PUC-Minas - Ciência da Computação
AED1 – Estudo Dirigido 02

Preparação

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

https://www.youtube.com/watch?v=84mgFRR_ODo&list=PL8iN9FQ7_jt4DJbeQqv--jpTy-2gTA3Cp&index=13
https://www.youtube.com/watch?v=YR-ku4OdPJU&list=PL8iN9FQ7_jt4DJbeQqv--jpTy-2gTA3Cp&index=14
https://www.youtube.com/watch?v=JBFgiNJevqc&list=PL8iN9FQ7_jt4DJbeQqv--jpTy-2gTA3Cp&index=15
https://www.youtube.com/watch?v=z395-PmpzII&list=PL8iN9FQ7_jt4DJbeQqv--jpTy-2gTA3Cp&index=17

Tema: Introdução à programação
Atividade: Testes, repetições e alternativas em C

01.) Editar e salvar um esboço de programa em C,
para usar alternativas simples:

```
/*
  Exemplo0201 - v0.0. - __ / __ / _____
  Author: _____

  Para compilar em terminal (janela de comandos):
  Linux     : gcc -o exemplo0201       exemplo0201.c
  Windows: gcc -o exemplo0201.exe exemplo0201.c

  Para executar em terminal (janela de comandos):
  Linux     :  ./exemplo0201
  Windows:    exemplo0201
 */
// dependencias
#include "io.h"            // para definicoes proprias

/*
  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 ( )
{
 // definir dado
    int x = 0;                 // definir variavel com valor inicial

 // identificar
    IO_id ( "EXEMPLO0201 - Programa - v0.0" );
```

```
 // ler do teclado
    x = IO_readint ( "Entrar com um valor inteiro: " );

 // testar valor
    if ( x == 0 )
    {
      IO_printf ( "%s (%d)\n", "Valor igual a zero", x );
    }
    if ( x != 0 )
    {
      IO_printf ( "%s (%d)\n", "Valor diferente de zero ", x );
    } // fim se

 // encerrar
    IO_pause ( "Apertar ENTER para terminar" );
    return ( 0 );
} // fim main( )



/*
---------------------------------------------- documentacao complementar

---------------------------------------------- notas / observacoes / comentarios



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

a.)  0
b.)  5
c.) -5

---------------------------------------------- historico

Versao      Data            Modificacao
  0.1       __/__           esboco

 ---------------------------------------------- testes

Versao      Teste
  0.1       01. ( OK )      identificacao de programa

*/
```

02.) 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.
Em caso de dúvidas, consultar a apostila, recorrer aos monitores ou apresentá-las ao professor.

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

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

04.) Copiar a versão atual do programa para outra nova – Exemplo0202.c.

05.) Editar mudanças no nome do programa e versão.
Alterar para usar uma alternativa dupla.
Prever novos testes.

```
/*
  Exemplo0202 - v0.0. - __ / __ / _____
  Author: _____
*/
// dependencias
#include "io.h"          // para definicoes proprias

/*
 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 ( )
{
 // definir dado
   int x = 0;            // definir variavel com valor inicial

 // identificar
   IO_id ( "EXEMPLO0202 - Programa - v0.0" );

 // ler do teclado
   x = IO_readint ( "Entrar com um valor inteiro: " );

 // testar valor
   if ( x == 0 )
   {
     IO_printf ( "%s (%d)\n", "Valor igual a zero", x );
   }
   else
   {
     IO_printf ( "%s (%d)\n", "Valor diferente de zero ", x );
   } // fim se

 // encerrar
   IO_pause ( "Apertar ENTER para terminar" );
   return ( 0 );
} // fim main( )
```

/*
---------- documentacao complementar

---------- notas / observacoes / comentarios

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

a.) 0
b.) 5
c.) -5

---------- historico

Versao	Data	Modificacao
0.1	__/__	esboco

---------- testes

Versao	Teste	
0.1	01. (OK)	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.

08.) Copiar a versão atual do programa para outra nova – Exemplo0203.c.

09.) Editar mudanças no nome do programa e versão.
Acrescentar outra alternativa dupla, aninhada.
Prever novos testes.
DICA: Será necessário definir outra variável, para não perder o dado (x).

```
/*
  Exemplo0203 - v0.0. - __ / __ / _____
  Author: _____
*/
// dependencias
#include "io.h"            // para definicoes proprias

/*
 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 ( )
{
 // definir dado
   int x = 0;              // definir variavel com valor inicial

 // identificar
   IO_id ( "EXEMPLO0203 - Programa - v0.0" );

 // ler do teclado
   x = IO_readint ( "Entrar com um valor inteiro: " );

 // testar valor
   if ( x == 0 )
   {
     IO_printf ( "%s (%d)\n", "Valor igual a zero", x );
   }
   else
   {
     IO_printf ( "%s (%d)\n", "Valor diferente de zero ", x );
     if ( x > 0 )
     {
       IO_printf ( "%s (%d)\n", "Valor maior que zero", x );
     }
     else
     {
       IO_printf ( "%s (%d)\n", "Valor menor que zero", x );
     } // fim se
   } // fim se

 // encerrar
   IO_pause ( "Apertar ENTER para terminar" );
   return ( 0 );
} // fim main( )
```

/*
---------- documentacao complementar

---------- notas / observacoes / comentarios

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

a.) 0
b.) 5
c.) -5

---------- historico

Versao	Data	Modificacao
0.1	__/__	esboco

---------- testes

Versao	Teste	
0.1	01. (OK)	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.

12.) Copiar a versão atual do programa para outra nova – Exemplo0204.c.

13.) Editar mudanças no nome do programa e versão.
Incluir condições compostas em condições.
Prever novos testes.
DICA: Será necessário definir outra variável, para controlar a repetição.

```
/*
   Exemplo0204 - v0.0. - __ / __ / _____
   Author: _____
*/
// dependencias
#include "io.h"            // para definicoes proprias

/*
  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 ( )
{
 // definir dado
    double x = 0.0;         // definir variavel com valor inicial

 // identificar
    IO_id ( "EXEMPLO0204 - Programa - v0.0" );

 // ler do teclado
    x = IO_readdouble ( "Entrar com um valor real: " );

 // testar valor
    if ( 1.0 <= x && x <= 10.0 )
    {
      IO_printf ( "%s (%lf)\n", "Valor dentro do intervalo [1:10] ", x );
    }
    else
    {
      IO_printf    ( "%s (%lf)\n", "Valor fora do intervalo [1:10] ", x );
      if ( x < 1.0 )
      {
        IO_printf ( "%s (%lf)\n", "Valor abaixo do intervalo [1:10] ", x );
      }
      else
      {
        IO_printf ( "%s (%lf)\n", "Valor acima  do intervalo [1:10]", x );
      } // fim se
    } // fim se

 // encerrar
    IO_pause ( "Apertar ENTER para terminar" );
    return ( 0 );
} // fim main( )
```

/*
---------- documentacao complementar

---------- notas / observacoes / comentarios

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

a.) 0
b.) 1
c.) 10
d.) -1
e.) 100

---------- historico

Versao	Data	Modificacao
0.1	__/__	esboco

---------- testes

Versao	Teste	
0.1	01. (OK)	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.

16.) Copiar a versão atual do programa para outra nova – Exemplo0205.c.

17.) Editar mudanças no nome do programa e versão.
Acrescentar várias condições compostas e aninhadas.
Prever novos testes.
DICA: Será necessário definir outra variável, para controlar a repetição.

```
/*
  Exemplo0205 - v0.0. - __ / __ / _____
  Author: _____
*/
// dependencias
#include "io.h"            // para definicoes proprias

/*
 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 ( )
{
 // definir dado
   char x = '_';           // definir variavel com valor inicial

 // identificar
   IO_id ( "EXEMPLO0205 - Programa - v0.0" );

 // ler do teclado
   x = IO_readchar ( "Entrar com um caractere: " );

 // testar valor
   if ( 'a' <= x && x <= 'z' )
   {
     IO_printf ( "%s (%c)\n", "Letra minuscula", x );
   }
   else
   {
     IO_printf ( "%s (%c)\n", "Valor diferente de minuscula", x );
     if ( 'A' <= x && x <= 'Z' )
     {
       IO_printf ( "%s (%c)\n", "Letra maiuscula", x );
     }
     else
     {
       if ( '0' <= x && x <= '9' )
       {
         IO_printf ( "%s (%c)\n", "Algarismo", x );
       }
       else
       {
         IO_printf ( "%s (%c)\n", "Valor diferente de algarismo", x );
       } // fim se
     } // fim se
   } // fim se

 // encerrar
   IO_pause ( "Apertar ENTER para terminar" );
   return ( 0 );
} // fim main( )
```

/*
---------- documentacao complementar

---------- notas / observacoes / comentarios

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

a.) a
b.) A
c.) 0
d.) #

---------- historico

Versao	Data	Modificacao
0.1	__/__	esboco

---------- testes

Versao	Teste	
0.1	01. (OK)	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.

20.) Copiar a versão atual do programa para outra nova – Exemplo0206.c.

21.) Editar mudanças no nome do programa e versão.
Combinar condições compostas mediante conectivos lógicos.
Prever novos testes.
DICA: Usar parênteses para identificar cada condição.

```
/*
   Exemplo0206 - v0.0. - __ / __ / _____
   Author: _____
*/
// dependencias
#include "io.h"            // para definicoes proprias

/*
  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 ( )
{
 // definir dado
    char x = '_';          // definir variavel com valor inicial

 // identificar
    IO_id ( "EXEMPLO0206 - Programa - v0.0" );

 // ler do teclado
    x = IO_readchar ( "Entrar com um caractere: " );

 // testar valor
    if ( ( 'a' <= x && x <= 'z' ) ||          // minuscula OU
         ( 'A' <= x && x <= 'Z' ) )           // maiuscula
    {
      IO_printf ( "%s (%c)\n", "Letra", x );
    }
    else
    {
      IO_printf ( "%s (%c)\n", "Valor diferente de letra", x );
    } // fim se

 // encerrar
    IO_pause ( "Apertar ENTER para terminar" );
    return ( 0 );
} // fim main( )
```

```
/*
---------------------------------------------- documentacao complementar

---------------------------------------------- notas / observacoes / comentarios



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

a.)   0
b.)   1
c.)  10
d.)  -1
e.) 100

---------------------------------------------- historico

Versao      Data              Modificacao
 0.1        __/__             esboco

 ---------------------------------------------- testes

Versao      Teste
 0.1        01. ( OK )        identificacao de programa

*/
```

22.) Copiar a versão atual do programa para outra nova – Exemplo0207.c.

23.) Editar mudanças no nome do programa e versão.
Modificar o teste para incluir o uso da negação.
Observar a inversão dos blocos de comandos.
Prever novos testes.
DICA: Usar parênteses para indicar tudo o que deverá ser negado.

```
/*
   Exemplo0207 - v0.0. - __ / __ / _____
   Author: _____
*/
// dependencias
#include "io.h"            // para definicoes proprias

/*
  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 ( )
{
 // definir dado
    char x = '_';            // definir variavel com valor inicial

 // identificar
    IO_id ( "EXEMPLO0207 - Programa - v0.0" );

 // ler do teclado
    x = IO_readchar ( "Entrar com um caractere: " );

 // testar valor
    if (  !  ( ( 'a' <= x && x <= 'z' ) ||          // NAO (minuscula OU
             ( 'A' <= x && x <= 'Z' ) )  )          //          maiuscula)
    {
      IO_printf ( "%s (%c)\n", "Valor diferente de letra", x );
    }
    else
    {
      IO_printf ( "%s (%c)\n", "Letra", x );
    } // fim se

 // encerrar
    IO_pause ( "Apertar ENTER para terminar" );
    return ( 0 );
} // fim main( )
```

```
/*
---------------------------------------------- documentacao complementar

---------------------------------------------- notas / observacoes / comentarios



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

a.)  0
b.)  1
c.)  10
d.)  -1
e.) 100

---------------------------------------------- historico

Versao      Data              Modificacao
 0.1        __/__             esboco

 ---------------------------------------------- testes

Versao      Teste
 0.1        01. ( OK )        identificacao de programa

*/
```

24.) 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.

25.) Executar o programa.
Observar as saídas.
Registrar os valores usados para testes e os resultados.

26.) Copiar a versão atual do programa para outra nova – Exemplo0208.c.

27.) Editar mudanças no nome do programa e versão.
Introduzir o uso de alternativa múltipla
Prever novos testes.

```
/*
  Exemplo0208 - v0.0. - __ / __ / _____
  Author: _____
*/
// dependencias
#include "io.h"            // para definicoes proprias

/*
 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 ( )
{
 // definir dado
   char x = '_';            // definir variavel com valor inicial

 // identificar
   IO_id ( "EXEMPLO0208 - Programa - v0.0" );

 // ler do teclado
   x = IO_readchar ( "Entrar com um caractere ['0','A','a']: " );

 // testar valor
   switch ( x )
   {
    case '0':
      IO_printf ( "%s (%c=%d)\n", "Valor igual do simbolo zero", x, x );
     break;
    case 'A':
      IO_printf ( "%s (%c=%d)\n", "Valor igual 'a letra A", x, x );
     break;
    case 'a':
      IO_printf ( "%s (%c=%d)\n", "Valor igual 'a letra a", x, x );
     break;
    default:   // se nao alguma das opcoes anteriores
      IO_printf ( "%s (%c=%d)\n", "Valor diferente das opcoes ['0','A','a']", x, x );
   } // fim escolher

 // encerrar
   IO_pause ( "Apertar ENTER para terminar" );
   return ( 0 );
} // fim main( )
```

```
/*
---------------------------------------------- documentacao complementar

---------------------------------------------- notas / observacoes / comentarios



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

a.)  0
b.)  A
c.)  a
d.)  1

---------------------------------------------- historico

Versao      Data            Modificacao
 0.1        __/__           esboco

 ---------------------------------------------- testes

Versao      Teste
 0.1        01. ( OK )      identificacao de programa

*/
```

28.) 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.

29.) Executar o programa.
Observar as saídas.
Registrar os valores usados para testes e os resultados.

30.) Copiar a versão atual do programa para outra nova – Exemplo0209.c.

31.) Editar mudanças no nome do programa e versão.
Alterar o tipo de dado usado na alternativa múltipla.
Prever novos testes.

```
/*
   Exemplo0209 - v0.0. - __ / __ / _____
   Author: _____
*/
// dependencias
#include "io.h"            // para definicoes proprias

/*
  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 ( )
{
 // definir dado
    int x = 0;                  // definir variavel com valor inicial

 // identificar
    IO_id ( "EXEMPLO0209 - Programa - v0.0" );

 // ler do teclado
    x = IO_readint ( "Entrar com um inteiro [0,1,2,3]: " );

 // testar valor
    switch ( x )
    {
      case 0:
        IO_printf ( "%s (%d)\n", "Valor igual a zero", x );
       break;
      case 1:
        IO_printf ( "%s (%d)\n", "Valor igual a um  ", x );
       break;
      case 2:
        IO_printf ( "%s (%d)\n", "Valor igual a dois", x );
       break;
      case 3:
        IO_printf ( "%s (%d)\n", "Valor igual a tres", x );
       break;
      default:    // se nao alguma das opcoes anteriores
        IO_printf ( "%s (%d)\n", "Valor diferente das opcoes [0,1,2,3]", x );
    } // fim escolher

 // encerrar
    IO_pause ( "Apertar ENTER para terminar" );
    return ( 0 );
} // fim main( )
```

```
/*
---------------------------------------------- documentacao complementar

---------------------------------------------- notas / observacoes / comentarios



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

a.)  0
b.)  1
c.)  2
d.)  3
e.)  4
 f.) -1

---------------------------------------------- historico

Versao      Data            Modificacao
 0.1        __/__           esboco

 ---------------------------------------------- testes

Versao      Teste
 0.1        01. ( OK )      identificacao de programa

*/
```

32.) 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.

33.) Executar o programa.
Observar as saídas.
Registrar os valores usados para testes e os resultados.

34.) Copiar a versão atual do programa para outra nova – Exemplo0210.c.

35.) Editar mudanças no nome do programa e versão.
Empregar a alternativa múltipla para controle de execução de métodos.
Prever novos testes.

```
/*
  Exemplo0210 - v0.0. - __ / __ / _____
  Author: _____
*/
// dependencias
#include "io.h"            // para definicoes proprias

/**
  Method00 - nao faz nada.
 */
void method00 ( )
{
 // nao faz nada
} // fim method00 ( )

/**
  Method01 - mostrar mensagem com texto constante.
 */
void method01 ( )
{
    IO_println ( "Valor igual a um" );
} // fim method01 ( )

/**
  Method02 - mostrar mensagem com texto constante e
             valor variavel
  @param x - valor a ser exibido
 */
void method02 ( int x )
{
    IO_println ( IO_concat ( "Valor igual a (",
                             IO_concat ( IO_toString_d ( x ), ")\n" ) ) );
} // fim method02 ( )

/**
  Method03 - mostrar mensagem com texto e
             valor variavel
  @param text1 - texto a ser exibido
  @param x     - valor a ser exibido
 */
void method03 ( char* text1, int x )
{
    IO_println ( IO_concat (
                   IO_concat ( text1, " (" ),
                   IO_concat ( IO_toString_d ( x ), ")\n" ) ) );
} // fim method03 ( )
```

```
/*
  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 ( )
{
 // definir dado
    int x = 0;                  // definir variavel com valor inicial

 // identificar
    IO_id ( "EXEMPLO0210 - Programa - v0.0" );

 // ler do teclado
    x = IO_readint ( "Entrar com um inteiro [0,1,2,3]: " );

 // testar valor
    switch ( x )
    {
      case 0:
        method00 ( );
       break;
      case 1:
        method01 ( );
       break;
      case 2:
        method02 ( x );
       break;
      case 3:
        method03 ( "Valor igual a tres", x );
       break;
      default:    // se nao alguma das opcoes anteriores
        IO_println ( IO_concat ( "Valor diferente das opcoes [0,1,2,3] (",
                        IO_concat ( IO_toString_d ( x ), ")" ) ) );
    } // fim escolher

 // encerrar
    IO_pause ( "Apertar ENTER para terminar" );
    return ( 0 );
} // fim main( )
```

/*
---------- documentacao complementar

---------- notas / observacoes / comentarios

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

a.) 0
b.) 1
c.) 2
d.) 3
e.) 4
f.) -1

---------- historico

Versao	**Data**	**Modificacao**
0.1	**__/__**	**esboco**

---------- testes

Versao	**Teste**	
0.1	**01. (OK)**	**identificacao de programa**

***/**

OBS.:
A função ***strcat (s1, s1)*** da biblioteca <***string.h***> não é considerada segura.
Recomendável usar equivalente que ofereça maior segurança.
Na falta de um conversor universal de qualquer tipo para cadeia de caracteres,
usar funções do tipo ***toString( )*** ou equivalentes são também recomendáveis.

36.) Compilar e testar o programa novamente.
Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.
Se não houver erros, testar o programa, anotar os dados e os resultados e seguir em frente.

Exercícios:

DICAS GERAIS: Consultar o Anexo C 02 na apostila para outros exemplos.
Prever, testar e registrar todos os resultados obtidos.
Montar todos os métodos em um único programa conforme o último exemplo.

01.) Incluir um procedimento (Exemplo0211) para:
- ler um valor inteiro do teclado e
- dizer se é par ou ímpar.

DICA: Considerar o zero como par. Testar o resto inteiro da divisão por 2.

02.) Incluir um procedimento (Exemplo0212) para:
- ler um valor inteiro do teclado e
- dizer se é par e maior que -100, ou ímpar e menor que 100.

03.) Incluir um procedimento (Exemplo0213) para:
- ler um valor inteiro do teclado e
- dizer se pertence ao intervalo aberto entre (20:64).

04.) Incluir um procedimento (Exemplo0214) para:
- ler um valor inteiro do teclado e
- dizer é par e também pertence ao intervalo fechado entre [20:64].

05.) Incluir um procedimento (Exemplo0215) para:
- ler um valor inteiro do teclado e
- dizer se pertence aos intervalos fechados [15:48] ou [33:75], ou a apenas a um deles.

06.) Incluir um procedimento (Exemplo0216) para:
- ler dois valores inteiros do teclado e
dizer se o primeiro é ímpar e o segundo é par.

07.) Incluir um procedimento (Exemplo0217) para:
- ler dois valores inteiros do teclado e
dizer se o primeiro é ímpar e positivo, e se o segundo é par e negativo.

08.) Incluir um procedimento (Exemplo0218) para:
- ler dois valores reais do teclado e
dizer se o dobro do primeiro é maior, menor ou igual ao segundo.

09.) Incluir um procedimento (Exemplo0219) para:
- ler três valores reais do teclado e
dizer se o primeiro está entre os outros dois, quando todos forem diferentes entre si.
OBS.: Notar a ordem dos testes.

10.) Incluir um procedimento (Exemplo0220) para:
- ler três valores reais do teclado e
dizer se a metade do primeiro não está entre os outros dois, quando forem diferentes entre si.

Tarefas extras

E1.) Incluir um procedimento (Exemplo02E1) para:
- ler três valores literais (caracteres) do teclado e
  dizer se o dobro do primeiro valor lido está entre os outros dois, ou se é igual a um deles.

E2.) Incluir um procedimento (Exemplo02E2) para:
- ler três valores literais (caracteres) do teclado e
  dizer se o primeiro valor lido está fora do intervalo definido pelos outros dois,
  se todos forem diferentes entre si.
  OBS.: Notar que não há garantias de ser o segundo menor que o terceiro.