

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=q51cHsgRHU4&list=PL8iN9FQ7_jt7pMKtbgoc0uUQjoJK-3dYu&index=2

Orientações gerais:

A melhor maneira de lidar com um estudo dirigido

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 Exemplo0000.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:

```
/*
Exemplo0000 - v0.0. - __ / __ / ____
Author: _____

Para compilar em terminal (janela de comandos):
Linux   : gcc -o exemplo0000   exemplo0000.c
Windows: gcc -o exemplo0000   exemplo0000.c

Para executar em terminal (janela de comandos):
Linux   : ./exemplo0000
Windows: exemplo0000
*/
// 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 dados / resultados

    // identificar
    printf ( "%s\n", "Exemplo0000 - Programa = v0.0" );
    printf ( "%s\n", "Autor: _____" );
    printf ( "\n" );    // mudar de linha

    // acoes

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

```

/*
----- documentacao complementar

----- notas / observacoes / comentarios

----- previsao de testes

```

- a.) 5
- b.) -5
- c.) 123456789

```

----- historico

Versao      Data      Modificacao
0.1         _/_/   esboco

----- testes

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

*/

```

02.) Compilar o programa.

Se houver erros, identificar, individualmente. a referência para a linha onde ocorre.
 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
0.0         00. ( 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) – Exemplo0001.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 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.

```
/*
Exemplo0001 - v0.0. - __ / __ / ____
Author: _____

Para compilar em terminal (janela de comandos):
Linux   : gcc -o exemplo0001 exemplo0001.c
Windows: gcc -o exemplo0001 exemplo0001.c

Para executar em terminal (janela de comandos):
Linux   : ./exemplo0001
Windows: exemplo0001
*/
// dependencias
#include <stdio.h>    // para as entradas e saidas
#include <stdlib.h>    // para outras funcoes de uso geral

/*
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 / resultados

    // identificar
    printf ( "%s\n", "Exemplo0001 - Programa = v0.0" );
    printf ( "%s\n", "Autor: _____" );
    printf ( "\n" );    // mudar de linha

    // acoes

    // encerrar
    printf ( "\n\nApertar ENTER para terminar." );
    getchar();    // aguardar por ENTER
    return ( 0 );    // voltar ao SO (sem erros)
} // end 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	
0.0	01. (OK)	identificacao de programa
0.1	01. (____)	identificacao de programa

*/

- 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	
0.0	01. (OK)	identificacao de programa
0.1	01. (OK)	identificacao de programa

08.) Copiar a versão atual do programa para outra (nova) – Exemplo0002.c.

09.) 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 presentes em comentários.
Incluir na documentação complementar as alterações feitas,
acrescentar indicações de mudança de versão e
prever novos testes.

```
/*
Exemplo0002 - v0.0. - __ / __ / ____
Author: _____

Para compilar em terminal (janela de comandos):
Linux   : gcc -o exemplo0002 exemplo0002.c
Windows: gcc -o exemplo0002 exemplo0002.c

Para executar em terminal (janela de comandos):
Linux   : ./exemplo0002
Windows: exemplo0002
*/
// dependencias
#include <stdio.h>    // para as entradas e saidas
#include <stdlib.h>    // para outras funcoes de uso geral

/*
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 / resultados
    int opcao = 0;

    // identificar
    printf ( "%s\n", "Exemplo0002 - Programa = v0.0" );
    printf ( "%s\n", "Autor: _____" );
    printf ( "\n" );    // mudar de linha

    // acoes
    // ler a opcao do teclado
    printf ( "\n%s", "Opcao = " );
    scanf ( "%d", &opcao );
    getchar();    // para limpar a entrada de dados

    // para mostrar a opcao lida
    printf ( "\n%s%d", "Opcao = ", opcao );

    // encerrar
    printf ( "\n\nApertar ENTER para terminar." );
    getchar();    // aguardar por ENTER
    return ( 0 );    // voltar ao SO (sem erros)
} // end 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	
0.0	00. (OK)	identificacao de programa leitura e exibicao de inteiro
0.1	01. (____)	identificacao de programa

*/

10.) 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.

11.) Executar o programa.

Observar as saídas.

Registrar os resultados.

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

12.) Copiar a versão atual do programa para outra (nova) – Exemplo0003.c.

13.) 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 presentes em comentários.

Incluir na documentação complementar as alterações feitas,

acrescentar indicações de mudança de versão e

prever novos testes.

```

/*
Exemplo0003 - v0.0. - __ / __ / ____
Author: _____

Para compilar em terminal (janela de comandos):
Linux   : gcc -o exemplo0003 exemplo0003.c
Windows: gcc -o exemplo0003 exemplo0003.c

Para executar em terminal (janela de comandos):
Linux   : ./exemplo0003
Windows: exemplo0003
*/
// dependencias
#include <stdio.h>    // para as entradas e saidas
#include <stdlib.h>    // para outras funcoes de uso geral

/*
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 / resultados
    int opcao = 0;

    // identificar
    printf ( "%s\n", "Exemplo0003 - Programa = v0.0" );
    printf ( "%s\n", "Autor: _____" );
    printf ( "\n" );    // mudar de linha

    // acoes

    // para mostrar opcoes
    printf ( "\n%s\n", "Opcoes:" );
    printf ( "\n%s" , "Terminar" );
    printf ( "\n" );

    // ler a opcao do teclado
    printf ( "\n%s", "Opcao = " );
    scanf ( "%d", &opcao );
    getchar( );    // para limpar a entrada de dados

    // para mostrar a opcao lida
    printf ( "\n%s%d", "Opcao = ", opcao );

    // escolher acao dependente da opcao
    switch ( opcao )
    {
        case 0: // nao fazer nada
            break;

        default: // comportamento padrao
            printf ( "\nERRO: Opcao invalida.\n" );
            break;
    } // end switch

```



```

// encerrar
printf ( "\n\nApertar ENTER para terminar." );
getchar();          // aguardar por ENTER
return ( 0 );       // voltar ao SO (sem erros)
} // end 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
0.0         00. ( OK )      identificacao de programa
0.1         01. ( ___ )      identificacao de programa

*/

```

- 14.) 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.

- 15.) Executar o programa.
 Observar as saídas.
 Registrar os resultados.

Versao	Teste	
0.0	00. (OK)	identificacao de programa
0.1	01. (OK)	identificacao de programa
0.2	02. (OK)	identificacao de programa
		leitura e exibicao de inteiro
0.3	03. (OK)	identificacao de programa
		leitura e exibicao mediante escolha

16.) Copiar a versão atual do programa para outra (nova) – Exemplo0004.c.

17.) 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 presentes em comentários.
Incluir na documentação complementar as alterações feitas,
acrescentar indicações de mudança de versão e
prever novos testes.

```
/*
Exemplo0004 - v0.0. - __ / __ / ____
Author: _____

Para compilar em terminal (janela de comandos):
Linux   : gcc -o exemplo0004    exemplo0004.c
Windows: gcc -o exemplo0004    exemplo0004.c

Para executar em terminal (janela de comandos):
Linux   : ./exemplo0004
Windows: exemplo0004
*/
// dependencias
#include <stdio.h>    // para as entradas e saidas
#include <stdlib.h>   // para outras funcoes de uso geral

/**
Metodo 01.
*/
void method_01 ( void )
{
    // identificar
    printf ( "%s\n", "Metodo 01" );

    // encerrar
    printf ( "\nApertar ENTER para continuar.\n" );
    getchar ( );
} // end method_01 ( )

/*
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 / resultados
    int opcao = 0;

    // identificar
    printf ( "%s\n", "Exemplo0004 - Programa = v0.0" );
    printf ( "%s\n", "Autor: _____" );
    printf ( "\n" );    // mudar de linha
}
```

```

// acoes

// para mostrar opcoes
printf ( "\n%s\n", "Opcoes:" );
printf ( "\n%s" , "0 - Terminar" );
printf ( "\n%s" , "1 - Metodo 01" );
printf ( "\n" );

// ler a opcao do teclado
printf ( "\n%s", "Opcao = " );
scanf ( "%d", &opcao );
getchar(); // para limpar a entrada de dados

// para mostrar a opcao lida
printf ( "\n%s%d", "Opcao = ", opcao );

// escolher acao dependente da opcao
switch ( opcao )
{
    case 0: // nao fazer nada
        break;
    case 1: // executar metodo 01
        method_01 ();
        break;
    default: // comportamento padrao
        printf ( "\nERRO: Opcao invalida.\n" );
        break;
} // end switch

// encerrar
printf ( "\n\nApertar ENTER para terminar." );
getchar(); // aguardar por ENTER
return ( 0 ); // voltar ao SO (sem erros)
} // end 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
0.0         00. ( OK )      identificacao de programa
0.1         01. ( ___ )      identificacao de programa

*/

```

18.) 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.

19.) Executar o programa.
Observar as saídas.
Registrar os resultados.

Versao	Teste	
0.0	00. (OK)	identificacao de programa
0.1	01. (OK)	identificacao de programa
0.2	02. (OK)	identificacao de programa leitura e exibicao de inteiro
0.3	03. (OK)	identificacao de programa leitura e exibicao mediante escolha
0.4	04. (OK)	identificacao de programa leitura e exibicao mediante escolha

20.) Copiar a versão atual do programa para outra (nova) – Exemplo0005.c.

21.) 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 presentes em comentários.
Incluir na documentação complementar as alterações feitas,
acrescentar indicações de mudança de versão e
prever novos testes.

```
/*
Exemplo0005 - v0.0. - __ / __ / ____
Author: _____

Para compilar em terminal (janela de comandos):
Linux   : gcc -o exemplo0005 exemplo0005.c
Windows: gcc -o exemplo0005 exemplo0005.c

Para executar em terminal (janela de comandos):
Linux   : ./exemplo0005
Windows: exemplo0005
*/
// dependencias
#include <stdio.h>    // para as entradas e saidas
#include <stdlib.h>   // para outras funcoes de uso geral
```

```

/**
Metodo 01.
*/
void method_01 ( void )
{
    // identificar
    printf ( "%s\n", "Metodo 01" );

    // encerrar
    printf ( "\nApertar ENTER para continuar.\n" );
    getchar ( );
} // end method_01 ( )

/*
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 / resultados
    int opcao = 0;

    // identificar
    printf ( "%s\n", "Exemplo0005 - Programa = v0.0" );
    printf ( "%s\n", "Autor: _____" );
    printf ( "\n" );      // mudar de linha

    // acoes

    // repetir
    do
    {
        // para mostrar opcoes
        printf ( "\n%s\n", "Opcoes:" );
        printf ( "\n%s" , "0 - Terminar" );
        printf ( "\n%s" , "1 - Metodo 01" );
        printf ( "\n" );

        // ler a opcao do teclado
        printf ( "\n%s", "Opcao = " );
        scanf ( "%d", &opcao );
        getchar();      // para limpar a entrada de dados

        // para mostrar a opcao lida
        printf ( "\n%s%d", "Opcao = ", opcao );
    }

```

```

// escolher acao dependente da opcao
switch ( opcao )
{
    case 0: // nao fazer nada
        break;
    case 1: // executar metodo 01
        method_01 ( );
        break;
    default: // comportamento padrao
        printf ( "\nERRO: Opcao invalida.\n" );
        break;
} // end switch
}
while ( opcao != 0 );

// encerrar
printf ( "\n\nApertar ENTER para terminar." );
getchar( ); // aguardar por ENTER
return ( 0 ); // voltar ao SO (sem erros)
} // end 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
0.0         00. ( OK )      identificacao de programa
0.1         01. ( ___ )      identificacao de programa

*/

```

22.) 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.

- 23.) Executar o programa.
Observar as saídas.
Registrar os resultados.

Versao	Teste	
0.0	00. (OK)	identificacao de programa
0.1	01. (OK)	identificacao de programa
0.2	02. (OK)	identificacao de programa leitura e exibicao de inteiro
0.3	03. (OK)	identificacao de programa leitura e exibicao mediante escolha
0.4	04. (OK)	identificacao de programa leitura e exibicao mediante escolha
0.5	05. (OK)	identificacao de programa leitura e exibicao mediante escolha com repeticao

Exercícios:

DICAS GERAIS: Consultar os Anexos C 01 e C 02 para outros exemplos.

Prever, testar e registrar todos os dados e os resultados obtidos.

01.) Fazer um programa (Exemplo00M0) para:

- montar um programa com opções para testar métodos;
- colocar em métodos **todos os 30 primeiros exemplos** em 2025-1_aed1_exemplos_C;
- para cada método acrescentar uma opção correspondente para teste no programa principal;
- testar todos os métodos e anotar suas saídas ao final (como comentários).

OBS.: Montar cada programa em uma única lista para torná-los métodos testáveis.

Programa isolado	Método a ser testado
<pre>int main (void) { // definicoes // comandos // ... return (0); } // end main ()</pre>	<pre>void method_xx (void) { // definicoes // comandos // ... } // end method_xx // programa principal int main (int argc, char* argv []) { // definir dados / resultados int opcao = 0; // ... switch (opcao) { case 0: // nao fazer nada break; case xx: // executar metodo xx method_xx (); break; default: // comportamento padrao printf ("\nERRO: Opcao invalida.\n"); break; } // end switch // ... // encerrar printf ("\n\nApertar ENTER para terminar."); getchar(); // aguardar por ENTER return (0); // voltar ao SO (sem erros) } // end main () /* ----- testes */</pre>

Tarefas extras

E1.) Fazer um programa (Exemplo00E1) para:

- montar um programa com o índice de **todos os 30 exemplos** de programas montados;
- testar todos os métodos e anotar suas saídas ao final (como comentários).

E2.) Fazer um programa (Exemplo00E2) para:

- montar uma lista com 10 métodos vazios (só com identificações e encerramentos);
- montar a parte principal para chamar **todos os 10 métodos em sequência**;
- testar todos os métodos e anotar suas saídas.

DICA: Alterar a repetição (teste no fim->teste no início com variação crescente).