

**INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PERNAMBUCO**

CAMPUS AFOGADOS DA INGAZEIRA

Introdução à Programação

Bibliotecas de Funções

Prof. Dr. Diego Rodrigues de Almeida
diego.rodrigues@afogados.ifpe.edu.br
<https://sites.google.com/site/ifpediego>

Biblioteca de Funções

- ▶ Os compiladores C oferecem diversas funções com objetivos predeterminados e que servem para facilitar a criação de programas
- ▶ Para que o programador utilizar as funções é necessário que ele conheça:
 - ▶ O *identificador* da função
 - ▶ O que a função faz
 - ▶ Quantos e de que tipo são os *argumentos* utilizados
 - ▶ Tipo de valor que ela *retorna* quando termina sua execução
- ▶ `identificador_da_funcao(arg1, arg2, arg3, ..., argN)`
- ▶ **IMPORTANTE:**
 - ▶ A lista de argumentos pode ser vazia

Algumas Funções de Biblioteca Matemática

Identificador	Tipo dos Argumentos	O que retorna
<code>fabs(x)</code>	float	Valor absoluto de x
<code>floor(x)</code>	float	Arredonda x para baixo
<code>ceil(x)</code>	float	Arredonda x para cima
<code>pow(x, y)</code>	float, float	x^y
<code>rand()</code>		Um numero aleatório
<code>sqrt(x)</code>	float	Raiz quadrada de x

Algumas Funções de Biblioteca

Caracteres

Identificador	Tipo dos Argumentos	O que retorna
<code>tolower(x)</code>	<code>char</code>	Converte o caractere <code>x</code> para minúsculo
<code>toupper(x)</code>	<code>char</code>	Converte o caractere <code>x</code> para maiúsculo
<code>isdigit(x)</code>	<code>char</code>	Retorna 0 se o caractere NÃO for um dígito (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
<code>isalpha(x)</code>	<code>char</code>	Retorna 0 se um caractere NÃO é uma letra do alfabeto
<code>isalnum(x)</code>	<code>char</code>	Retorna 0 se o caractere não for alfabeto ou número (caractere especial), ou um valor maior que zero caso contrário.

Exemplos

- Os exemplos em código estão no site

Exercício

- ▶ Crie um programa que o usuário digita um número e o programa diz quanto é esse número arredondado para cima e arredondado para baixo

Resolução

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <locale.h>
4
5  int main(int argc, char *argv[]) {
6      setlocale(LC_ALL, "Portuguese");
7      float numero;
8      printf("Digite um número: ");
9      scanf("%f", &numero);
10     float numero_arredondado_cima;
11     float numero_arredondado_baixo;
12     numero_arredondado_cima = ceil(numero);
13     numero_arredondado_baixo = floor(numero);
14     printf("%f arredondado para cima é %f\n", numero, numero_arredondado_cima);
15     printf("%f arredondado para baixo é %f\n", numero, numero_arredondado_baixo);
16 }
```

Exercício

- ▶ Crie um programa que o usuário digita um número e o programa calcula quanto é esse número ao quadrado e quanto é a raiz desse número

Resolução

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <locale.h>
4
5  int main(int argc, char *argv[]) {
6      setlocale(LC_ALL, "Portuguese");
7      float numero;
8      printf("Digite um número: ");
9      scanf("%f", &numero);
10     float raiz_quadrada;
11     float numero_ao_quadrado;
12     numero_ao_quadrado = pow(numero, 2);
13     raiz_quadrada = sqrt(numero);
14     printf("O número ao quadrado é: %f\n", numero_ao_quadrado);
15     printf("A raiz quadrada do número é: %f\n", raiz_quadrada);
16 }
```

Exercício

- Crie uma função que o usuário digita x e y e a função retorna $\sqrt{x + y}$

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <locale.h>
4
5  □ int main(int argc, char *argv[]) {
6      setlocale(LC_ALL, "Portuguese");
7      float x, y;
8      printf("Digite x: ");
9      scanf("%f", &x);
10     printf("Digite y: ");
11     scanf("%f", &y);
12     float resultado = sqrt(x + y);
13     printf("Resultado = %f\n", resultado);
14     return (0);
15 }
```

Exercício

- Faça um programa que o usuário digita um valor x e o programa calcula quanto é $\sqrt{x^3 - 4}$

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <locale.h>
4
5  int main(int argc, char *argv[]) {
6      setlocale(LC_ALL, "Portuguese");
7      float numero, resultado;
8      printf("Digite um número: ");
9      scanf("%f", &numero);
10     resultado = sqrt(pow(numero, 3) - 4);
11     printf("resultado: %f", resultado);
12     return 0;
13 }
```

Exercício

- ▶ Crie um programa em que o usuário digita uma letra e o programa diz se é consoante ou vogal
- ▶ O programa só para quando o usuário digitar um caractere que não é letra

Resolução

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <locale.h>
4
5  int main(int argc, char *argv[]) {
6      setlocale(LC_ALL, "Portuguese");
7      char letra;
8      do{
9          printf("Digite uma letra: ");
10         fflush(stdin);
11         scanf("%c", &letra);
12         if(isalpha(letra) == 0){
13             printf("Você não digitou uma letra!!!\n");
14         }else if(letra == 'a' || letra == 'e' || letra == 'i' || letra == 'o' || letra == 'u'){
15             printf("%c é uma vogal\n", letra);
16         }else{
17             printf("%c é uma consoante\n", letra);
18         }
19     }while(isalpha(letra) != 0);
20     return 0;
21 }
```

Funções de String

- ▶ Para facilitar a programação, foram criadas algumas funções para manipular strings
- ▶ Exemplo:
 - ▶ Crie um programa em que o usuário digita uma frase e o programa escreve essa mesma frase toda com letras maiúsculas

Solução sem usar funções de String

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <locale.h>
4
5  int main(int argc, char *argv[]) {
6      setlocale(LC_ALL, "Portuguese");
7      char frase[101];
8      int i, tamanho, qte_vogais, qte_consoantes;
9      printf("Digite uma frase: ");
10     gets(frase);
11
12     for(i = 0; frase[i] != '\0'; i++){
13         switch(frase[i]){
14             case 'a':{
15                 frase[i] = 'A';
16                 break;
17             }
18             case 'b':{
19                 frase[i] = 'B';
20                 break;
21             }
22             case 'c':{
23                 frase[i] = 'C';
24                 break;
25             }
26             case 'd':{
27                 frase[i] = 'D';
28                 break;
29             }
30             case 'e':{
31                 frase[i] = 'E';
32                 break;
33             }
34             case 'f':{
35                 frase[i] = 'F';
36                 break;
37             }
38             case 'g':{
39                 frase[i] = 'G';
40                 break;
41             }
42             case 'h':{
43                 frase[i] = 'H';
44                 break;
45             }
46             case 'i':{
47                 frase[i] = 'I';
48                 break;
49             }
50             case 'j':{
51                 frase[i] = 'J';
52                 break;
53             }
54             case 'k':{
55                 frase[i] = 'K';
56                 break;
57             }
58             case 'l':{
59                 frase[i] = 'L';
60                 break;
61             }
62             case 'm':{
63                 frase[i] = 'M';
64                 break;
65             }
66             case 'n':{
67                 frase[i] = 'N';
68                 break;
69             }
70             case 'o':{
71                 frase[i] = 'O';
72                 break;
73             }
74             case 'p':{
75                 frase[i] = 'P';
76                 break;
77             }
78             case 'q':{
79                 frase[i] = 'Q';
80                 break;
81             }
82             case 'r':{
83                 frase[i] = 'R';
84                 break;
85             }
86             case 's':{
87                 frase[i] = 'S';
88                 break;
89             }
90             case 't':{
91                 frase[i] = 'T';
92                 break;
93             }
94             case 'u':{
95                 frase[i] = 'U';
96                 break;
97             }
98             case 'v':{
99                 frase[i] = 'V';
100                break;
101            }
102            case 'w':{
103                frase[i] = 'W';
104                break;
105            }
106            case 'x':{
107                frase[i] = 'X';
108                break;
109            }
110            case 'y':{
111                frase[i] = 'Y';
112                break;
113            }
114            case 'z':{
115                frase[i] = 'Z';
116                break;
117            }
118        }
119    }
120    printf("A frase digitada em caixa alta é: %s", frase);
121    return 0;
122 }
```

Solução usando funções de String

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <locale.h>
4
5  int main(int argc, char *argv[]) {
6      setlocale(LC_ALL, "Portuguese");
7      char frase[101];
8      printf("Digite uma frase: ");
9      gets(frase);
10     strupr(frase);
11     printf("A frase digitada em caixa alta é: %s", frase);
12     return 0;
13 }
```

Funções de String

Equivale a
strupr(frase);

consoantes;

```

1  #include <stdio.h>
2  #include <string.h>
3  #include <ctype.h>
4
5  int main()
6  {
7      char frase[100];
8      int i;
9      printf("Digite uma frase: ");
10     gets(frase);
11
12     for(i = 0; frase[i] != '\0'; i++){
13         switch(frase[i]){
14             case 'a':{
15                 frase[i] = 'A';
16                 break;
17             }
18             case 'b':{
19                 frase[i] = 'B';
20                 break;
21             }
22             case 'c':{
23                 frase[i] = 'C';
24                 break;
25             }
26             case 'd':{
27                 frase[i] = 'D';
28                 break;
29             }
30             case 'e':{
31                 frase[i] = 'E';
32                 break;
33             }
34             case 'f':{
35                 frase[i] = 'F';
36                 break;
37             }
38             case 'g':{
39                 frase[i] = 'G';
40                 break;
41             }
42             case 'h':{
43                 frase[i] = 'H';
44                 break;
45             }
46             case 'i':{
47                 frase[i] = 'I';
48                 break;
49             }
50             case 'j':{
51                 frase[i] = 'J';
52                 break;
53             }
54             case 'k':{
55                 frase[i] = 'K';
56                 break;
57             }
58             case 'l':{
59                 frase[i] = 'L';
60                 break;
61             }
62             case 'm':{
63                 frase[i] = 'M';
64                 break;
65             }
66             case 'n':{
67                 frase[i] = 'N';
68                 break;
69             }
70             case 'o':{
71                 frase[i] = 'O';
72                 break;
73             }
74             case 'p':{
75                 frase[i] = 'P';
76                 break;
77             }
78             case 'q':{
79                 frase[i] = 'Q';
80                 break;
81             }
82             case 'r':{
83                 frase[i] = 'R';
84                 break;
85             }
86             case 's':{
87                 frase[i] = 'S';
88                 break;
89             }
90             case 't':{
91                 frase[i] = 'T';
92                 break;
93             }
94             case 'u':{
95                 frase[i] = 'U';
96                 break;
97             }
98             case 'v':{
99                 frase[i] = 'V';
100                break;
101            }
102            case 'w':{
103                frase[i] = 'W';
104                break;
105            }
106            case 'x':{
107                frase[i] = 'X';
108                break;
109            }
110            case 'y':{
111                frase[i] = 'Y';
112                break;
113            }
114            case 'z':{
115                frase[i] = 'Z';
116                break;
117            }
118        }
119    }
120
121    printf("A frase digitada em caixa alta é: %s", frase);
122    return 0;
123 }

```

Funções de String

► As funções mais comuns são as seguintes:

Função	Descrição
strcat (dest, orig)	Concatena string origem no final de destino.
strncat (dest, orig, n)	Concatena string orig no final de dest, usando no máximo n caracteres de orig.
strcmp (str1, str2)	Compara as duas strings. Retorna zero se iguais, menor que 0 se str1 < str2, maior que 0 se str1 > str2.
strcmpi (str1, str2)	Compara as duas strings sem levar em conta maiúsculas e minúsculas.
strlen (str)	Calcula o comprimento da string sem o caractere nulo.
strlwr (str)	Converte string para minúsculas.
strupr (str)	Converte string para maiúsculas.
strcpy (dest, orig)	Copia string origem para destino.

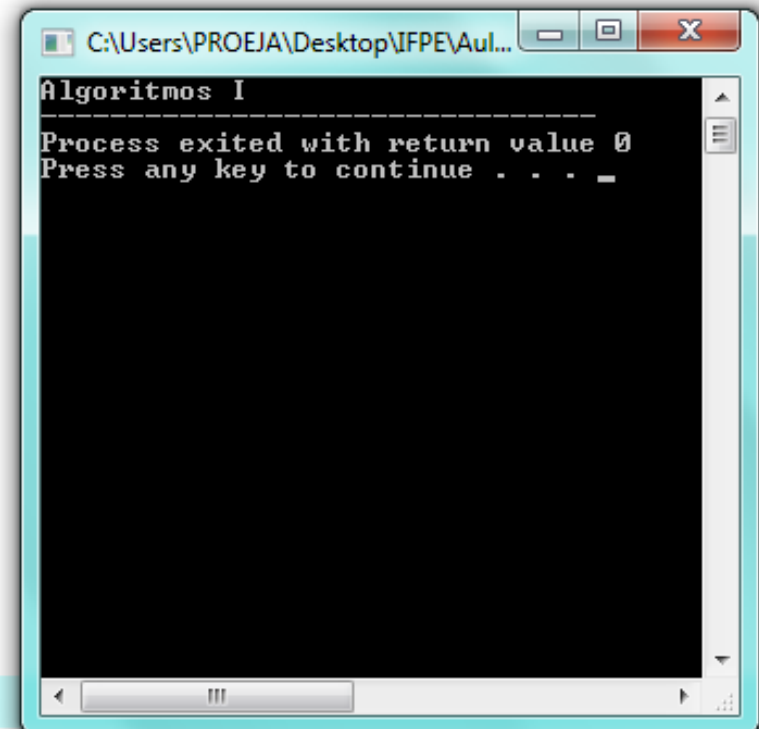
strcat (dest, orig)

- ▶ A função **strcat()** tem o objetivo de concatenar strings
- ▶ Concatenar é o ato de unir duas strings
- ▶ Exemplo:
 - ▶ A string “**Informática**” concatenada com a string “**Básica**” resultará na string “**InformáticaBásica**” (sem o espaço, pois não há espaço depois de “**Informática**” nem antes de “**Básica**”)
- ▶ A função **strcat()** tem a seguinte forma geral:
 - ▶ *strcat (string_destino,string_origem);*
- ▶ A string de origem permanecerá inalterada e será anexada ao fim da string de destino

strcat (dest, orig)

► Exemplo:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <locale.h>
4
5  int main(int argc, char *argv[]) {
6      setlocale(LC_ALL, "Portuguese");
7      char str1[20] = "Algoritmos";
8      char str2[20] = " I";
9      strcat(str1, str2);
10     printf(str1);
11     return (0);
12 }
13
14
```



strcat (dest, orig)

► Exemplo:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <locale.h>
4
5  int main(int argc, char *argv[]) {
6      setlocale(LC_ALL, "Portuguese");
7      char frase1[101], frase2[101];
8      int i;
9      printf("Digite uma frase: ");
10     gets(frase1);
11     printf("Digite uma outra frase a ser concatenada a primeira: ");
12     gets(frase2);
13     strcat(frase1, frase2);
14     printf("As frases concatenadas formam a string: %s\n", frase1);
15     return 0;
16 }
```

strcat (dest, orig)

► Exemplo:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <locale.h>
4
5  int main(int argc, char *argv[]) {
6      setlocale(LC_ALL, "Portuguese");
7      char cidade[21], estado[3];
8      printf("Digite a sua cidade: ");
9      gets(cidade);
10     printf("Digite o estado: ");
11     gets(estado);
12     strcat(cidade, " - ");
13     strcat(cidade, estado);
14     printf("Você mora em %s", cidade);
15     return 0;
16 }
```

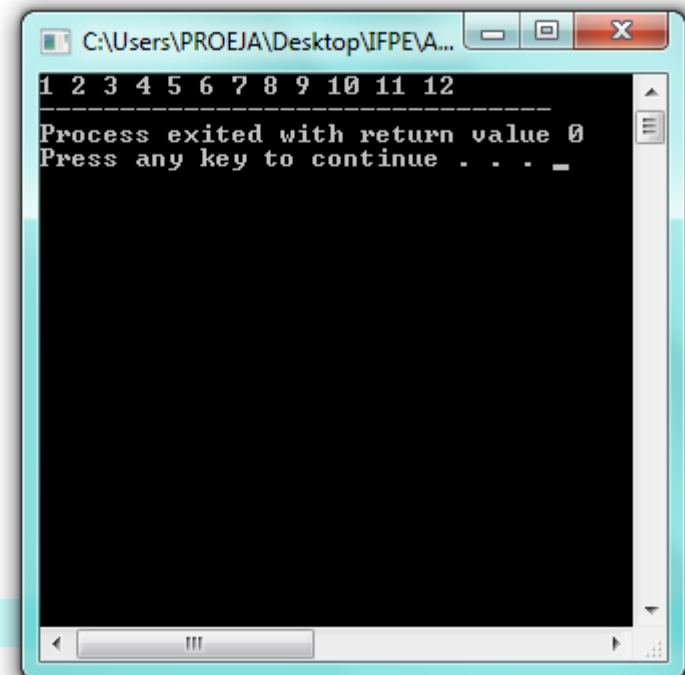

strncat (dest, orig, n)

- ▶ A função **strncat()** tem o objetivo de concatenar as n primeiras letras de uma segunda string com todas da primeira
- ▶ Exemplo:
 - ▶ `strncat("Informática", "Básica", 2)`
 - ▶ Resultará em: "InformáticaBá"
- ▶ A função **strncat()** tem a seguinte forma geral:
 - ▶ `strncat (string_destino, string_origem, n);`
- ▶ A string de origem permanecerá inalterada e suas n primeiras letras serão anexadas ao fim da string de destino

strncat (dest, orig, n)

► Exemplo

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <locale.h>
4
5 int main(int argc, char *argv[]) {
6     setlocale(LC_ALL, "Portuguese");
7     char str1[100] = "1 2 3 4 5 6 7 8 9 10 ";
8     char str2[100] = "11 12 13 14 15";
9     strncat(str1, str2, 5);
10    printf(str1);
11    return (0);
12 }
13
14
```



strncat (dest, orig, n)

► Exemplo

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <locale.h>
4
5  int main(int argc, char *argv[]) {
6      setlocale(LC_ALL, "Portuguese");
7      char frase1[101], frase2[101];
8      int i;
9      printf("Digite uma frase: ");
10     gets(frase1);
11     printf("Digite uma outra frase a ser concatenada a primeira: ");
12     gets(frase2);
13     strncat(frase1, frase2, 2);
14     printf("As frases concatenadas formam a string: %s\n", frase1);
15     return 0;
16 }
```

strcmp (str1, str2)

- ▶ Sua forma geral é:
 - ▶ *strcmp (string1, string2);*
- ▶ A função **strcmp()** compara a string 1 com a string 2.
- ▶ Retorna 0 se as duas forem idênticas
- ▶ Retorna -1 se a string 1 estiver antes que string 2 na ordem alfabética
 - ▶ Exemplo: string 1 = “avião” e string 2 = “borboleta”
- ▶ Retorna 1 se a string 1 estiver depois que string 2 na ordem alfabética
 - ▶ Exemplo: string 1 = “cavalo” e string 2 = “borboleta”

strcmp (str1, str2)

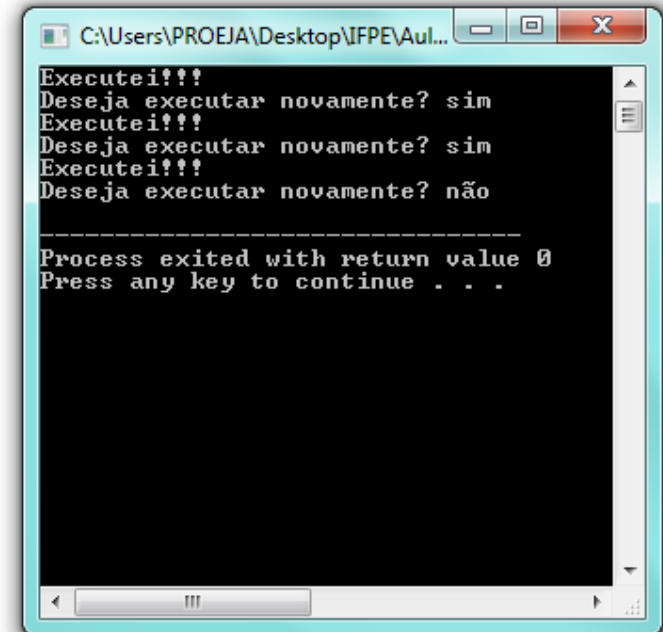
► Comparando:

Função	Resultado
strcmp("avião", "brilho")	-1
strcmp("zebra", "cavalo")	1
strcmp("natureza", "natureza")	-1
strcmp("violão", "violão")	0
strcmp("Violão", "violão")	1
strcmp("claro", "escuro")	-1
strcmp("ifpe", "ifpe")	0
strcmp("ifpe", "IFPE")	-1

strcmp (str1, str2)

► Exemplo

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <locale.h>
4
5  int main(int argc, char *argv[]) {
6      setlocale(LC_ALL, "Portuguese");
7      char resposta[3];
8      do{
9          printf("Executei!!!\n");
10         printf("Deseja executar novamente? ");
11         gets(resposta);
12     }while(strcmp(resposta, "sim") == 0);
13     return (0);
14 }
15
16
```



```
C:\Users\PROEJA\Desktop\IFPE\Aul...
Executei!!!
Deseja executar novamente? sim
Executei!!!
Deseja executar novamente? sim
Executei!!!
Deseja executar novamente? não

-----
Process exited with return value 0
Press any key to continue . . .
```

strcmp (str1, str2)

► Exemplo

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <locale.h>
4
5  int main(int argc, char *argv[]) {
6      setlocale(LC_ALL, "Portuguese");
7      char frase1[101], frase2[101];
8      printf("Digite uma frase: ");
9      gets(frase1);
10     printf("Digite uma outra frase: ");
11     gets(frase2);
12     int resultado = strcmp(frase1, frase2);
13     switch(resultado){
14     case 0:{
15         printf("As strings são idênticas\n");
16         break;
17     }
18     case -1:{
19         printf("A primeira frase está antes da segunda frase na ordem alfabética\n");
20         break;
21     }
22     case 1:{
23         printf("A primeira frase está depois da segunda frase na ordem alfabética\n");
24         break;
25     }
26     }
27     return 0;
28 }
```

strcmp (str1, str2)

► Exemplo

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <locale.h>
4
5  int main(int argc, char *argv[]) {
6      setlocale(LC_ALL, "Portuguese");
7      char resposta[5];
8      printf("Este programa vai se repetir até quando você quiser\n");
9      do{
10         printf("Deseja repetir novamente?\n");
11         printf("Resposta: <sim> ou <nao>\n");
12         gets(resposta);
13     }while(strcmp(resposta, "sim") == 0);
14     return 0;
15 }
```


strcmpi (str1, str2)

- ▶ Idêntica a função **strcmp()** mas não leva em consideração letras maiúsculas e minúsculas

strcmpi(str1, str2)

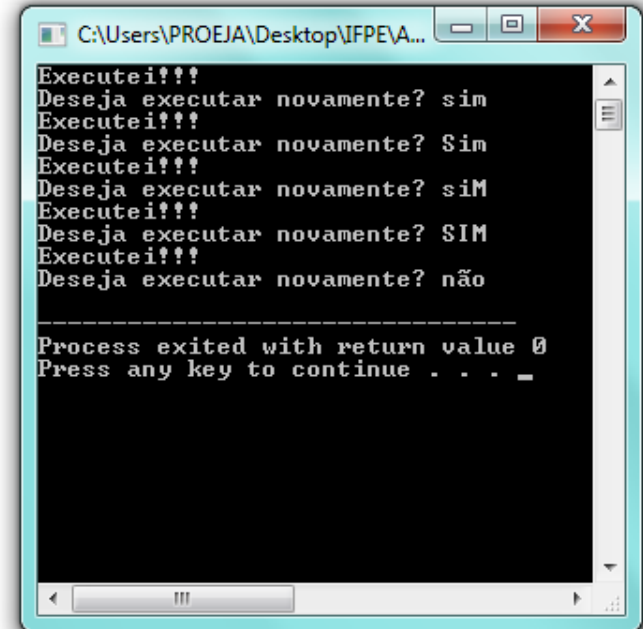
► Comparando:

Função	Resultado
strcmpi("avião", "brilho")	-1
strcmpi("zebra", "cavalo")	1
strcmpi("natureza", "natureza")	-1
strcmpi("violão", "violão")	0
strcmpi("Violão", "violão")	0
strcmpi("claro", "escuro")	-1
strcmpi("ifpe", "ifpe")	0
strcmpi("ifpe", "IFPE")	0

strcmpi (str1, str2)

► Exemplo

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <locale.h>
4
5  int main(int argc, char *argv[]) {
6      setlocale(LC_ALL, "Portuguese");
7      char resposta[3];
8      do{
9          printf("Executei!!!\n");
10         printf("Deseja executar novamente? ");
11         gets(resposta);
12     }while(strcmpi(resposta, "sim") == 0);
13     return (0);
14 }
15
16
```



```
C:\Users\PROEJA\Desktop\IFPE\A...
Executei!!!
Deseja executar novamente? sim
Executei!!!
Deseja executar novamente? Sim
Executei!!!
Deseja executar novamente? siM
Executei!!!
Deseja executar novamente? SIM
Executei!!!
Deseja executar novamente? não
-----
Process exited with return value 0
Press any key to continue . . . _
```

strlen (str)

- ▶ Sua forma geral é:
 - ▶ *strlen (string);*
- ▶ A função **strlen()** retorna o tamanho da string fornecida
- ▶ O terminador nulo ('/0') não é contado. Isto quer dizer que, de fato, o comprimento do vetor da string deve ser um a mais que o inteiro retornado por **strlen()**.

strlen (str)

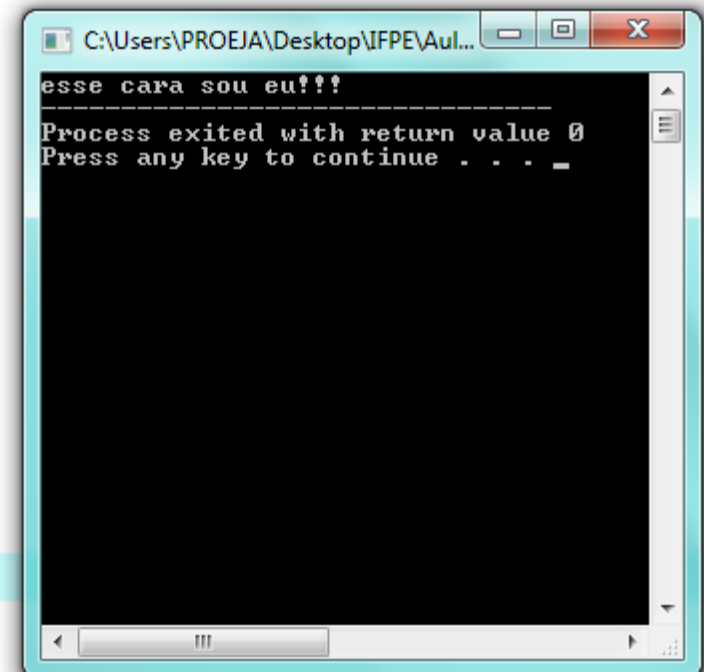
► Exemplo

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <locale.h>
4
5  int main(int argc, char *argv[]) {
6      setlocale(LC_ALL, "Portuguese");
7      char frase[101];
8      int i, tamanho;
9      printf("Digite uma frase: ");
10     gets(frase);
11     tamanho = strlen(frase);
12     printf("A frase <%s> tem tamanho %i\n", frase, tamanho);
13     return 0;
14 }
```

strlwr (str)

- ▶ A função `strlwr()` converte letras maiúsculas em minúsculas
- ▶ Exemplo:
 - ▶ `strlwr("Esse Cara sou Eu!!!")` resultará em `"esse cara sou eu!!!"`

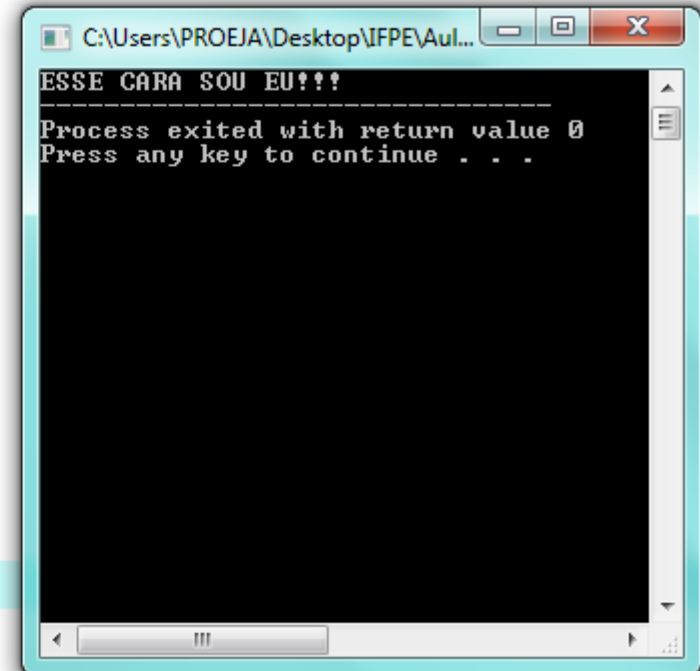
```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <locale.h>
4
5  int main(int argc, char *argv[]) {
6      setlocale(LC_ALL, "Portuguese");
7      char frase[30] = "Esse Cara sou Eu!!!";
8      strlwr(frase);
9      printf(frase);
10     return (0);
11 }
12
13
```



strupr (str)

- ▶ A função `strupr()` converte letras minúsculas em maiúsculas
- ▶ Exemplo:
 - ▶ `strlwr("Esse Cara sou Eu!!!")` resultará em **"ESSE CARA SOU EU!!!"**


```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <locale.h>
4
5  int main(int argc, char *argv[]) {
6      setlocale(LC_ALL, "Portuguese");
7      char frase[30] = "Esse Cara sou Eu!!!";
8      strupr(frase);
9      printf(frase);
10     return (0);
11 }
12
13
```



strcpy (dest, orig)

- ▶ Sua forma geral é:
 - ▶ `strcpy (string_destino, string_origem);`

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <locale.h>
4
5  int main(int argc, char *argv[]) {
6      setlocale(LC_ALL, "Portuguese");
7      char str1[100], str2[100], str3[100];
8      printf ("Entre com uma string: ");
9      gets (str1);
10     strcpy (str2, str1); /* Cópia str1 em str2 */
11     strcpy (str3, "Voce digitou a string ");
12     printf ("str2 = %s\n", str2);
13     printf ("str3 = %s\n", str3);
14     return 0;
15 }
```



`strcpy (str2, str1);`



`strcpy (str3, "Voce digitou a string ");`

strcpy (dest, orig)

► Exemplo

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <locale.h>
4
5  int main(int argc, char *argv[]) {
6      setlocale(LC_ALL, "Portuguese");
7      char mensagem[100], mensagem_codificada[100];
8      int i;
9      printf ("Digite uma mensagem: ");
10     gets (mensagem);
11     strcpy (mensagem_codificada,mensagem);
12     for(i = 0; mensagem_codificada[i] != '\0'; i++){
13         mensagem_codificada[i]++;
14     }
15     printf("A mensagem <%s> codificada é: <%s>", mensagem, mensagem_codificada);
16     return 0;
17 }
```

Obrigado

