

Tema: Introdução à programação  
Atividade: Montagem de programas em C

### Preparação

Vídeos recomendados:

Antes de iniciar as atividades, recomenda-se assistir aos seguintes vídeos:

[https://www.youtube.com/watch?v=GiCt0Cwcp-U&list=PL8iN9FQ7\\_jt7pMKtbgoc0uUQjoJK-3dYu&index=1](https://www.youtube.com/watch?v=GiCt0Cwcp-U&list=PL8iN9FQ7_jt7pMKtbgoc0uUQjoJK-3dYu&index=1)  
[https://www.youtube.com/watch?v=q51cHsgRHU4&list=PL8iN9FQ7\\_jt7pMKtbgoc0uUQjoJK-3dYu&index=2](https://www.youtube.com/watch?v=q51cHsgRHU4&list=PL8iN9FQ7_jt7pMKtbgoc0uUQjoJK-3dYu&index=2)  
[https://www.youtube.com/watch?v=07YPObbEpU8&list=PL8iN9FQ7\\_jt7pMKtbgoc0uUQjoJK-3dYu&index=3](https://www.youtube.com/watch?v=07YPObbEpU8&list=PL8iN9FQ7_jt7pMKtbgoc0uUQjoJK-3dYu&index=3)  
[https://www.youtube.com/watch?v=yQx8sD6vK6M&list=PL8iN9FQ7\\_jt7pMKtbgoc0uUQjoJK-3dYu&index=4](https://www.youtube.com/watch?v=yQx8sD6vK6M&list=PL8iN9FQ7_jt7pMKtbgoc0uUQjoJK-3dYu&index=4)  
[https://www.youtube.com/watch?v=tQhnuVR2gc4&list=PL8iN9FQ7\\_jt7pMKtbgoc0uUQjoJK-3dYu&index=5](https://www.youtube.com/watch?v=tQhnuVR2gc4&list=PL8iN9FQ7_jt7pMKtbgoc0uUQjoJK-3dYu&index=5)  
[https://www.youtube.com/watch?v=GdjGrVjRgTI&list=PL8iN9FQ7\\_jt7pMKtbgoc0uUQjoJK-3dYu&index=6](https://www.youtube.com/watch?v=GdjGrVjRgTI&list=PL8iN9FQ7_jt7pMKtbgoc0uUQjoJK-3dYu&index=6)  
[https://www.youtube.com/watch?v=NsRwpFNZhJs&list=PL8iN9FQ7\\_jt7pMKtbgoc0uUQjoJK-3dYu&index=7](https://www.youtube.com/watch?v=NsRwpFNZhJs&list=PL8iN9FQ7_jt7pMKtbgoc0uUQjoJK-3dYu&index=7)  
[https://www.youtube.com/watch?v=8PAWmHdreoc&list=PL8iN9FQ7\\_jt7pMKtbgoc0uUQjoJK-3dYu&index=8](https://www.youtube.com/watch?v=8PAWmHdreoc&list=PL8iN9FQ7_jt7pMKtbgoc0uUQjoJK-3dYu&index=8)  
[https://www.youtube.com/watch?v=kaivxmdkyTg&list=PL8iN9FQ7\\_jt7pMKtbgoc0uUQjoJK-3dYu&index=11](https://www.youtube.com/watch?v=kaivxmdkyTg&list=PL8iN9FQ7_jt7pMKtbgoc0uUQjoJK-3dYu&index=11)  
[https://www.youtube.com/watch?v=TIIEIMmutQo&list=PL8iN9FQ7\\_jt7pMKtbgoc0uUQjoJK-3dYu&index=12](https://www.youtube.com/watch?v=TIIEIMmutQo&list=PL8iN9FQ7_jt7pMKtbgoc0uUQjoJK-3dYu&index=12)

Orientações gerais:

A melhor maneira de lidar com o guia é ler todos os enunciados e digitá-los aos poucos, e não copiá-los.

Após digitação, prever testes e registrar os valores escolhidos ao final do programa.

Testar cada um dos testes previstos e registrar os resultados.

Depois de todos os testes concluídos, iniciar a confecção dos exercícios.

Lidar com erros de compilação ou de execução faz parte do processo.

Caso necessitar de ajuda, primeiro, rever o código original e as referências indicadas;

quando esgotadas, buscar ajuda externa. Anotar as soluções ao final do código, também.

Manter cópias e controle de versões. Não descartar soluções incompletas ou interrompidas.

Solicitar (e prestar-se à) revisão de código é uma excelente prática formativa e profissional.

- 01.) Editar e salvar um esboço de programa em C, com o nome do arquivo Exemplo0101.c  
(não usar espaços em branco em nomes de pastas ou arquivos), observar o uso de pontuação,  
maiúsculas e minúsculas, espaços em branco entre operações e não usar acentos ou cedilha:

```
/*  
Exemplo0101 - v0.0. - __ / __ / ____  
Author: _____  
  
Para compilar em terminal (janela de comandos):  
Linux   : gcc -o exemplo0101 exemplo0101.c  
Windows: gcc -o exemplo0101 exemplo0101.c  
  
Para executar em terminal (janela de comandos):  
Linux   : ./exemplo0101  
Windows: exemplo0101  
*/  
  
// dependencias  
#include <stdio.h>    // para as entradas e saidas
```



**DICA:** O melhor lugar para se colocar definições de dados é próximo ao início, junto aos cabeçalhos (assinaturas) dos procedimentos ou das funções. Recomenda-se, sempre que possível, definir com valores iniciais, principalmente os dados que servirão como constantes ou variáveis, segundo os tipos de valores que armazenarão.

**SUGESTÃO:**

Se quiser experimentar poderá experimentar outra forma alternativa de definição, como a mostrada a seguir que, ao ser usada, não deverá ter qualquer consequência sobre o resultado da execução; embora seja muito menos recomendada. A atribuição (ou transferência) de valor será geralmente indicada pela referência para o dado (nome ou destino) à esquerda do sinal de atribuição ('='); e o valor a ser transferido (fonte), à direita desse.

```
int x; // forma alternativa, sem definir o valor inicial

...

x = 0; // e definir o valor depois, portanto: x <- 0 (ler como: o lugar x receberá zero)

printf ( "%s%\n", "x = ", x );
```

**02.) Compilar o programa.**

Se houver erros, identificar individualmente a referência para a linha onde ocorrem. Consultar atentamente o modelo acima, na linha onde ocorreu o erro (e também linhas próximas), editar as modificações necessárias. Compilar novamente e proceder assim até que todos os erros tenham sido resolvidos. Se não houver erros, seguir para o próximo passo.

**DICA:** Se precisar de ajuda sobre como proceder a compilação, consultar os vídeos com as demonstrações sobre algumas formas para fazê-lo.

**SUGESTÃO:** Para se acostumar ao tratamento de erros, registrar a mensagem de erro (como comentário) e quais as medidas encontradas para resolvê-lo.

**03.) Executar o programa.**

Observar as saídas.  
Registrar os resultados.

Versao	Teste	
0.1	01. ( OK )	identificacao de programa

Em caso de erro (ou dúvida), usar comentários para registrar a ocorrência e, posteriormente, tentar resolvê-lo (ou para esclarecer dúvidas).

**04.) Copiar a versão atual do programa para outra (nova) – Exemplo0102.c.**

- 05.) Editar mudanças no nome do programa e versão,  
para manipular um valor real,  
conforme as indicações a seguir,  
tomando o cuidado de modificar todas as indicações,  
inclusive as presentes em comentários.  
Incluir na documentação complementar as alterações feitas,  
acrescentar indicações de mudança de versão e  
prever novos testes.

SUGESTÃO: Recomenda-se uma rápida compilação,  
após a troca do nome, antes de outras alterações mais significativas,  
para verificar se as modificações iniciais ocorreram  
sem inserir erros no programa existente.

```
/*
Exemplo0102 - v0.0. - __ / __ / ____
Author: _____

Para compilar em terminal (janela de comandos):
Linux   : gcc -o exemplo0102    exemplo0102.c
Windows: gcc -o exemplo0102    exemplo0102.c

Para executar em terminal (janela de comandos):
Linux   : ./exemplo0102
Windows: exemplo0102
*/
// dependencias
#include <stdio.h>    // para as entradas e saidas

/*
Funcao principal.
@return codigo de encerramento
@param argc - quantidade de parametros na linha de comandos
@param argv - arranjo com o grupo de parametros na linha de comandos
*/
int main ( int argc, char* argv [ ] )
{
    // definir dado
    double x = 0.0;        // definir variavel com valor inicial
                          // OBS.: Definir a parte fracionaria e' util

    // identificar
    printf ( "%s\n", "EXEMPLO0102 - Programa - v0.0" );
    printf ( "%s\n", "Autor: _____" );
    printf ( "\n" );      // mudar de linha

    // mostrar valor inicial
    printf ( "%s%lf\n", "x = ", x );
                          // OBS.: O formato para double -> %lf

    // ler do teclado
    printf ( "Entrar com um valor real: " );
    scanf ( "%lf", &x );
                          // OBS.: Necessario indicar o endereco -> &
    getchar ( );          // OBS.: Limpar a entrada de dados
}
```

```
// mostrar valor lido
printf ( "%s%lf\n", "x = ", x );

// encerrar
printf ( "\n\nApertar ENTER para terminar.\n" );
getchar( );      // aguardar por ENTER
return ( 0 );     // voltar ao SO (sem erros)
} // fim main( )
```

```
/*
----- documentacao complementar
----- notas / observacoes / comentarios
```

```
----- previsao de testes
```

- a.) 0.5
- b.) -0.5
- c.) 1.23456789

```
----- historico
```

Versao	Data	Modificacao
0.1	__/__/__	esboco
0.2	__/__/__	mudanca de versao

```
----- testes
```

Versao	Teste	
0.1	01. ( OK )	identificacao de programa leitura e exibicao de inteiro
0.2	01. ( ___ )	identificacao de programa

```
*/
```

DICA: A exibição (ou transferência para a saída padrão) de valor de um dado poderá ser feita, sempre que necessário, para se consultar o que estiver armazenado. Como a saída exige uma conversão para os símbolos correspondentes aos padrões da língua do usuário, faz-se necessário converter valores numéricos em equivalentes literais (caracteres), o que será indicado pelo formato aspas, que antecederá a referência para o valor a ser convertido (x). A operação de composição (chamada de *formatação*) também providenciará a *concatenação* (junção) da sequência com a conversão do valor. Para essa operação ser bem sucedida, a sequência recomenda-se usar uma cadeia de caracteres, conteúdo constante ou não, seguida de valor(es).

SUGESTÃO: Recomenda-se preceder a exibição do valor pelo nome escolhido para o mesmo.

06.) Compilar o programa novamente.

Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.

Se não houver erros, seguir para o próximo passo.

07.) Executar o programa.

Observar as saídas.

Registrar os resultados.

Versao	Teste	
0.1	01. ( OK )	identificacao de programa leitura e exibicao de inteiro
0.2	01. ( <b>OK</b> )	identificacao de programa leitura e exibicao de real

08.) Copiar a versão atual do programa para outra (nova) – Exemplo0103.c.

09.) Acrescentar ao programa a definição de outro tipo de dado (x):

```
/*
Exemplo0103 - v0.0. - __ / __ / ____
Author: _____

Para compilar em terminal (janela de comandos):
Linux   : gcc -o exemplo0103  exemplo0103.c
Windows: gcc -o exemplo0103  exemplo0103.c

Para executar em terminal (janela de comandos):
Linux   : ./exemplo0103
Windows: exemplo0103
*/
// dependencias
#include <stdio.h>    // para as entradas e saidas

/*
Funcao principal.
@return codigo de encerramento
@param argc - quantidade de parametros na linha de comandos
@param argv - arranjo com o grupo de parametros na linha de comandos
*/
int main ( int argc, char* argv [ ] )
{
    // definir dado
    char x = 'A';           // definir variavel com valor inicial
                          // OBS.: Indispensavel usar apostrofos

    // identificar
    printf ( "%s\n", "EXEMPLO0103 - Programa - v0.0" );
    printf ( "%s\n", "Autor: _____" );
    printf ( "\n" );       // mudar de linha

    // mostrar valor inicial
    printf ( "%s%c\n", "x = ", x );
                          // OBS.: O formato para char -> %c
}
```

```

// ler do teclado
printf ( "Entrar com um caractere: " );
scanf ( "%c", &x );
                                // OBS.: Necessario indicar o endereco -> &
getchar ( );                    // OBS.: Limpar a entrada de dados

// mostrar valor lido
printf ( "%s%c\n", "x = ", x );

// encerrar
printf ( "\n\nApertar ENTER para terminar.\n" );
getchar ( );                    // aguardar por ENTER
return ( 0 );                   // voltar ao SO (sem erros)
} // fim main( )

```

- 10.) Compilar o programa.  
Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.  
Se não houver erros, seguir para o próximo passo.
- 11.) Executar o programa.  
Observar as saídas.  
Registrar os resultados e realizar novos testes.
- 12.) Copiar a versão atual do programa para outra (nova) – Exemplo0104.c.
- 13.) Acrescentar ao programa uma outra definição de dado ( x ):

```

/*
Exemplo0104 - v0.0. - __ / __ / ____
Author: _____

Para compilar em terminal (janela de comandos):
Linux   : gcc -o exemplo0104  exemplo0104.c
Windows: gcc -o exemplo0104  exemplo0104.c

Para executar em terminal (janela de comandos):
Linux   : ./exemplo0104
Windows: exemplo0104
*/
// dependencias
#include <stdio.h>    // para as entradas e saidas
#include <stdbool.h>  // para valores logicos
// Nota: Em alguns compiladores pode haver problemas
//       com relacao a essa biblioteca, caso acontecer
//       considerar o uso de definicoes alternativas.

```

```

/*
Funcao principal.
@return codigo de encerramento
@param argc - quantidade de parametros na linha de comandos
@param argv - arranjo com o grupo de parametros na linha de comandos
*/
int main ( int argc, char* argv [ ] )
{
// definir dado
    bool x = false;           // definir variavel com valor inicial
                                // OBS.: Indispensavel usar minusculas
    int  y = 0;               // definir variável auxiliar

// identificar
    printf ( "%s\n", "EXEMPLO0104 - Programa - v0.0" );
    printf ( "%s\n", "Autor: _____" );
    printf ( "\n" );          // mudar de linha

// mostrar valor inicial
    printf ( "%s%d\n", "x = ", x );
                                // OBS.: O formato para equivalente inteiro -> %d

// ler do teclado
    printf ( "Entrar com um valor logico: " );
    scanf ( "%d", &y );
                                // OBS.: Usar equivalente inteiro -> 0 = false
    getchar ( );               // OBS.: Limpar a entrada de dados

// garantir valor logico no intervalo [0:1]
    x = (y!=0);

// mostrar valor lido
    printf ( "%s%d\n", "x = ", x );

// encerrar
    printf ( "\n\nApertar ENTER para terminar.\n" );
    getchar ( );               // aguardar por ENTER
    return ( 0 );              // voltar ao SO (sem erros)
} // fim main( )

```

```

/*
----- documentacao complementar

----- notas / observacoes / comentarios

```

```

----- previsao de testes

```

- a.) 1
- b.) 0
- c.) true

```

----- historico

```

Versao	Data	Modificacao
0.1	__/__/	esboco

```

----- testes

```



<b>Versao</b>	<b>Teste</b>	
0.1	01. ( OK )	identificacao de programa

\*/

14.) Compilar o programa.

Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.

Se não houver erros, seguir para o próximo passo.

15.) Executar o programa.

Observar as saídas.

Registrar os resultados e realizar novos testes.

16.) Copiar a versão atual do programa para outra (nova) – Exemplo0105.c.

17.) Acrescentar ao programa uma outra definição de dado ( x ):

```
/*
Exemplo0105 - v0.0. - __ / __ / ____
Author: _____

Para compilar em terminal (janela de comandos):
Linux   : gcc -o exemplo0105   exemplo0105.c
Windows: gcc -o exemplo0105   exemplo0105.c

Para executar em terminal (janela de comandos):
Linux   : ./exemplo0105
Windows: exemplo0105
*/
// dependencias
#include <stdio.h>    // para as entradas e saidas
#include <stdbool.h>   // para valores logicos
#include <string.h>    // para cadeias de caracteres

/*
Funcao principal.
@return codigo de encerramento
@param argc - quantidade de parametros na linha de comandos
@param argv - arranjo com o grupo de parametros na linha de comandos
*/
int main ( int argc, char* argv [ ] )
{
    // definir dado
    char x [80] = "abc"; // definir variavel com tamanho e valor inicial
    char *px    = &x[0]; // definir alternativa para acesso via endereco

    // identificar
    printf ( "%s\n", "EXEMPLO0105 - Programa - v0.0" );
    printf ( "%s\n", "Autor: _____" );
    printf ( "\n" );    // mudar de linha

    // mostrar valor inicial
    printf ( "%s%s\n", "x = ", x );
    // OBS.: O formato para caracteres -> %s
}
```

```

// ler do teclado
printf ( "Entrar com uma cadeia de caracteres: " );
scanf ( "%s", x );
                                // OBS.: Nao devera' ser usado o endereco dessa vez !
                                //      O tamanho do valor NAO devera' ultrapassar 80 símbolos.
getchar ( );                    // OBS.: Limpar a entrada de dados

// mostrar valor lido
printf ( "%s%s\n", "x = ", x );

// ler do teclado      (forma alternativa para acesso via endereco)
printf ( "Entrar com outra cadeia de caracteres: " );
scanf ( "%s", px );
getchar ( );           // OBS.: Limpar a entrada de dados

// mostrar valor lido  (forma alternativa para acesso via endereco)
printf ( "%s%s\n", "x = ", px );

// encerrar
printf ( "\n\nApertar ENTER para terminar.\n" );
getchar ( );           // aguardar por ENTER
return ( 0 );          // voltar ao SO (sem erros)
} // fim main( )

/*
----- documentacao complementar

----- notas / observacoes / comentarios

----- previsao de testes

a.) def
b.) d e f
c.) d_e_f

----- historico

Versao      Data      Modificacao
0.1         __/___     esboco

----- testes

Versao      Teste
0.1         01. ( OK )  identificacao de programa

*/

```

18.) Compilar o programa.

Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.

Se não houver erros, seguir para o próximo passo.

19.) Executar o programa.

Observar as saídas.

Registrar os resultados e realizar novos testes.

21.) Acrescentar novos dados e manipulações de seus valores:

```

// mostrar valor resultante
printf ( "%s(%i)*(%i) = (%d)\n", "z = ", x, y, z );

// encerrar
printf ( "\n\nApertar ENTER para terminar.\n" );
getchar( );          // aguardar por ENTER
return ( 0 );        // voltar ao SO (sem erros)
} // fim main( )

/*
----- documentacao complementar

----- notas / observacoes / comentarios

----- previsao de testes

a.) 3 e 5
b.) -3 e 5
c.) -3 e -5

----- historico

Versao      Data      Modificacao
0.1         _/_/___     esboco

----- testes

Versao      Teste
0.1         01. ( OK )   identificacao de programa

*/

```

22.) Compilar o programa.

Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.  
Se não houver erros, seguir para o próximo passo.

23.) Executar o programa.

Observar as saídas.  
Registrar os resultados e realizar novos testes.

24.) Copiar o Exemplo0102.c para outra versão – Exemplo0107.c.

25.) Acrescentar novos dados e manipulações de seus valores:

```
/*
Exemplo0107 - v0.0. - __ / __ / ____
Author: _____

Para compilar em terminal (janela de comandos):
Linux   : gcc -o exemplo0107   exemplo0107.c
Windows: gcc -o exemplo0107   exemplo0107.c

Nota:
Para alguns compiladores poderá ser necessário indicar a inclusão da biblioteca no comando para compilar:

gcc -o exemplo0107 exemplo0107.c -lm

Para executar em terminal (janela de comandos):
Linux   : ./exemplo0107
Windows: exemplo0107
*/
// dependencias
#include <stdio.h>    // para as entradas e saidas
#include <stdbool.h>   // para valores logicos
#include <string.h>    // para cadeias de caracteres
#include <math.h>      // para funcoes matemáticas: pow( ), sqrt( ), sin( ), cos( ) ...

/*
Funcao principal.
@return codigo de encerramento
@param argc - quantidade de parametros na linha de comandos
@param argv - arranjo com o grupo de parametros na linha de comandos
*/
int main ( int argc, char* argv [ ] )
{
    // definir dados
    double x = 0.0;    // definir variavel com valor inicial
    double y = 0.0;    // definir variavel com valor inicial
    double z = 0.0;    // definir variavel com valor inicial

    // identificar
    printf ( "%s\n", "EXEMPLO0107 - Programa - v0.0" );
    printf ( "%s\n", "Autor: _____" );
    printf ( "\n" );    // mudar de linha

    // mostrar valores iniciais
    printf ( "%s%lf\n", "x = ", x );
    printf ( "%s%lf\n", "y = ", y );
    // OBS.: O formato para int -> %d (ou %i)

    // ler do teclado
    printf ( "Entrar com um valor real: " );
    scanf ( "%lf", &x );
    // OBS.: Necessario indicar o endereco -> &

    getchar ( );    // OBS.: Limpar a entrada de dados
    printf ( "Entrar com outro valor real: " );
    scanf ( "%lf", &y );
    // OBS.: Necessario indicar o endereco -> &

    getchar ( );    // OBS.: Limpar a entrada de dados
```

```

// operar valores
z = pow( x, y );      // elevar a base (x) 'a potencia (y)

// mostrar valor resultante
printf ( "%s(%f) elevado a (%f) = (%f)\n", "z = ", x, y, z );

// operar valores
x = pow( z, 1.0/y );  // elevar a base (x) 'a potencia inversa de (y) (raiz)

// mostrar valor resultante
printf ( "%s(%f) elevado a (1/%f) = (%f)\n", "z = ", z, y, x );

// operar valores
z = sqrt( x );        // raiz quadrada do argumento

// mostrar valor resultante
printf ( "%sraiz(%f) = (%f)\n", "z = ", x, z );

// encerrar
printf ( "\n\nApertar ENTER para terminar.\n" );
fflush ( stdin );     // limpar a entrada de dados
getchar( );           // aguardar por ENTER
return ( 0 );          // voltar ao SO (sem erros)
} // fim main( )

/*
----- documentacao complementar

----- notas / observacoes / comentarios

----- previsao de testes

a.) 2.0 e 3.0
b.) 3.0 e 2.0
c.) -3.0 e 2.0
d.) -2.0 e -3.0

----- historico

Versao      Data      Modificacao
0.1         __/___      esboco

----- testes

Versao      Teste
0.1         01. ( OK )      identificacao de programa

*/

```

26.) Compilar o programa.

Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.

Se não houver erros, seguir para o próximo passo.

27.) Executar o programa.

Observar as saídas.

Registrar os resultados e realizar novos testes.

28.) Copiar o Exemplo0105.c para outra versão – Exemplo0108.c.

29.) Acrescentar novos dados e manipulações de seus valores:

```
/*
Exemplo0108 - v0.0. - __ / __ / ____
Author: _____

Para compilar em terminal (janela de comandos):
Linux   : gcc -o exemplo0108   exemplo0108.c
Windows: gcc -o exemplo0108   exemplo0108.c

Para executar em terminal (janela de comandos):
Linux   : ./exemplo0108
Windows: exemplo0108
*/
// dependencias
#include <stdio.h>    // para as entradas e saidas
#include <stdbool.h>   // para valores logicos
#include <string.h>    // para cadeias de caracteres: strcat( ), strcpy( ), strdup( ), strlen( )
#include <math.h>      // para funcoes matematicas

/*
Funcao principal.
@return codigo de encerramento
@param argc - quantidade de parametros na linha de comandos
@param argv - arranjo com o grupo de parametros na linha de comandos
*/
int main ( int argc, char* argv [ ] )
{
    // definir dados
    char x [80] = "abc"; // definir variavel com tamanho e valor inicial
    char y [80] = "def"; // definir variavel com tamanho e valor inicial
    char z [80];         // definir variavel com tamanho inicial
    strcpy ( z, "" );    // e copiar para (z) a representacao de vazio

    // identificar
    printf ( "%s\n", "EXEMPLO0108 - Programa - v0.0" );
    printf ( "%s\n", "Autor: _____" );
    printf ( "\n" );    // mudar de linha

    // mostrar valores iniciais e comprimentos das cadeias
    printf ( "%s%s (%d)\n", "x = ", x, strlen( x ) );
    printf ( "%s%s (%d)\n", "y = ", y, strlen( y ) );
    // OBS.: O formato para int -> %d (ou %i)

    // ler do teclado
    printf ( "Entrar com caracteres: " );
    scanf ( "%s", x );

    // OBS.: Nao indicar o endereco -> &
    getchar ( );    // OBS.: Limpar a entrada de dados
    printf ( "Entrar com outros caracteres: " );
    scanf ( "%s", y );

    // OBS.: Nao indicar o endereco -> &
    getchar ( );    // OBS.: Limpar a entrada de dados
}
```

```

// operar valores
strcpy ( z, x );      // copiar (x) para (z)
strcat ( z, y );      // concatenar (juntar) (y) a (z)
                        // OBS.: Forma mais eficiente

// mostrar valor resultante
printf ( "%s[%s]*[%s] = [%s]\n", "z = ", x, y, z );

// operar valores (forma alternativa)
strcpy ( z, strcat ( strdup(x), y ) );
                        // copiar para (z)
                        // o resultado de concatenar
                        // a copia de (x) com (y)
                        // OBS.: Se nao duplicar, o valor (x) sera' alterado.

// mostrar valor resultante
printf ( "%s[%s]*[%s] = [%s]\n", "z = ", x, y, z );

// encerrar
printf ( "\n\nApertar ENTER para terminar.\n" );
getchar();             // aguardar por ENTER
return ( 0 );          // voltar ao SO (sem erros)
} // fim main( )

/*
----- documentacao complementar

----- notas / observacoes / comentarios

----- previsao de testes

a.) 12 e 24
b.) ab e cd
c.) a e bc
d.) ab e c

----- historico

Versao      Data      Modificacao
0.1         __/___     esboco

----- testes

Versao      Teste
0.1         01. ( OK )   identificacao de programa

*/

```

30.) Compilar o programa.

Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.  
Se não houver erros, seguir para o próximo passo.



- 31.) Executar o programa.  
Observar as saídas.  
Registrar os resultados e realizar novos testes.
- 32.) Copiar o programa atual para outra versão – Exemplo0109.c.
- 33.) Acrescentar novos dados e manipulações de seus valores:

```
/*
Exemplo0109 - v0.0. - __ / __ / ____
Author: _____

Para compilar em terminal (janela de comandos):
Linux   : gcc -o exemplo0109   exemplo0109.c
Windows: gcc -o exemplo0109   exemplo0109.c

Para executar em terminal (janela de comandos):
Linux   : ./exemplo0109
Windows: exemplo0109
*/
// dependencias
#include <stdio.h>    // para as entradas e saidas
#include <stdbool.h>   // para valores logicos
#include <string.h>    // para cadeias de caracteres
#include <math.h>      // para funcoes matematicas

/*
Funcao principal.
@return codigo de encerramento
@param argc - quantidade de parametros na linha de comandos
@param argv - arranjo com o grupo de parametros na linha de comandos
*/
int main ( int argc, char* argv [ ] )
{
    // definir dados singulares
    int    x = 0;      // definir variavel com valor inicial para guardar inteiro
    double y = 3.5;    // definir variavel com valor inicial para guardar real
    char   z = 'A';    // definir variavel com valor inicial para guardar caractere (simbolo)
    bool   w = false;  // definir variavel com valor inicial para guardar falso ou verdadeiro

    // definir dados com mais de um valor
    char   s [80] = ""; // definir espaco de armazenamento para ate' 80 caracteres (simbolos)

    // identificar
    printf ( "%s\n", "EXEMPLO0109 - Programa - v0.0" );
    printf ( "%s\n", "Autor: _____" );
    printf ( "\n" );      // mudar de linha

    // mostrar valores iniciais
    printf ( "01. %s%d\n", "x = ", x );
    printf ( "02. %s%lf\n", "y = ", y );
    printf ( "03. %s%c\n", "z = ", z );
}
```

```

// converter entre tipos de dados (type casting)
x = (int) z; // código inteiro equivalente ao caractere
printf ( "04. %s%d -> %c\n" , "x = ", x, z );

x = (int) y; // parte inteira de real
printf ( "05. %s%d -> %lf\n" , "x = ", x, y );

x = 97;
z = (char) x; // símbolo equivalente ao código inteiro
printf ( "06. %s%c -> %d\n" , "z = ", z, x );

x = (int) '0'; // código inteiro equivalente ao caractere
z = (char) x; // caractere equivalente ao código inteiro
printf ( "07. %s%c -> %d\n" , "z = ", z, x );

x = w; // código inteiro equivalente ao lógico
printf ( "08. %s%d -> %d\n" , "x = ", x, w );

w = true;
x = w; // código inteiro equivalente ao lógico
printf ( "09. %s%d -> %d\n" , "x = ", x, w );

x = (w==false); // equivalente 'a comparação de igualdade (true igual a false)
printf ( "10. %s%d -> %d\n" , "x = ", x, w );

x = !(w==false); // equivalente ao contrário da comparação de valores (true igual a false)
printf ( "11. %s%d -> %d\n" , "x = ", x, w );

x = (w!=false); // equivalente 'a comparação de diferença (true diferente de false)
printf ( "12. %s%d -> %d\n" , "x = ", x, w );

w = (x == 0); // equivalente 'a comparação de igualdade entre (x) e zero
printf ( "13. %s%d == %d = %d\n" , "w = ", x, 0, w );

w = (x != 0); // equivalente 'a comparação de diferença entre (x) e zero
printf ( "14. %s%d != %d = %d\n" , "w = ", x, 0, w );

w = (x < y); // equivalente 'a comparação entre (x) e (y)
printf ( "15. %s%d < %lf = %d\n" , "w = ", x, y, w );

w = (x <= y); // equivalente 'a comparação entre (x) e (y)
printf ( "16. %s%d <= %lf = %d\n" , "w = ", x, y, w );

w = (y > x); // equivalente 'a comparação entre (x) e (y)
printf ( "17. %s%lf > %d = %d\n" , "w = ", y, x, w );

w = (y >= x); // equivalente 'a comparação entre (x) e (y)
printf ( "18. %s%lf >= %d = %d\n" , "w = ", y, x, w );

x = 4;
w = (x % 2 == 0); // equivalente a testar se é par ou
// resto inteiro (%) da divisão por 2 igual a zero
printf ( "19. %s (%d%%2) ? %d\n" , "e' par ", x, w );

x = 4;
w = (x % 2 != 0); // equivalente a testar se é ímpar ou
// resto inteiro (%) da divisão por 2 diferente de zero
printf ( "20. %s (%d%%2) ? %d\n" , "e' ímpar ", x, w );

```

```

z = '5';
w = ('0'<=z && z<='9');           // equivalente a testar se e' algarismo/digito
                                   // pertence a ['0':'9'] (é igual ou esta' entre '0' e '9')
printf ( "21. %s (%c) ? %d\n" , "e' digito" , z, w );

z = 'x';
w = ('a'<=z && z<='z');           // equivalente a testar se e' letra minuscula,
                                   // pertence a ['a':'z'] (é igual ou esta' entre 'a' e 'z')
printf ( "22. %s (%c) ? %d\n" , "e' minuscula " , z, w );

z = 'X';
w = ( ! ('a'<=z && z<='z' ) );    // equivalente a testar se NAO (!) e' letra minuscula
printf ( "23. %s (%c) ? %d\n" , "nao e' minuscula " , z, w );

z = 'x';
w = ('A'<=z && z<='Z');           // equivalente a testar se e' letra maiuscula
printf ( "24. %s (%c) ? %d\n" , "e' maiuscula " , z, w );

z = 'X';
w = ( ( z < 'A' ) || ( 'Z' < z ) ); // equivalente a testar se NAO e' letra maiuscula,
                                   // esta' fora do intervalo ['a':'z'], ou e' menor que 'a' ou e' maior que 'z'
printf ( "25. %s (%c) ? %d\n" , "nao e' maiuscula " , z, w );

z = '0';
w = ('0'==z || '1'==z);           // equivalente a testar se e' igual a '0' ou a '1'
printf ( "26. %s (%c) ? %d\n" , "e' 0 ou 1 " , z, w );

strcpy ( s, "zero" );             // copiar para (s) <- "zero" (NAO usar '=' com caracteres);
printf ( "27. palavra  = %s\n" , s );

w = (strcmp ( "zero" , s ) == 0); // comparar se caracteres iguais (NAO usar '==' com caracteres);
                                   // Nota: O resultado da comparação sempre devera' ser avaliado
                                   //      em relacao a zero, e sera' igual caso coincida.
printf ( "28. palavra  == %s ? %d\n" , s, w );

strcpy ( s, "um e dois" );        // copiar para (s) <- "outras palavras" (NAO usar '=' com caracteres);
printf ( "29. palavras = %s\n" , s );

w = (strcmp ( "zero" , s ) != 0); // comparar se caracteres diferentes (NAO usar '!=' com caracteres);
                                   // Nota: O resultado da comparação sempre devera' ser avaliado
                                   //      em relacao a zero, e sera' diferente caso NAO coincidir.
printf ( "30. palavra  == %s ? %d\n" , s, w );

// encerrar
printf ( "\n\nApertar ENTER para terminar." );
getchar();                       // aguardar por ENTER
return ( 0 );                    // voltar ao SO (sem erros)
} // fim main( )

```

```

/*
----- documentacao complementar

----- notas / observacoes / comentarios

----- previsao de testes

----- historico

Versao      Data      Modificacao
0.1         _/_      esboco

----- testes

Versao      Teste
0.1         01. ( OK )      identificacao de programa

*/

```

34.) Executar o programa.

Observar as saídas.

Registrar os resultados e realizar novos testes.

35.) Copiar o programa atual para outra versão – Exemplo0110.c.

36.) A versão atual será dependente de uma biblioteca externa io.h,  
que deverá estar presente na mesma pasta do programa.

Seu objetivo é minimizar as dependências e normalizar o uso de conceitos.

```

/*
Exemplo0110 - v0.0. - __ / __ / ____
Author: _____

Para compilar em terminal (janela de comandos):
Linux   : gcc -o exemplo0110   exemplo0110.c
Windows: gcc -o exemplo0110   exemplo0110.c

Para executar em terminal (janela de comandos):
Linux   : ./exemplo0110
Windows: exemplo0110

*/

// dependencias
#include "io.h"      // para definicoes proprias ( na mesma pasta )

```

```

/*
Funcao principal.
@return codigo de encerramento
@param argc - quantidade de parametros na linha de comandos
@param argv - arranjo com o grupo de parametros na linha de comandos
*/
int main ( int argc, char* argv [ ] )
{
// definir dados
int    x = 5 ;      // definir variavel com valor inicial
double y = 3.5;     // definir variavel com valor inicial
char   z = 'A';     // definir variavel com valor inicial
bool   w = TRUE;    // definir variavel com valor inicial
chars  a = IO_new_chars(STR_SIZE); // definir variavel com tamanho inicial
chars  b = IO_new_chars(STR_SIZE); // definir variavel com tamanho inicial
chars  c = IO_new_chars(STR_SIZE); // definir variavel com tamanho inicial

// identificar
IO_id ( "EXEMPLO0110 - Programa - v0.0" );

// concatenar (juntar) cadeias de caracteres
strcpy ( a, "abc" );      // atribuir a variavel (a) o valor constante ("abc")
strcpy ( b, "def" );      // OBS.: a atribuicao de cadeia de caracteres NAO usa (=)

IO_printf ( "\na = %s b = %s\n", a, b );

c = IO_concat ( a, b );   // alternativa melhor para a funcao nativa strcat (a,b)
IO_printf ( "\nc = [%s]+[%s] = [%s]\n", a, b, c );

strcpy ( a, "c = " );
strcpy ( c, STR_EMPTY );  // limpar a cadeia de caracteres

IO_printf ( "%s\n", IO_concat ( a, IO_toString_c ( z ) ) );

IO_printf ( IO_concat ( "x = ", IO_toString_d ( x ) ) );

IO_printf ( IO_concat ( "w = ", IO_toString_b ( w ) ) );

strcpy ( b, STR_EMPTY );
IO_print ( "y = " );
IO_print ( IO_concat ( b, IO_toString_f ( y ) ) );
IO_print ( "\n" );

z = IO_readchar ( "char = " );
IO_printf ( IO_concat ( a, IO_toString_c ( z ) ) );

y = IO_readdouble ( "double = " );
IO_printf ( IO_concat ( a, IO_toString_f ( y ) ) );

x = IO_readint ( "int = " );
IO_printf ( IO_concat ( a, IO_toString_d ( x ) ) );

w = IO_readbool ( "bool = " );
IO_printf ( IO_concat ( a, IO_toString_b ( w ) ) );

b = IO_readstring ( "chars = " );
IO_printf ( IO_concat ( a, b ) );

b = IO_readln ( "line = " );
IO_printf ( IO_concat ( a, b ) );

```

```

// encerrar
IO_pause ( "Apertar ENTER para terminar" );
// chamar metodo para pausar
return ( 0 ); // voltar ao SO (sem erros)
} // fim main( )

/*
----- documentacao complementar

----- notas / observacoes / comentarios

----- previsao de testes

a.) a
b.) 4.2
c.) 10
d.) 1
e.) abc def
f.) abc def

----- historico

Versao      Data      Modificacao
0.1         _/_      esboco

----- testes

Versao      Teste
0.1         01. ( OK )      identificacao de programa

*/

```

37.) Compilar o programa novamente.

Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.

Se não houver erros, seguir para o próximo passo.

38.) Executar o programa.

Observar as saídas.

Registrar os resultados.

Exercícios:

DICAS GERAIS: Consultar os Anexos C 01 e C 02 ou na apostila o capítulo 05 para outros exemplos.  
Prever, testar e registrar todos os dados e os resultados obtidos.

01.) Fazer um programa (Exemplo0111) para:

- definir e ler um valor inteiro do teclado;
- supor que esse valor represente o lado de um quadrado, calcular e mostrar um sexto da área do mesmo.

DICA: Usar constantes reais em expressões que envolvam valores reais.

Exemplo: valor = 5

02.) Fazer um programa (Exemplo0112) para:

- definir e ler um valor inteiro do teclado;
- supor que esse valor represente o lado de um quadrado, calcular e mostrar a área e o perímetro de um quadrado com o dobro do tamanho do lado.

Exemplo: valor = 5

03.) Fazer um programa (Exemplo0113) para:

- definir e ler dois valores inteiros do teclado;
- supor que esses dois valores representem lados de um retângulo, calcular e mostrar um quarto da área do mesmo.

DICA: Usar constantes reais em expressões que envolvam valores reais.

Exemplo: valores = 3 e 5

04.) Fazer um programa (Exemplo0114) para:

- definir e ler dois valores inteiros do teclado;
- supor que esses dois valores representem lados de um retângulo, calcular e mostrar a área e o perímetro de um retângulo com a metade do tamanho dos lados.

Exemplo: valores = 3 e 5

05.) Fazer um programa (Exemplo0115) para:

- definir e ler dois valores reais do teclado;
- supor que esses dois valores representem base e altura de um triângulo, calcular e mostrar a área de um triângulo com um terço da altura do mesmo.

DICA: Usar constantes reais em expressões que envolvam valores reais.

Exemplo: valores = 3 e 5

06.) Fazer um programa (Exemplo0116) para:

- definir e ler um valor real do teclado;
  - supor que esse valor represente o lado de um triângulo equilátero, calcular e mostrar a altura, área e o perímetro do triângulo com o dobro do lado.
- DICA: Usar constantes reais em expressões que envolvam valores reais.

Exemplo: valor = 5.0

07.) Fazer um programa (Exemplo0117) para:

- definir e ler um valor real do teclado;
  - supor que esse valor represente a medida de lados de um cubo, calcular e mostrar o volume do sólido com um quarto da medida do lado.
- DICA: Usar constantes reais em expressões que envolvam valores reais.

Exemplo: valor = 5.0

08.) Fazer um programa (Exemplo0118) para:

- definir e ler três valores reais do teclado;
- supor que esses valores correspondam ao comprimento, à largura e à altura de um paralelepípedo, respectivamente,
- calcular e mostrar o volume do sólido com sete vezes esses valores.

Exemplo: valores = 3.0, 4.0 e 5.0

09.) Fazer um programa (Exemplo0119) para:

- definir e ler um valor real do teclado;
  - supor que esse valor represente o raio de um círculo, calcular e mostrar a área de um círculo com um quarto do raio.
- DICA: Na biblioteca <math.h> há definição da constante equivalente a PI (M\_PI).

Exemplo: valor = 5.0

10.) Fazer um programa (Exemplo0120) para:

- definir e ler um valor real do teclado;
  - supor que esse valor represente o raio de uma esfera, calcular e mostrar o volume de uma esfera com cinco oitavos do raio.
- DICA: Usar constantes reais em expressões que envolvam valores reais.

Exemplo: valor = 5.0



## Tarefas extras

E1.) Fazer um programa (Exemplo01E1) para:

- definir e ler um valor real do teclado;
- supor que esse valor informe a área de uma circunferência,
- calcular e mostrar o raio do triplo dessa área.

Exemplo: valor = 5.0

E2.) Fazer um programa (Exemplo01E2) para:

- definir e ler um valor real do teclado;
- supor que esse valor represente o volume de uma esfera;
- calcular e mostrar o raio de um oitavo do volume da esfera e a área de sua superfície.

DICA: Usar constantes reais em expressões que envolvam valores reais.

Exemplo: valor = 5.0