Delegados e Eventos

? EXERCÍCIO 1: DELEGADO PARA RESTOS

```
using System;
namespace DE1
{delegate int DelegadoParaExecFuncoes (int X);

 public class RestosDivisao
 {public static int Dois(int X)
   {return X % 2;}
  public static int Tres(int X)
   {return X % 3; }}

 public class RestosPorDoisETres
  {static void Main(string[] args)
   {DelegadoParaExecFuncoes D;
    int X = 4;
    D = new DelegadoParaExecFuncoes(RestosDivisao.Dois);
    Console.WriteLine("Resto de {0} a dividir por 2={1}",X,D(X));
    D = new DelegadoParaExecFuncoes(RestosDivisao.Tres);
    Console.WriteLine("Resto de {0} a dividir por
    3={1}",X,D(X));}}}
```

? EXERCÍCIO 2: VETOR DE FUNÇÕES

```
using System;
namespace DE2
{delegate int DelegadoParaExecFuncoes( int X);

 public class Multiplos
  {public static int Dobro( int X)
  {return X * 2; }
  public static int Triplo( int X)
  {return X * 3; }
  public static int Quadruplo( int X)
  {return X * 4; }}

 public class VectorDeFuncoes
  {static void Main(string[] args)
   {int X = 5;
   int Total = 0;
```

```
    DelegadoParaExecFuncoes[] D = {Multiplos.Dobro,
    Multiplos.Triplo, Multiplos.Quadruplo};
    for (int I = 0; I < D.Length; I++)
     Total += D[I](X);
    Console.WriteLine("Somatório do dobro, triplo e quadrúplo de
    {0}={1}",X, Total);}}}
```

? EXERCÍCIO 3: COMPOSIÇÃO DE FUNÇÕES

```
using System;
namespace DE3
{delegate int DelegadoParaComposicaoFunc( int X);

 public class ContentorDeFunc
  {public static int Dobro( int X)
   {return X * 2;}
  public static int Triplo( int X)
   {return X * 3;}}

 public class ComposicaoDeFuncoes
 {static void Main(string[] args)
  {DelegadoParaComposicaoFunc[] D={ContentorDeFunc.Triplo,
  ContentorDeFunc.Dobro};
   int X= 5;
   Console.WriteLine("Triplo do dobro de {0}={1}", X,
   D[0](D[1](X)));}}}
```

? EXERCÍCIO 4: COMBINAÇÃO DE DELEGADOS

```
using System;
namespace DE4
{delegate void DelegadoParaMensagens(string X);

public class Mensagens
 {public static void Mensag1(string X)
  {Console.WriteLine(X+ "do primeiro método"); }
   public static void Mensag2(string X)
    {Console.WriteLine(X+"do segundo método"); }
   public static void Mensag3(string X)
    {Console.WriteLine(X + "do terceiro método"); }}

  public class CombinacaoDeDelegados
   {static void Main(string[] args)
    {DelegadoParaMensagens[] R={Mensagens.Mensag1,
    Mensagens.Mensag2, Mensagens.Mensag3};
    DelegadoParaMensagens Sequencia;
```

```
      string X="Primeira invocação ";
      Sequencia = R[0] + R[1];
      Sequencia(X);
      X="Segunda invocação ";
      Sequencia += R[2];
      Sequencia(X);}}}
```

? EXERCÍCIO 5: REMOÇÃO DE DELEGADOS

```
using System;
namespace DE5
{delegate void DelegadoParaMensagens(string X);

 public class Mensagens
 {public static void Mensag1(string X)
  {Console.WriteLine(X+ "do primeiro método"); }
   public static void Mensag2(string X)
    {Console.WriteLine(X+"do segundo método"); }
   public static void Mensag3(string X)
    {Console.WriteLine(X + "do terceiro método"); }}

  public class RemocaoDeDelegados
   {static void Main(string[] args)
    {DelegadoParaMensagens[] R={Mensagens.Mensag1,
    Mensagens.Mensag2, Mensagens.Mensag3};
    DelegadoParaMensagens Sequencia;
     string X="Primeira invocação ";
     Sequencia = R[0] + R[1];
     Sequencia(X);
     X="Segunda invocação ";
     Sequencia-=R[1];
     Sequencia(X);}}}
```

? EXERCÍCIO 6: PASSAGEM DE PARÂMETROS A DELEGADOS

```
using System;
namespace DE6
{delegate void DelegadoParaLinhas(ref string Cadeia, string
 Caract);

 public class Contentor
 {public static void CadeiaCaract(ref string Cadeia, string
 Caract)
  {Cadeia+=Caract;
```

```
    Console.WriteLine(Cadeia);}}

    public class TrianguloPorDelegado
     {static void Main(string[] args)
      {DelegadoParaLinhas LinhasD;
       int NLinhas=10;
       LinhasD = new DelegadoParaLinhas(Contentor.CadeiaCaract);
       for(int I=1; I<NLinhas; I++)
       LinhasD+=new DelegadoParaLinhas(Contentor.CadeiaCaract);
       string Linha="";
       LinhasD(ref Linha, "a");}}}
```

? EXERCÍCIO 7: CLASSE COM DELEGADO

```
public class ParDeObjectos
 {private object[] OPar = new object[2];
  public delegate int Ordenar(object O1, object O2);

  public ParDeObjectos(object O1, object O2)
   {OPar[0] = O1;
    OPar[1] = O2;}
  public void Ordem(Ordenar FuncaoDel)
   {if (FuncaoDel(OPar[0], OPar[1])== 2)
    {object Temporaria = OPar[0];
     OPar[0] = OPar[1];
     OPar[1] = Temporaria;}}
  public override string ToString()
  {return OPar[0].ToString() + " e " + OPar[1].ToString();}}
```

? EXERCÍCIO 8: ORDENAÇÃO DE PESOS COM DELEGADO

```
using System;
namespace DE8
{public class ParDeObjectos
 {private object[] OPar = new object[2];
  public delegate int Ordenar(object O1, object O2);
  public ParDeObjectos(object O1, object O2)
   {OPar[0] = O1;
    OPar[1] = O2;}

  public void Ordem(Ordenar FuncaoDel)
   {if (FuncaoDel(OPar[0], OPar[1])== 2)
    {object Temporaria = OPar[0];
     OPar[0] = OPar[1];
```

```
      OPar[1] = Temporaria;}}
   public override string ToString()
   {return OPar[0].ToString() + " e " + OPar[1].ToString(); }}

 public class Gato
  {private int Peso;
 public Gato(int P)
  {Peso = P; }
 public static int OrdemPesos(Object O1, object O2)
  {Gato G1 = (Gato) O1;
   Gato G2 = (Gato) O2;
   return (G1.Peso < G2.Peso ? 1 : 2);}
 public override string ToString()
  {return Peso.ToString()+"kg";}}
public class OrdenarPesos
 {static void Main(string[] args)
  {Gato G1 = new Gato(20);
   Gato G2 = new Gato(15);
   ParDeObjectos Pares = new ParDeObjectos(G1, G2);
   ParDeObjectos.Ordenar Delegado = new
   ParDeObjectos.Ordenar(Gato.OrdemPesos);
   Pares.Ordem(Delegado);
   Console.WriteLine(Pares.ToString());}}}
```

? EXERCÍCIO 9: ORDENAÇÃO DE NOMES COM DELEGADO

```
using System;
namespace DE9
{public class ParDeObjectos
 {private object[] OPar = new object[2];
  public delegate int Ordenar(object O1, object O2);
  public ParDeObjectos(object O1, object O2)
   {OPar[0] = O1;
    OPar[1] = O2;}

  public void Ordem(Ordenar FuncaoDel)
   {if (FuncaoDel(OPar[0], OPar[1])== 2)
    {object Temporaria = OPar[0];
     OPar[0] = OPar[1];
     OPar[1] = Temporaria;}}
  public override string ToString()
  {return OPar[0].ToString() + " e " + OPar[1].ToString(); }}

public class Pessoa
 {public string Nome;
```

```
  public Pessoa(string N)
   {Nome = N; }
  public static int OrdemNomes(Object O1, object O2)
   {Pessoa P1 = (Pessoa)O1;
    Pessoa P2 = (Pessoa)O2;
    return (String.Compare(P1.Nome, P2.Nome) < 0 ? 1 : 2);}
  public override string ToString()
   {return Nome;}}

public class OrdenarNomes
 {static void Main(string[] args)
  {Pessoa P1 = new Pessoa("Joana");
   Pessoa P2 = new Pessoa("Abel");
   ParDeObjectos Pares = new ParDeObjectos(P1, P2);
   ParDeObjectos.Ordenar Delegado = new
   ParDeObjectos.Ordenar(Pessoa.OrdemNomes);
   Pares.Ordem(Delegado);
   Console.WriteLine(Pares.ToString());}}}
```

? EXERCÍCIO 10: OCORRÊNCIA DE MÚLTIPLOS DE 5 E RESPOSTA

```
using System;
namespace DE10
{public delegate void RespMultiplo5(object produtor, EventArgs
arg);

 public class Produtor
  {public event RespMultiplo5 Multiplo5;
  public void GerarNumerosENotificar()
  {Random A = new Random();
  int Aleat;
  for (int I = 1; I <= 50; I++)
   {Aleat = A.Next() % 100;
    if (Aleat % 5 == 0)
     Multiplo5(this, new EventArgs());}}}

 public class Consumidor
  {public static int M5=0;
   public void Subscricao(Produtor P)
   {P.Multiplo5 += new RespMultiplo5(RecebiUmMultiplo); }
   public void RecebiUmMultiplo(object produtor, EventArgs arg)
   {Console.WriteLine("Recebi um múltiplo de 5");
   M5++;}}

 public class RespostaAMultiplosDeCinco
  {static void Main(string[] args)
```

```
    {Consumidor C = new Consumidor();
     Produtor P = new Produtor();
     C.Subscricao(P);
     P.GerarNumerosENotificar();
     Console.WriteLine("Recebi {0} múltiplos de 5",
     Consumidor.M5);}}}
```

? EXERCÍCIO 11: DETERMINADOS TELEFONEMAS

```
using System;
namespace DE11
{public class QuemLigou : EventArgs
 {private string Nome;
  private int Horas;
  public QuemLigou(string A, int H)
   {Nome=A;
    Horas=H;}
  public string DadosLigacao
   {get
    {return Nome + " telefonou às " + Horas;}}}

public class Voicemail
 {private string PDesejada;
  public delegate void Telefonemas(object produtor, QuemLigou A);
  public event Telefonemas Ligou;
  public Voicemail(string P)
   {PDesejada=P;}
  public void VerificarOcorrencia(string A, int H)
   {if (A == PDesejada)
    Ligou(this, new QuemLigou(A, H));}}

public class Email
 {public Email()
  { }
  public void subscricao(Voicemail C)
   {C.Ligou+=new Voicemail.Telefonemas(Resposta);}

  public void Resposta(object VoiceMail, QuemLigou A)
   {Console.WriteLine("Recebi notificação: {0} e vou enviar
   email!", A.DadosLigacao);}}

public class ControloVoiceMail
 {public static void Main()
  {string[] Nomes = { "Silva", "Paiva", "Ferreira", "Ferro",
   "Paiva", "Pinho", "Paiva" };
   int[] Horas = {8, 12, 14, 17, 18, 19, 20};
```

```
    Voicemail C = new Voicemail("Paiva");
    Email L = new Email();
    L.subscricao(C);
    for (int I = 0; I < Nomes.Length; I++)
     C. VerificarOcorrencia(Nomes[I], Horas[I]);}}}
```

? EXERCÍCIO 12: SEGUNDOS E CICLOS

```
using System;
namespace DE12
{public class InfHoras : EventArgs
 {public int Hora;
  public int Minuto;
  public int Segundo;
  public string Ciclos;

  public InfHoras(int H, int M, int S, int C)
  {Hora = H;
   Minuto = M;
   Segundo = S;
   Ciclos = " Ciclos "+C;}}

public class Relogio
 {private int Hora;
  private int Minuto;
  private int Segundo;
  public delegate void IncrementacaoSeg(object Relogio, InfHoras
  T);
  public event IncrementacaoSeg UmSegMais;
  public void Run()
  {int Ciclos=0;
  while (Ciclos<=9000000)
   {System.DateTime Agora = System.DateTime.Now;
    if (Agora.Second != this.Segundo)
     {InfHoras T = new InfHoras(Agora.Hour, Agora.Minute,
      Agora.Second, Ciclos);
      UmSegMais(this, T);}
      this.Segundo = Agora.Second;
      this.Minuto = Agora.Minute;
      this.Hora = Agora.Hour;
      Ciclos++;}}}

public class Consumidor1
 {public void Subscricao(Relogio Relagora)
 {Relagora.UmSegMais +=new
  Relogio.IncrementacaoSeg(NovoSegPara1);}
```

```
   public void NovoSegPara1(object Relagora, InfHoras ta)
    {Console.WriteLine("Consumidor 1-->{0}:{1}:{2}{3}", ta.Hora,
     ta.Minuto, ta.Segundo, ta.Ciclos);}}

public class Consumidor2
 {public void Subscricao(Relogio Relagora)
  {Relagora.UmSegMais +=new
   Relogio.IncrementacaoSeg(NovoSegPara2);}
  public void NovoSegPara2(object Relagora, InfHoras ta)
  {Console.WriteLine("Consumidor 2-->{0}:{1}:{2}", ta.Hora,
   ta.Minuto, ta.Segundo);}}

public class IncrementacaoDeSegundos
 {public static void Main()
  {Relogio Relagora = new Relogio();
   Consumidor1 C1 = new Consumidor1();
   C1.Subscricao(Relagora);
   Consumidor2 C2 = new Consumidor2();
   C2.Subscricao(Relagora);
   Relagora.Run();}}}
```