



# SCC0221 - Introdução à Ciência de Computação I

---

**Prof.: Dr. Rudinei Goularte**

(rudinei@icmc.usp.br)

## Strings

Instituto de Ciências Matemáticas e de Computação - ICMC  
Sala 4-229



# 1. Introdução

---

- Uma **string** é um array unidimensional de caracteres.
- Strings terminam com o caracter especial '\0', ou caracter nulo (NULL character)
  - NULL possui 1 byte com todos os bits setados para zero.
- C não possui o **tipo de dado** string. Permite apenas constantes string.
- Strings podem ser constantes ou variáveis.
- Variáveis string podem ser alocadas na stack e também na heap.



## 2. Strings como constantes

---

- Contantes string
  - "a" é uma constante string -> 2 bytes
    - Termina com '\0'
  - 'a' é uma constante caracter -> 1 byte

```
int main (void){  
char* c = "a";  
printf("Isto é uma constante string \n");  
printf ("%s - %d", c, sizeof(c));  
    /*Imprime: a - 8*/  
}
```



## 2. Strings como constantes

---

- Constantes Strings como ponteiros para char.
  - `char* str;`  
`str = "abcd";`  
`printf("%s", str);`
    - Compilador aloca espaço para `str`, coloca a constante `"abcd"` em algum lugar da memória, faz `str` apontar para a constante `"abcd"`.
  - `scanf("%s", str);` */\*erro\*/*
    - `scanf` espera um `char[]` como parâmetro. `str` aqui é uma constante.



## 3. Strings como variáveis

---

- Definição de uma variável string de caracteres:

```
char str[100];
```

- Uma string é um vetor de caracteres terminado pelo caracter nulo `'\0'`.
- Deve-se reservar espaço no vetor para o caracter nulo.
- Uma string de 10 elementos: `char str2[11];`



## 3. Strings como variáveis

---

```
char str[5], str2[5];  
str[0] = 'a';  
str[1] = 'b';  
str[2] = 'c';  
printf("\ %s ",str);  
str2[0] = str[0];  
str2[1] = str[1];  
printf("\ %s ", str2);
```



## 3. Strings como variáveis

---

```
char str[6], str2[6];  
str[0] = 'a';  
str[1] = 'b';  
str[2] = 'c';  
str[3] = '\0';  
printf("%s",str);  
str2[0] = str[0];  
str2[1] = str[1];  
str2[2] = '\0';  
printf("%s", str2);
```

# 4. Funções para manipular strings



---

- Definidas em `<stdio.h>`:

`char str[80];`

- `gets(str);` – lê uma string do teclado e armazena em `str`.
  - Não usar! Unsafe!
- `puts(str);` – imprime o conteúdo de `str` no monitor.
- `printf("%s", str);`
- `scanf("%s", str)` – sem o `&!!!`



# 4. Funções para manipular strings

- Exemplos de uso (e de problemas) com scanf():

```
char str[20], str2[20];  
char c;  
scanf("%s", str);  
scanf("%c", &c);  
scanf("%s", str2);  
printf("%s %c %s", str, c, str2);
```

\* Ao executar o scanf o programa espera algo ser digitado. O usuário digita um valor e pressiona a tecla <Enter>. "Enter" é um caracter!

Solução: scanf(" %c", &c);



## 4. Funções para manipular strings

---

- Definidas em `<string.h>`:
  - `strcpy(s1,s2)` – copia `s2` em `s1`.
  - `strcat(s1, s2)` – concatena `s2` ao final de `s1`.
  - `strlen(s1)` – retorna o tamanho de `s1`.
  - `strcmp(s1,s2)` – retorna 0 se `s1` e `s2` são iguais; menor que 0 se `s1 < s2` (lexicograficamente); maior que 0 se `s1 > s2` (lexicograficamente).

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

```
int main (void){
    char str[20], str2[20];

    printf("Digite a frase 1:\n");
    scanf("%s", str);

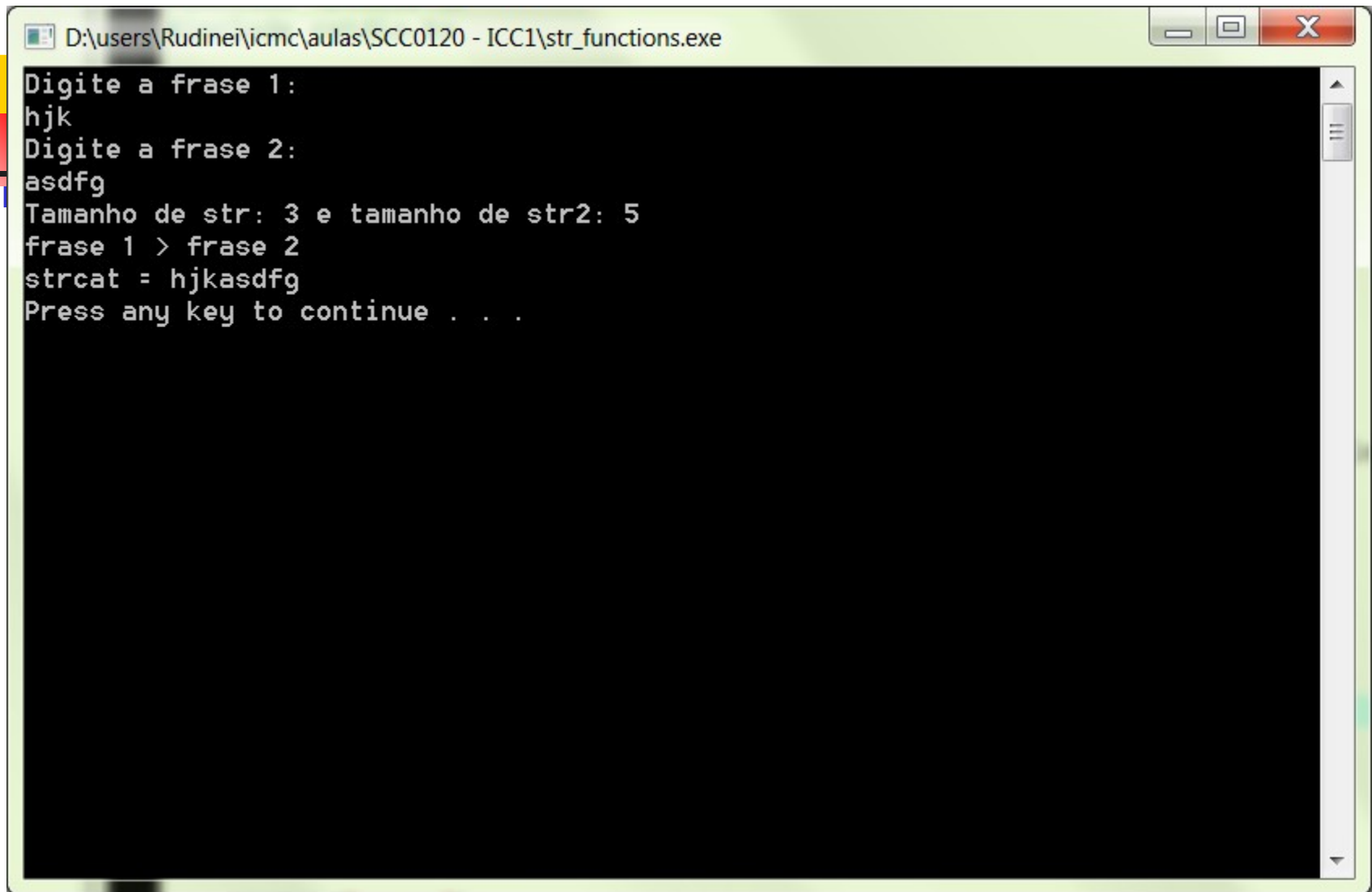
    printf("Digite a frase 2:\n");
    scanf("%s", str2);

    printf("Tamanho de str: %lu e tamanho de str2: %lu\n", strlen(str), strlen(str2));

    if (!strcmp(str, str2))
        printf("frase 1 = frase 2\n");
    else if (strcmp(str, str2) > 0)
        printf("frase 1 > frase 2\n");
    else
        printf("frase 1 < frase 2\n");

    strcat(str, str2);
    printf("strcat = %s\n", str);

    return(0);
}
```



```
D:\users\Rudinei\icmc\aulas\SCC0120 - ICC1\str_functions.exe
Digite a frase 1:
hjk
Digite a frase 2:
asdfg
Tamanho de str: 3 e tamanho de str2: 5
frase 1 > frase 2
strcat = hjkasdfg
Press any key to continue . . .
```



## 4. Funções para manipular strings

---

- No programa anterior, o que acontece se digitar frases com espaços?
- Testar!

# 4. Funções para manipular strings



---

- Solução:

...

```
printf("Digite a frase 1:\n");  
scanf("%[^\\n]s", str);
```

```
printf("Digite a frase 2:\n");  
scanf(" %[^\\n]s", str2);
```

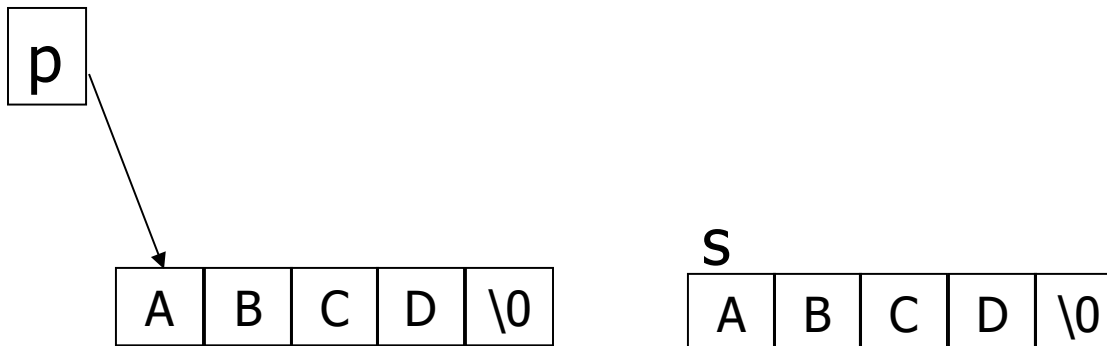
...



## 5. Strings e ponteiros

---

- `char *p = "ABCD";` e `char s[5] = "ABCD";`
  - `char` = 1 byte.
  - Um ponteiro = 8 bytes.





# Exercício

---

- Faça um programa que imprima o tamanho de uma string fornecida pelo usuário. Não use `strlen()`!



