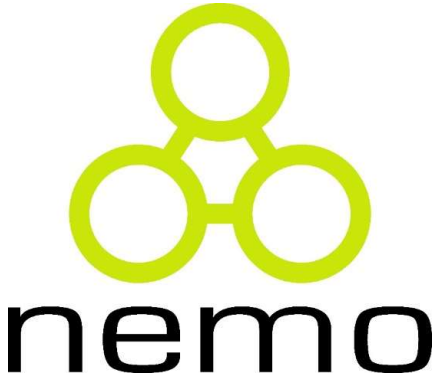


Programação 2

Jordana S. Salamon

jssalamon@inf.ufes.br

DEPARTAMENTO DE INFORMÁTICA

CENTRO TECNOLÓGICO

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

Strings

- Até agora, utilizáva-mos variáveis do tipo *char* para identificar letras do alfabeto;

- Mas como armazenar mais de um caracter?

Strings

- O termo string serve para identificar uma sequência de caracteres.
- Na prática, as strings são usadas para representar textos.
- Em linguagem C, ao contrário de outras linguagens, não existe um tipo de dados string nativo.
- Para representar uma string em C, devemos criar um vetor de caracteres, ou seja um vetor do tipo char.

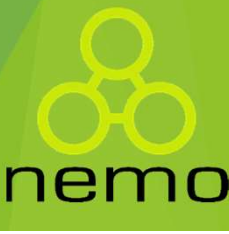
Strings

- Exemplo de declaração de string:

```
char nome_cliente[61];
```

- O último caractere de uma string deve ser sempre o caracter nulo “\0” que serve para indicar o final da string.

- Na prática teríamos então 60 espaços para armazenar o nome do cliente.



Strings

- **Inicializando o valor de strings**

```
char nome_cliente[30] = "Fulano";

char nome_cliente[30] = {'F','u','l','a','n','o'};
```

- Inicializando uma string sem definir o tamanho do vetor:

```
char nome_cliente[] = "Fulano";
```

- Neste caso, a quantidade de caracteres de armazenamento é calculada automaticamente de forma a ter a dimensão exata para conter a string que está sendo atribuída.

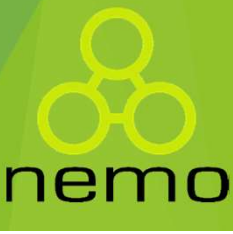
Strings

- **Lendo uma string em C**

- *Usando scanf*

- A função scanf permite fazer leitura de strings usando %s.

- Em relação ao uso de scanf para armazenar string devemos observar duas coisas:
  - A função scanf realiza a leitura até encontrar um espaço, depois encerra a leitura e coloca o caracter terminador \0.
  - A variável que vai armazenar a string não necessita ser precedida por &, porém se colocar não dará erro.



Strings

- *Usando scanf*

```c
#include<stdio.h>

int main(){

char nome[10];

printf("Digite seu nome:\n");
scanf(" %s", nome);

printf("Imprimindo seu nome:\n");
printf(" %s", nome);

return 0;
}
```

Strings

- *Usando fgets*

- Esta função armazena tudo que foi digitado, inclusive os espaços, até que a tecla ENTER seja pressionada.

```c
#include <stdio.h>

int main(){
char nome[10];

printf("Digite seu nome: \n");
fgets(nome, 10, stdin);

printf("Imprimindo seu nome: \n");
printf(" %s ", nome);
}
```

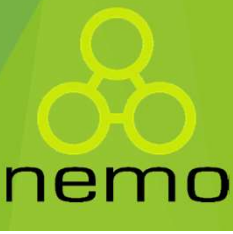
Strings

- **Biblioteca string.h**

- A linguagem fornece uma biblioteca chamada string.h que possui várias funções úteis para a manipulação de strings;

- **Strlen**: retorna o tamanho da string, sem contar o caracter indicador de final da string

```
char nome[15] = "Maria da Silva";
int s = strlen (nome);
// s conterá o valor 14
```



Strings

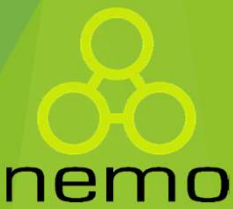
- **Biblioteca string.h**

- **Strcpy**: copia o conteúdo de uma string para outra e coloca um terminador de string.

```
char nome[]  = "Clarice Lispector";
char nome2[] = "Oswald de Andrade";
```

```
strcpy (nome, nome2);
// agora nome conterá "Oswald de Andrade"
```

- **Lembrando que o tamanho da string copiada deve ser menor ou igual ao da string de destino!**



Strings

- **Biblioteca string.h**

- **Strcat**: concatena duas strings, adicionando o conteúdo da segunda ao final da primeira, além do terminador (\0).

```
char nome[50] = "Maria";
char sobrenome[] = " da Silva";
strcat (nome, sobrenome);
// agora nome contém "Maria da Silva"
```

- Note que a primeira string deve ter espaço suficiente para conter a segunda.

Strings

- **Biblioteca string.h**

- **Strcmp**: Compara o tamanho de duas strings.

- Se você tentar criar duas strings com o mesmo conteúdo e compará-las como faria como números, verá que elas "não são iguais". Isso ocorre porque, na verdade, o que está sendo comparado são os endereços de memória onde estão guardadas as strings.

- O valor de retorno é:
  - menor que zero se s1 for menor que s2;
  - igual a zero se s1 e s2 são iguais;
  - maior que zero se s1 for maior que s2.

Strings

- **Biblioteca string.h**
- **Strcmp**: Compara o tamanho de duas strings.
- A comparação é entre a primeira letra que difere nas duas strings.
- Assim, se tivermos s1 = "abc" e s2 = "abd", s2 é maior que s1 pois na primeira posição em que as duas strings diferem, a letra em s2 é "maior".

Strings

- **Biblioteca string.h**
- **Strcmp**: Compara o tamanho de duas strings.
- A comparação é entre a primeira letra que difere nas duas strings.
- Assim, se tivermos s1 = "abc" e s2 = "abd", s2 é maior que s1 pois na primeira posição em que as duas strings diferem, a letra em s2 é "maior".

Strings

- **Biblioteca string.h**

- **Strupr** : converte uma string para maiúsculas;

- **Strlwr** : converte uma string para minúsculas;

```
#include< stdio.h>

main() {
   char s1[] = “abc";
   strlupr(s1);
   printf ("%s\n", s1);
}
```

```
#include< stdio.h>

main() {
   char s2[] = "ABC";
   strlwr(s2);
   printf ("%s\n", s2);
}
```

nemo

Exercícios

1. Faça um programa que leia um nome, calcule e retorne quantas letras tem esse nome.
2. Faça um programa que leia um nome e imprima o nome somente se a primeira letra do nome for ”a”(maiúscula ou minúscula).
3. Faça um programa que leia uma palavra e a imprima de trás para frente.
4. Faça um programa que leia uma palavra, calcule quantas vogais (a, e, i, o, u) possui essa palavra. Depois, leia um caractere (vogal ou consoante) e substitua todas as vogais da palavra dada por esse caractere.

That’s all Folks!