Programação Imperativa e Funcional

Strings

Diego Bezerra

dfb2@cesar.school

C.E.S.A.R school

Na aula anterior...

- Vetores e matrizes

Objetivos da aula

- Strings e funções de manipulação
- Tabela ASCII
- Exercícios

Strings

- Uma string é um array de caracteres terminado por um caractere nulo **'\0'** (código ASCII igual a 0)

‘s’	‘t’	‘r’	‘i’	‘n’	‘g’	‘\0’

- As funções básicas para manipulação de strings estão prototipadas na biblioteca **string.h**

Declaração de Strings

```c
#include <stdio.h>
#include <string.h> // necessária para funções com string

int main() {
    // declaração de variável
    char nome[100];
    // a maneira tediosa de declarar
    char nome1[] = {'j','o','s','e',' ','s','i','l','v','a','\o'};
    // a maneira mais fácil (como constante)
    char nome2[] = "jose silva";

    return 0;
}
```

Declaração e inicialização de Strings

- Necessário reservar uma posição no array para o caractere nulo
    - Ex.: Para uma string de tamanho 21, 20 são caracteres e 1 é reservado para o caractere nulo
- Quando declaradas como constantes (usando aspas) o compilador fica encarregado de alocar o caractere nulo

Tabela ASCII

- **Caracteres são**, na verdade, **números disfarçados** e seguem uma **codificação específica**. Uma pessoa pode decidir que o ’a’ será o 1, o ’b’ será o 2 e assim por diante. Mas como outra pessoa que receber a informação saberá disso? Para evitar este problema a representação de caracteres como números foi padronizada. Os principais padrões existentes são:
    - **ASCII** - American Standard Code for **I**nformation **I**nterchange
    - **EBCDIC** - Extended Binary Coded Decimal Interchange Code
    - **UNICODE**.

Imprimindo Strings

- Uma alternativa ao **printf** usando o formatador **%s** seria o procedimento **puts**

```
#include <stdio.h>
#include <string.h> // necessária para funções com string

int main() {
    char mensagem[100] = “Ola mundo!”;
    printf(“%s”, mensagem);

    return 0;
}
```

Lendo Strings pelo teclado

- O **scanf**() lê somente a primeira palavra até a ocorrência do separador
- Quando usamos vetores como argumento, o **&** não deve ser utilizado

```
#include <stdio.h>
#include <string.h>
int main() {

    char nome[100];
    scanf("%s", nome);
    printf("%s\n", nome);
    return 0;
}
```

Lendo Strings pelo teclado - Dica

- É possível configurar o scanf() para que seja possível ler além do caractere separador
    - Basta definir a a leitura até a ocorrência da quebra de linha "%[^\n]"

```
#include <stdio.h>
#include <string.h>
int main() {

    char nomeCompleto[100];
    scanf( "%[^\n]", nomeCompleto);
    return 0;
}
```

Lendo Strings pelo teclado - Dica

- É possível configurar o scanf() para que seja possível ler além do caractere separador
    - Basta definir a a leitura até a ocorrência da quebra de linha "**%[^\n]**"
    - Alternativamente podemos usar o procedimento **gets** e **puts** para saída. No entanto, elas não são funções seguras

Lendo Strings pelo teclado - Dica

```c
#include <stdio.h>
#include <string.h> // necessária para funções com string

int main() {
    char mensagem1[100];
    char mensagem2[100];

    scanf("%[^\n]", mensagem1);
    gets(mensagem2);

    printf("%s", mensagem1);
    puts(mensagem2);

    return 0;
}
```

Funções de String

Função	Retorno	Argumentos
strlen	Inteiro que representa o tamanho da String	String original
strcpy	Uma cópia da string original	String original e destino
strcmp	**0** se s1 == s2; **<0** se s1 < s2; **>0** se s1 > s2	String1 e string2

Funções de String

```
#include<stdio.h>
#include<string.h>
int main() {
    char nome1[100], nome2[100], nome3[100];
    int comprimento, resultado;
    scanf("%s", nome1);
    scanf("%s", nome2);
    /* Pegando tamanho da string */
    comprimento = strlen(nome1);
    /* Verificando se as strings são iguais */
    resultado = strcmp(nome1, nome2);
    /* Copiando string nome2 em nome3 */
    strcpy(nome3, nome2);
    return 0;
}
```

Mais funções de String

Função	Retorno	Argumentos
strchr	Primeira ocorrência de um caracter na string.	String original, caracter
strstr	Primeira ocorrência da string s2 em s1	String1 e string2
strupr	String com letras maiúsculas	String
strlwr	String com letras minúsculas	String

Biblioteca ctype.h

- A biblioteca ctype.h também oferece um conjunto de funções úteis para a manipulação de strings em C
    - Pode ser mais adequada quando se busca manipular os caracteres individualmente (isspace(), isalpha(), isdigit())
    - Já a biblioteca string.h pode ser mais adequada para manipular a string inteira (strupr(), strlwr())

Biblioteca ctype.h

Função	Descrição	Retorno
int toupper(int c)	Converte um caractere para maiúsculo, se for uma letra minúscula.	O caractere convertido ou o próprio caractere, se não for uma letra.
int tolower(int c)	Converte um caractere para minúsculo, se for uma letra maiúscula.	O caractere convertido ou o próprio caractere, se não for uma letra.
int isalpha(int c)	Verifica se o caractere é uma letra (A-Z ou a-z).	Retorna um valor diferente de zero (verdadeiro) se for uma letra; caso contrário, 0.
int isdigit(int c)	Verifica se o caractere é um dígito (0-9).	Retorna um valor diferente de zero se for um dígito; caso contrário, 0.
int isalnum(int c)	Verifica se o caractere é uma letra ou um dígito.	Retorna um valor diferente de zero se for alfanumérico; caso contrário, 0.
int isspace(int c)	Verifica se o caractere é um espaço em branco (espaço, tab, nova linha, etc.).	Retorna um valor diferente de zero se for um espaço em branco; caso contrário, 0.

Biblioteca ctype.h

Função	Descrição	Retorno
int isupper(int c)	Verifica se o caractere é uma letra maiúscula.	Retorna um valor diferente de zero se for uma letra maiúscula; caso contrário, 0.
int islower(int c)	Verifica se o caractere é uma letra minúscula.	Retorna um valor diferente de zero se for uma letra minúscula; caso contrário, 0.
int ispunct(int c)	Verifica se o caractere é um sinal de pontuação.	Retorna um valor diferente de zero se for um sinal de pontuação; caso contrário, 0.
int isxdigit(int c)	Verifica se o caractere é um dígito hexadecimal (0-9, a-f, A-F).	Retorna um valor diferente de zero se for um dígito hexadecimal; caso contrário, 0.

Exercícios

- Fazer um programa que leia uma lista de N nomes de pessoas de até 10 caracteres e imprima esta mesma lista de nomes todos em letras maiúsculas usando a biblioteca **ctype.h**.
- Fazer um programa que leia uma lista de N nomes de até 10 caracteres, calcule e imprima a quantidade de nomes masculinos, femininos e indefinidos, da seguinte forma:
  - Nome terminado com a letra ‘o’ - masculino;
  - Nome terminado com a letra ‘a’ - feminino;
  - Nomes terminados com outras letras - indefinido

Referências

- Rangel Netto, J. L. M., Cerqueira, R. D. G., & Celes Filho, W. (2004). **Introdução a estrutura de dados: com técnicas de programação em C**.