```
//
// OBS.: RETIRAR OS COMENTARIOS /* */
//           PARA TESTAR CADA EXEMPLO INDIVIDUALMENTE.
//

/*
// -------------------------- EXEMPLO0101
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas

int main ( )
{
 // identificar
    printf ( "\n%s\n", "EXEMPLO0101 - PRIMEIRO EXEMPLO EM C" );
 // encerrar
    printf ( "\n%s\n", "APERTAR <Enter> PARA TERMINAR."  );
    getchar( );                 // para esperar
    return ( 0 );
} // end main ( )
*/
/*
// -------------------------- EXEMPLO0102
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral

int main ( )
{
 // identificar
                            // (dependente do sistema operacional)
    system ( "cls" );       // ( Windows ) para limpar a tela
//  system ( "clear" );     // ( Linux   ) para limpar a tela

    printf ( "\n%s\n", "EXEMPLO0102 - PRIMEIRO EXEMPLO EM C" );
 // encerrar
    printf ( "\n%s\n", "APERTAR <Enter> PARA TERMINAR."     );
    getchar( );                 // para esperar
    return ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO0103
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral

// metodo para uso local
   void clrscr ( ) {  system ( "cls"    ); } // para Windows
// void clrscr ( ) {  system ( "clear" ); } // para Linux

int main ( )
{
 // identificar
    printf ( "\n%s\n", "EXEMPLO0103 - PRIMEIRO EXEMPLO EM C" );
                                // (dependente do sistema operacional)
    clrscr ( );                 // para limpar a tela
    printf ( "\n%s\n", "MATRICULA: ______ ALUNO : ____________________" );
 // encerrar
    printf ( "\n%s\n", "APERTAR <Enter> PARA TERMINAR."     );
    getchar( );                 // para esperar
    return ( EXIT_SUCCESS );
} // end main ( )
*/
/*
// -------------------------- EXEMPLO0104
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral

int main ( )
{
 // identificar
    printf ( "\n%s\n", "EXEMPLO0104 - PRIMEIRO EXEMPLO EM C" );
    printf ( "\n" );            // para mudar de linha (="\n")
    printf ( "\n%s\n", "MATRICULA: ______ ALUNO : ____________________" );
    printf ( "\n" );  // para mudar de linha
 // encerrar
    printf ( "\n%s\n", "APERTAR <Enter> PARA TERMINAR."     );
    getchar( );                 // para esperar
    return ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO0105
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral

int main ( )
{
 // identificar
    printf ( "\n%s\n", "EXEMPLO0105 - PRIMEIRO EXEMPLO EM C" );
    printf ( "\n%s\n", "MATRICULA: ______ ALUNO : _____________________" );
    printf ( "\nEXEMPLOS DE VALORES : " );
    printf ( "\nCARACTERE : %c", 'A'    ); // letra ou simbolo
    printf ( "\nINTEIRO   : %d",  10    );     // valor sem parte fracionaria
    printf ( "\nREAL      : %f", 3.1415 );     // valor com parte fracionaria
 // encerrar
    printf ( "\n%s\n", "APERTAR <Enter> PARA TERMINAR."     );
    getchar( );                 // para esperar
    return ( EXIT_SUCCESS );
} // end main ( )
*/
/*
// -------------------------- EXEMPLO0106
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral

#define  PI 3.1415          // definicao de macro (nome para substituir constante global)

int main ( )
{
 // identificar
    printf ( "\n%s\n", "EXEMPLO0106 - PRIMEIRO EXEMPLO EM C" );
    printf ( "\n%s\n", "MATRICULA: ______ ALUNO : _____________________" );
    printf ( "\nEXEMPLOS DE VALORES : " );
    printf ( "\nCARACTERE : %c", 'A' );     // letra ou simbolo
    printf ( "\nINTEIRO   : %d", 10  );         // valor sem parte fracionaria
    printf ( "\nREAL      : %f", PI  );           // emprego de macro
 // encerrar
    printf ( "\n%s\n", "APERTAR <Enter> PARA TERMINAR."     );
    getchar( );                 // para esperar
    return ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO0107
// bibliotecas de funcoes auxiliares
#include <stdio.h>         // para entradas e saídas
#include <stdlib.h>        // para outras funcoes de uso geral

int main ( )
{
 // definicao de constante (local)
    const float PI = 3.14;  // com nome e tipo (melhor)

 // identificar
    printf ( "\n%s\n", "EXEMPLO0107 - PRIMEIRO EXEMPLO EM C" );
    printf ( "\n%s\n", "MATRICULA: ______ ALUNO : _____" );
    printf ( "\nEXEMPLOS DE VALORES : " );
    printf ( "\nCARACTERE : %c", 'A' );     // letra ou simbolo
    printf ( "\nINTEIRO   : %d", 10  );     // valor sem parte fracionaria
    printf ( "\nREAL      : %f", PI  );     // constante real
 // encerrar
    printf ( "\n%s\n", "APERTAR <Enter> PARA TERMINAR."     );
    getchar( );                // para esperar
    return ( EXIT_SUCCESS );
} // end main ( )
*/
/*
// -------------------------- EXEMPLO0108
// bibliotecas de funcoes auxiliares
#include <stdio.h>         // para entradas e saídas
#include <stdlib.h>        // para outras funcoes de uso geral

int main ( )
{
 // definicao de constante
    const float PI = 3.14;  // com nome e tipo (melhor)
 // definicao de variavel real
    float   X = 10.01;         // com atribuicao de valor inicial

 // identificar
    printf ( "\n%s\n", "EXEMPLO0108 - PRIMEIRO EXEMPLO EM C" );
    printf ( "\n%s\n", "MATRICULA: ______ ALUNO : _____" );
    printf ( "\nEXEMPLOS DE VALORES : " );
    printf ( "\nCARACTERE : %c", 'A' );     // letra ou simbolo
    printf ( "\nINTEIRO   : %d", 10  );     // valor sem parte fracionaria
    printf ( "\nREAL      : %f", PI  );     // constante real
    printf ( "\nREAL      : %f", X   );     // variavel real
 // encerrar
    printf ( "\n%s\n", "APERTAR <Enter> PARA TERMINAR."     );
    getchar( );                // para esperar
    return ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO0109
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral

int main ( )
{
 // definicao de constante
    const float PI = 3.14;
 // definicao de variavel real
    float   X = 10.01;
 // definicao de variavel inteira
    int       I  = 10;

 // identificar
    printf ( "\n%s\n", "EXEMPLO0109 - PRIMEIRO EXEMPLO EM C" );
    printf ( "\n%s\n", "MATRICULA: ______ ALUNO : _____________________" );
    printf ( "\nEXEMPLOS DE VALORES : " );
    printf ( "\nINTEIRO : %i" , I  );
    printf ( "\nREAL    : %f" , X  );
    printf ( "\nREAL    : %f" , PI );
 // encerrar
    printf  ( "\n%s\n", "APERTAR <Enter> PARA TERMINAR."     );
    getchar ( );                // para esperar
    return ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO0110
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral

int main ( )
{
 // definicao de constante
    const float PI = 3.14;
 // definicao de variavel real
    float   X = 10.01;
 // definicao de variavel inteira
    int       I  = 10;
 // definicao de variavel caractere
    char   N = '\n';            // mudar de linha

 // identificar
    printf ( "\n%s\n", "EXEMPLO0106 - PRIMEIRO EXEMPLO EM C" );
    printf ( "\n%s\n", "MATRICULA: ______ ALUNO : _____________________" );
    printf ( "%c%s"  , N, "EXEMPLOS DE VALORES : " );
    printf ( "%c%s%i", N, "INTEIRO : " , I  );
    printf ( "%c%s%f", N, "REAL    : " , X  );
    printf ( "%c%s%f", N, "REAL    : " , PI );
 // encerrar
    printf  ( "\n%s\n", "APERTAR <Enter> PARA TERMINAR."     );
    getchar ( );                // para esperar
    return ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
//
// OBS.: RETIRAR OS COMENTARIOS /* */
//           PARA TESTAR CADA EXEMPLO INDIVIDUALMENTE.
//

/*
// -------------------------- EXEMPLO0201
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral

int main ( )
{
 // PROGRAMA PARA LER E IMPRIMIR UM VALOR INTEIRO
 // VARIAVEL:
    int X = 0;

 // identificar
    printf ( "EXEMPLO0201 - LER E IMPRIMIR UM VALOR INTEIRO" );
    printf ( "\nFORNECER UM VALOR INTEIRO QUALQUER: " );
    scanf ( "%d", &X );
    getchar( );                 // limpar a entrada de dados
    printf ( "\nO VALOR DIGITADO FOI: %d", X );
 // encerrar
    printf ( "\n%s\n", "APERTAR <Enter> PARA TERMINAR."     );
    getchar( );                 // para esperar
    return ( EXIT_SUCCESS );
} // end main ( )
*/
/*
// ------------------------------------ EXEMPLO0202
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral

int main ( )
{
 // PROGRAMA PARA LER E IMPRIMIR UM VALOR REAL
 // VARIAVEL:
    float X = 0.0;

 // identificar
    printf ( "EXEMPLO0202 - LER E IMPRIMIR UM VALOR REAL" );
    printf ( "\nFORNECER UM VALOR REAL QUALQUER: " );
    scanf ( "%f", &X );
    getchar( );                 // limpar a entrada de dados
    printf ( "\nO VALOR DIGITADO FOI: %f", X );
 // encerrar
    printf ( "\n%s\n", "APERTAR <Enter> PARA TERMINAR."     );
    getchar( );                 // para esperar
    return ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// ------------------------------------ EXEMPLO0203
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral

int main ( )
{
 // PROGRAMA PARA LER E IMPRIMIR UM CARACTERE
 // VARIAVEL:
    char X = '0';

 // identificar
    printf ( "EXEMPLO0203 - LER E IMPRIMIR UM CARACTERE" );
    printf ( "\nFORNECER UM CARACTERE QUALQUER: " );
    scanf ( "%c", &X );
    getchar( );                 // limpar a entrada de dados
    printf ( "\nO VALOR DIGITADO FOI: %c", X );
 // encerrar
    printf ( "\n%s\n", "APERTAR <Enter> PARA TERMINAR."     );
    getchar( );                 // para esperar
    return ( EXIT_SUCCESS );
} // end main ( )
*/
/*
// ------------------------------------ EXEMPLO0204
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral

int main ( )
{
 // PROGRAMA PARA LER E IMPRIMIR, NO MAXIMO, 10 CARACTERES
 // VARIAVEL:
    char X [10];

 // identificar
    printf ( "EXEMPLO0204 - LER E IMPRIMIR, NO MAXIMO, 09 CARACTERES" );
    printf ( "\nDIGITE, NO MAXIMO, 09 CARACTERES QUAISQUER: " );
    scanf ( "%s", X );          // OBS.: NAO usar o (&) para caracteres !
    getchar( );                 // limpar a entrada de dados
    printf ( "\nFOI DIGITADO: %s", X );
 // encerrar
    printf ( "\n%s\n", "APERTAR <Enter> PARA TERMINAR."     );
    getchar( );                 // para esperar
    return ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// ------------------------------------ EXEMPLO0205
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral

int main ( )
{
 // PROGRAMA PARA LER E SOMAR DOIS VALORES INTEIROS
 // VARIAVEIS:
    int  X=0, Y=0, Z=0;

 // identificar
    printf ( "EXEMPLO0205 - LER E SOMAR DOIS VALORES INTEIROS" );
    printf ( "\nFORNECER UM VALOR INTEIRO QUALQUER: " );
    scanf ( "%d", &X );
    getchar( );                 // limpar a entrada de dados
    printf ( "\nFORNECER OUTRO VALOR INTEIRO QUALQUER: " );
    scanf  ( "%d", &Y );
    getchar( );                 // limpar a entrada de dados
    Z = X + Y;
    printf ( "\nA SOMA DOS DOIS = %d", Z );
 // encerrar
    printf ( "\n%s\n", "APERTAR <Enter> PARA TERMINAR."     );
    getchar( );                 // para esperar
    return ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// ------------------------------------ EXEMPLO0206
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral

int main ( )
{
 // PROGRAMA PARA LER E SUBTRAIR DOIS VALORES REAIS
 // VARIAVEIS:
    float X=0.0, Y=0.0, Z=0.0;

 // identificar
    printf ( "EXEMPLO0206 - LER E SUBTRAIR DOIS VALORES REAIS" );
    printf ( "\nFORNECER UM VALOR REAL QUALQUER: " );
    scanf ( "%f", &X );
    getchar( );                 // limpar a entrada de dados
    printf ( "\nFORNECER OUTRO VALOR REAL QUALQUER: " );
    scanf ( "%f", &Y );
    getchar( );                 // limpar a entrada de dados
    Z = X - Y;
    printf ( "\nA DIFERENCA ENTRE OS DOIS = %f", Z );
 // encerrar
    printf ( "\n%s\n", "APERTAR <Enter> PARA TERMINAR."     );
    getchar( );                 // para esperar
    return ( EXIT_SUCCESS );
} // end main ( )
*/
/*
// ------------------------------------ EXEMPLO0207
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral
#include <stdbool.h>        // para valores logicos

int main ( )
{
 // PROGRAMA PARA OPERAR VALORES LOGICOS
 // VARIAVEIS:
    bool X=false, Y=false, Z=false;

 // identificar
    printf ( "EXEMPLO0207 - OPERAR VALORES LOGICOS" );
    X = true;
    Y = false;
    Z = X || Y;                 // X ou Y
    printf ( "\nA DISJUNCAO ENTRE VERDADEIRO E FALSO = %d", Z );
    Z = X && Y;                 // X e  Y
    printf ( "\nA CONJUNCAO ENTRE VERDADEIRO E FALSO = %d", Z );
 // encerrar
    printf ( "\n%s\n", "APERTAR <Enter> PARA TERMINAR."     );
    getchar( );                 // para esperar
    return ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// ------------------------------------ EXEMPLO0208
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral

int main ( )
{
 // PROGRAMA PARA CALCULAR A VELOCIDADE DE UM VEICULO
 // VARIAVEIS:
    float  D = 0.0, // Distancia
           T = 0.0, // Tempo
           V = 0.0; // Velocidade

 // identificar
    printf ( "EXEMPLO0208 - CALCULAR A VELOCIDADE DE UM VEICULO" );
    printf ( "\nFORNECER UMA DISTANCIA QUALQUER EM METROS: " );
    scanf ( "%f", &D );
    getchar( );                 // limpar a entrada de dados
    printf ( "\nFORNECER O TEMPO PARA PERCORRE-LA EM SEGUNDOS: " );
    scanf ( "%f", &T );
    getchar( );                 // limpar a entrada de dados
    V = D / T;
    printf ( "\nV = D / T = %f%s", V, " m/s " );
 // encerrar
    printf ( "\n%s\n", "APERTAR <Enter> PARA TERMINAR."     );
    getchar( );                 // para esperar
    return ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// ------------------------------------ EXEMPLO0209
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral
#include <string.h>         // para lidar com caracteres

int main ( )
{
 // PROGRAMA PARA COMPARAR CARACTERES COM UMA SENHA
 // CONSTANTE:
    const char SENHA[5] = "XXXX";
 // VARIAVEL:
    char S [10];

 // identificar
    printf ( "EXEMPLO0209 - COMPARAR CARACTERES COM UMA SENHA" );
    printf ( "\nFORNECER UMA CADEIA DE CARACTERES QUALQUER: " );
    scanf ( "%s", S ); // OBS.: NAO usar o (&) para caracteres !
    getchar( );                 // limpar a entrada de dados
    printf ( "\nA COMPARACAO COM A SENHA = %d", (strcmp(S,SENHA)==0)?1:0 );
//              strcmp(S1,S2) compara S1 com S2
//              igual a 0: S1 =  S2 => 1 ( verdadeiro )
//              diferente: S1 <> S2 => 0 (   falso    )
 // encerrar
    printf ( "\n%s\n", "APERTAR <Enter> PARA TERMINAR."     );
    getchar( );                 // para esperar
    return ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// ------------------------------------ EXEMPLO0210
// bibliotecas de funcoes auxiliares
#include <stdio.h>         // para entradas e saídas
#include <stdlib.h>        // para outras funcoes de uso geral
#include <math.h>          // para operacoes matematicas

int main ( )
{
 // PROGRAMA PARA CALCULAR O ARCO TRIGONOMETRICO DE UM SENO
 // CONSTANTE:
    const double PI       = 3.14;
 // VARIAVEIS:
    double ARCO         = 0.0,
           COSSENO  = 0.0,
           SENO         = 0.0,
           TANGENTE = 0.0;

 // identificar
    printf ( "EXEMPLO0210 - CALCULAR O ARCO TRIGONOMETRICO DE UM SENO" );
    printf ( "\nFORNECER O VALOR DO SENO: " );
    scanf ( "%lf", &SENO );
    getchar( );                // limpar a entrada de dados
    COSSENO  = sqrt( 1.0 - pow(SENO,2) ); // raiz quadrada
    TANGENTE = SENO / COSSENO;
    ARCO     = atan( TANGENTE );          // arcotangente
    printf ( "\nO ARCO TRIGONOMETRICO EM GRAUS = %lf", (ARCO*180.0/PI) );
 // encerrar
    printf ( "\n%s\n", "APERTAR <Enter> PARA TERMINAR."     );
    getchar( );                // para esperar
    return ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
//
// OBS.: RETIRAR OS COMENTARIOS /* */
//           PARA TESTAR CADA EXEMPLO INDIVIDUALMENTE.
//

/*
// ------------------------------------ EXEMPLO301
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral

int main ( )
{
 // PROGRAMA PARA LER UM VALOR INTEIRO E VERIFICAR SE E' ZERO
 // VARIAVEL:
    int X = 0;

    printf ( "EXEMPLO301 - LER E TESTAR UM VALOR INTEIRO" );
    printf ( "\nFORNECER UM VALOR INTEIRO QUALQUER: " );
    scanf ( "%d", &X );
    getchar( );                 // limpar a entrada de dados
    if ( X == 0 )
      printf ( "\nO VALOR DIGITADO FOI ZERO" );
    else
      printf ( "\nO VALOR DIGITADO NAO FOI ZERO" );
    printf    ( "\nPRESSIONAR <Enter> PARA TERMINAR." );
    getchar( );                 // para esperar
    return   ( EXIT_SUCCESS );
} // fim do programa
*/
/*
// ------------------------------------ EXEMPLO302
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral

int main ( )
{
 // PROGRAMA PARA LER UM REAL E TESTAR SE DIFERENTE DE ZERO
 // VARIAVEL:
    float X = 0.0;

    printf ( "EXEMPLO302 - LER E TESTAR UM VALOR REAL" );
    printf ( "\nFORNECER UM VALOR REAL DIFERENTE DE ZERO: " );
    scanf ( "%f", &X );
    getchar( );                 // limpar a entrada de dados
    if( X != 0.0 )
      printf  ( "\nO VALOR DIGITADO FOI DIFERENTE DE ZERO" );
    printf    ( "\nPRESSIONAR <Enter> PARA TERMINAR." );
    getchar( );                 // para esperar
    return   ( EXIT_SUCCESS );
} // fim do programa
*/
```

```
/*
// ------------------------------------ EXEMPLO303
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral

int main ( )
{
 // PROGRAMA PARA LER CARACTERE E VERIFICAR SE E' UM ALGARISMO
 // VARIAVEL:
    char X = '0';

    printf ( "EXEMPLO303 - LER E TESTAR UM CARACTERE" );
    printf ( "\nFORNECER UM ALGARISMO QUALQUER: " );
    scanf ( "%c", &X );
    getchar( );                 // limpar a entrada de dados
    if( X >= '0' && X <= '9' )
    {
      printf ( "\nO VALOR DIGITADO FOI UM ALGARISMO" );
      printf ( "\nO ALGARISMO DIGITADO FOI: %c", X );
    } // if ALGARISMO
    printf ( "\nPRESSIONAR <Enter> PARA TERMINAR." );
    getchar( );                 // para esperar
    return ( EXIT_SUCCESS );
} // fim do programa
*/
/*
// ------------------------------------ EXEMPLO304
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral

int main ( )
{
 // PROGRAMA PARA LER CARACTERE E TESTAR SE NAO E' ALGARISMO
 // VARIAVEL:
    char X =  '0';

    printf ( "EXEMPLO304 - LER E TESTAR CARACTERE" );
    printf ( "\nFORNECER UM CARACTERE QUALQUER: " );
    scanf ( "%c", &X );
    getchar( );                 // limpar a entrada de dados
    if( !( X >= '0' && X <= '9') )
    {
       printf ( "\nNAO FOI DIGITADO UM ALGARISMO" );
       printf ( "\nFOI DIGITADO O CARACTERE: %c", X );
    } // if NAO ALGARISMO
    printf    ( "\nPRESSIONAR <Enter> PARA TERMINAR." );
    getchar( );                 // para esperar
    return   ( EXIT_SUCCESS );
} // fim do programa
*/
```

```
/*
// ------------------------------------ EXEMPLO305
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral

int main ( )
{
 // PROGRAMA PARA LER E TESTAR A IGUALDADE DE DOIS INTEIROS
 // VARIAVEIS:
    int X=0, Y=0;

    printf ( "EXEMPLO305 - LER E TESTAR DOIS VALORES INTEIROS" );
    printf ( "\nFORNECER UM VALOR INTEIRO QUALQUER: " );
    scanf ( "%d", &X );
    getchar( );                 // limpar a entrada de dados
    printf ( "\nFORNECER OUTRO VALOR INTEIRO QUALQUER: " );
    scanf ( "%d", &Y );
    getchar( );                 // limpar a entrada de dados
    if( X == Y )
      printf ( "\nDOIS VALORES IGUAIS" );
    else
    {
      printf ( "\n%d", X );
      printf ( " DIFERENTE DE " );
      printf ( "%d", Y );
    }
    printf   ( "\nPRESSIONAR <Enter> PARA TERMINAR." );
    getchar( );                 // para esperar
    return   ( EXIT_SUCCESS );
} // fim do programa
*/
```

```
/*
// ------------------------------------ EXEMPLO306
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral

int main ( )
{
 // PROGRAMA PARA LER E TESTAR DOIS VALORES REAIS
 // VARIAVEIS:
    double X=0.0, Y=0.0;

    printf ( "EXEMPLO306 - LER E TESTAR DOIS VALORES REAIS" );

    printf ( "\nFORNECER UM VALOR REAL QUALQUER: " );
    scanf ( "%lf", &X );
    getchar( );                 // limpar a entrada de dados
    printf ( "\nFORNECER OUTRO VALOR REAL QUALQUER: " );
    scanf ( "%lf", &Y );
    getchar( );                 // limpar a entrada de dados
    if( ! (X == Y) )
    {
       printf ( "\n%lf", X );
       printf ( " DIFERENTE DE " );
       printf ( "%lf", Y );
    }
    else
    {
       printf ( "VALORES IGUAIS" );
    } // if VALORES DIFERENTES
    printf    ( "\nPRESSIONAR <Enter> PARA TERMINAR." );
    getchar( );                 // para esperar
    return   ( EXIT_SUCCESS );
} // fim do programa
*/
```

```
/*
// ------------------------------------ EXEMPLO307
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral
#include <stdbool.h>        // para variaveis logicas

int main ( )
{
 // PROGRAMA PARA TRATAR ALTERNATIVAS COM VALORES LOGICOS
 // VARIAVEIS:
    int    X=0, Y=0;
    bool Z=false;

    printf ( "EXEMPLO307 - TRATAR VALORES LOGICOS" );
    printf ( "\nFORNECER UM VALOR INTEIRO QUALQUER: " );
    scanf ( "%d", &X );
    getchar( );                 // limpar a entrada de dados
    printf ( "\nFORNECER OUTRO VALOR INTEIRO QUALQUER: " );
    scanf ( "%d", &Y );
    getchar( );                 // limpar a entrada de dados
    Z = (X == Y);
    if( Z )
       printf ( "VALORES IGUAIS" );
    else
       printf ( "VALORES DIFERENTES" );
    printf    ( "\nPRESSIONAR <Enter> PARA TERMINAR." );
    getchar( );                 // para esperar
    return   ( EXIT_SUCCESS );
} // fim do programa
*/
```

```
/*
// ------------------------------------ EXEMPLO308
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral

int main ( )
{
 // PROGRAMA PARA LER E TESTAR UMA LETRA
 // VARIAVEL:
    char X = '0';

    printf ( "EXEMPLO308 - LER E TESTAR UMA LETRA" );
    printf ( "\nFORNECER UMA LETRA QUALQUER: " );
    scanf ( "%c", &X );
    getchar( );                 // limpar a entrada de dados
    if( X >= 'A' && X <= 'Z' )
      printf ( "FOI DIGITADA UMA LETRA MAIUSCULA" );
    else
      if( X >= 'a' && X <= 'z' )
        printf ( "FOI DIGITADA UMA LETRA MINUSCULA" );
      else
        printf ( "NAO FOI DIGITADA UMA LETRA" );
    printf    ( "\nPRESSIONAR <Enter> PARA TERMINAR." );
    getchar( );                 // para esperar
    return   ( EXIT_SUCCESS );
} // fim do programa
*/
```

```
/*
// ------------------------------------ EXEMPLO309
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral

int main ( )
{
 // PROGRAMA PARA COMPARAR CARACTERES < , = , >
 // VARIAVEL:
    char X = '0';

    printf ( "EXEMPLO309 - COMPARAR CARACTERES < , = , >" );
    printf ( "\nFORNECER UM DOS CARACTERES CITADOS: " );
    scanf ( "%c", &X );
    getchar( );                 // limpar a entrada de dados
    switch( X )
    {
      case '>':  printf ( "FOI DIGITADO O SINAL DE MAIOR" );
       break;
      case '=':  printf ( "FOI DIGITADO O SINAL DE IGUAL" );
       break;
      case '<':  printf ( "FOI DIGITADO O SINAL DE MENOR" );
       break;
      default :  printf ( "FOI DIGITADO UM OUTRO CARACTERE QUALQUER" );
    } // COMPARACAO DE X COM < , = , >
    printf     ( "\nPRESSIONAR <Enter> PARA TERMINAR." );
    getchar ( );                // para esperar
    return    ( EXIT_SUCCESS );
} // fim do programa
*/
```

```
/*
// ------------------------------------ EXEMPLO310
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral

int main ( )
{
 // PROGRAMA PARA IDENTIFICAR CARACTERES
 // VARIAVEL
    char X = '0';

    printf  ( "EXEMPLO310 - IDENTIFICAR CARACTERES" );
    printf  ( "\nFORNECER UM CARACTERE QUALQUER: " );
    scanf ( "%c", &X );
    getchar( );                 // limpar a entrada de dados
    switch ( X )
    {
      case 'A':
      case 'E':
      case 'I':
      case 'O':
      case 'U': printf ( "FOI DIGITADO UMA VOGAL" );
         break;

      case '0':
      case '1':
      case '2':
      case '3':
      case '4':
      case '5':
      case '6':
      case '7':
      case '8':
      case '9': printf ( "FOI DIGITADO UM ALGARISMO" );
                printf ( "\nO NUMERO CORRESPONDENTE = %d", (X-48) );
      break;

      default: printf ( "FOI DIGITADO UM OUTRO CARACTERE QUALQUER" );
    } // IDENTIFICACAO DE UM CARACTERE
    printf     ( "\nPRESSIONAR <Enter> PARA TERMINAR." );
    getchar ( );                // para esperar
    return    ( EXIT_SUCCESS );
} // fim do programa
*/
```

```
//
// OBS.: RETIRAR OS COMENTARIOS /* */
//           PARA TESTAR CADA EXEMPLO INDIVIDUALMENTE.
//

/*
// -------------------------- EXEMPLO0401
// bibliotecas de funcoes auxiliares
#include <stdio.h>            // para entradas e saídas
#include <stdlib.h>           // para outras funcoes de uso geral

int main ( )
{
 // PROGRAMA PARA LER E IMPRIMIR 03 VALORES INTEIROS
 // VARIAVEIS :
    int X                 = 0,
        CONTADOR = 0;

 // identificar
    printf ( "EXEMPLO0401 - LER E IMPRIMIR 03 VALORES INTEIROS" );
    printf ( "\n" );              // mudar de linha
    CONTADOR = 1;
    while( CONTADOR <= 3 ) // REPETIR
    {
      printf ( "\n" );            // mudar de linha
      printf ( "%d. FORNECER UM VALOR INTEIRO : ", CONTADOR );
      scanf  ( "%d", &X );
      getchar( );                 // limpar a entrada de dados
      printf ( "\nO VALOR DIGITADO FOI : %d\n", X );
      CONTADOR = CONTADOR + 1;
    }                             // ENQUANTO ( CONTADOR <= 3 )
 // encerrar
    printf ( "\n%s\n", "APERTAR <Enter> PARA TERMINAR."     );
    getchar( );                   // para esperar
    return ( EXIT_SUCCESS );
} // end main ( )
} // fim do programa
*/
```

```
/*
// -------------------------- EXEMPLO0402
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral

int main ( )
{
 // PROGRAMA PARA LER E IMPRIMIR (N) VALORES INTEIROS
 // VARIAVEIS :
    int X = 0, N = 0, CONTADOR = 0;

 // identificar
    printf ( "EXEMPLO0402 - LER E IMPRIMIR (N) VALORES INTEIROS\n" );
    printf ( "\nFORNECER O NUMERO DE VEZES (N) : " );
    scanf ( "%d", &N );
    getchar( );                 // limpar a entrada de dados
    CONTADOR = 1;
    while( CONTADOR <= N )
    {
      printf ( "\n%d", CONTADOR );
      printf ( " FORNECER UM VALOR INTEIRO QUALQUER : " );
      scanf  ( "%d", &X );
      getchar( );               // limpar a entrada de dados
      printf ( "\nO VALOR DIGITADO FOI : %d\n", X );
      CONTADOR = CONTADOR + 1;
    } // ENQUANTO ( CONTADOR <= N )
 // encerrar
    printf ( "\n%s\n", "APERTAR <Enter> PARA TERMINAR."     );
    getchar( );                 // para esperar
    return ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO0403
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral

int main ( )
{
 // PROGRAMA PARA LER E IMPRIMIR (N) VALORES INTEIROS
 // VARIAVEIS :
    int X = 0, N = 0;

 // identificar
    printf ( "EXEMPLO0403 - LER E IMPRIMIR (N) VALORES INTEIROS\n" );
    printf ( "\nFORNECER O NUMERO DE VEZES (N) : " );
    scanf  ( "%d", &N );
    getchar( );                 // limpar a entrada de dados
    while( N > 0 )              // REPETIR
    {
      printf ( "\n%d", N );
      printf ( " FORNECER UM VALOR INTEIRO QUALQUER : " );
      scanf  ( "%d", &X );
      getchar( );               // limpar a entrada de dados
      printf ( "\nO VALOR DIGITADO FOI : %d\n", X );
      N = N - 1;
    }                           // ENQUANTO N > 0
 // encerrar
    printf ( "\n%s\n", "APERTAR <Enter> PARA TERMINAR."     );
    getchar( );                 // para esperar
    return ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO0404
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral

int main ( )
{
 // PROGRAMA PARA LER E IMPRIMIR 03 VALORES INTEIROS
 // VARIAVEIS :
    int X                = 0,
        CONTADOR = 0;

 // identificar
    printf ( "EXEMPLO0404 - LER E IMPRIMIR 03 VALORES INTEIROS\n" );
    for( CONTADOR = 1; CONTADOR <= 3; CONTADOR = CONTADOR+1)
    {
      printf ( "\n%d. FORNECER UM VALOR INTEIRO : ", CONTADOR );
      scanf ( "%d", &X );
      getchar( );               // limpar a entrada de dados
      printf ( "\nO VALOR DIGITADO FOI : %d\n", X );
    } // PARA CONTADOR EM [1:3]
 // encerrar
    printf ( "\n%s\n", "APERTAR <Enter> PARA TERMINAR."     );
    getchar( );                 // para esperar
    return ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO0405
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral

int main ( )
{
 // PROGRAMA PARA LER E IMPRIMIR (N) VALORES INTEIROS
 // VARIAVEIS :
    int X                = 0,
        N                = 0,
        CONTADOR = 0;

 // identificar
    printf ( "EXEMPLO0405 - LER E IMPRIMIR (N) VALORES INTEIROS\n" );
    printf ( "\nFORNECER O NUMERO DE VEZES (N) : " );
    scanf ( "%d", &N );
    getchar( );                 // limpar a entrada de dados
    for( CONTADOR = 1; CONTADOR <= N; CONTADOR++ )
    {
      printf ( "\n%d. FORNECER UM VALOR INTEIRO : ", CONTADOR );
      scanf  ( "%d", &X );
      getchar( );               // limpar a entrada de dados
      printf ( "\nO VALOR DIGITADO FOI : %d", X );
    } // PARA CONTADOR EM [1:N]
 // encerrar
    printf ( "\n%s\n", "APERTAR <Enter> PARA TERMINAR."     );
    getchar( );                 // para esperar
    return ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO0406
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral

int main ( )
{
 // PROGRAMA PARA LER E IMPRIMIR 03 VALORES INTEIROS
 // VARIAVEIS :
    int X                = 0,
        CONTADOR = 0;

 // identificar
    printf ( "EXEMPLO0406 - LER E IMPRIMIR 03 VALORES INTEIROS\n" );
    CONTADOR = 1;
    do                          // REPETIR
    {
      printf ( "\n%d. FORNECER UM VALOR INTEIRO : ", CONTADOR );
      scanf  ( "%d", &X );
      getchar( );               // limpar a entrada de dados
      printf ( "\nO VALOR DIGITADO FOI : %d\n", X );
      CONTADOR = CONTADOR + 1;
    }
    while( CONTADOR <= 3 ); // ATE' ( CONTADOR > 3 )
 // encerrar
    printf ( "\n%s\n", "APERTAR <Enter> PARA TERMINAR."     );
    getchar( );                 // para esperar
    return ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO0407
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral

int main ( )
{
 // PROGRAMA PARA LER E IMPRIMIR (N) VALORES INTEIROS
 // VARIAVEIS :
    int X                = 0,
        CONTADOR = 0;

 // identificar
    printf ( "EXEMPLO0407 - LER E IMPRIMIR (N) VALORES INTEIROS\n" );
    printf ( "\nFORNECER O NUMERO DE VEZES (N) : " );
    scanf ( "%d", &CONTADOR );
    getchar( );                 // limpar a entrada de dados
    do                          // REPETIR
    {
      printf ( "\n%d. FORNECER UM VALOR INTEIRO : ", CONTADOR );
      scanf  ( "%d", &X );
      printf ( "\nO VALOR DIGITADO FOI : %d\n", X );
      CONTADOR = CONTADOR - 1;
    }
    while( CONTADOR > 0 ); // ATE' ( CONTADOR <= 3 )
 // encerrar
    printf ( "\n%s\n", "APERTAR <Enter> PARA TERMINAR."     );
    getchar( );                 // para esperar
    return ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO0408
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral

int main ( )
{
 // PROGRAMA PARA LER E IMPRIMIR INTEIROS DIFERENTES DE ZERO
 // VARIAVEL :
    int X = 0;

 // identificar
    printf ( "EXEMPLO0408 - LER E IMPRIMIR INTEIROS NAO NULOS\n" );
    printf ( "\nFORNECER UM VALOR INTEIRO (0 = PARAR) : " );
    scanf ( "%d", &X );
    getchar( );                 // limpar a entrada de dados
    while ( X != 0 )            // REPETIR
    {
      printf ( "\nO VALOR DIGITADO FOI : %d\n", X );
      printf ( "\nDIGITE UM VALOR INTEIRO QUALQUER : " );
      scanf ( "%d", &X );
    }                           // ENQUANTO X DIFERENTE DE ZERO
 // encerrar
    printf ( "\n%s\n", "APERTAR <Enter> PARA TERMINAR."     );
    getchar( );                 // para esperar
    return ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO0409
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral

int main ( )
{
// PROGRAMA PARA LER UM INTEIRO DIFERENTE DE ZERO
// VARIAVEL :
   int X = 0;

 // identificar
   printf ( "EXEMPLO0409 - PARA LER UM INTEIRO NAO NULO\n" );
   printf ( "\nFORNECER UM VALOR DIFERENTE DE ZERO : " );
   scanf ( "%d", &X );
   getchar( );                  // limpar a entrada de dados
   while( X == 0 )              // REPETIR
   {
      printf ( "\nFORNECER UM VALOR DIFERENTE DE ZERO : " );
      scanf  ( "%d", &X );
      getchar( );               // limpar a entrada de dados
   }                            // ENQUANTO X IGUAL A ZERO
   printf ( "\nDIGITADO UM NUMERO DIFERENTE DE ZERO\n" );
 // encerrar
    printf ( "\n%s\n", "APERTAR <Enter> PARA TERMINAR."     );
    getchar( );                 // para esperar
    return ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO0410
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral

int main ( )
{
// PROGRAMA PARA LER UM INTEIRO DIFERENTE DE ZERO
// VARIAVEL :
   int X = 0;

 // identificar
   printf ( "EXEMPLO0410 - LER UM INTEIRO NAO NULO\n" );
   do                           // REPETIR
   {
     printf ( "\nFORNECER UM VALOR DIFERENTE DE ZERO : " );
     scanf ( "%d", &X );
     getchar( );                // limpar a entrada de dados
   }
   while( X == 0 );             // ATE' X DIFERENTE DE ZERO
   printf ( "\nDIGITADO UM NUMERO DIFERENTE DE ZERO\n" );
 // encerrar
    printf ( "\n%s\n", "APERTAR <Enter> PARA TERMINAR."     );
    getchar( );                 // para esperar
    return ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
//
// OBS.: RETIRAR OS COMENTARIOS /* */
//           PARA TESTAR CADA EXEMPLO INDIVIDUALMENTE.
//

/*
// -------------------------- EXEMPLO0501
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral

void P1 ( void )
{
   printf ( "\n" );
   printf ( "\nCHAMADO O PROCEDIMENTO P1 SEM PARAMETROS" );
   printf ( "\n" );
} // fim procedimento P1( )

int main ( void )
{
 // PROGRAMA PARA CHAMADA DE PROCEDIMENTO SEM PARAMETROS

 // identificar
    printf  ( "EXEMPLO0501 - CHAMADA A UM PROCEDIMENTO" );
    P1 ( );                     // chamada ao procedimento
    printf  ( "\n" );
 // encerrar
    printf    ( "\nAPERTAR <Enter> PARA TERMINAR." );
    getchar( );                 // para esperar
    return   ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO0502
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral

int X = 0;                  // VARIAVEL EM CONTEXTO GLOBAL

void P1 ( void )
{
 printf ( "\n" );
 printf ( "\nCHAMADO O PROCEDIMENTO P1 %d VEZ(ES)", X );
 printf ( "\n" );
} // fim procedimento P1( )

int main ( void )
{
 // PROGRAMA PARA MOSTRAR PASSAGENS DE PARAMETROS

 // identificar
    printf ( "EXEMPLO0502 - CHAMADA COM VARIAVEL GLOBAL\n" );
    for ( X = 1; X <= 5; X = X + 1 )
        P1 ( );             // chamar 5 vezes
    printf  ( "\n" );
 // encerrar
    printf    ( "\nAPERTAR <Enter> PARA TERMINAR." );
    getchar( );             // para esperar
    return   ( EXIT_SUCCESS );
} // end main ( )
*/
/*
// -------------------------- EXEMPLO0503
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral

int X = 0;                  // VARIAVEL EM CONTEXTO GLOBAL

void P1 ( void )
{
   X = X + 1;               // AVANCAR O CONTEXTO

   printf ( "\nCHAMADO O PROCEDIMENTO P1 %d VEZ(ES)", X );
   printf ( "\n" );

   if ( X < 5 )
     P1 ( );                // CHAMAR O PROCEDIMENTO RECURSIVAMENTE

   printf ( "\nRETORNANDO AO PROCEDIMENTO P1 PARA A CHAMADA %d", X );
   X = X - 1;               // RETORNAR AO CONTEXTO ANTERIOR
   getchar ( );             // para esperar
} // fim procedimento P1( )
```

```
int main ( void )
{
 // PROGRAMA PARA MOSTRAR PASSAGENS DE PARAMETROS

 // identificar
    printf ( "EXEMPLO0503 - CHAMADA/RETORNO COM VARIAVEL GLOBAL\n" );
    X = 0;
    P1 ( );                    // OBSERVAR A RECURSIVIDADE !
    printf  ( "\n" );
 // encerrar
    printf    ( "\nAPERTAR <Enter> PARA TERMINAR." );
    getchar( );                // para esperar
    return   ( EXIT_SUCCESS );
} // end main ( )
*/

/*
// -------------------------- EXEMPLO0504
// bibliotecas de funcoes auxiliares
#include <stdio.h>         // para entradas e saídas
#include <stdlib.h>        // para outras funcoes de uso geral

void P1 ( int X )
{
   printf ( "\nCHAMADO O PROCEDIMENTO P1 %d VEZ(ES)\n", X );
   if ( X < 5 )
     P1( X + 1 );          // chamar recursivamente com parametro
   printf  ( "\nRETORNANDO AO PROCEDIMENTO P1 PARA A CHAMADA %d", X );
   getchar ( );            // para esperar
} // fim procedimento P1( )

int main ( void )
{
 // PROGRAMA PARA MOSTRAR PASSAGENS DE PARAMETROS

 // identificar
    printf ( "EXEMPLO0904 - CHAMADA/RETORNO COM PARAMETRO\n" );
    P1 ( 1 );                  // OBSERVAR REPETICAO FINITA, SEM VARIAVEL GLOBAL !
    printf  ( "\n" );
 // encerrar
    printf    ( "\nAPERTAR <Enter> PARA TERMINAR." );
    getchar( );                // para esperar
    return   ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO0505
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral

void P1 ( int X )
{
   printf ( "\nCHAMADO O PROCEDIMENTO P1 %d VEZ(ES)\n", X );
   if ( X > 1 )
     P1 ( X - 1 );
   printf  ( "\nRETORNANDO AO PROCEDIMENTO P1 PARA A CHAMADA %d", X );
   getchar ( );                 // para esperar
} // fim procedimento P1( )

int main ( void )
{
 // PROGRAMA PARA MOSTRAR PASSAGEM DE PARAMETRO POR VALOR

 // identificar
    printf ( "EXEMPLO0505 - CHAMADA/RETORNO COM PARAMETRO\n" );
    P1 ( 5 );                   // OBSERVAR REPETICAO FINITA, SEM VARIAVEL GLOBAL !
    printf  ( "\n" );
 // encerrar
    printf    ( "\nAPERTAR <Enter> PARA TERMINAR." );
    getchar( );                 // para esperar
    return   ( EXIT_SUCCESS );
} // end main ( )
*/

/*
// -------------------------- EXEMPLO0506
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral

void P2 (int X);            // PROTOTIPO DE PROCEDIMENTO

void P1 (int X)
{
   printf ( "CHAMADO O PROCEDIMENTO P1 COM X = %d\n", X );
   if ( X < 5 )
      P2 ( X );
   printf    ( "RETORNANDO AO PROCEDIMENTO P1 PARA A CHAMADA COM X = %d\n", X );
   getchar( );                  // para esperar
} // fim do procedimento P1( )

void P2 (int X)
{
   printf ( "CHAMADO O PROCEDIMENTO P2 COM X = %d\n", X );
   X = X+1;
   printf     ( "RETORNANDO AO PROCEDIMENTO P2 PARA A CHAMADA COM X = %d\n", X );
   getchar ( );                 // para esperar
   P1 ( X );
} // fim do procedimento P2( )
```

```
int main ( void )
{
 // PROGRAMA PARA MOSTRAR PASSAGENS DE PARAMETROS

 // identificar
    printf ( "EXEMPLO0506 - CHAMADA/RETORNO COM PARAMETRO\n\n" );
    P1 ( 1 ); // OBSERVAR RECURSIVIDADE INDIRETA !
    printf  ( "\n" );
 // encerrar
    printf    ( "\nAPERTAR <Enter> PARA TERMINAR." );
    getchar( );                 // para esperar
    return   ( EXIT_SUCCESS );
} // end main ( )
*/
/*
// -------------------------- EXEMPLO0507
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral

void P1 ( int* X )
{
   *X = *X + 1;             // AVANCAR O CONTEXTO NA REFERENCIA

   printf ( "\nCHAMADO O PROCEDIMENTO P1 %d VEZ(ES)\n", *X );
   if ( *X < 5 )
     P1 ( X );

   printf  ( "\nRETORNANDO AO PROCEDIMENTO P1 PARA A CHAMADA %d", *X );
   *X = *X - 1;             // RETORNAR AO CONTEXTO DA REFERENCIA ANTERIOR
   getchar ( );             // para esperar
} // fim procedimento P1( )

int main ( void )
{
 // PROGRAMA PARA MOSTRAR PASSAGEM DE PARAMETRO POR REFERENCIA
 // VARIAVEL LOCAL
    int X;

 // identificar
    printf ( "EXEMPLO0507 - CHAMADA/RETORNO COM REFERENCIA\n" );
    X = 0;
    P1 ( &X );              // OBSERVAR REPETICAO FINITA !
    printf  ( "\n" );
 // encerrar
    printf     ( "\nAPERTAR <Enter> PARA TERMINAR." );
    getchar ( );            // para esperar
    return    ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO0508
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral

void P2 ( int X );          // PROTOTIPO DE PROCEDIMENTO

void P1 ( int X )
{
   X = X + 1;
   printf ( "CHAMADO O PROCEDIMENTO P1 COM X = %d\n", X );

   if ( X < 4 )
   {
     P1 ( X );
     P2 ( X );
   }

   printf  ( "\nRETORNANDO AO PROCEDIMENTO P1 PARA A CHAMADA COM X = %d\n", X );
   getchar ( ) ;               // para esperar
} // fim do procedimento P1( )

void P2 ( int X )
{
   printf ( "CHAMADO O PROCEDIMENTO P2 COM X = %d\n", X );
   if ( X > 1 )
     P2 ( X - 1 );
   printf  ( "RETORNANDO AO PROCEDIMENTO P2 PARA A CHAMADA COM X = %d", X );
   getchar ( );                // para esperar
} // fim procedimento P2( )

int main ( void )
{
 // PROGRAMA PARA MOSTRAR PASSAGENS DE PARAMETROS

 // identificar
    printf ( "EXEMPLO0508 - MULTIPLAS CHAMADAS/RETORNOS\n\n" );
    P1 ( 1 );
    printf ( "\n" );
 // encerrar
    printf    ( "\nAPERTAR <Enter> PARA TERMINAR." );
    getchar( );                 // para esperar
    return   ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*

// Digitar o conteudo abaixo em um arquivo com o nome io.h :

// ------------------------------------ my_lib.h

// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral

 void pause ( char message [ ] )
 {
    printf  ( "\n%s", message );
    getchar ( );                // para esperar
 } // fim pause ( )

 void println ( char text [ ] )
 { printf ( "%s\n", text ); }
*/

// Digitar o conteudo abaixo em outro arquivo:

/*
// -------------------------- EXEMPLO0509
// bibliotecas de funcoes auxiliares
#include "my_lib.h"         // para funcoes proprias

int main ( void )
{
 // PROGRAMA PARA MOSTRAR O USO DE MODULOS

 // identificar
    println ( "EXEMPLO0509 - USO DE MODULOS" );

 // encerrar
    pause  ( "APERTAR <Enter> PARA TERMINAR." );
    return  ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
// Digitar o conteudo abaixo em um arquivo com o nome my_string.h :

/*

// DEFINICOES GLOBAIS

#define EOL '\n'
#define EOS '\0'

// CONSTANTES GLOBAIS

   const int STR_SIZE = 80;        // quantidade maxima de caracteres

// TIPOS GLOBAIS

   typedef char* chars;            // tipo similar 'a cadeia de caracteres

*/

// Digitar o conteudo abaixo em outro arquivo:

/*
// -------------------------- EXEMPLO0510
// bibliotecas de funcoes auxiliares
#include "my_def.h"             // para definicoes globais, constantes ...
#include "my_lib.h"

int main ( void )
{
 // PROGRAMA PARA MOSTRAR O USO DE MODULOS
 // VARIAVEIS :
    chars text = "MUDAR DE LINHA";

 // identificar
    println ( "EXEMPLO0510 - USO DE MODULOS" );

    printf  ( "%c"     , EOL );
    printf  ( "%s%c", text, EOL );

//  encerrar
    pause ( "PRESSIONAR <Enter> PARA TERMINAR." );
    return ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
//
// OBS.: RETIRAR OS COMENTARIOS /* */
//           PARA TESTAR CADA EXEMPLO INDIVIDUALMENTE.
//

/*
// -------------------------- EXEMPLO0601
// bibliotecas de funcoes auxiliares
#include <stdio.h>         // para entradas e saídas
#include <stdlib.h>        // para outras funcoes de uso geral

void CONTAR ( int X )
{
   if ( X > 0 )
   {
     CONTAR ( X-1 );
     printf ( "\n%d\n", X );
   }
} // fim procedimento CONTAR( )

int main ( void )
{
 // PROGRAMA PARA CONTAR DE 1 ATE' 5, RECURSIVAMENTE

 // identificar
    printf  ( "EXEMPLO0601 - CONTAR DE 1 A 5 RECURSIVAMENTE\n" );
    CONTAR  ( 5 );
 // encerrar
    printf    ( "\nAPERTAR <Enter> PARA TERMINAR" );
    getchar( );                // para esperar
    return   ( EXIT_SUCCESS );
} // end main ( )
*/
/*
// -------------------------- EXEMPLO0602
// bibliotecas de funcoes auxiliares
#include <stdio.h>         // para entradas e saídas
#include <stdlib.h>        // para outras funcoes de uso geral

void CONTAR ( int X )
{
   if ( X > 0 )
   {
     printf ( "\n%d\n", X );
     CONTAR ( X-1 );
   }
} // fim procedimento CONTAR( )
```

```
int main ( void )
{
 // PROGRAMA PARA CONTAR 5 10 ATE' 1, RECURSIVAMENTE

 // identificar
    printf  ( "EXEMPLO0602 - CONTAR DE 5 A 1 RECURSIVAMENTE\n" );
    CONTAR  ( 5 );
 // encerrar
    printf    ( "\nAPERTAR <Enter> PARA TERMINAR" );
    getchar( );                 // para esperar
    return   ( EXIT_SUCCESS );
} // end main ( )
*/
/*
// -------------------------- EXEMPLO0603
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral

void PARES ( int X )
{
   if ( X > 0 )
     if ( X % 2 == 0 )
     {
       PARES ( X-2 );
        printf ( "\n%d\n", X );
     }
     else
       PARES ( X-1 );
} // fim procedimento PARES ( )

int main ( void )
{
 // PROGRAMA RECURSIVO PARA MOSTRAR PARES

 // identificar
    printf  ( "EXEMPLO0603 - MOSTRAR OS PARES <= 10\n" );
    PARES   ( 10 );
 // encerrar
    printf    ( "\nAPERTAR <Enter> PARA TERMINAR" );
    getchar( );                 // para esperar
    return   ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO0604
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral

void PARES ( int X )
{
   if ( X > 0 )
   {
     PARES ( X-1 );
     printf ( "\n%d%c%d\n", X, " ", 2*X );
   }
} // fim procedimento PARES ( )

int main ( void )
{
 // PROGRAMA RECURSIVO PARA MOSTRAR PARES

 // identificar
    printf  ( "EXEMPLO0604 - MOSTRAR OS 5 PRIMEIROS PARES\n" );
    PARES   ( 5 );
 // encerrar
    printf    ( "\nAPERTAR <Enter> PARA TERMINAR" );
    getchar( );                 // para esperar
    return   ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO0605
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral

void PARES ( int X, int* S )
{
   if ( X > 0 )
   {   PARES ( X-1, S );   *S = *S + 2*X;  }
   else
       S = 0;
} // fim procedimento PARES ( )

int main ( void )
{
 // PROGRAMA RECURSIVO PARA SOMAR PARES
 // DADO:
    int SOMA;

 // identificar
    printf  ( "EXEMPLO0605 - SOMAR OS 5 PRIMEIROS PARES\n" );
    PARES   ( 5, &SOMA );
    printf  ( "\nSOMA DOS 5 PRIMEIROS PARES = %d\n", SOMA );
 // encerrar
    printf    ( "\nAPERTAR <Enter> PARA TERMINAR" );
    getchar( );                 // para esperar
    return   ( EXIT_SUCCESS );
} // end main ( )
*/
/*
// -------------------------- EXEMPLO0606
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral

int PARES ( int X )
{
 // DADO:
    int S = 0;
    if ( X > 0 )
      S = 2*X + PARES( X-1 );
    else
      S = 0;
    return ( S );
} // fim funcao PARES ( )
```

```
int main ( void )
{
 // PROGRAMA RECURSIVO PARA SOMAR PARES
 // DADO :
    int SOMA = 0;

 // identificar
    printf  ( "EXEMPLO0606 - SOMAR OS 5 PRIMEIROS PARES\n" );
    SOMA  =  PARES ( 5 );
    printf  ( "\nSOMA DOS 5 PRIMEIROS PARES = %d\n", SOMA );
 // encerrar
    printf    ( "\nAPERTAR <Enter> PARA TERMINAR" );
    getchar( );                 // para esperar
    return   ( EXIT_SUCCESS );
} // end main ( )
*/

/*
// -------------------------- EXEMPLO0607
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral

int PARES ( int X )
{
 // DADO :
    int S = 0;

    if ( X > 0 )
      if ( X % 2 == 0 )
        S = 1 + PARES ( X-2 );
      else
        S = PARES ( X-1 );
    else
      S = 0;

    return ( S );
} // fim funcao PARES ( )

int main ( void )
{
 // PROGRAMA RECURSIVO PARA CONTAR PARES

 // identificar
    printf  ( "EXEMPLO0607 - CONTAR OS PARES <= 10\n" );
    printf  ( "\nPARES <= 10 = %d\n", PARES( 10 ) );
 // encerrar
    printf    ( "\nAPERTAR <Enter> PARA TERMINAR" );
    getchar( );                 // para esperar
    return   ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO0608
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral
#include <stdbool.h>    // para definicoes logicas
#include <string.h>         // para strlen()

typedef   char  STRING30 [30];

bool PROCURAR
    ( char LETRA, STRING30 S, int POSICAO )
{
 // DADO :
   bool R = false;

   if ( POSICAO <= strlen( S ) )
     R = ( S [ POSICAO ] == LETRA) || PROCURAR ( LETRA,S,POSICAO+1 );
   else
     R = false;

   return ( R );
} // fim funcao PROCURAR ( )

int main ( void )
{
 // PROGRAMA RECURSIVO PARA ACHAR A POSICAO DE UMA LETRA
 // DADO :
   char           L = '_';
   STRING30 S = "";

 // identificar
   printf  ( "EXEMPLO0608 - PROCURAR UMA LETRA EM UMA SENTENCA\n" );
   printf  ( "\nFORNECER UMA SENTENCA COM MENOS DE 30 CARACTERES : " );
   gets    ( S );
   printf  ( "\nFORNECER UMA LETRA PARA SER PROCURADA : " );
   L = getchar ( );
   getchar ( );     // para limpar a entrada de dados
   printf  ( "\nRESPOSTA = %d\n", PROCURAR( L,S,0 ) );
 // encerrar
   printf    ( "\nAPERTAR <Enter> PARA TERMINAR" );
   getchar( );                 // para esperar
   return   ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO0609
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral
#include <string.h>         // para strlen()

typedef   char  STRING30[30];

int PROCURAR
  ( char LETRA, STRING30 S, int POSICAO )
{
 // DADO :
    int R = 0;

    if ( POSICAO <= strlen(S) )
      if ( S [ POSICAO ] == LETRA )
        R = POSICAO + 1; // a primeira posicao e' 0 !
      else
        R = PROCURAR ( LETRA,S,POSICAO+1 );
    else
      R = 0;

    return ( R );
} // fim funcao PROCURAR ( )

int main ( void )
{
 // PROGRAMA RECURSIVO PARA PROCURAR UMA LETRA
 // DADO :
    char     L = '_';
    STRING30 S = "" ;

 // identificar
    printf  ( "EXEMPLO0609 - POSICAO DE UMA LETRA EM UMA SENTENCA\n" );
    printf  ( "\nFORNECER UMA SENTENCA COM MENOS DE 30 CARACTERES : " );
    gets    ( S );
    printf  ( "\nFORNECER UMA LETRA PARA SER PROCURADA : " );
    L = getchar ( );
    getchar ( );                // para limpar a entrada de dados
    printf  ( "\nRESPOSTA = %d\n", PROCURAR ( L,S,0 ) );
 // encerrar
    printf    ( "\nAPERTAR <Enter> PARA TERMINAR" );
    getchar( );                 // para esperar
    return   ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO0610
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral
#include <string.h>         // para strlen()

typedef   char  STRING30[30];

int PROCURAR
  ( char LETRA, STRING30 S, int POSICAO )
{
 // DADO :
    int R = 0;

    if ( POSICAO <= strlen(S) )
      if ( S [ POSICAO ] == LETRA )
        R = 1 + PROCURAR ( LETRA, S, POSICAO+1 );
      else
        R = PROCURAR ( LETRA,S,POSICAO+1 );
    else
      R = 0;

    return ( R );
} // fim funcao PROCURAR ( )

int main ( void )
{
 // PROGRAMA RECURSIVO PARA PROCURAR OCORRENCIAS DE UMA LETRA
 // DADO :
    char           L = '_';
    STRING30 S = "";

    printf  ( "EXEMPLO0610 - PROCURAR OCORRENCIAS DE UMA LETRA\n" );
    printf  ( "\nFORNECER UMA SENTENCA COM MENOS DE 30 CARACTERES : " );
    gets    ( S );
    printf  ( "\nFORNECER UMA LETRA PARA SER PROCURADA : " );
    L = getchar ( );
    getchar ( );                // para limpar a entrada de dados
    printf  ( "\nRESPOSTA = %d\n", PROCURAR( L,S,0 ) );
 // encerrar
    printf    ( "\nAPERTAR <Enter> PARA TERMINAR" );
    getchar( );                 // para esperar
    return   ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
//
// OBS.: RETIRAR OS COMENTARIOS /* */
//           PARA TESTAR CADA EXEMPLO INDIVIDUALMENTE.
//

/*
// -------------------------- EXEMPLO0701
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saidas
#include <stdlib.h>         // para outras funcoes de uso geral

typedef  int  TABELA [10];

int main ( void )
{
 // PROGRAMA PARA LER UMA TABELA DE INTEIROS

 // VARIAVEIS:
    TABELA V;
    int          X = 0;

 // identificar
    printf ( "EXEMPLO0701 - LER UM TABELA DE 10 INTEIROS\n" );
 // REPETIR PARA CADA POSICAO
    for ( X = 0; X < 10; X++ )
    { // a primeira posicao e' zero !
        printf ( "\nFORNECER O %d o. INTEIRO : ",(X+1) );
        scanf ( "%d", &V[ X ] );
        getchar ( );            // para limpar a entrada de dados
    } // FIM REPETIR
    printf ( "\nVETOR LIDO: \n" );
 // REPETIR PARA CADA POSICAO
    for ( X = 0; X < 10; X++ )
    {
      printf ( "%d ", V[ X ] );
    } // FIM REPETIR
 // encerrar
    printf    ( "\n\nAPERTAR <Enter> PARA TERMINAR" );
    getchar( );                 // para esperar
    return   ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO0702
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral

typedef  int  TABELA[10];

int main ( void )
{
 // PROGRAMA PARA SOMAR UMA TABELA DE INTEIROS
 // VARIAVEIS :
    TABELA V;
    int          X        = 0,
                 SOMA = 0;

 // identificar
    printf ( "EXEMPLO0702 - SOMAR UM TABELA DE 10 INTEIROS\n" );

 // REPETIR PARA CADA POSICAO
    for ( X = 0; X < 10; X++ )
    {
       printf ( "\nFORNECER O %do. INTEIRO : ",(X+1) );
       scanf ( "%d", &V[ X ] );
       getchar ( );              // para limpar a entrada de dados
    } // FIM REPETIR
    SOMA = 0;
 // REPETIR PARA CADA POSICAO
    for ( X = 0; X < 10; X++ )
      SOMA = SOMA + V[ X ];
    printf  ( "\nSOMA = %d",SOMA );
 // encerrar
    printf    ( "\n\nAPERTAR <Enter> PARA TERMINAR" );
    getchar( );                  // para esperar
    return   ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO0703
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral

typedef  int  TABELA[10];

int main ( void )
{
 // PROGRAMA PARA CALCULAR A MEDIA DE UMA TABELA DE INTEIROS
 // VARIAVEIS :
    TABELA V;
    float  MEDIA = 0.0;
    int     X         =  0 ,
            SOMA  =  0 ;

 // identificar
    printf ( "EXEMPLO0703 - MEDIA DE UMA TABELA DE 10 INTEIROS\n" );
 // REPETIR PARA CADA POSICAO
    for ( X = 0; X < 10; X++ )
    {
       printf ( "\nFORNECER O %do. INTEIRO : ",(X+1) );
       scanf ( "%d", &V[ X ] );
       getchar ( );             // para limpar a entrada de dados
    }                           // FIM REPETIR
    SOMA = 0;
 // REPETIR PARA CADA POSICAO
    for ( X = 0; X < 10; X++ )
       SOMA = SOMA + V[ X ];
    MEDIA = SOMA / 10.0;
    printf  ( "\nMEDIA = %f\n", MEDIA );
 // encerrar
    printf    ( "\n\nAPERTAR <Enter> PARA TERMINAR" );
    getchar( );                 // para esperar
    return   ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO0704
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral
#include <string.h>         // para strlen()

typedef char STRING10[10];

int main ( void )
{
 // PROGRAMA PARA LER UMA PALAVRA
 // VARIAVEIS :
    STRING10 PALAVRA;
    int              X = 0;

 // identificar
    printf ( "EXEMPLO0704 - LER UMA PALAVRA\n" );
    printf ( "\nFORNECER UMA PALAVRA (NO MAXIMO 09 LETRAS) : " );
    gets  ( PALAVRA );
    printf ( "\nLETRAS DA PALAVRA LIDA : " );
 // REPETIR PARA CADA POSICAO
    for ( X = 0; X < strlen( PALAVRA ); X++ )
      printf ( "%c ", PALAVRA[ X ] );
 // encerrar
    printf    ( "\n\nAPERTAR <Enter> PARA TERMINAR" );
    getchar( );                    // para esperar
    return   ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO0705
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral
#include <stdbool.h>        // para definicoes logicas
#include <string.h>         // para strlen()

typedef char STRING10[10];

int main ( void )
{
 // PROGRAMA PARA PROCURAR LETRA EM PALAVRA
 // VARIAVEIS :
    STRING10 PALAVRA;
    char          LETRA  = '_';
    int           X          =  0 ;
    bool          ACHAR = false;

 // identificar
    printf ( "EXEMPLO0705 - PROCURAR LETRA EM UMA PALAVRA\n" );

    printf ( "\nDIGITAR UMA PALAVRA (NO MAXIMO 09 LETRAS) : " );
    gets   ( PALAVRA );

    printf ( "\nFORNECER A LETRA A SER PROCURADA : " );
    LETRA = getchar ( );
    getchar ( );                // para limpar a entrada de dados

    ACHAR =  false;
    X          =  0;
 // REPETIR PARA CADA POSICAO
    while ( X < strlen( PALAVRA ) && ! ACHAR )
    {
      if( PALAVRA[ X ] == LETRA )
        ACHAR = true;
      else
        X = X + 1;
    } // FIM REPETIR
    if ( ACHAR )
      printf ( "LETRA ENCONTRADA" );
    else
     printf ( "LETRA NAO ENCONTRADA" );
 // encerrar
    printf    ( "\n\nAPERTAR <Enter> PARA TERMINAR" );
    getchar( );                 // para esperar
    return   ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO0706
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral

typedef   double POLINOMIO[10+1];    // posicoes de 0:10

int main ( void )
{
 // PROGRAMA PARA AVALIAR UM POLINOMIO
 // VARIAVEIS :
    POLINOMIO P;
    int             Y =  0 ,  N   =  0 ;
    double          X = 0.0, PX = 0.0;

 // identificar
    printf ( "EXEMPLO0706 - LER COEFICIENTES DE UM POLINOMIO\n" );
    printf ( "\nFORNECER O GRAU DO POLINOMIO : " );
    scanf ( "%d", &N );
    getchar ( );                // para limpar a entrada de dados
 // REPETIR PARA CADA POSICAO
    for ( Y = 0; Y <= N; Y++ )
    {
      printf ( "\nFORNECER O %do. COEFICIENTE : ", (Y+1) );
      scanf ( "%lf", &P[ Y ] );
      getchar ( );              // para limpar a entrada de dados
    } // FIM REPETIR
    printf ( "\nFORNECER O PONTO DE AVALIACAO : " );
    scanf  ( "%lf", &X );
    getchar ( );                // para limpar a entrada de dados
    PX = 0.0;
 // REPETIR PARA CADA POSICAO
 // DA ULTIMA ATE' A PRIMEIRA
    for ( Y = N; Y >= 0; Y-- )
      PX = PX * X + P[ Y ];
    printf  ( "\nP(%lf) = %lf", X, PX );
 // encerrar
    printf    ( "\n\nAPERTAR <Enter> PARA TERMINAR" );
    getchar( );                 // para esperar
    return   ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO0707
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral
#include <math.h>

typedef   int  VETOR[3]; // X,Y,Z

int main ( void )
{
 // PROGRAMA PARA AVALIAR O COMPRIMENTO DE UM VETOR
 // VARIAVEIS :
    VETOR V;
    int         X        =  0 ;
    double   SOMA = 0.0;

 // identificar
    printf ( "EXEMPLO0707 - COMPRIMENTO DE UM VETOR\n" );
    printf ( "\nFORNECER O VALOR DE X : " );
    scanf ( "%d", &V[0] );
    getchar ( );                // para limpar a entrada de dados
    printf ( "\nFORNECER O VALOR DE Y : " );
    scanf ( "%d", &V[1] );
    getchar ( );                // para limpar a entrada de dados
    printf ( "\nFORNECER O VALOR DE Z : " );
    scanf ( "%d", &V[2] );
    getchar ( );                // para limpar a entrada de dados
    SOMA = 0.0;
 // REPETIR PARA CADA POSICAO
    for( X = 0; X < 3; X++ )
      SOMA = SOMA + V[ X ]*V[ X ];
    printf  ( "\nCOMPRIMENTO = %lf", sqrt( SOMA ) );
 // encerrar
    printf    ( "\n\nAPERTAR <Enter> PARA TERMINAR" );
    getchar( );                 // para esperar
    return   ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO0708
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral

typedef   int MATRIZ[2][2];

int main ( void )
{
 // PROGRAMA PARA LER UMA MATRIZ
 // VARIAVEIS :
    MATRIZ M;
    int          X = 0, Y = 0;

 // identificar
    printf ( "EXEMPLO0708 - LER UMA MATRIZ INTEIRA 2x2\n" );
 // REPETIR PARA CADA LINHA
    for ( X = 0; X < 2; X++ )
    {                           //  REPETIR PARA CADA COLUNA
      for ( Y = 0; Y < 2; Y++ )
      {
        printf ( "\nFORNECER ELEMENTO %d, %d : ",(X+1), (Y+1) );
        scanf  ( "%d", &M[ X ][ Y ] );
        getchar ( );            // para limpar a entrada de dados
      } //  FIM REPETIR
    } // FIM REPETIR
    printf (  "\n" );
 // REPETIR PARA CADA LINHA
    for ( X = 0; X < 2; X++ )
    {                           //  REPETIR PARA CADA COLUNA
      for ( Y = 0; Y < 2; Y++ )
        printf ( "%d ", M[ X ][ Y ] );
      printf ( "\n" );
    } // FIM REPETIR
 // encerrar
    printf    ( "\n\nAPERTAR <Enter> PARA TERMINAR" );
    getchar( );                 // para esperar
    return   ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO0709
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral

typedef   int MATRIZ[2][2];

int main ( void )
{
 // PROGRAMA PARA MONTAR A TRANSPOSTA DE UMA MATRIZ
 // VARIAVEIS :
    MATRIZ M,               // MATRIZ ORIGINAL
           MT;              // MATRIZ TRANSPOSTA
    int    X = 0, Y = 0;

 // identificar
    printf ( "EXEMPLO0709 - TRANSPOR UMA MATRIZ INTEIRA 2x2\n" );
 // REPETIR PARA CADA LINHA
    for ( X = 0; X < 2; X++ )
    {                           //  REPETIR PARA CADA COLUNA
      for ( Y = 0; Y < 2; Y++ )
      {
         printf ( "\nFORNECER ELEMENTO %d, %d : ",(X+1), (Y+1) );
         scanf  ( "%d", &M[ X ][ Y ] );
         getchar ( );  // para limpar a entrada de dados
         MT[ Y ][ X ] = M[ X ][ Y ];
      } //  FIM REPETIR
    } // FIM REPETIR
    printf ( "\n" );
 // REPETIR PARA CADA LINHA
    for ( X = 0; X < 2; X++ )
    {                           //  REPETIR PARA CADA COLUNA
       for ( Y = 0; Y < 2; Y++ )
          printf ( "%d ", M[ X ][ Y ] );
       printf (  "\n" );
    } // FIM REPETIR
 // encerrar
    printf    ( "\n\nAPERTAR <Enter> PARA TERMINAR" );
    getchar( );             // para esperar
    return   ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO0710
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral

#define   ORDEM   3

typedef   int MATRIZ [ ORDEM ][ ORDEM ];

int main ( void )
{
 // PROGRAMA PARA MOSTRAR A DIAGONAL DE UMA MATRIZ
 // VARIAVEIS :
    MATRIZ M;
    int    X = 0, Y = 0;

 // identificar
    printf ( "EXEMPLO0710 - MOSTRAR A DIAGONAL DE UMA MATRIZ\n" );
 // REPETIR PARA CADA LINHA
    for ( X = 0; X < ORDEM; X++ )
    {                           //  REPETIR PARA CADA COLUNA
      for ( Y = 0; Y < ORDEM; Y++ )
      {
          printf ( "\nFORNECER ELEMENTO %d, %d : ",(X+1), (Y+1) );
          scanf  ( "%d", &M[ X ][ Y ] );
          getchar ( );          // para limpar a entrada de dados
      } //  FIM REPETIR
    } // FIM REPETIR
    printf ( "\n" );
 // REPETIR PARA CADA LINHA
    for ( X = 0; X < ORDEM; X++ )
    {                           //  REPETIR PARA CADA COLUNA
      for ( Y = 0; Y < ORDEM; Y++ )
      {
          if ( X == Y )         //   SE ESTIVER NA DIAGONAL
             printf ( "%d ", M[ X ][ Y ] );
      } //  FIM REPETIR
    } // FIM REPETIR
 // encerrar
    printf    ( "\n\nAPERTAR <Enter> PARA TERMINAR" );
    getchar( );                 // para esperar
    return   ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
//
// OBS.: RETIRAR OS COMENTARIOS /* */
//           PARA TESTAR CADA EXEMPLO INDIVIDUALMENTE.
//

/*
// -------------------------- EXEMPLO0801
// bibliotecas de funcoes auxiliares
#include <stdio.h>         // para entradas e saídas
#include <stdlib.h>        // para outras funcoes de uso geral
#include <math.h>          // para funcoes matematicas

typedef
  struct SPONTOS
  {
    double X,Y,Z;
  }
PONTOS;

int main ( void )
{
 // PROGRAMA PARA CALCULAR A DISTANCIA ENTRE PONTOS
 // DADOS:
    PONTOS P1, P2, P3;
    double     D = 0.0;

 // identificar
    printf  ( "EXEMPLO0801 - DISTANCIA ENTRE PONTOS\n" );
    printf  ( "\n ENTRE COM O PRIMEIRO PONTO : \n " );
    scanf ( "%lf %lf %lf", &P1.X, &P1.Y, &P1.Z );
    getchar ( );               // para limpar a entrada de dados
    printf  ( "\n ENTRE COM O SEGUNDO PONTO : \n " );
    scanf  ( "%lf %lf %lf", &P2.X, &P2.Y, &P2.Z );
    getchar ( );               // para limpar a entrada de dados
    P3.X = P2.X - P1.X;
    P3.Y = P2.Y - P1.Y;
    P3.Z = P2.Z - P1.Z;
    D = sqrt ( pow(P3.X, 2.0) +
               pow(P3.Y, 2.0) +
               pow(P3.Z, 2.0) );
    printf  ( "\n DISTANCIA = %lf", D );
 // encerrar
    printf    ( "\n\nAPERTAR <Enter> PARA TERMINAR" );
    getchar( );                // para esperar
    return   ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO0802
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral
#include <math.h>           // para funcoes matematicas

typedef struct SPONTOS { float X,Y,Z; } PONTOS;

int main ( void )
{
 // PROGRAM PARA CALCULAR A DISTANCIA ENTRE PONTOS
 // DADOS:
    PONTOS P1, P2;
    double     D = 0.0;

 // identificar
    printf  ("EXEMPLO0802 - DISTANCIA ENTRE PONTOS\n" );
    printf  ( "\n ENTRE COM O PRIMEIRO PONTO : \n " );
    scanf ( "%lf %lf %lf", &P1.X, &P1.Y, &P1.Z );
    getchar ( );                // para limpar a entrada de dados
    printf  ( "\n ENTRE COM O SEGUNDO PONTO : \n " );
    scanf ( "%lf %lf %lf", &P2.X, &P2.Y, &P2.Z );
    getchar ( );                // para limpar a entrada de dados
    D = sqrt ( pow(P2.X-P1.X, 2.0) +
               pow(P2.Y-P1.Y, 2.0) +
               pow(P2.Z-P1.Z, 2.0) );
    printf  ( "\n DISTANCIA = %lf", D );
 // encerrar
    printf    ( "\n\nAPERTAR <Enter> PARA TERMINAR" );
    getchar( );                 // para esperar
    return   ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO0803
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral
#include <math.h>           // para funcoes matematicas

typedef struct SPONTOS { float X,Y,Z; } PONTOS;
typedef PONTOS VETOR[2];

int main ( void )
{
 // PROGRAMA PARA CALCULAR A DISTANCIA ENTRE PONTOS
 // DADOS:
    VETOR  V;
    double   D = 0.0;

 // identificar
    printf  ( "EXEMPLO0803 - DISTANCIA ENTRE PONTOS\n" );
    printf  ( "\n ENTRE COM O PRIMEIRO PONTO : \n " );
    scanf ( "%lf %lf %lf", &V[0].X, &V[0].Y, &V[0].Z );
    getchar ( );                // para limpar a entrada de dados
    printf  ( "\n ENTRE COM O SEGUNDO PONTO : \n " );
    scanf ( "%lf %lf %lf", &V[1].X, &V[1].Y, &V[1].Z );
    getchar ( );                // para limpar a entrada de dados
    D = sqrt ( pow(V[1].X-V[0].X, 2.0) +
               pow(V[1].Y-V[0].Y, 2.0) +
               pow(V[1].Z-V[0].Z, 2.0) );
    printf  ("\n DISTANCIA = %lf", D );
 // encerrar
    printf    ( "\n\nAPERTAR <Enter> PARA TERMINAR" );
    getchar( );                 // para esperar
    return   ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO0804
// bibliotecas de funcoes auxiliares
#include <stdio.h>         // para entradas e saídas
#include <stdlib.h>        // para outras funcoes de uso geral
#include <math.h>          // para funcoes matematicas

typedef double PONTOS[3]; // X, Y, Z
typedef
   struct SVETOR
   { PONTOS P1, P2; }
VETOR;

int main ( void )
{
 // PROGRAMA PARA CALCULAR A DISTANCIA ENTRE PONTOS
 // DADOS:
    VETOR  V;
    double    D = 0.0;

 // identificar
    printf  ( "EXEMPLO0804 - DISTANCIA ENTRE PONTOS\n" );
    printf  ( "\n ENTRE COM O PRIMEIRO PONTO : \n " );
    scanf ( "%lf %lf %lf", &V.P1[0], &V.P1[1], &V.P1[2] );
    getchar ( );               // para limpar a entrada de dados
    printf  ( "\n ENTRE COM O SEGUNDO PONTO : \n " );
    scanf ( "%lf %lf %lf", &V.P2[0], &V.P2[1], &V.P2[2] );
    getchar ( );               // para limpar a entrada de dados
    D = sqrt ( pow(V.P2[0]-V.P1[0], 2.0)+
               pow(V.P2[1]-V.P1[1], 2.0)+
               pow(V.P2[2]-V.P1[2], 2.0) );
    printf  ( "\n DISTANCIA = %lf", D );
 // encerrar
    printf  ( "\n\nAPERTAR <Enter> PARA TERMINAR" );
    getchar ( );                   // para esperar
    return  ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO0805
// bibliotecas de funcoes auxiliares
#include <stdio.h>         // para entradas e saídas
#include <stdlib.h>        // para outras funcoes de uso geral
#include <math.h>          // para funcoes matematicas

typedef struct SPONTO  { double X,Y,Z; } PONTO ;
typedef struct SPONTOS { PONTO  P1,P2; } PONTOS;

int main ( void )
{
 // PROGRAMA PARA CALCULAR A DISTANCIA ENTRE PONTOS
 // DADOS:
    PONTOS P;
    double     D = 0.0;

 // identificar
    printf  ( "EXEMPLO0805 - DISTANCIA ENTRE PONTOS\n" );
    printf  ( "\n ENTRE COM O PRIMEIRO PONTO : \n " );
    scanf ( "%lf %lf %lf", &P.P1.X, &P.P1.Y, &P.P1.Z );
    getchar ( );               // para limpar a entrada de dados
    printf  ( "\n ENTRE COM O SEGUNDO  PONTO : \n " );
    scanf   ( "%lf %lf %lf", &P.P2.X, &P.P2.Y, &P.P2.Z );
    getchar ( );               // para limpar a entrada de dados
    D = sqrt ( pow(P.P2.X-P.P1.X, 2.0) +
               pow(P.P2.Y-P.P1.Y, 2.0) +
               pow(P.P2.Z-P.P1.Z, 2.0) );
    printf  ( "\n DISTANCIA = %lf", D );
 // encerrar
    printf    ( "\n\nAPERTAR <Enter> PARA TERMINAR" );
    getchar( );                // para esperar
    return   ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO0806
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral
#include <math.h>           // para funcoes matematicas

typedef struct SPONTO  { double X,Y,Z; } PONTO ;
typedef double VETOR[3];// X, Y, Z
typedef
   struct SPONTOS
  { PONTO P1;   VETOR P2; }
PONTOS;

int main ( void )
{
 // PROGRAMA PARA CALCULAR A DISTANCIA ENTRE PONTOS
 // DADOS:
    PONTOS P;
    double     D = 0.0;

 // identificar
    printf  ( "EXEMPLO0806 - DISTANCIA ENTRE PONTOS\n" );
    printf  ( "\n ENTRE COM O PRIMEIRO PONTO : \n " );
    scanf  ( "%lf %lf %lf", &P.P1.X, &P.P1.Y, &P.P1.Z );
    getchar ( );                // para limpar a entrada de dados
    printf  ( "\n ENTRE COM O SEGUNDO  PONTO : \n " );
    scanf  ( "%lf %lf %lf", &P.P2[0], &P.P2[1], &P.P2[2] );
    getchar ( );                // para limpar a entrada de dados
    D = sqrt ( pow(P.P2[0]-P.P1.X, 2.0) +
               pow(P.P2[1]-P.P1.Y, 2.0) +
               pow(P.P2[2]-P.P1.Z, 2.0) );
    printf  ( "\n DISTANCIA = %lf", D );
 // encerrar
    printf    ( "\n\nAPERTAR <Enter> PARA TERMINAR" );
    getchar( );                 // para esperar
    return   ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO0807
// bibliotecas de funcoes auxiliares
#include <stdio.h>         // para entradas e saídas
#include <stdlib.h>        // para outras funcoes de uso geral
#include <math.h>          // para funcoes matematicas

typedef double VETOR [3]; // X, Y, Z
typedef VETOR  PONTOS[2];

int main ( void )
{
 // PROGRAMA PARA CALCULAR
 // A DISTANCIA ENTRE PONTOS
 // DADOS:
    PONTOS P;
    double     D = 0.0;

 // identificar
    printf  ( "EXEMPLO0807 - DISTANCIA ENTRE PONTOS\n" );
    printf  ( "\n ENTRE COM O PRIMEIRO PONTO : \n " );
    scanf  ( "%lf %lf %lf", &P[0][0], &P[0][1], &P[0][2] );
    getchar ( );                // para limpar a entrada de dados
    printf  ( "\n ENTRE COM O SEGUNDO  PONTO : \n " );
    scanf   ( "%lf %lf %lf", &P[1][0], &P[1][1], &P[1][2] );
    getchar ( );                // para limpar a entrada de dados
    D = sqrt ( pow(P[1][0]-P[0][0], 2.0) +
               pow(P[1][1]-P[0][1], 2.0) +
               pow(P[1][2]-P[0][2], 2.0) );
    printf  ( "\n DISTANCIA = %lf", D );
 // encerrar
    printf    ( "\n\nAPERTAR <Enter> PARA TERMINAR" );
    getchar( );                     // para esperar
    return   ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO0808
// bibliotecas de funcoes auxiliares
#include <stdio.h>         // para entradas e saídas
#include <stdlib.h>        // para outras funcoes de uso geral
#include <math.h>          // para funcoes matematicas

typedef double VETOR1[3];// X, Y, Z
typedef VETOR1 VETOR [2];

int main ( void )
{
 // PROGRAMA PARA CALCULAR A DISTANCIA ENTRE PONTOS
 // DADOS:
    VETOR  P;
    double   D = 0.0;

    printf  ( "EXEMPLO0808 - DISTANCIA ENTRE PONTOS\n" );
    printf  ( "\n ENTRE COM O PRIMEIRO PONTO : \n " );
    scanf  ( "%lf %lf %lf", &P[0][0], &P[0][1], &P[0][2] );
    getchar ( );                // para limpar a entrada de dados
    printf  ( "\n ENTRE COM O SEGUNDO  PONTO : \n " );
    getchar ( );                // para limpar a entrada de dados
    scanf   ( "%lf %lf %lf", &P[1][0], &P[1][1], &P[1][2] );
    D = sqrt( pow(P[1][0]-P[0][0], 2.0) +
              pow(P[1][1]-P[0][1], 2.0) +
              pow(P[1][2]-P[0][2], 2.0) );
    printf  ( "\n DISTANCIA = %lf", D );
 // encerrar
    printf    ( "\n\nAPERTAR <Enter> PARA TERMINAR" );
    getchar( );                    // para esperar
    return   ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO0809
// bibliotecas de funcoes auxiliares
#include <stdio.h>         // para entradas e saídas
#include <stdlib.h>        // para outras funcoes de uso geral
#include <math.h>          // para funcoes matematicas

typedef struct SPONTO  { double X,Y,Z; } PONTO ;
typedef PONTO  VETOR[2];// X, Y, Z

int main ( void )
{
 // PROGRAMA PARA CALCULAR A DISTANCIA ENTRE PONTOS
 // DADOS:
    VETOR  P;
    double   D = 0.0;

 // identificar
    printf  ( "EXEMPLO0809 - DISTANCIA ENTRE PONTOS\n" );
    printf  ( "\n ENTRE COM O PRIMEIRO PONTO : \n " );
    scanf  ( "%lf%lf%lf", &(P[0].X), &(P[0].Y), &(P[0].Z));
    getchar ( );                // para limpar a entrada de dados
    printf  ( "\n ENTRE COM O SEGUNDO  PONTO : \n " );
    scanf  ( "%lf%lf%lf", &(P[1].X), &(P[1].Y), &(P[1].Z));
    getchar ( );                // para limpar a entrada de dados
    D = sqrt ( pow(P[1].X-P[0].X, 2.0) +
               pow(P[1].Y-P[0].Y, 2.0) +
               pow(P[1].Z-P[0].Z, 2.0) );
    printf  ( "\n DISTANCIA = %lf", D );
 // encerrar
    printf    ( "\n\nAPERTAR <Enter> PARA TERMINAR" );
    getchar( );                 // para esperar
    return   ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO0810
// bibliotecas de funcoes auxiliares
#include <stdio.h>         // para entradas e saídas
#include <stdlib.h>        // para outras funcoes de uso geral
#include <math.h>          // para funcoes matematicas

typedef struct SPONTO  { double X,Y,Z; } PONTO ;
typedef struct SPONTOS { PONTO  P1,P2; } PONTOS;

int main ( void )
{
 // PROGRAMA PARA CALCULAR A DISTANCIA ENTRE PONTOS
 // DADOS:
    PONTOS P;
    double     D = 0.0;

 // identificar
    printf  ( "EXEMPLO0810 - DISTANCIA ENTRE PONTOS\n");
    printf  ( "\n ENTRE COM O PRIMEIRO PONTO : \n ");
    scanf  ( "%lf%lf%lf", &(P.P1.X), &(P.P1.Y), &(P.P1.Z));
    getchar ( );                // para limpar a entrada de dados
    printf  ( "\n ENTRE COM O SEGUNDO  PONTO : \n ");
    scanf   ( "%lf%lf%lf", &(P.P2.X), &(P.P2.Y), &(P.P2.Z));
    getchar ( );                // para limpar a entrada de dados
    D = sqrt ( pow(P.P2.X-P.P1.X, 2.0) +
               pow(P.P2.Y-P.P1.Y, 2.0) +
               pow(P.P2.Z-P.P1.Z, 2.0) );
    printf  ( "\n DISTANCIA = %lf", D );
 // encerrar
    printf    ( "\n\nAPERTAR <Enter> PARA TERMINAR" );
    getchar( );                    // para esperar
    return   ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
//
// OBS.: RETIRAR OS COMENTARIOS /* */
//           PARA TESTAR CADA EXEMPLO INDIVIDUALMENTE.
//

/*
// -------------------------- EXEMPLO0901
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral
#include <math.h>           // para funcoes matematicas
#include <fcntl.h>          // para arquivos

typedef struct SPONTOS { double X,Y,Z; } PONTOS;

int main ( void )
{
 // PROGRAMA PARA GRAVAR COORDENADAS DE PONTOS
 // DADOS:
    PONTOS P;
    int           X = 0;
    FILE      *A;

 // identificar
    printf  ( "EXEMPLO0901 - GRAVAR COORDENADAS DE PONTOS \n" );
    A = fopen ( "PONTOS1.TXT", "wt" );
    for ( X = 1; X <= 2; X++ )
    {
        printf  ( "\nENTRE COM AS COORDENADAS DE UM PONTO : \n" );
        scanf  ( "%lf%lf%lf", &P.X, &P.Y, &P.Z );
        getchar ( );             // para limpar a entrada de dados
        fprintf ( A, "%lf  %lf  %lf\n", P.X, P.Y, P.Z );
    } // FIM REPETIR
    fclose  ( A );
 // encerrar
    printf    ( "\nAPERTAR <Enter> PARA TERMINAR" );
    getchar( );                 // para esperar
    return   ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO0902
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral
#include <math.h>           // para funcoes matematicas
#include <fcntl.h>          // para arquivos

typedef struct SPONTOS { double X,Y,Z; } PONTOS;

int main ( void )
{
 // PROGRAMA PARA LER ARQUIVO DE PONTOS
 // VARIAVEIS :
    PONTOS  P;
    int           X = 0;
    FILE      * A ;

 // identificar
    printf  ( "EXEMPLO0902 - LER ARQUIVO DE PONTOS \n" );
    A = fopen ( "PONTOS1.TXT", "rt" );
    for ( X = 1; X <= 2; X++ )
    {
       fscanf ( A, "%lf%lf%lf", &P.X, &P.Y, &P.Z );
       printf  ( "\nPONTO %d : %lf   %lf  %lf", X, P.X, P.Y, P.Z );
    } // FIM REPETIR
    fclose  ( A );
 // encerrar
    printf  ( "\nAPERTAR <Enter> PARA TERMINAR" );
    getchar ( );                // para esperar
    return  ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO0903
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral
#include <math.h>           // para funcoes matematicas
#include <fcntl.h>          // para arquivos

typedef struct SPONTOS { double X,Y,Z; } PONTOS;

int main ( void )
{
 // PROGRAMA PARA GRAVAR COORDENADAS DE PONTOS
 // DADOS:
    PONTOS P;
    int           X = 0;
    FILE      * A;

 // identificar
    printf ( "EXEMPLO0903 - GRAVAR COORDENADAS DE PONTOS \n" );
    A = fopen ( "PONTOS2.DAT", "wb" );
    for ( X = 1; X <= 2; X++ )
    {
        printf ( "\nENTRE COM AS COORDENADAS DE UM PONTO : \n" );
        scanf  ( "%lf%lf%lf", &P.X, &P.Y, &P.Z );
        getchar( );             // para limpar a entrada de dados
        fwrite ( &P, sizeof(PONTOS), 1, A );
    } // FIM REPETIR
    fclose ( A );
 // encerrar
    printf  ( "\nAPERTAR <Enter> PARA TERMINAR" );
    getchar ( );                // para esperar
    return  ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO0904
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral
#include <math.h>           // para funcoes matematicas
#include <fcntl.h>          // para arquivos

typedef struct SPONTOS { double X,Y,Z; } PONTOS;

int main ( void )
{
 // PROGRAMA PARA LER ARQUIVO DE PONTOS
 // DADOS:
    PONTOS P;
    int           X = 0;
    FILE       *A;

 // identificar
    printf ( "EXEMPLO0904 - LER ARQUIVO DE PONTOS \n" );
    A = fopen ( "PONTOS2.DAT", "rb" );
    for( X = 1; X <= 2; X++ )
    {
      fread  ( &P, sizeof(PONTOS), 1, A );
      printf ( "\nPONTO %d : %6.2lf %6.2lf %6.2lf\n", X, P.X, P.Y, P.Z );
    } // FIM REPETIR
    fclose(A);
 // encerrar
    printf    ( "\nAPERTAR <Enter> PARA TERMINAR" );
    getchar( );                     // para esperar
    return   ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO0905
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral
#include <math.h>           // para funcoes matematicas
#include <fcntl.h>          // para arquivos

typedef struct SPONTOS { double X,Y,Z; } PONTOS;

int main ( void )
{
 // PROGRAMA PARA COPIAR O ARQUIVO COM COORDENADAS DE PONTOS
 // DADOS:
    PONTOS P;
    int           X = 0;
    FILE       *A1, *A2;

 // identificar
    printf ( "EXEMPLO0905 - COPIAR COORDENADAS DE PONTOS \n" );
    A1 = fopen ( "PONTOS1.TXT", "rt" );
    A2 = fopen ( "NOVO1.DAT"  , "wb" );
    for ( X = 1; X <= 2; X++ )
    {
      fscanf ( A1, "%lf%lf%lf\n", &P.X, &P.Y, &P.Z );
      fwrite  ( &P, sizeof(PONTOS), 1, A2 );
      printf   ( "\nCOPIADO %d : %6.2lf %6.2lf %6.2lf\n", X, P.X, P.Y, P.Z );
    } // FIM REPETIR
    fclose  ( A1 );
    fclose  ( A2 );
 // encerrar
    printf    ( "\nAPERTAR <Enter> PARA TERMINAR" );
    getchar( );                 // para esperar
    return   ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO0906
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral
#include <math.h>           // para funcoes matematicas
#include <fcntl.h>          // para arquivos

typedef struct SPONTOS { double X,Y,Z; } PONTOS;

int main ( void )
{
 // PROGRAMA PARA ACRESCENTAR COORDENADAS DE PONTOS
 // DADOS:
    PONTOS P;
    int           X = 0;
    FILE       *A;

 // identificar
    printf ( "EXEMPLO0906 - ACRESCENTAR COORDENADAS DE PONTOS \n" );
    A = fopen ( "PONTOS2.DAT", "r+b" );
    while( ! feof(A) )
      fread ( &P, sizeof(PONTOS), 1, A );  // LER ATE' O FIM DE ARQUIVO
    fseek ( A, 0L, SEEK_CUR );              // MARCAR A POSICAO
    for ( X = 3; X <= 4; X++ )
    {
      printf ( "\nENTRE COM AS COORDENADAS DE OUTRO PONTO (%d): \n", X );
      scanf  ( "%lf%lf%lf", &P.X, &P.Y, &P.Z );
      getchar ( );     // para limpar a entrada de dados
      fwrite ( &P, sizeof(PONTOS), 1, A );
    } // FIM REPETIR
    fseek ( A, 0L, SEEK_SET );              // VOLTAR AO INICIO
    fread ( &P, sizeof(PONTOS), 1, A );    // LER O PRIMEIRO DO ARQUIVO
    while( ! feof(A) )
    {
      printf ( "\nPONTO %d : %6.2lf %6.2lf %6.2lf\n", X, P.X, P.Y, P.Z );
      fread ( &P, sizeof(PONTOS), 1, A );  // LER ATE' O FIM DE ARQUIVO
    } // FIM REPETIR
    fclose  ( A );
 // encerrar
    printf    ( "\nAPERTAR <Enter> PARA TERMINAR" );
    getchar( );                             // para esperar
    return   ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO0907
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral
#include <math.h>           // para funcoes matematicas
#include <fcntl.h>          // para arquivos

typedef struct SPONTOS { double X,Y,Z; } PONTOS;

int main ( void )
{
 // PROGRAMA PARA ACRESCENTAR COORDENADAS DE PONTOS
 // DADOS:
    PONTOS P;
    int           X = 0;
    FILE       *A;

 // identificar
    printf ( "EXEMPLO0907 - ACRESCENTAR COORDENADAS DE PONTOS \n" );
    A = fopen ( "PONTOS1.TXT", "at" );
    for( X = 1; X <= 2; X++ )
    {
      printf  ( "\nENTRE COM AS COORDENADAS DE OUTRO PONTO : \n" );
      scanf   ( "%lf%lf%lf", &P.X, &P.Y, &P.Z );
      getchar ( );              // para limpar a entrada de dados
      fprintf ( A, "%Lf  %Lf  %Lf", P.X, P.Y, P.Z );
      fprintf ( A, "%s", "\n" );
    } // FIM REPETIR
    fclose  ( A );
 // encerrar
    printf    ( "\nAPERTAR <Enter> PARA TERMINAR" );
    getchar( );                 // para esperar
    return   ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO0908
// bibliotecas de funcoes auxiliares
#include <stdio.h>         // para entradas e saídas
#include <stdlib.h>        // para outras funcoes de uso geral
#include <stdbool.h>       // para definicoes logicas
#include <math.h>          // para funcoes matematicas
#include <fcntl.h>         // para arquivos

typedef struct SPONTOS { double X,Y,Z; } PONTOS;

int main ( void )
{
 // PROGRAMA PARA PROCURAR COORDENADAS DE PONTOS
 // DADOS:
    PONTOS  P,
            PROCURADO;
    bool    ACHAR = false;
    FILE    *A;

 // identificar
    printf ( "EXEMPLO0908 - PROCURAR COORDENADAS DE PONTOS \n" );
    printf ( "\nENTRE COM AS COORDENADAS DO PONTO A PROCURAR : \n" );
    scanf ( "%lf%lf%lf", &PROCURADO.X, &PROCURADO.Y, &PROCURADO.Z );
    ACHAR = false;
    A = fopen ( "PONTOS1.TXT", "rt" );
    fscanf ( A, "%lf %lf %lf", &P.X, &P.Y, &P.Z );
    while ( ! feof( A ) && ! ACHAR )
    {
      if ( P.X==PROCURADO.X && P.Y==PROCURADO.Y && P.Z==PROCURADO.Z )
        ACHAR = true;
      fscanf ( A, "%lf%lf%lf", &P.X, &P.Y, &P.Z );
    } // FIM REPETIR
    fclose ( A );
    if ( ACHAR )
      printf ( "\nCOORDENADAS ENCONTRADAS" );
    else
      printf ( "\nCOORDENADAS NAO ENCONTRADAS" );
 // encerrar
    printf    ( "\nAPERTAR <Enter> PARA TERMINAR" );
    getchar( );                // para esperar
    return   ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO0909
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral
#include <math.h>           // para funcoes matematicas
#include <fcntl.h>          // para arquivos

typedef struct SPONTOS { float X,Y,Z; } PONTOS;

int main ( void )
{
 // PROGRAMA PARA ALTERAR COORDENADAS DE PONTOS
 // DADOS:
    PONTOS P;
    int           X = 0;
    FILE       *A;

 // identificar
    printf ( "EXEMPLO0909 - ALTERAR COORDENADAS DE PONTOS \n" );
    A = fopen( "PONTOS2.DAT", "r+b" );
    for ( X = 1; X <= 2; X++ )
    {
      fseek ( A, (X-1)*sizeof(PONTOS), SEEK_SET ); // o primeiro e' zero
      fread ( &P, sizeof(PONTOS), 1, A );
      printf ( "\nATUAL %d : %6.2Lf %6.2Lf %6.2Lf\n", X, P.X, P.Y, P.Z );
      printf ( "\nENTRE COM AS NOVAS COORDENADAS DO PONTO : \n" );
      scanf ( "%lf%lf%lf", &P.X, &P.Y, &P.Z );
      getchar( );               // para limpar a entrada de dados
      fseek  ( A, (X-1)*sizeof(PONTOS), SEEK_SET );
      fwrite ( &P, sizeof(PONTOS), 1, A );
    } // FIM REPETIR
    fclose(A);
 // encerrar
    printf    ( "\nAPERTAR <Enter> PARA TERMINAR" );
    getchar( );                 // para esperar
    return   ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO0910
// bibliotecas de funcoes auxiliares
#include <stdio.h>         // para entradas e saídas
#include <stdlib.h>        // para outras funcoes de uso geral
#include <math.h>          // para funcoes matematicas
#include <fcntl.h>         // para arquivos

typedef struct SPONTOS { double X,Y,Z; } PONTOS;

int main ( void )
{
 // PROGRAMA PARA LER ARQUIVO DE PONTOS DIRETAMENTE
 // DADOS:
    PONTOS P;
    int           X = 0;
    FILE       *A;

 // identificar
    printf ( "EXEMPLO0910 - LER ARQUIVO DE PONTOS DIRETAMENTE \n" );
    A = fopen ( "PONTOS2.DAT", "r" );
    for ( X = 2; X > 0; X-- )
    {
      fseek  ( A, (X-1) * sizeof (PONTOS), SEEK_SET ); // o primeiro e' zero
      fread  ( &P, sizeof(PONTOS), 1, A );
      printf ( "\nPONTO %d : %6.2lf %6.2lf %6.2lf\n", X, P.X, P.Y, P.Z );
    } // FIM REPETIR
    fclose( A );
 // encerrar
    printf    ( "\nAPERTAR <Enter> PARA TERMINAR" );
    getchar( );                // para esperar
    return   ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
//
// OBS.: RETIRAR OS COMENTARIOS /* */
//           PARA TESTAR CADA EXEMPLO INDIVIDUALMENTE.
//

/*
// -------------------------- EXEMPLO1001
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral
#include <math.h>           // para funcoes matematicas

typedef
 struct S_CELULA { int VALOR; struct S_CELULA *LINK; }
CELULA;

int main ( void )
{
 // PROGRAMA PARA DEFINIR E MONTAR CELULA DE DADOS
 // DADOS:
    CELULA      P;
    P.VALOR = 0;
    P.LINK     = NULL;

 // identificar
    printf  ( "EXEMPLO1001 - MONTAR CELULA \n" );
    printf  ( "VALOR = " );
    scanf ( "%d", &P.VALOR );
    getchar ( );
    printf  ( "VALOR = %d LINK = %p\n", P.VALOR, P.LINK );
    printf  ( "\n" );
 // encerrar
    printf    ( "\nAPERTAR <Enter> PARA TERMINAR" );
    getchar( );                     // para esperar
    return   ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO1002
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral
#include <math.h>           // para funcoes matematicas

typedef
 struct S_CELULA { int VALOR; struct S_CELULA *LINK; }
CELULA;

int main ( void )
{
 // PROGRAMA PARA DEFINIR E MONTAR CELULA DE DADOS
 // DADOS:
    CELULA   *P;                          // APONTADOR PARA CELULA
    P = malloc ( 1*sizeof(CELULA) ); // RESERVAR ESPACO
    (*P).VALOR = 0;
    (*P).LINK     = NULL;

 // identificar
    printf  ( "EXEMPLO1002 - MONTAR CELULA \n" );
    printf  ( "VALOR = " );
    scanf ( "%d", &(*P).VALOR );
    getchar ( );
    printf  ( "VALOR = %d LINK = %p\n", (*P).VALOR, (*P).LINK );
    printf  ( "\n" );
 // encerrar
    printf    ( "\nAPERTAR <Enter> PARA TERMINAR" );
    getchar( );                 // para esperar
    return   ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO1003
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral
#include <math.h>           // para funcoes matematicas

typedef
 struct S_CELULA { int VALOR; struct S_CELULA *LINK; }
CELULA;

int main ( void )
{
 // PROGRAMA PARA DEFINIR E MONTAR CELULA DE DADOS
 // DADOS:
    CELULA   *P;                        // APONTADOR PARA CELULA
    P = malloc ( 1*sizeof(CELULA) ); // RESERVAR ESPACO
    P->VALOR = 0;                       // OUTRA NOTACAO
    P->LINK     = NULL;

 // identificar
    printf  ( "EXEMPLO1003 - MONTAR CELULA \n" );
    printf  ( "VALOR = " );
    scanf ( "%d", &P->VALOR );
    getchar ( );
    printf  ( "VALOR = %d LINK = %p\n", P->VALOR, P->LINK );
    printf  ( "\n" );
 // encerrar
    printf    ( "\nAPERTAR <Enter> PARA TERMINAR" );
    getchar( );                 // para esperar
    return   ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO1004
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral
#include <math.h>           // para funcoes matematicas

typedef
 struct S_CELULA { int VALOR; struct S_CELULA *LINK; }
CELULA;

int main ( void )
{
 // PROGRAMA PARA DEFINIR E MONTAR CELULA DE DADOS
 // DADOS:
    CELULA  *P;                         // APONTADOR PARA CELULA
    P = calloc ( 1, sizeof(CELULA) ); // OUTRA FORMA DE RESERVAR
    P->VALOR = 0;                       // OUTRA NOTACAO
    P->LINK     = NULL;

 // identificar
    printf  ( "EXEMPLO1004 - MONTAR CELULA \n" );
    printf  ( "VALOR = " );
    scanf ( "%d", &P->VALOR );
    getchar ( );
    printf  ( "VALOR = %d LINK = %p\n", P->VALOR, P->LINK );
    printf  ( "\n" );
 // encerrar
    printf    ( "\nAPERTAR <Enter> PARA TERMINAR" );
    getchar( );                 // para esperar
    return   ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO1005
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral
#include <math.h>           // para funcoes matematicas

typedef
 struct S_CELULA { int VALOR; struct S_CELULA *LINK; }
CELULA;

typedef
 struct S_CELULA *
REF_CELULA;

int main ( void )
{
 // PROGRAMA PARA DEFINIR E MONTAR CELULA DE DADOS
 // DADOS:
    REF_CELULA P;                    // APONTADOR PARA CELULA
    P = calloc ( 1, sizeof(CELULA) ); // OUTRA FORMA DE RESERVAR
    P->VALOR = 0;                    // OUTRA NOTACAO
    P->LINK     = NULL;

 // identificar
    printf  ( "EXEMPLO1005 - MONTAR CELULA \n" );
    printf  ( "VALOR = " );
    scanf  ( "%d", &P->VALOR );
    getchar ( );
    printf  ( "VALOR = %d LINK = %p\n", P->VALOR, P->LINK );
    printf  ( "\n" );
 // encerrar
    printf    ( "\nAPERTAR <Enter> PARA TERMINAR" );
    getchar( );                      // para esperar
    return   ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO1006
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral
#include <math.h>           // para funcoes matematicas

typedef
 struct S_CELULA { int VALOR; struct S_CELULA *LINK; }
CELULA;

typedef
 struct S_CELULA *
REF_CELULA;

REF_CELULA NEW_CELULA ( int INICIAL )
{
   REF_CELULA P;                    // APONTADOR PARA CELULA
   P = calloc ( 1, sizeof(CELULA) ); // OUTRA FORMA DE RESERVAR
   P->VALOR = INICIAL;              // OUTRA NOTACAO
   P->LINK     = NULL;
   return ( P );
} // fim funcao NEW_CELULA ( )

int main ( void )
{
 // PROGRAMA PARA DEFINIR E MONTAR CELULA DE DADOS
 // DADOS:
   REF_CELULA P = NEW_CELULA ( 0 );  // APONTADOR PARA CELULA

 // identificar
   printf  ( "EXEMPLO1006 - MONTAR CELULA \n" );
   printf  ( "VALOR = " );
   scanf  ( "%d", &P->VALOR );
   getchar ( );
   printf  ( "VALOR = %d LINK = %p\n", P->VALOR, P->LINK );
   printf  ( "\n" );
 // encerrar
   printf    ( "\nAPERTAR <Enter> PARA TERMINAR" );
   getchar( );                // para esperar
   return   ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO1007
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral
#include <math.h>           // para funcoes matematicas

typedef
 struct S_CELULA { int VALOR; struct S_CELULA *LINK; }
CELULA;

typedef
 struct S_CELULA *
REF_CELULA;

REF_CELULA NEW_CELULA ( int INICIAL )
{
   REF_CELULA P;                    // APONTADOR PARA CELULA
   P = calloc ( 1, sizeof(CELULA) ); // OUTRA FORMA DE RESERVAR
   P->VALOR = INICIAL;              // OUTRA NOTACAO
   P->LINK     = NULL;
   return ( P );
} // fim funcao NEW_CELULA ( )

int main ( void )
{
 // PROGRAMA PARA DEFINIR E MONTAR CELULAS DE DADOS
 // DADOS:
   REF_CELULA P1 = NEW_CELULA ( 0 );  // APONTADOR PARA CELULA
   REF_CELULA P2 = NEW_CELULA ( 0 );  // APONTADOR PARA CELULA

 // identificar
   printf  ( "EXEMPLO1007 - MONTAR CELULAS \n" );
   printf  ( "P1: VALOR = " );
   scanf  ( "%d", &P1->VALOR );
   getchar ( );
   printf  ( "P2: VALOR = " );
   scanf  ( "%d", &P2->VALOR );
   getchar ( );
   printf  ( "P1: VALOR = %d LINK = %p\n", P1->VALOR, P1->LINK );
   printf  ( "P2: VALOR = %d LINK = %p\n", P2->VALOR, P2->LINK );
   P1->LINK = P2;         // ligar uma celula a outra
   printf  ( "P1: VALOR = %d LINK = %p\n", P1->VALOR, P1->LINK );
   printf  ( "P2: VALOR = %d LINK = %p\n",
               P1->LINK->VALOR, P1->LINK->LINK );
   printf  ( "\n" );
 // encerrar
   printf    ( "\nAPERTAR <Enter> PARA TERMINAR" );
   getchar( );                // para esperar
   return   ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO1008
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral
#include <math.h>           // para funcoes matematicas

typedef
 struct S_CELULA { int VALOR; struct S_CELULA *LINK; }
CELULA;

typedef  struct S_CELULA *  REF_CELULA;

REF_CELULA NEW_CELULA ( int INICIAL )
{
   REF_CELULA P;                    // APONTADOR PARA CELULA
   P = calloc ( 1, sizeof(CELULA) ); // OUTRA FORMA DE RESERVAR
   P->VALOR = INICIAL;              // OUTRA NOTACAO
   P->LINK     = NULL;
   return ( P );
} // fim funcao NEW_CELULA ( )

int main ( void )
{
 // PROGRAMA PARA DEFINIR E MONTAR CELULAS DE DADOS
 // DADOS:
   REF_CELULA P1 = NEW_CELULA ( 0 );  // APONTADOR PARA CELULA
   REF_CELULA P2 = NEW_CELULA ( 0 );  // APONTADOR PARA CELULA

 // identificar
   printf  ( "EXEMPLO1008 - MONTAR CELULAS \n" );
   printf  ( "P1: VALOR = " );
   scanf  ( "%d", &P1->VALOR );   getchar ( );
   printf  ( "P2: VALOR = " );
   scanf  ( "%d", &P2->VALOR );   getchar ( );
   printf  ( "P1: VALOR = %d LINK = %p\n", P1->VALOR, P1->LINK );
   printf  ( "P2: VALOR = %d LINK = %p\n", P2->VALOR, P2->LINK );
   P1->LINK = P2;                   // ligar uma celula a outra
   P2 = NEW_CELULA ( 0 );
   printf  ( "P3: VALOR = " );
   scanf  ( "%d", &P2->VALOR );   getchar ( );
   P1->LINK->LINK = P2;             // ligar 'a terceira
   printf  ( "P1: VALOR = %d LINK = %p\n", P1->VALOR, P1->LINK );
   printf  ( "P2: VALOR = %d LINK = %p\n",
               P1->LINK->VALOR, P1->LINK->LINK );
   printf  ( "P3: VALOR = %d LINK = %p\n",
               P1->LINK->LINK->VALOR, P1->LINK->LINK->LINK );
   printf  ( "\n" );
 // encerrar
   printf    ( "\nAPERTAR<Enter> PARA TERMINAR" );
   getchar( );                  // para esperar
   return   ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO1009
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral
#include <math.h>           // para funcoes matematicas

typedef
 struct S_CELULA { int VALOR; struct S_CELULA *LINK; }
CELULA;

typedef  struct S_CELULA *  REF_CELULA;

REF_CELULA NEW_CELULA ( int INICIAL )
{
   REF_CELULA P;                    // APONTADOR PARA CELULA
   P = calloc ( 1, sizeof(CELULA) ); // OUTRA FORMA DE RESERVAR
   P->VALOR = INICIAL;              // OUTRA NOTACAO
   P->LINK  = NULL;
   return ( P );
} // fim funcao NEW_CELULA ( )

int main ( void )
{
 // PROGRAMA PARA DEFINIR E MONTAR CELULAS DE DADOS
 // DADOS:
   REF_CELULA P1 = NEW_CELULA ( 0 );  // APONTADOR PARA CELULA
   REF_CELULA P2 = NEW_CELULA ( 0 );  // APONTADOR PARA CELULA

 // identificar
   printf  ( "EXEMPLO1009 - MONTAR CELULAS \n" );
   printf  ( "P1: VALOR = " );
   scanf  ( "%d", &P1->VALOR );  getchar ( );
   printf  ( "P2: VALOR = " );
   scanf  ( "%d", &P2->VALOR );  getchar ( );
   printf  ( "P1: VALOR = %d LINK = %p\n", P1->VALOR, P1->LINK );
   printf  ( "P2: VALOR = %d LINK = %p\n", P2->VALOR, P2->LINK );
   P1->LINK = P2;                   // ligar uma celula a outra
   P2 = NEW_CELULA ( 0 );
   printf  ( "P3: VALOR = " );
   scanf  ( "%d", &P2->VALOR );  getchar ( );
   P1->LINK->LINK = P2;             // ligar 'a terceira
   P2 = P1;                         // comecar no primeiro
   while ( P2 != NULL )
   {                                // mostrar valor
     printf  ( "VALOR = %d LINK = %p\n", P2->VALOR, P2->LINK );
     P2 = P2->LINK;                 // passar ao proximo
   } // end while
   printf  ( "\n" );
 // encerrar
   printf  ( "\nAPERTAR <Enter> PARA TERMINAR" );
   getchar ( );                 // para esperar
   return  ( EXIT_SUCCESS );
} // end main ( )
*/
```

```
/*
// -------------------------- EXEMPLO1010
// bibliotecas de funcoes auxiliares
#include <stdio.h>          // para entradas e saídas
#include <stdlib.h>         // para outras funcoes de uso geral
#include <math.h>           // para funcoes matematicas

typedef
 struct S_CELULA { int VALOR; struct S_CELULA *LINK; }
CELULA;
typedef  struct S_CELULA *  REF_CELULA;

REF_CELULA NEW_CELULA ( int INICIAL )
{
   REF_CELULA P;                    // APONTADOR PARA CELULA
   P = calloc ( 1, sizeof(CELULA) ); // OUTRA FORMA DE RESERVAR
   P->VALOR = INICIAL;              // OUTRA NOTACAO
   P->LINK     = NULL;
   return ( P );
} // fim funcao NEW_CELULA ( )

int main ( void )
{
 // PROGRAMA PARA DEFINIR E MONTAR CELULAS DE DADOS
 // DADOS:
   REF_CELULA P1 = NULL;  // APONTADOR ESTRUTURAL
   REF_CELULA P2 = NULL;  // APONTADOR DE SERVICO
   int x = 0,  y = 0;

 // identificar
   printf  ( "EXEMPLO1010 - MONTAR CELULAS \n" );
   printf  ( "VALOR = " );
   scanf  ( "%d", &x   );    getchar ( );
   P1 = NEW_CELULA ( x );       // montar o primeiro
   for ( y=0; y<4; y=y+1 )
   {
      printf  ( "VALOR = " );
      scanf  ( "%d", &x   );  getchar ( );
      P2 = P1;                      // comecar no primeiro
      while ( P2->LINK != NULL ) // procurar o ultimo
         P2 = P2->LINK;
      P2->LINK = NEW_CELULA ( x );  // anexar mais outro
   } // end for
   P2 = P1;                         // comecar no primeiro
   while ( P2 != NULL )
   {                                // mostrar valor
     printf  ( "VALOR = %d LINK = %p\n", P2->VALOR, P2->LINK );
     P2 = P2->LINK;                 // passar ao proximo
   } // end while
   printf  ( "\n" );
 // encerrar
   printf    ( "\nAPERTAR <Enter> PARA TERMINAR" );
   getchar( );                  // para esperar
   return   ( EXIT_SUCCESS );
} // end main ( )
*/
```