

# Caracteres em C

---

- Entrada de caracteres

```
scanf("%c", &x);
```

```
x = getchar();
```

- Saída de caracteres

```
printf("%c", x);
```

```
putchar(x);
```

```
#include <stdio.h>

int main(){
    char x;

    printf("\ndigite um caractere: ");
    x = getchar();

    printf("\no caractere digitado eh: ");
    putchar(x);

    printf("\n\n");
    return 0;
}
```

```
digite um caractere: s
o caractere digitado eh: s
```

# Strings em C

---

- vetor de caracteres com terminador '\0'
- Ex: string **str** de 9 posições, com as posições de 0 a 6 ocupadas (demais posições com “lixo”)

|            | 0        | 1        | 2        | 3        | 4        | 5        | 6         | 7        | 8        |
|------------|----------|----------|----------|----------|----------|----------|-----------|----------|----------|
| <b>str</b> | <b>B</b> | <b>r</b> | <b>a</b> | <b>s</b> | <b>i</b> | <b>l</b> | <b>\0</b> | <b>?</b> | <b>?</b> |

Declaração:

```
char <identificador> [tamanho+1];
```

---

# Strings: inicialização

---

- **char nome[15] = "Ivo";**  
insere os caracteres entre "" na string nome, a partir da posição 0 e, ao final, acrescenta '\0' (no ex. posição 3)
- **char nome[15] = {'I', 'v', 'o'};**  
insere os caracteres entre {} a partir da posição 0. Se o tamanho da string > nro de caracteres armazenados, as demais posições são preenchidas com zeros
- **char nome[] = "Ivo";**      **// \*nome**  
determina nro de caracteres entre "", soma 1 e cria a string com esse tamanho

# Entrada e saída de strings – uso do scanf() e printf()

---

- Entrada

`scanf("%s", s1);`



sem &

```
#include <stdio.h>

int main(){
    char s1[20];

    printf("\ndigite uma string: ");
    scanf("%s", s1);

    printf("\na string eh: %s", s1);

    printf("\n\n");
    return 0;
}
```

- Saída

`printf("%s", s1);`

scanf – leitura até  
espaço em branco

```
digite uma string: Ola mundo
a string eh: Ola
```

# Entrada e saída de strings: uso do gets(), fgets() e puts()

---

- Entrada

`gets(s1);` Warning → gets  
`fgets(s1, 10, stdin);`

```
#include <stdio.h>

int main(){
    char s1[20];

    printf("\ndigite uma string: ");
    gets(s1);

    printf("\na string eh: ");
    puts(s1);

    printf("\n\n");
    return 0;
}
```

digite uma string: ola mundo

a string eh: ola mundo

- Saída

`puts(s1);`

```
#include <stdio.h>

int main(){
    char s1[20];

    printf("\ndigite uma string: ");
    fgets(s1, 10, stdin);

    printf("\na string eh: ");
    puts(s1);

    printf("\n\n");
    return 0;
}
```

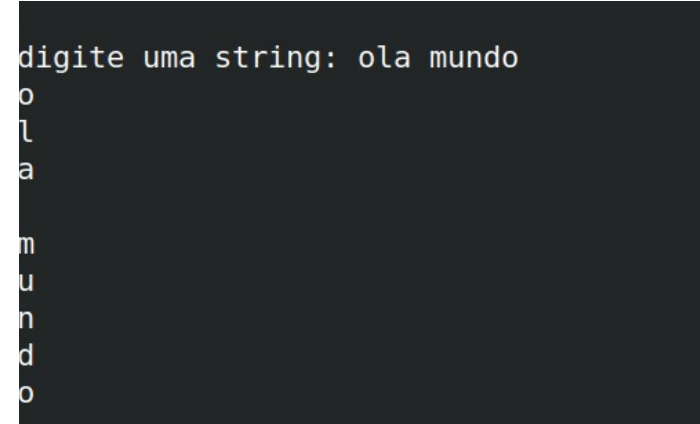
digite uma string: ola mundo

a string eh: ola mundo

# Entrada e saída de strings: caractere por caractere

---

```
1  #include <stdio.h>
2
3  int main(){
4      char s1[20];
5      int i;
6
7      printf("\n\ndigite uma string: ");
8      fgets(s1, sizeof(s1), stdin);
9
10     for(i=0; s1[i] != '\0'; i++)
11         printf("%c \n", s1[i]);
12
13     printf("\n\\");
14     return 0;
15 }
16
```



A terminal window showing the execution of the program. The prompt is 'digite uma string: ola mundo'. The program then prints each character of the string 'ola mundo' on a new line, followed by a double backslash '\\'. The output is: 'o', 'l', 'a', ' ', 'm', 'u', 'n', 'd', 'o', '\\'. The characters are printed in a light green color on a dark background.

# Manipulação de strings

---

- String não é um tipo primitivo da linguagem C, logo as seguintes operações **NÃO** são válidas:

```
str1 = str2;           // ERRO! Não copia str2 em str1
if (str1 == str2)      // ERRO! Não compara str1 com str2
```

- Porém, seus elementos individuais admitem:

```
str[3] = 'b';
if (str[0] == 'a')
```

Funções para manipulação de strings

- Biblioteca string.h**

# Função strcpy (string.h)

---

Copia a *string\_origem* na *string\_destino*

**strcpy(string\_destino, string\_origem);**

```
1  #include <stdio.h>
2  #include <string.h>
3
4  int main(){
5      char s1[10], s2[10];
6
7
8      printf("\ndigite 2 palavras: ");
9      fgets(s1, sizeof(s1), stdin);
10     fgets(s2, sizeof(s2), stdin);
11
12     printf("\nANTES\n string s1: %s string s2: %s", s1, s2);
13     strcpy(s2, s1);
14     printf("\nAPOS\n string s1: %s string s2: %s", s1, s2);
15
16     printf("\n\n");
17     return 0;
18 }
```

```
digite 2 palavras: cao
gato
```

ANTES

```
string s1: cao
string s2: gato
```

APOS

```
string s1: cao
string s2: cao
```



# Função strcat (string.h)

---

Anexa a *string\_origem* ao final da *string\_destino*

**strcat(string\_destino, string\_origem);**

```
1  #include <stdio.h>
2  #include <string.h>
3
4  int main(){
5      char s1[10], s2[20];
6
7
8      printf("\n\ndigite 2 palavras: ");
9      scanf("%s", s1);
10     scanf("%s", s2);
11
12     printf("\nANTES\n string s1: %s\n string s2: %s", s1, s2);
13     strcat(s2, s1);
14     printf("\nAPOS\n string s1: %s\n string s2: %s", s1, s2);
15
16     printf("\n\n");
17     return 0;
18 }
```

Tamanho da s2 deve ser compatível com s1

```
digite 2 palavras: cao
gato
```

```
ANTES
 string s1: cao
 string s2: gato
APOS
 string s1: cao
 string s2: gatocao
```

# Função strlen (string.h)

---

Retorna o tamanho de uma string (despreza o '\0')

**strlen(str);**

```
1  #include <stdio.h>
2  #include <string.h>
3
4  ▼ int main(){
5      char s1[10];
6      int tam;
7
8      printf("\n\ndigite uma palavra: ");
9      scanf("%s", s1);
10
11     tam = strlen(s1);
12     printf("\nstring %s tem tamanho %d\n", s1, tam);
13
14     printf("\n\n");
15     return 0;
16 }
17
```

```
digite uma palavra: caderno
string caderno tem tamanho 7
```

## Indo além... outras funções (string.h)

---

- **strstr**(str1, str2) — retorna um ponteiro para primeira ocorrência de str2 em str1
- **strchr**(str1, ch) — retorna um ponteiro para primeira ocorrência de ch em str1
- **strcmp**(str1, str2) – compara duas strings e:
  - retorna 0 caso str1 seja igual a str2
  - retorna < 0 caso str1 seja menor que str2
  - retorna > 0 caso str1 seja maior que str2
- **strtok**, **strncpy**, **strncat**, **strupr**, **strlwr**...

# Exercícios

Obs: nos exercícios 1, 2 e 4 não podem ser usadas funções da biblioteca string.h

- 1) Faça uma função que leia uma string, e 2 caracteres c1 e c2. A função deve substituir todas as ocorrências do caractere c1 por c2. Ao final, escreva as 2 strings.
- 2) Faça um programa que leia um caractere C e uma string S. O programa deve armazenar, em um vetor V, os índices onde c1 aparece em S. Ao final, imprima o vetor.
- 3) Faça uma função booleana que receba uma string e identifique se ela é um palíndromo (ex: “RADAR” e “SAIAS” são palíndromos).
- 4) Faça um programa que leia 10 palavras, verifique o tamanho de cada uma, armazenando os tamanhos em um vetor. Ao final escreva o vetor, a maior palavra e seu tamanho.
- 5) Faça um programa que leia uma string de até 30 caracteres, gere uma cópia e substitua, na cópia, os espaços em branco por \*. Ao final, escreva a cópia o número de posições alteradas.

# Exercícios

---

- 5) Um dos métodos de encriptação mais antigos é o de Júlio César: se uma letra a ser criptografada é a letra N do alfabeto, substitua-a com a letra  $N+K$ , onde K é um número inteiro constante (César usava  $K = 3$ ). Ex.: para  $K = 1$ , a string “Adoro programar em C” se torna “Bepsp! qsphsbnbs!fn!D”. Faça um programa que leia uma string e um valor K e criptografe a string utilizando o método de César.
  - 6) Faça uma função que receba 2 strings e um valor inteiro, representando uma posição. A seguir, insira a segunda string na primeira, na posição indicada pelo valor.
  - 7) Faça uma função que receba 2 strings (A e B) e retorne uma terceira string (C) formada pelos caracteres de A e B intercalados. Ex.: Se  $A = \text{'Quarta'}$  e  $B = \text{'Segunda'}$ , a resposta deve ser  $\text{'Qsueagrutnada'}$ . A seguir, faça uma outra função que dadas as strings C e B, retorne a string A.
-