Linguagens de Programação I

Tema # 9

Cadeias de Caracteres - Strings

Susana M Iglesias

STRINGS - INTRODUÇÃO

- Strings (cadeia de caracteres): é uma serie de caracteres que podem ser tratados como uma unidade simples,
- Numerosas entidades do dia a dia são representados computacionalmente utilizando strings: nomes, endereços, números de telefones, CPF, ...
- Uma string pode incluir caracteres alfanuméricos (letras ou dígitos), e caracteres especiais (+, -, *, _, $, @, #),

STRINGS - INTRODUÇÃO

- Dados strings são utilizados em quase todas as aplicações computacionais (editores de texto, bancos de dados, redes, internet, ...)
- A diferença da maioria das linguagens de programação, a linguagem C não fornece um tipo string como tipo de dado básico. Limitação?
- Na linguagem C uma string é um ponteiro para seu primeiro caractere, i.e. o valor da string é o endereço do primeiro caractere,

STRINGS - INTRODUÇÃO

- Para manipular strings a linguagem C utiliza vetores de tipo caractere,
- STRING = conjunto de caracteres + caractere nulo
- caractere nulo = '\0', é utilizado para indicar o final da string,
- O caractere nulo esta na posição 0 (zero) da tabela ASCII,
- O caratere nulo é diferente do caractere zero (posição 48 da tabela ASCII).

CONSTANTES STRINGS

- Strings literais ou constantes de strings são escritas em C utilizando aspas duplas:

```
"Paulo P. Silva"
"Rua Alberto Rangel, s/n, Vila Nova, RJ"
"456.789.534.002"
```

- Como é armazenada uma constante string na memória?

DECLARANDO - STRINGS

- Declaração de uma string:

```
char cor[] = "azul";
char cor[] = {'a', 'z', 'u', 'l', '\0'};
char cor[5];
```

- Em C, `'a'` é diferente de `"a"`?

- Qual é a diferença entre um vetor de caracteres e uma string?

I/O (E/S) USANDO STRINGS

- **printf**

```
char cor[] = "azul";
printf("%s", cor);
```

- **scanf**

```
char cor[5];
printf("Digite uma cor: ");
scanf("%s", cor);
```

- a função **scanf()** lerá os caracteres até que um espaço ou um indicador de nova linha seja encontrado,
- Não é recomendável utilizarmos a função **scanf()** para leitura de strings.

I/O (E/S) USANDO STRINGS

- **`puts(string1)`**, imprime os caracteres contidos em **`string1`** seguidos de um caractere nova linha

```
char cor[] = "azul";
puts(cor);
```

- **`gets(string1)`**, obtém caracteres do dispositivo de entrada e os coloca em **`string1`** até que o caractere nova linha seja encontrado, adiciona o caractere nulo no final de **`string1`**.

```
char cor[5];
printf("Digite uma cor: ");
gets(cor);
```

I/O (E/S) USANDO STRINGS

- como as variáveis strings em C são vetores de caracteres, o nome do vetor e o subscrito correspondente pode ser utilizado para acessar cada caractere por separado,

```
char cor[] = "azul";
char ch;

ch = cor[0];
ch = cor[4];
ch = cor[10];
```

I/O (E/S) USANDO STRINGS

- as funções I/O para caracteres podem ser utilizadas para manipular strings,
- `getchar()`, obtém um caractere do dispositivo de entrada e retorna seu valor, lembre se ao criar uma string usando `getchar()` colocar o caractere `NULL` ao final da string,
- `putchar(c)`, envia o caractere c para o dispositivo de saída (video),

EXEMPLO 1

```
#define N 10

int main()
{
  char cor1[N];

  int i=0;

  printf("Digite uma cor: ");
  scanf("%s", cor1);
  printf("Cor: %s\n", cor1);

  while(cor1[i]!='\0')
    printf("%c\n", cor1[i++]);

  system("PAUSE");
  return 0;
}
```

```
Digite uma cor: verde
Cor: verde
v
e
r
d
e
Press any key to continue . . .
```

EXEMPLO 2

```
#define N 50

int main()
{
  char frase[N];
  int i=0;

  printf("Digite uma frase: ");
  gets(frase);

  printf("Vc digitou:\n");
  while(frase[i]!='\0'){
    if (frase[i]!=' ')
      putchar(frase[i]);
    else
      putchar('\n');
    i++;
  }
  printf("\n");
  return 0;
```

```
Digite uma frase: A vida e bela
Vc digitou:
A
vida
e
bela
Press any key to continue . . .
```

BIBLIOTECA DE MANIPULAÇÃO DE CARACTERES (`ctype.h`)

- **`int isdigit(int c)`**, retorna verdadeiro se c for um dígito,
- **`int isalpha(int c)`**, retorna verdadeiro se c for uma letra,
- **`int isalnum(int c)`**, retorna verdadeiro se c for um caractere alfanúmerico (letra ou dígito),
- **`int isspace(int c)`**, retorna verdadeiro se c for um espaço em branco (“ ”, “\n”, “\t”)
- **`int islower(int c)`**, retorna verdadeiro se c for uma letra minúscula,

BIBLIOTECA DE MANIPULAÇÃO DE CARACTERES (`ctype.h`)

- `int isupper(int c)`, retorna verdadeiro se c for uma letra maiúscula,
- `int tolower(int c)`, se c for uma letra maiúscula retorna a minúscula correspondente, senão retorna o mesmo caractere inalterado,
- `int toupper(int c)`, se c for uma letra minúscula retorna a maiúscula correspondente, senão retorna o mesmo caractere inalterado,

BIBLIOTECA DE MANIPULAÇÃO DE CARACTERES (`ctype.h`)

- Crie uma implementação própria para a função **`isalpha(char)`.**

```
int my_isalpha(char);
```

```
int my_isalpha(char c){
  if (c>61 && c<94)
    return 1;
  else
    return 0;
}
```

```
int my_isalpha(char c){
  return (c>61 && c<94);
}
```

BIBLIOTECA DE MANIPULAÇÃO DE STRINGS (`string.h`)

- `int strlen(char s1[])`, determina o comprimento da string `s1`. Retorna o número de caracteres que antecedem ao caractere `NULL`.

```
int main(){
  char str[] = "Brasil Hexacampeao";

  printf("Quantidade de caracteres: %d\n", strlen(str));

  system("PAUSE");
  return 0;
}
```

BIBLIOTECA DE MANIPULAÇÃO DE STRINGS (`string.h`)

- Crie uma função que determine o número de caracteres de uma *string* sem utilizar a função **`strlen`**().

```
int my_strlen(char []);
```

```
int my_strlen(char str[]){
  int i=0;

  while(str[i]!='\0')
    i++;

  return i;
}
```

FUNÇÕES DE CONVERSÃO DE STRINGS

- `int atoi(char s[])`, retorna o valor inteiro representado pela string `s`, se s não representar um valor inteiro gerara um erro e retorna zero,

- `double atof(char s[])`, retorna o valor de ponto flutuante representado pela string `s`, se s não representar um valor de ponto flutuante gerara um erro e retorna zero,

FUNÇÕES DE CONVERSÃO DE STRINGS

```c
int main(){
  char str1[] = "45622", str2[] = "3.14", str3[] = "abc.2";
  int a;
  float b;

  a = atoi(str1);
  b = atof(str2);
  printf("Valor a: %d\nValor b: %.2f\n", a, b);

  a = atoi(str3);
  b = atof(str3);
  printf("Valor a: %d\nValor b: %.2f\n", a, b);

  system("PAUSE");
  return 0;
}
```

```
Valor a: 45622
Valor b: 3.14
Valor a: 0
Valor b: 0.00
```

- Qual é a saída do programa?
- Como garantir que a funções sejam utilizadas com *strings* apropriadas?

BIBLIOTECA DE MANIPULAÇÃO DE STRINGS (`string.h`)

- **Exemplo**: Crie um programa que converta uma string que representa um valor inteiro positivo no número correspondente, se a string não representar um inteiro imprima uma mensagem de erro e aborte a execução do programa.

```c
int my_atoi(char []);

int my_atoi(char str[]){
  int val = 0, prod = 1, i;

  for (i=strlen(str)-1; i>=0; i--, prod*=10){
    if (str[i]<48 || str[i]>57){
      printf("A string nao representa um inteiro.\n");
      system("PAUSE");
      exit(-1);
    }
    val += (str[i]-48)*prod;
  }

  return val;
}
```

- Quais modificações seriam necessárias na função anterior para permitir inteiros negativos?

BIBLIOTECA DE MANIPULAÇÃO DE STRINGS (`string.h`)

- **`char * strcpy(char s1[], char s2[])`**, copia a string **`s2`** para a string **`s1`** e retorna um ponteiro a **`s1`**,

```
int main(){
  char str1[10]={}, str2[] = "Brasil";

  printf("%s\t%s\n", str1, str2);
  strcpy(str1, str2);
  printf("%s\t%s\n", str1, str2);

  system("PAUSE");
  return 0;
}
```

```
        Brasil
Brasil  Brasil
```

- A função **`strcpy()`** não realiza verificação de subscritos, tenha certeza que **`s1`** têm espaço suficiente para conter **`s2`**,

BIBLIOTECA DE MANIPULAÇÃO DE STRINGS (`string.h`)

- `char * strcat(char s1[], char s2[])`, concatena a string `s2` a string `s1`. O primeiro caractere de `s2` é sobrescrito ao caractere `NULL` de `s1`. O valor de `s1` é retornado,

```
int main(){
  char str1[20]="Brasil ", str2[] = "Hexacampeao";

  printf("%s\t%s\n", str1, str2);
  strcat(str1, str2);
  printf("%s\t%s\n", str1, str2);

  system("PAUSE");
  return 0;
}
```

```
Brasil   Hexacampeao
Brasil Hexacampeao      Hexacampeao
```

BIBLIOTECA DE MANIPULAÇÃO DE STRINGS (`string.h`)

```
char str1[]="ABC", str2[]="ABC";
if (str1==str2){
...
}
```

- Qual é o resultado da condição anterior?
- Como compararmos duas strings?
- **`int strcmp(char s1[], char s2[])`**, retorna 0 se a string **`s2`** é igual a string **`s1`**; -1 se **`s1<s2`** e 1 se **`s1>s2`**,

```c
int main(){
  char str1[20]="ABC", str2[]="ABC", str3[]="DEF";

  printf("Comparacao str1 e str2: %d\n", strcmp(str1, str2));
  printf("Comparacao str1 e str3: %d\n", strcmp(str1, str3));
  printf("Comparacao str3 e str1: %d\n", strcmp(str3, str1));

  system("PAUSE");
  return 0;
}
```

```
Comparacao str1 e str2: 0
Comparacao str1 e str3: -1
Comparacao str3 e str1: 1
```

- A função `strcmp()` pode ser utilizada para ordenar *strings* alfabeticamente.

BIBLIOTECA DE MANIPULAÇÃO DE STRINGS (`string.h`)

- Existem muitas outras funções de manipulação de strings, pesquise sobre elas:

- `char *strncat(char *str1, const char *str2, size_t n)`
- `int strncmp(const char *str1, const char *str2, size_t n)`
- `char *strncpy(char *str1, const char *str2, size_t n)`
- `char *strchr(const char *str, int c)`
- `char *strrchr(const char *str, int c)`

ESTOURO DE MEMORIA EM STRINGS

- A linguagem C não realiza nenhum controle sobre os subscritos de um vetor,
- Ao trabalharmos com strings devemos tomar cuidado para não estourar a memória reservada ao vetor,
- Funções de leitura e manipulação de strings não verificam os limites da memória alocada.

ESTOURO DE MEMORIA EM STRINGS

```c
#define N 20

int main(){
  char str[N];

  printf("Digite a string: ");
  gets(str);

  if(strlen(str)>N-1){
    printf("ERRO!!! Estouro de memoria.\n");
    return -1;
  }

  /*
    Processamento da string
  */

  system("PAUSE");
  return 0;
}
```