Strings no C

Profa.: Andréia Rodrigues Casare
E-mail: casareandreia@gmail.com

Trabalhando com Strings

- Trabalhar com *strings*, em C, não é nada óbvio.
- Essa dificuldade se torna ainda maior se você já tiver estudado outra linguagem de programação, como Java, Python, PHP, onde a manipulação de texto é extremamente óbvia e simples.

Biblioteca <string.h>

- Essa biblioteca fornece várias funções para trabalhar com as strings no C.
- strlen, strcmp, strcat, etc

Função strcmp

- Essa função compara duas strings segundo sua ordem alfabética e retorna um inteiro.
- Se for negativo, é porque a primeira string é menor que a segunda.
- Se for positivo, é porque a segunda string é maior que a segunda.
- Se retornar 0, é porque as strings são idênticas.
- Compara caractere por caractere começando do primeiro caractere até que os caracteres em ambas as strings sejam iguais ou um caractere NULL seja encontrado.
- Este processo continuará até que um caractere em qualquer string seja NULL ou os caracteres sejam desiguais.

Exemplo 1

```c
# include <stdio.h>
# include <string.h>
# include <locale.h>

int main(){
    setlocale(LC_ALL,"Portuguese");
    char msg[10], msg2[10];
    int resu;
    printf("Digite a 1a palavra:\n");
    gets(msg);
    printf("Digite a 2a palavra:\n");
    gets(msg2);
    resu=strcmp(msg,msg2);
    if (resu == 0){
        printf("as strings são iguais");
    }else{
        printf("as strings são diferentes");
    }
    return 0;
}
```

Exemplo 2

```
int main(){
    char str1[10], str2[10];
    int i, achou=2;
    printf("\n Entre com a 1a string: ");
    gets(str1);
    printf("\n Entre com a 2a string: ");
    gets(str2);
    for (i = 0; str1[i] == str2[i]; i++) {
        if (str1[i] == 0) {
            achou= 1;
        }
    }
    if (achou == 1){
        printf("\n As strings são iguais !!");
    }else{
        printf("\n As strins são diferentes");
    }
    return 0;
}
```

Exemplo 3

```c
int main() {
    setlocale(LC_ALL,"Portuguese");
    char aNomes[10][15];
    int i, j, achou;
    char nome[15];
    printf("Digite os 10 nomes:\n");
    for (i = 0; i < 10; i++) {
        strupr(gets(aNomes[i]));
    }
    while (1) {
        printf("Digite um nome para pesquisar (ou 'FIM' para sair):\n");
        strupr(gets(nome));
        if (strcmp(nome, "FIM") == 0) {
            break;
        }
        achou = 0;
        for (i = 0; i < 10; i++) {
            if (strcmp(nome, aNomes[i]) == 0) {
                printf("%s foi encontrado no índice %d\n", nome, i);
                achou = 1;
                break;
            }
        }
        if (!achou) {
            printf("%s não foi encontrado\n", nome);
        }
    }
    return 0;
}
```

Funções strupr() e strlwr()

- A função **strupr()** converte uma string em maiúsculas.
- A função **strlwr()** converte uma string em minúsculas.

- Obs: estas funções não fazem parte da biblioteca padrão da linguagem C, portanto elas funcionam apenas no SO Windows.

Função strlen()

- Retorna o tamanho de uma string.
- Sintaxe:
  - tam = strlen(string);
- Exemplo:

```
int main() {
    setlocale(LC_ALL,"Portuguese");
    int i, tam;
    char nome[15];
    printf("\n Digite o nome: ");
    gets(nome);
    tam= strlen(nome);

    printf("A palavra %s tem %i caracteres",nome,tam);
    return 0;
}
```

Funções toupper() e tolower()

- A função toupper() recebe como parâmetro um caracter e retorna sua versão em maiúsculo enquanto a função tolower() retorna a versão em minúsculo.

Exemplo toupper()

```
int main() {
    setlocale(LC_ALL,"Portuguese");
    int i, tam;
    char nome[15];
    printf("\n Digite o nome: ");
    gets(nome);
    i=0;
    while (nome[i] !='\0'){
        nome[i] = toupper(nome[i]);
        i++;
    }

    printf("\n %s",nome);
    return 0;
}
```

Exemplo tolower()

```
int main() {
    setlocale(LC_ALL,"Portuguese");
    int i, tam;
    char nome[15];
    printf("\n Digite o nome: ");
    gets(nome);
    i=0;
    while (nome[i] !='\0'){
        nome[i] = tolower(nome[i]);
        i++;
    }

    printf("\n %s",nome);
    return 0;
}
```

Função strcat()

- Concatena a segunda string no final da primeira string.
- Exemplo:

```
int main() {
    setlocale(LC_ALL,"Portuguese");
    char nome[30], sobrenome[15];
    printf("\n Digite o nome: ");
    gets(nome);
    printf("\n Digite o sobrenome: ");
    gets(sobrenome);
    strcat(nome," ");
    strcat(nome,sobrenome);

    printf("\n %s",nome);
    return 0;
}
```

Problema com scanf e gets

```
int main() {
    setlocale(LC_ALL,"Portuguese");
    int i, idade;
    char nome[15];
    printf("Digite os 5 nomes e idades:\n");
    for (i = 0; i < 5; i++) {
        printf("\n Digite o nome: ");
        gets(nome);
        printf("\n Digite a idade: ");
        scanf("%i",&idade);
    }
    return 0;
}
```

Problema com scanf e gets

- Quando usamos scanf, o programa salva o que foi digitado em um buffer de entrada, e depois grava o que estiver neste buffer na próxima variável.
- O principal problema com o scanf, é que ele deixa dentro do buffer o "ENTER" que você pressiona para confirmar o que acabou de digitar.
- Então, quando você digita a primeira idade, ele vai "consumir" o numero que você digitou, mas vai ficar um ENTER gravado no buffer de entrada; Quando o laço volta para "gets(nome)", o gets lê o ENTER que ficou alocado no buffer, e pula esse comando, deixando o nome vazio.
- O programa entende que você não quis colocar nome nenhum, e simplesmente digitou ENTER... Porém esse ENTER você digitou antes), e ele vai direto para o scanf("%d", &idade).

Função fflush(stdin)

- Para solucionar deve ser usado o comando fflush(stdin), para limpar buffer do teclado.

Exemplo com fflush(stdin)

```
int main() {
    setlocale(LC_ALL,"Portuguese");
    int i, idade;
    char nome[15];
    printf("Digite os 5 nomes e idades:\n");
    for (i = 0; i < 5; i++) {
        printf("\n Digite o nome: ");
        gets(nome);
        printf("\n Digite a idade: ");
        scanf("%i",&idade);
        fflush(stdin);
    }
    return 0;
}
```