

```
63         } else if ( strcmp( argv[3], "/" ) == 0 ) {
64             printf( "%d / %d = %d", n1, n2, n1 / n2 ) ;
65         } else if ( strcmp( argv[3], "pow" ) == 0 ) {
66             printf( "%d pow %d = %d", n1, n2, (int) pow( n1, n2 )
67                 ↵ ) ;
68         } else {
69             printf( "Operador invalido!" );
70             return 1;
71         }
72     }
73 }
74 }
75 }
76 }
77
78 return 0;
79
80 }
81
82 bool ehInteiro( const char *string ) {
83
84     for ( int i = 0; i < strlen( string ); i++ ) {
85         if ( !isdigit( string[i] ) ) {
86             return false;
87         }
88     }
89
90     return true;
91
92 }
```

9.2 Exercícios

Atenção! Para todos os exercícios a seguir, considere que as Strings possuem, no máximo, 40 caracteres válidos.

Exercício 9.1: Escreva um programa para ler uma string e apresentar seus quatro primeiros caracteres.

Arquivo com a solução: [ex9.1.c](#)

Entrada

```
String: essa eh uma string
```

Saída

```
e, s, s, a.
```

Exercício 9.2: Escreva um programa para ler uma sentença e apresentar:

- O primeiro caractere da sentença;
- O último caractere da sentença;
- O número de caracteres existente na sentença.

Arquivo com a solução: [ex9.2.c](#)

Entrada

```
Sentenca: ola, como vai, tudo bem?
```

Saída

```
Primeiro caractere: o  
Ultimo caractere: ?  
Numero de caracteres: 24
```

Exercício 9.3: Escreva um programa para ler uma sentença e imprimir todos os seus caracteres das posições pares. Se algum caractere for um espaço, imprima-o cercado de aspas simples.

Arquivo com a solução: [ex9.3.c](#)

Entrada

```
Sentenca: um dois tres
```

Saída

```
u, ' ', o, s, t, e
```

Exercício 9.4: Escreva um programa para ler uma sentença e imprimir todos os seus caracteres das posições ímpares.

Arquivo com a solução: [ex9.4.c](#)

Entrada

Sentença: um dois tres

Saída

m, d, i, ' ', r, s

Exercício 9.5: Escreva um programa para ler um nome e imprimi-lo 5 vezes, um por linha.

Arquivo com a solução: [ex9.5.c](#)

Entrada

Nome: Fernanda

Saída

Fernanda
Fernanda
Fernanda
Fernanda
Fernanda

Exercício 9.6: Escreva um programa que receba um nome e que o imprima tantas vezes quanto forem seus caracteres.

Arquivo com a solução: [ex9.6.c](#)

Entrada

Nome: Aurora

Saída

Aurora
Aurora
Aurora
Aurora
Aurora
Aurora

Exercício 9.7: Escreva um programa que leia 5 pares de strings e que imprima:

- IGUAIS, se as strings do par forem iguais;
- ORDEM CRESCENTE, se as strings do par foram fornecidas em ordem crescente;
- ORDEM DECRESCENTE, se as strings do par foram fornecidas em ordem decrescente.

Arquivo com a solução: [ex9.7.c](#)

Entrada

```
Par 1, palavra 1: Joao
Par 1, palavra 2: Maria
Par 2, palavra 1: Fernanda
Par 2, palavra 2: David
Par 3, palavra 1: Rafaela
Par 3, palavra 2: Rafaela
Par 4, palavra 1: Renata
Par 4, palavra 2: Cecilia
Par 5, palavra 1: Joana
Par 5, palavra 2: Zelia
```

Saída

```
Joao - Maria: ORDEM CRESCENTE
Fernanda - David: ORDEM DECRESCENTE
Rafaela - Rafaela: IGUAIS
Renata - Cecilia: ORDEM DECRESCENTE
Joana - Zelia: ORDEM CRESCENTE
```

Exercício 9.8: Escreva um programa que leia três strings. A seguir imprimir as 3 strings em ordem alfabética.

Arquivo com a solução: [ex9.8.c](#)

Entrada

```
String 1: Luiz
String 2: Everton
String 3: Breno
```

Saída

Breno, Everton e Luiz

Exercício 9.9: Escreva um programa para ler uma string. A seguir copie para outra string a string informada na ordem inversa e imprima-a. Para isso, implemente a função `void inverter(char *destino, const char *origem)`.

Arquivo com a solução: [ex9.9.c](#)

Entrada

String: abacate verde

Saída

Invertida: edrev etacaba

Exercício 9.10: Escreva um programa para ler uma frase e imprimir o número de caracteres dessa frase. Você deve implementar a função `int tamanho(const char *str)` que fará o trabalho de contar a quantidade de caracteres. **Atenção, não use a função `int strlen(const char *str)`.**

Arquivo com a solução: [ex9.10.c](#)

Entrada

Frase: vou comprar um lanche

Saída

21 caractere(s)!

Exercício 9.11: Escreva um programa para ler um caractere e logo após uma frase. Para cada frase informada, imprimir o número de ocorrências do caractere na frase. O programa deve ser encerrado quando a frase digitada for a palavra “fim”, que por sua vez não deve ter as ocorrências do caractere informado contadas. A contagem de ocorrências deve ser feita pela função `int contarOcorrencias(const char *str, char c)`.

Arquivo com a solução: [ex9.11.c](#)

Entrada

```
Caractere: a
Frase: camarao assado
Frase: mas que cabelo sujo!
Frase: fim
```

Saída

```
"camarao assado" tem 5 ocorrencia(s) do caractere 'a'
"mas que cabelo sujo!" tem 2 ocorrencia(s) do caractere 'a'
```

Exercício 9.12: Escreva um programa para ler uma frase e contar o número de ocorrências de cada uma das 5 primeiras letras do alfabeto (tanto maiúsculas quanto minúsculas) e imprimir essas contagens. Você pode usar a função `contarOcorrencias` implementada no exercício anterior.

Arquivo com a solução: [ex9.12.c](#)

Entrada

```
Frase: UI, QUE medo DO white walker!
```

Saída

```
A/a: 1
B/b: 0
C/c: 0
D/d: 2
E/e: 4
```

Exercício 9.13: Escreva um programa para ler uma frase e contar o número de ocorrências das letras A, E, I, O e U (tanto maiúsculas quanto minúsculas) e imprimir essas contagens. Você pode usar a função `contarOcorrencias` implementada no exercício anterior.

Arquivo com a solução: [ex9.13.c](#)

Entrada

```
Frase: UI, QUE medo DO white walker!
```

Saída

```
A/a: 1
E/e: 4
I/i: 2
O/o: 2
U/u: 2
```

Exercício 9.14: Escreva um programa para ler uma frase. A seguir converter todas as letras minúsculas existentes na frase para maiúsculas e apresentar a frase modificada. Essa conversão deve ser feita por meio da função `void tornarMaiuscula(char *str)`.

Arquivo com a solução: [ex9.14.c](#)

Entrada

```
Frase: Fui comprar um COMPUTADOR.
```

Saída

```
FUI COMPRAR UM COMPUTADOR.
```

Exercício 9.15: Escreva um programa para ler uma frase. A seguir converter todas as letras maiúsculas existentes na frase para minúsculas e apresentar a frase modificada. Essa conversão deve ser feita por meio da função `void tornarMinuscula(char *str)`.

Arquivo com a solução: [ex9.15.c](#)

Entrada

```
Frase: Fui comprar um COMPUTADOR.
```

Saída

```
fui comprar um computador.
```

Exercício 9.16: Escreva um programa para ler uma frase e um caractere. A seguir retirar da frase todas as letras iguais (tanto as ocorrências maiúsculas quanto minúsculas) à informada e imprimir a frase modificada. A remoção deve ser feita usando a função `void removerLetra(char *str, char c)`.

Arquivo com a solução: [ex9.16.c](#)**Entrada**

```
Frase: ja acabou, JESSICA?  
Caractere: a
```

Saída

```
j cbou, JESSIC?
```

Exercício 9.17: Escreva um programa para ler uma frase e contar o número de palavras existentes na frase. Considere palavra um conjunto qualquer de caracteres separados por um conjunto qualquer de espaços em branco. A contagem deve ser feita por meio da função `int contarPalavras(const char *str)`.

Arquivo com a solução: [ex9.17.c](#)**Entrada**

```
Frase: A aranha arranha a ra.
```

Saída

```
Quantidade de palavras: 5
```

Exercício 9.18: Escreva um programa que leia uma string e verifique se a mesma é um palíndromo. Um palíndromo é toda sentença que pode ser lida da mesma forma da esquerda para a direita e vice-versa. Letras maiúsculas e minúsculas não devem ser diferenciadas. Implemente para isso a função `bool ehPalindromo(const char *str)`.

Arquivo com a solução: [ex9.18.c](#)**Entrada**

```
String: arara
```

Saída

```
"arara" eh um palindromo!
```


Entrada

String: Arara

Saída

"Arara" nao eh um palindromo!

Entrada

String: ArarA

Saída

"ArarA" eh um palindromo!

Entrada

String: Macaco

Saída

"Macaco" nao eh um palindromo!

Exercício 9.19: Escreva um programa que recorte uma string com base em uma posição inicial e uma posição final. Para isso, implemente a função `void substring(char *recorte, const char *origem, int inicio, int fim)`. O índice inicial é inclusivo e o final é exclusivo. Caso algum índice inválido seja fornecido, a string não deve ser recortada, sendo copiada inteiramente para o destino.

Arquivo com a solução: [ex9.19.c](#)

Entrada

String: Girafa
Inicio: 0
Fim: 3

Saída

Recorte: Gir

Entrada

```
String: Girafa  
Inicio: 5  
Fim: 3
```

Saída

```
Recorte: Girafa
```

Entrada

```
String: Girafa  
Inicio: 5  
Fim: 10
```

Saída

```
Recorte: Girafa
```

Entrada

```
String: Girafa  
Inicio: 10  
Fim: 12
```

Saída

```
Recorte: Girafa
```

Exercício 9.20: Escreva um programa que leia duas strings e que verifique se a segunda está contida na primeira. Considere que letras maiúsculas e minúsculas são diferentes. Para isso, implemente a função `bool contem(const char *fonte, const char *aPesquisar)`.

Arquivo com a solução: [ex9.20.c](#)

Entrada

```
String fonte: rabanete  
String a pesquisar: bane
```

Saída

```
"bane" esta contida em "rabanete"
```

Entrada

```
String: Hamburger  
String a pesquisar: Hambo
```

Saída

```
"Hambo" nao esta contida em "Hamburger"
```

Exercício 9.21: Escreva um programa que leia uma string e que a imprima centralizada no terminal. Para isso, implemente a função `void imprimirCentralizado(const char *str)`. Considere que o terminal tem 80 colunas.

Arquivo com a solução: [ex9.21.c](#)

Entrada

```
String: um texto qualquer
```

Saída

```
um texto qualquer
```

Exercício 9.22: Escreva um programa que leia uma string e que a imprima alinhada à direita no terminal. Para isso, implemente a função `void imprimirDireita(const char *str)`. Considere que o terminal tem 80 colunas.

Arquivo com a solução: [ex9.22.c](#)

Entrada

```
String: um texto qualquer
```

Saída

```
um texto qualquer
```

Exercício 9.23: Escreva um programa que leia uma string e que a imprima dentro de

uma caixa desenhada usando os símbolos =, + e |. Para isso, implemente a função `void imprimirCaixa(const char *str)`.

Arquivo com a solução: [ex9.23.c](#)

Entrada

```
String: um texto qualquer
```

Saída

```
++=====++  
| um texto qualquer |  
++=====++
```