

Tema: Introdução à programação II
Atividade: Funções e procedimentos em C

01.) Editar e salvar um esboço de programa em C, cujo nome será Exemplo0401.c, para ler e mostrar certa quantidade de valores:

```
/*
    Exemplo0401 - v0.0. - __ / __ / ____
    Author: _____
*/

// dependencias
#include "io.h"          // para definicoes proprias

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

/**
    Method01 - Repeticao para ler certa quantidade de valores.
*/
void method01 ( )
{
    // definir dado
    int quantidade = 0;
    int valor      = 0;
    int controle   = 0;

    // identificar
    IO_id ( "EXEMPLO0401 - Method01 - v0.0" );

    // ler do teclado
    quantidade = IO_readint ( "Entrar com uma quantidade: " );

    // repetir para a quantidade de vezes informada
    controle = 1;
    while ( controle <= quantidade )
    {
        // ler valor do teclado
        valor = IO_readint ( IO_concat (
                                IO_concat ( "", IO_toString_d ( controle ) ),
                                ". " ) );

        // passar ao proximo valor
        controle = controle + 1;
    } // fim repetir

    // encerrar
    IO_pause ( "Apertar ENTER para continuar" );
} // fim method01 ( )
```

```

/**
    Method02.
*/
void method02 ( )
{
    // identificar
    IO_id ( "EXEMPLO0401 - Method02 - v0.0" );

    // encerrar
    IO_pause ( "Apertar ENTER para continuar" );
} // fim method02 ( )

/**
    Method03.
*/
void method03 ( )
{
    // identificar
    IO_id ( "EXEMPLO0401 - Method03 - v0.0" );

    // encerrar
    IO_pause ( "Apertar ENTER para continuar" );
} // fim method03 ( )

/**
    Method04.
*/
void method04 ( )
{
    // identificar
    IO_id ( "EXEMPLO0401 - Method04 - v0.0" );

    // encerrar
    IO_pause ( "Apertar ENTER para continuar" );
} // fim method04 ( )

/**
    Method05.
*/
void method05 ( )
{
    // identificar
    IO_id ( "EXEMPLO0401 - Method05 - v0.0" );

    // encerrar
    IO_pause ( "Apertar ENTER para continuar" );
} // fim method05 ( )

```

```

/**
    Method06.
*/
void method06 ( )
{
    // identificar
    IO_id ( "EXEMPLO0401 - Method06 - v0.0" );

    // encerrar
    IO_pause ( "Apertar ENTER para continuar" );
} // fim method06 ( )

/**
    Method07.
*/
void method07 ( )
{
    // identificar
    IO_id ( "EXEMPLO0401 - Method07 - v0.0" );

    // encerrar
    IO_pause ( "Apertar ENTER para continuar" );
} // fim method07 ( )

/**
    Method08.
*/
void method08 ( )
{
    // identificar
    IO_id ( "EXEMPLO0401 - Method08 - v0.0" );

    // encerrar
    IO_pause ( "Apertar ENTER para continuar" );
} // fim method08 ( )

/**
    Method09.
*/
void method09 ( )
{
    // identificar
    IO_id ( "EXEMPLO0401 - Method09 - v0.0" );

    // encerrar
    IO_pause ( "Apertar ENTER para continuar" );
} // fim method09 ( )

/**
    Method10.
*/
void method10 ( )
{
    // identificar
    IO_id ( "EXEMPLO0401 - Method10 - v0.0" );

    // encerrar
    IO_pause ( "Apertar ENTER para continuar" );
} // fim method10 ( )

```

```

/*
Funcao principal.
@return codigo de encerramento
*/
int main ( )
{
// definir dado
int x = 0;

// repetir até desejar parar
do
{
// identificar
IO_id ( "EXEMPLO0401 - Programa - v0.0" );

// ler do teclado
IO_println ( "Opcoes" );
IO_println ( " 0 - parar" );
IO_println ( " 1 - ler certa quantidade de valores" );
IO_println ( " 2 - " );
IO_println ( " 3 - " );
IO_println ( " 4 - " );
IO_println ( " 5 - " );
IO_println ( " 6 - " );
IO_println ( " 7 - " );
IO_println ( " 8 - " );
IO_println ( " 9 - " );
IO_println ( "10 - " );
IO_println ( "" );

x = IO_readint ( "Entrar com uma opcao: " );

// testar valor
switch ( x )
{
case 0:
method00 ( );
break;
case 1:
method01 ( );
break;
case 2:
method02 ( );
break;
case 3:
method03 ( );
break;
case 4:
method04 ( );
break;
case 5:
method05 ( );
break;
case 6:
method06 ( );
break;
case 7:
method07 ( );
break;

```

```

    case 8:
        method08 ( );
        break;
    case 9:
        method09 ( );
        break;
    case 10:
        method10 ( );
        break;
    default:
        IO_pause ( "ERRO: Valor invalido." );
    } // fim escolher
}
while ( x != 0 );

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

/*
----- documentacao complementar

----- notas / observacoes / comentarios

----- previsao de testes

a.) -1
b.) 0
c.) 5 e { 1, 2, 3, 4, 5 }

----- historico

Versao      Data      Modificacao
0.1         _/_/    esboco

----- testes

Versao      Teste
0.1         01. ( OK )    identificacao de programa

*/

```

02.) 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.

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

03.) Executar o programa.

Observar as saídas.

Registrar os dados e 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 – Exemplo0402.c.

05.) Editar mudanças no nome do programa e versão.

Acrescentar uma função para testar se um valor é positivo e um método para testá-la com vários valores.

Na parte principal, editar a chamada do método para isso.

Prever novos testes.

```
/**
 * Funcao para determinar se valor inteiro e' positivo.
 * @return true, se positivo; false, caso contrario
 * @param x - valor a ser testado
 */
bool positive ( int x )
{
    // definir dado local
    bool result = false;
    // testar a condicao
    if ( x > 0 )
    {
        result = true;
    } // fim se
    return ( result );
} // fim positive ( )
```

```
/**
 * Method02 - Ler valores e contar positivos.
 */
void method02 ( )
{
    // definir dado
    int quantidade = 0;
    int valor      = 0;
    int controle   = 0;
    int contador   = 0;

    // identificar
    IO_id ( "EXEMPLO0402 - Method02 - v0.0" );

    // ler do teclado
    quantidade = IO_readint ( "Entrar com uma quantidade: " );
```

```

// repetir para a quantidade de vezes informada
controle = 1;
while ( controle <= quantidade )
{
    // ler valor do teclado
    valor = IO_readint ( IO_concat (
                                IO_concat ( "", IO_toString_d ( controle ) ),
                                ": " ) );

    // testar e contar se valor for positivo
    if ( positive ( valor ) )
    {
        contador = contador + 1;
    } // fim se

    // passar ao proximo valor
    controle = controle + 1;
} // fim repetir
// mostrar a quantidade de valores positivos
IO_printf ( "%s%d\n", "Positivos = ", contador );

// encerrar
IO_pause ( "Apertar ENTER para continuar" );
} // fim method02 ( )

```

DICA: Desnecessário comparar o resultado da função com a constante verdadeira (**true**), visto que o resultado da comparação sempre será igual ao valor da função. Dessa forma, evita-se o aumento do custo computacional.

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 dados e os resultados.

08.) Copiar a versão atual do programa para outra nova – Exemplo0403.c.

09.) Editar mudanças no nome do programa e versão.

Acrescentar uma função para testar se um valor pertence a certo intervalo, e um método para testá-la com vários valores.

Na parte principal, editar a chamada do método para isso.

Prever novos testes.

```
/**
  Funcao para determinar se valor inteiro pertence a intervalo aberto.
  @return true, se pertencer; false, caso contrario
  @param x      - valor a ser testado
  @param inferior - limite inferior do intervalo
  @param superior - limite superior do intervalo
*/
bool belongsTo ( int x, int inferior, int superior )
{
  // definir dado local
  bool result = false;
  // testar a condicao
  if ( inferior < x && x < superior )
  {
    result = true;
  } // fim se
  return ( result );
} // fim belongsTo ( )

/**
  Method03 - Ler valores e contar positivos menores que 100.
*/
void method03 ( )
{
  // definir dado
  int quantidade = 0;
  int valor      = 0;
  int controle   = 0;
  int contador   = 0;

  // identificar
  IO_id ( "EXEMPLO0403 - Method03 - v0.0" );

  // ler do teclado
  quantidade = IO_readint ( "Entrar com uma quantidade: " );

  // repetir para a quantidade de vezes informada
  controle = 1;
  while ( controle <= quantidade )
  {
    // ler valor do teclado
    valor = IO_readint ( IO_concat (
      IO_concat ( "", IO_toString_d ( controle ) ),
      ": " ) );

    // testar e contar se valor for positivo
    if ( belongsTo ( valor, 0, 100 ) )
    {
      contador = contador + 1;
    } // fim se
  }
}
```



```

    // passar ao proximo valor
    controle = controle + 1;
} // fim repetir
// mostrar a quantidade de valores positivos
IO_printf ( "%s%d\n", "Positivos menores que 100 = ", contador );

// encerrar
IO_pause ( "Apertar ENTER para continuar" );
} // fim method03 ( )

```

- 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 dados e os resultados.
- 12.) Copiar a versão atual do programa para outra nova – Exemplo0404.c.
- 13.) Editar mudanças no nome do programa e versão.
Acrescentar uma função para testar se um é par,
e um método para testá-la com vários valores.
Na parte principal, editar a chamada do método para isso.
Prever novos testes.

```

/**
Funcao para determinar se valor inteiro e' par.
@return true, se par; false, caso contrario
@param x - valor a ser testado
*/
bool even ( int x )
{
    // definir dado local
    bool result = false;
    // testar a condicao ( resto inteiro (%) da divisao por 2 igual a zero )
    if ( x % 2 == 0 )
    {
        result = true;
    } // fim se
    return ( result );
} // fim even ( )

```

```

/**
  Method04 - Ler valores e contar positivos menores que 100 e pares.
 */
void method04 ( )
{
  // definir dado
  int quantidade = 0;
  int valor      = 0;
  int controle   = 0;
  int contador   = 0;

  // identificar
  IO_id ( "EXEMPLO0404 - Method04 - v0.0" );

  // ler do teclado
  quantidade = IO_readint ( "Entrar com uma quantidade: " );

  // repetir para a quantidade de vezes informada
  controle = 1;
  while ( controle <= quantidade )
  {
    // ler valor do teclado
    valor = IO_readint ( IO_concat (
                          IO_concat ( "", IO_toString_d ( controle ) ),
                          ": " ) );

    // testar e contar se valor for positivo menor que 100 e par
    if ( belongsTo ( valor, 0, 100 ) && even ( valor ) )
    {
      contador = contador + 1;
    } // fim se

    // passar ao proximo valor
    controle = controle + 1;
  } // fim repetir
  // mostrar a quantidade de valores positivos
  IO_printf ( "%s%d\n", "Positivos menores que 100 e pares = ", contador );

  // encerrar
  IO_pause ( "Apertar ENTER para continuar" );
} // fim method04 ( )

```

- 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 dados e os resultados.

- 16.) Copiar a versão atual do programa para outra nova – Exemplo0405.c.

- 17.) Editar mudanças no nome do programa e versão.
Acrescentar testes para combinar funções,
e um método para testá-las.
Na parte principal, editar a chamada do método para isso.
Prever novos testes.

```
/**
  Method05 - Ler valores e contar positivos menores que 100 e pares (alternativo).
 */
void method05 ( )
{
  // definir dado
  int quantidade = 0;
  int valor      = 0;
  int controle   = 0;
  int contador   = 0;
  bool ok        = false;

  // identificar
  IO_id ( "EXEMPLO0405 - Method05 - v0.0" );

  // ler do teclado
  quantidade = IO_readint ( "Entrar com uma quantidade: " );

  // repetir para a quantidade de vezes informada
  controle = 1;
  while ( controle <= quantidade )
  {
    // ler valor do teclado
    valor = IO_readint ( IO_concat (
                          IO_concat ( " ", IO_toString_d ( controle ) ),
                          ": " ) );

    // testar e contar se valor for positivo menor que 100 e par
    ok = belongsTo ( valor, 0, 100 );
    ok = ok && even ( valor );
    if ( ok )
    {
      contador = contador + 1;
    } // fim se

    // passar ao proximo valor
    controle = controle + 1;
  } // fim repetir

  // mostrar a quantidade de valores positivos
  IO_printf ( "%s%d\n", "Positivos menores que 100 e pares = ", contador );

  // encerrar
  IO_pause ( "Apertar ENTER para continuar" );
} // fim method05 ( )
```

- 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 dados e os resultados.
- 20.) Copiar a versão atual do programa para outra nova – Exemplo0406.c.
- 21.) Editar mudanças no nome do programa e versão.
Acrescentar função para testar se um caractere é uma letra minúscula,
e um método para testá-la com vários valores pertencente a uma palavra.
Na parte principal, editar a chamada do método para isso.
Prever novos testes.

```
/**  
    Funcao para determinar se caractere e' letra minuscula.  
    @return true, se par; false, caso contrario  
    @param x - valor a ser testado  
*/  
bool isLowerCase ( char x )  
{  
    // definir dado local  
    bool result = false;  
    // testar a condicao  
    if ( 'a' <= x && x <= 'z' )  
    {  
        result = true;  
    } // fim se  
    return ( result );  
} // fim isLowerCase ( )
```

```

/**
 * Method06 - Ler palavra e contar letras minúsculas.
 */
void method06 ( )
{
    // definir dado
    chars palavra = IO_new_chars ( STR_SIZE );
    int tamanho = 0;
    int posicao = 0;
    char simbolo = '_';
    int contador = 0;

    // identificar
    IO_id ( "EXEMPLO0406 - Method06 - v0.0" );

    // ler do teclado
    palavra = IO_readstring ( "Entrar com uma palavra: " );

    // determinar a quantidade de símbolos na palavra
    tamanho = strlen ( palavra );

    // repetir para a quantidade de vezes informada
    for ( posicao = 0; posicao < tamanho; posicao = posicao + 1 )
    {
        // isolar um símbolo por vez
        simbolo = palavra [ posicao ];
        // testar e contar se caractere é letra minúscula
        if ( isLowerCase ( simbolo ) )
        {
            contador = contador + 1;
        } // fim se
    } // fim repetir

    // mostrar a quantidade de minúsculas
    IO_printf ( "%s%d\n", "Minúsculas = ", contador );

    // encerrar
    IO_pause ( "Apertar ENTER para continuar" );
} // fim method06 ( )

```

- 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 dados e os resultados.
- 24.) Copiar a versão atual do programa para outra nova – Exemplo0407.c.

25.) Editar mudanças no nome do programa e versão.

Acrescentar ao exemplo anterior a exibição de cada letra minúscula encontrada, e um método para testá-la com vários valores pertencente a uma palavra.

Na parte principal, editar a chamada do método para isso.

Prever novos testes.

```
/**
 * Method07 - Ler palavra, contar e mostrar letras minusculas.
 */
void method07 ( )
{
    // definir dado
    chars palavra = IO_new_chars ( STR_SIZE );
    int tamanho = 0;
    int posicao = 0;
    char simbolo = '_';
    int contador = 0;

    // identificar
    IO_id ( "EXEMPLO0407 - Method07 - v0.0" );

    // ler do teclado
    palavra = IO_readstring ( "Entrar com uma palavra: " );

    // determinar a quantidade de simbolos na palavra
    tamanho = strlen ( palavra );

    // repetir para a quantidade de vezes informada
    for ( posicao = 0; posicao < tamanho; posicao = posicao + 1 )
    {
        // isolar um simbolo por vez
        simbolo = palavra [ posicao ];
        // testar e contar se caractere e' letra minuscula
        if ( isLowerCase ( simbolo ) )
        {
            // mostrar
            IO_printf ( "%c ", simbolo );
            // contar
            contador = contador + 1;
        } // fim se
    } // fim repetir

    // mostrar a quantidade de minusculas
    IO_printf ( "\n%s%d\n", "Minusculas = ", contador );

    // encerrar
    IO_pause ( "Apertar ENTER para continuar" );
} // fim method07 ( )
```

26.) 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.

- 27.) Executar o programa.
Observar as saídas.
Registrar os dados e os resultados.
- 28.) Copiar a versão atual do programa para outra nova – Exemplo0408.c.
- 29.) Editar mudanças no nome do programa e versão.
Acrescentar ao exemplo anterior a concatenação de cada letra minúscula encontrada, e um método para testá-la com vários valores pertencente a uma palavra.
Na parte principal, editar a chamada do método para isso.
Prever novos testes.

```
/**
 * Method08 - Ler palavra, contar e mostrar letras minusculas (alternativo).
 */
void method08 ( )
{
    // definir dado
    chars palavra      = IO_new_chars ( STR_SIZE );
    int  tamanho       = 0;
    int  posicao         = 0;
    char simbolo        = '_';
    int  contador       = 0;
    chars minusculas    = IO_new_chars ( STR_SIZE );

    strcpy ( minusculas, STR_EMPTY ); // vazio

    // identificar
    IO_id ( "EXEMPLO0408 - Method08 - v0.0" );

    // ler do teclado
    palavra = IO_readstring ( "Entrar com uma palavra: " );

    // determinar a quantidade de simbolos na palavra
    tamanho = strlen ( palavra );

    // repetir para a quantidade de vezes informada
    for ( posicao = 0; posicao < tamanho; posicao = posicao + 1 )
    {
        // isolar um simbolo por vez
        simbolo = palavra [ posicao ];
        // testar e contar as letras minusculas de uma palavra
        if ( isLowerCase ( simbolo ) )
        {
            // concatenar simbolo encontrado
            minusculas = IO_concat ( minusculas, IO_toString_c ( simbolo ) );
            // contar
            contador = contador + 1;
        } // fim se
    } // fim repetir
}
```

```
// mostrar a quantidade de minúsculas
IO_printf ( "\n%s%d [%s]\n", "Minúsculas = ", contador, minúsculas );

// encerrar
IO_pause ( "Apertar ENTER para continuar" );
} // fim method08 ( )
```

- 30.) 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.
- 31.) Executar o programa.
Observar as saídas.
Registrar os dados e os resultados.
- 32.) Copiar a versão atual do programa para outra nova – Exemplo0409.c.
- 33.) Editar mudanças no nome do programa e versão.
Acrescentar uma função para testar se um caractere é um algarismo,
e um método para testá-la com vários valores pertencente a uma palavra.
Na parte principal, editar a chamada do método para isso.
Prever novos testes.

```
/**
Funcao para determinar se caractere e' digito.
@return true, se par; false, caso contrario
@param x - valor a ser testado
*/
bool isDigit ( char x )
{
// definir dado local
bool result = false;
// testar a condicao
if ( '0' <= x && x <= '9' )
{
result = true;
} // fim se
return ( result );
} // fim isDigit ( )
```



```

/**
  Method09 - Ler palavra e contar os algarismos.
 */
void method09 ( )
{
  // definir dado
  chars palavra = IO_new_chars ( STR_SIZE );
  int tamanho = 0;
  int posicao = 0;
  char simbolo = '_';
  int contador = 0;

  // identificar
  IO_id ( "EXEMPLO0409 - Method09 - v0.0" );

  // ler do teclado
  palavra = IO_readstring ( "Entrar com caracteres: " );

  // determinar a quantidade de simbolos
  tamanho = strlen ( palavra );

  // repetir para a quantidade de vezes informada
  for ( posicao = 0; posicao < tamanho; posicao = posicao + 1 )
  {
    // isolar um simbolo por vez
    simbolo = palavra [ posicao ];
    // testar e contar os algarismos em uma cadeia de caracteres
    if ( isDigit ( simbolo ) )
    {
      // mostrar
      IO_printf ( "%c ", simbolo );
      // contar
      contador = contador + 1;
    } // fim se
  } // fim repetir

  // mostrar a quantidade de digitos
  IO_printf ( "\n%s%d\n", "Algarismos = ", contador );

  // encerrar
  IO_pause ( "Apertar ENTER para continuar" );
} // fim method09 ( )

```

34.) Compilar o programa novamente. Se houver erros, resolvê-los; senão seguir para o próximo passo.

35.) Executar o programa.
 Observar as saídas.
 Registrar os dados e os resultados.

36.) Copiar a versão atual do programa para outra nova – Exemplo0410.c.

37.) Editar mudanças no nome do programa e versão.

Acrescentar uma função alternativa para testar se um caractere é um algarismo, e um método para testá-la com vários valores pertencente a uma palavra.

Na parte principal, editar a chamada do método para isso.

Prever novos testes.

```
/**
  Funcao para determinar se caractere e' digito.
  @return true, se par; false, caso contrario
  @param x - valor a ser testado
*/
bool isADigit ( char x )
{
  return ( '0' <= x && x <= '9' );
} // fim isADigit ( )

/**
  Funcao para concatenar 'a cadeia de caracteres mais um digito.
  @return cadeia de caracteres acrescida de mais um digito
  @param digits - cadeia de caracteres
  @param digit - simbolo a ser acrescentado 'a cadeia de caracteres
*/
chars concatADigit ( chars string, char digit )
{
  return ( IO_concat ( string, IO_toString_c ( digit ) ) );
} // fim concatADigit ( )

/**
  Method10.
*/
void method10 ( )
{
  // definir dado
  chars palavra = IO_new_chars ( STR_SIZE );
  int tamanho = 0;
  int posicao = 0;
  char simbolo = '_';
  chars digitos = IO_new_chars ( STR_SIZE );

  strcpy ( digitos, STR_EMPTY ); // vazio

  // identificar
  IO_id ( "EXEMPLO0410 - Method10 - v0.0" );

  // ler do teclado
  palavra = IO_readstring ( "Entrar com uma palavra: " );

  // determinar a quantidade de simbolos na palavra
  tamanho = strlen ( palavra );
```

```

// repetir para a quantidade de vezes informada
for ( posicao = 0; posicao < tamanho; posicao = posicao + 1 )
{
    // isolar um simbolo por vez
    simbolo = palavra [ posicao ];
    // testar e contar os algarismos em uma cadeia de caracteres
    if ( isADigit ( simbolo ) )
    {
        // concatenar simbolo encontrado
        digitos = concatADigit ( digitos, simbolo );
    } // fim se
} // fim repetir

// mostrar a quantidade de digitos
IO_printf ( "\n%s%d [%s]\n", "Algarismos = ", strlen( digitos ), digitos );

// encerrar
IO_pause ( "Apertar ENTER para continuar" );
} // fim method10 ( )

```

Uma outra maneira para se acrescentar os dígitos a uma cadeia de caracteres poderia ser

```

/**
    Funcao para concatenar 'a cadeia de caracteres mais um digito.
    @return cadeia de caracteres acrescida de mais um digito
    @param digits - cadeia de caracteres
    @param digit - simbolo a ser acrescentado 'a cadeia de caracteres
*/
chars concatADigit ( chars string, char digit )
{
    // testar a existencia da cadeia de caracteres
    if ( string )
    {
        string [ strlen(string) + 1 ] = '\0';    // avançar o terminador
        string [ strlen(string) ] = digit;    // guardar o digito
    } // fim se
    return ( string );
} // fim concatADigit ( )

```

Exercícios:

DICAS GERAIS: Consultar o Anexo C 02 na apostila para outros exemplos.

Montar todos os métodos em um único programa conforme o último exemplo.

Incluir ao final desse programa os valores usados para testes.

01.) Incluir um método (Exemplo0411) para:

- ler dois valores reais para definir um intervalo fechado;
- ler certa quantidade de valores reais e
- contar quantos desses valores estão dentro do intervalo, e quantos estão fora dele.

Exemplo: $n = 10$ e $[14.2 : 23.1]$ com $\{ 5.1, 10.5, 12.4, 14.2, 15.3, 18.3, 20.4, 21.7, 23.1, 24.2 \}$

02.) Incluir um método (Exemplo0412) para:

- ler uma sequência de caracteres do teclado;
 - contar e mostrar a quantidade de letras maiúsculas menores que 'K'.
- DICA: Definir uma função para determinar se um caractere é letra maiúscula.

Exemplo: sequência = AaKkLmM0*Nx

03.) Incluir um método (Exemplo0413) para:

- ler uma sequência de caracteres do teclado;
- mostrar a quantidade de letras maiúsculas menores que 'K' contadas por uma função definida para receber uma cadeia de caracteres como parâmetro.

Exemplo: sequência = AaKkLmM0*Nx

04.) Incluir um método (Exemplo0414) para:

- ler uma sequência de caracteres do teclado;
- mostrar as letras maiúsculas menores que 'K' separadas por uma função definida para receber uma cadeia de caracteres como parâmetro.

Exemplo: sequência = AaKkLmM0*Nx

05.) Incluir um método (Exemplo0415) para:

- ler uma sequência de caracteres do teclado;
- mostrar a quantidade de letras (tanto maiúsculas, quanto minúsculas) menores que 'K' e 'k' contadas por uma função definida para receber uma cadeia de caracteres como parâmetro.

Exemplo: sequência = AaKkLmM0*Nx

06.) Incluir um método (Exemplo0416) para:

- ler uma sequência de caracteres do teclado;
- mostrar as letras (tanto maiúsculas, quanto minúsculas) menores que 'K' e 'k' separadas por uma função definida para receber uma cadeia de caracteres como parâmetro.

Exemplo: sequência = AaKkLmM0*Nx

07.) Incluir um método (Exemplo0417) para:

- ler uma sequência de caracteres do teclado;
- mostrar a quantidade de dígitos ímpares em uma cadeia de caracteres contados por uma função definida para receber uma cadeia de caracteres como parâmetro.

DICA: Considerar o valor inteiro do código equivalente (***type casting***) para teste.

Exemplo: sequência = A1b2C3d4E5f6

08.) Incluir um método (Exemplo0418) para:

- ler uma sequência de caracteres do teclado;
- mostrar todos os símbolos não alfanuméricos (letras e dígitos) em uma cadeia de caracteres separados por meio de uma função.

Exemplo: sequência = (A1b2+C3d4)*E5f6

09.) Incluir um método (Exemplo0419) para:

- ler uma sequência de caracteres do teclado;
- mostrar todos os símbolos alfanuméricos (letras e dígitos) em uma cadeia de caracteres separados por meio de uma função.

Exemplo: sequência = (A1b2+C3d4)*E5f6

10.) Incluir um método (Exemplo0420) para:

- ler certa quantidade de cadeias de caracteres do teclado, uma por vez;
- mostrar e contar a quantidade de símbolos alfanuméricos (letras e dígitos) em cada palavra, por meio de uma função, e calcular o total acumulado de todas as palavras.

Exemplo: sequências = { (A1b2+C3d4)*E5f6, [P&&Q]||[R&&!S], (a<b&&b<c) }

Tarefas extras

E1.) Incluir um método (Exemplo04E1) para:

- ler certa quantidade de cadeias de caracteres do teclado;
 - contar a quantidade de símbolos alfanuméricos, incluindo espaços em branco, em cada palavra, e calcular o total de todas as palavras, por meio de uma função.
- OBS.: Para a leitura incluir espaços em branco, usar `IO_readln()`, ou `gets()`, menos recomendado.

Exemplo: sequência = (A1b2 + C3d4) * E5f6

E2.) Incluir um método (Exemplo04E2) para:

- ler duas cadeias de caracteres do teclado;
- calcular qual das duas sequências possui a menor quantidade de dígitos, por meio de uma função.

Exemplo: sequência = { A1b2, C3d4E5 }