Linguagem C
String

Prof. Edwar Saliba Júnior
Maio de 2011

Manipulando Caracteres

- Se queremos ler um caractere, o código a seguir nos atende perfeitamente:

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

int main()
{
    char letra;

    printf("Digite um caractere: ");
    scanf("%c",&letra);

    printf("\n\nA letra digitada foi: %c", letra);

    return 0;
}
```

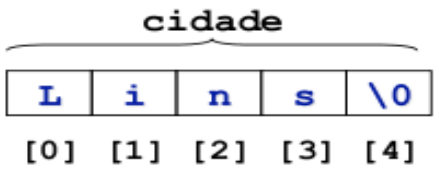
- E se quisermos ler uma sequência de caracteres?

Lendo Sequências de Caracteres

- Uma sequência de caracteres do tipo `char` é denomidada `string`;
- Uma `string` nada mais é do que um vetor de caracteres do tipo `char`;
- Toda string é terminada pelo caracter `'\0'`;
- Exemplo:

```
char cidade[5];

...
cidade[0] = 'L';
cidade[1] = 'i';
cidade[2] = 'n';
cidade[3] = 's';
```



Lendo *Strings*

- Ou você atribui posição por posição:

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

int main()
{
    char palavra[5];

    palavra[0] = 'L';
    palavra[1] = 'i';
    palavra[2] = 'n';
    palavra[3] = 's';
    palavra[4] = '\0';

    printf("\n\nA palavra digitada foi: %s", palavra);

    return 0;
}
```

Lendo *Strings*

- Ou você lê do teclado com o comando `gets`:

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

int main()
{
    char palavra[5];

    printf("Digite um caractere: ");
    gets(palavra);

    printf("\n\nA palavra digitada foi: %s", palavra);

    return 0;
}
```

Atribuindo e Copiando *Strings*

- Exemplo:

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

int main()
{
    // Atribuição.
    char palavra[] = "Lins";

    printf("\n\nA palavra é: %s", palavra);


    // Cópia.
    strcpy(palavra,"Itu");

    printf("\n\nA palavra é: %s", palavra);

    return 0;
}
```

- No caso de cópia, é responsabilidade do programador verificar se a variável comporta o tamanho de `string` que está sendo atribuída.

Exibindo *Strings*

- Comando `printf` e `puts`:

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

int main()
{
    char nome[] = "Paullo";
    char sobrenome[] = "Sampaio Penna";

    // Imprime em uma linha, com uma linha de comando.
    printf("%s %s \n\n", nome, sobrenome);

    // Imprime em diversas linhas com diversas linhas de comando.
    puts(nome);
    puts(sobrenome);

    return 0;
}
```

Funções Auxiliares

- `strlen(string)` – Retorna o número de caracteres existentes antes do `'\0'`;
- `strcpy(destino, origem)` – Atribui a uma variável do tipo `string` uma constante ou o valor de uma outra variável do tipo `string`;
- `strcmp(str1, str2)` – Compara duas variáveis ou valores do tipo `string`. Retorna maior que zero (`str1 > str2`), zero se `(str1 == str2)` e menor que zero se (`str1 < str2`);
- `strcat(str1, str2)` – Retorna a concatenação das `strings` passadas como parâmetro.

Exercício

- Implemente programas que simulem cada uma das funções apresentadas no *slide* anterior. Cada programa deverá possuir uma das funções apresentadas a seguir:
  - `int tamanho_string(char str[])` - deverá retornar o tamanho de uma `string` digitada pelo usuário;
  - `void copia_string(char str1[], char str2)` – deverá copiar o valor de `str2` para `str1` e mostrar o valor de `str2`;
  - `void concatena_string(char str1[], char str2[])` - deverá concatenar `str1` e `str2` e mostrar o valor da concatenação.

Bibliografia

- LAUREANO, Marcos. **Programação em C para ambiente Linux**. Disponível em: <http://br-c.org/doku.php>. Acesso em: 06 fev. 2011.
- MURTA, Cristina Duarte. *Slides* da disciplina de Programação de Computadores I. CEFET-MG, 2010.
- SENNE, Edson Luiz França. **Primeiro Curso de Programação em C**. 2. ed. Florianópolis: Visual Books, 2006.
- SOARES, Gabriela Eleutério. *Slides* da disciplina de Programação de Computadores I. CEFET-MG, 2011.