome1, nome2); /* faz com que a
string nome1 tenha o valor “Paulo Souza”.
Não é verificado o tamanho de nome1.*/
```

Função strcmp

- Recebe dois strings e diz se são iguais. A função retorna:
  - 0 se os strings são iguais;
  - um número < 0 se o primeiro caracter diferente nos dois strings tem valor menor no primeiro string;
  - um número > 0 se o primeiro caracter diferente nos dois strings tem valor maior no primeiro string;

- Exemplo:

```
char nome1[80] = “Paulo”;
char nome2[80] = “Paulo”;
strcmp (nome1, nome2); /* retorna 0 se são iguais. Fazendo-se nome1 == nome2 seriam comparados os endereços dos vetores. */
```

Função strcpy

- Recebe dois strings e copia o segundo para o primeiro.

- Exemplo:

```
char nome1[80] = “Paulo”;
char nome2[80] = “Souza”;
strcpy (nome1, nome2); /* nome1 passa a
valer “Souza” */
```

Funções strncat, strncmp e strncpy

- O mesmo que as funções anteriores, no entanto tem um terceiro parâmetro que diz quantos caracteres serão processados (concatenados, comparados, copiados, etc).

```
char nome1[80] = "Paulo";
char nome2[80] = "Souza";
strncpy (nome1, nome2, 3); /* nome1 passa a
valer "Soulo" */
strncmp (nome1, nome2, 3); /* retorna 0,
uma vez que as 3 primeiras letras de
nome1(Soulo) são iguais às 3 primeiras de
nome2(Souza)*/
strncat (nome1, nome2, 3); /*nome1 passa a
valer "SouloSou" */
```