مـلاحـق الكـتاب

الـمـحـتـويات:

الملحق أ: أنماط القيمة (Value Types)

النمط	خصائص النمط
bool	بولياني
byte	عددى صحيح بدون إشارة
char	عددى صحيح بدون إشارة
decimal	عددى عشرى
double	عددى حقيقى مضاعف الدقة
enum	منشأ متعدد
float	عددى حقيقى
int	عددى صحيح بإشارة
long	عددى صحيح بإشارة
sbyte	عددى صحيح بإشارة
short	عددى صحيح بإشارة
struct	مبتكر
uint	عددى صحيح بدون إشارة
ulong	عددى صحيح بدون إشارة
ushort	عددى صحيح بدون إشارة

ملاحظة:

- ❖ مـن بـاب الاختصـار قد يطلق اسم النمط الحقيقى على النمط الحقيقى ذى الدقة المضاعفة double ، علاوة على النمط الحقيقى float.
- ❖ قد يطلق اسم النمط الصحيح على أى بيان من الأنماط Integral Types ، علاوة على النمط الصحيح int.

الملحق ب: القيم سابقة التعريف لأنماط القيمة (Default Value Types)

النمط	القيمة سابقة التعريف
bool	false
byte	0
char	'\0'
decimal	0.0M
double	0.0D
enum	هى القيمة الناتجة عن التعبير 0(E) حيث E هو اسم متغير المنشأ المتعدد.
float	0.0F
int	0
long	0L
sbyte	0
short	0
struct	هى القيمة الناتجة من شحن أنماط القيمة بالقيم الابتدائية وأنماط المرجع بالقيمة null.
uint	0

ulong	0
ushort	0

الملحق ج: أولويات المؤثرات (Operator Precedence)

تـبدأ أولوية المؤثرات فى هذا الجدول من أعلى إلى أسفل تحت عمود المؤثرات.

الفئة	المؤثرات
الابتدائية Primary	x.y f(x) a[x] x++ x -- new typeof checked unchecked
الأحادية Unary	+ - ! ~ ++x --x (T)x
الضرب والقسمة Multiplicative	* / %
الجمع والطرح Additive	+ -
الإزاحة Shift	<< >>
العلاقية ومؤثرات الاختبار Relational and type testing	< > <= >= is as
التساوى Equality	== !=
الضرب المنطقى Logical AND	&
الجمع المنطقى الاستثنائى Logical XOR	^
الجمع المنطقى Logical OR	|
الضرب المنطقى الشرطى Conditional AND	&&

الجمع المنطقى الشرطى Conditional OR	||
الشرطية Conditional	?:
التخصيص Assignment	= *= /= %= += -= <<= >>= &= ^= |=

الملحق د: التحويلات الضمنية للبيانات (Implicit Conversions)

من	إلى
sbyte	short, int, long, flo at, double, decimal
byte	short, ushort, int, uint, long, ulong, float, double, decimal
short	int, long, float, d ouble, decimal
ushort	int, uint, long, ulong, float, double, decimal
int	long, float, double, decimal
uint	long, ulong, float, double, decimal
long	float, double, decimal
char	ushort, int, uint, long, ulong, float, double, decimal
float	double
ulong	float, double, decimal

ملاحظات:

❖ قد يتسبب التحويل من الأنماط int أو uint أو long إلى النمط float أو مـن الـنمط long إلـى double فى نقص درجة الدقة (وليس قيمة العدد).

- ❖ لايوجد تحويل ضمنى إلى النمط char
- ❖ لايوجد تحويل ضمنى بين الأنماط الحقيقية والنمط العشري
- ❖ يجـوز التحويل من النمط int إلى الأنماط sbyte, byte, short, ushort, uint, ulong بشرط أن يتسع المخزن للعدد المحوّل.

الملحق ه:التحويلات الصريحة للبيانات (Explicit Conversions)

من	إلى
sbyte	byte, ushort, uint, ulong, char
byte	sbyte, char
short	sbyte, byte, ushort, uint, ulong, char
ushort	sbyte, byte, short, char
int	sbyte, byte, short, ushort, uint, ulong, char
uint	sbyte, byte, short, ushort, int, char
long	sbyte, byte, short, ushort, int, uint, ulong, char
ulong	sbyte, byte, short, ushort, int, uint, long, char
char	sbyte, byte, short
float	sbyte, byte, short, ushort, int, uint, long, ulong, char, decimal
double	sbyte, byte, short, ushort, int, uint, long, ulong, char, float, decimal
decimal	sbyte, byte, short, ushort, int, uint, long, ulong, char, float, double

ملاحظات:

- قـد تتسـبب الـتحويلات الصـريحة فـى نقص الدقة أو إلقاء الاستثناءات.
- عند التحويل من float أو double أو decimal إلى أحد الأنماط الصـحيحة فـإن القـيمة المحوّلـة يتم تقريبها إلى أقرب قيمة صحيحة. أما إذا كانت القيمة المحوّلة أكبر من سعة المخزن يتم إلقاء الاستثناء InvalidCastException .
- عـند الـتحويل مـن النمط مضاعف الدقة (double) إلى النمط الحقـيقى (float) فـإن القيمة العددية يتم تقريبها إلى أقرب عدد حقـيقى. أما إذا كانت القيمة المحوّلة أكبر من سعة المخزن فإن النتيجة تصبح صفراً أو مالانهاية.
- عـند الـتحويل من النمط العشرى (decimal) إلى أحد الأنماط الحقيقية (float أوdouble) فإن القيمة المحولة يتم تقريبها.
- عـند التحويل من الأنماط الحقيقة إلى النمط العشرى فإن العدد المحول قد يتم تقريبه إلى 28 خانة عشرية إذا تطلب الأمر. وقد تحدث النتائج الآتية بحسب قيمة العدد المحوّل:
  - إذا كـان العـدد أصغر من أن يمثل بالفورمات العشرية يصبح صفراً.
  - إذا كان العدد أكبر من اللازم أو مالانهاية أو ليس عدداً (NaN) فإن العملية تلقى الاستثناء InvalidCastException

الملحق والصور المختلفة للفورمات (Formatting Results)

اسم الفورمات	لبنة الفورمات	مثال	النتيجة
العملة Currency	C	Console.Write("{0:C}", 1.2); Console.Write("{0:C}", -1.2);	$1.20 ($1.20)
العشرية Decimal	D	Console.Write("{0:D5}", 123);	00123
العلمية Scientific	E	Console.Write("{0:E}", 12300000);	1.230000E+007
العلامة العشرية الثابتة Fixed-point	F	Console.Write("{0:F2}", 12); Console.Write("{0:F0}", 12);	12.00 12
العامة General	G	Console.Write("{0:G}", 1.2);	1.2
العددى Number	N	Console.Write("{0:N}", 1230000000);	1,230,000,000.00
السداسى عشر Hexadecimal	X	Console.Write("{0:X}", 123); Console.Write("{0:X}", 65535);	7B FFFF

حلول التمارين

الباب الثانى:

تدريب (2-1)

```
// Drill 2-1
// WriteLine and Write

using System;

class HelloWorld
{
  static void Main()
  {
    // Using WriteLine:
    Console.WriteLine("Hello World!");
    Console.WriteLine("Hello C# user!");

     // Using Write:
    Console.Write("Hello World!");
    Console.Write("Hello C# user!");
  }
}

/*
Output:
Hello World!
Hello C# user!
Hello World!Hello C# user!
*/
```

تدريب (2-2)

```
// Drill 2-2.cs
// Adding two integers

using System;

class MyClass
{
```

```
  static void Main()
  {
    int myInt=123;
    int yourInt=123;
    int sum = myInt + yourInt;

    Console.Write("The sum = "+ sum.ToString());
  }
}

/*
Output:
The sum = 246
*/
```

الباب الثالث:

تدريب (3-1)

```
// Drill 3-1.cs
// Decrement operator

using System;

class ArithmeticOperators
{
  public static void Main()
  {
    int x = 10;
    int y = 100;
    int z = y;

    y = y++ + x;
    Console.WriteLine(y);
    z = ++z + x;
    Console.WriteLine(z);
  }
}

/*
Output:
110
```

```
111
*/
```

تدريب (3-2)

```
// Drill 3-2.cs
// Get types 1

using System;

class MyTypes
{
  static void Main()
  {
        Console.WriteLine(123.GetType());
        Console.WriteLine(3.14.GetType());

  }
}

/*
Output:
System.Int32
System.Double
*/
```

تدريب (3-3)

```
// Drill 3-3.cs
// Get types 2

using System;

class MyTypes
{
  static void Main()
  {
    decimal myDc = 23.4M;
    float myFl = 23.4F;
    double myDb = 23.4;

    Console.Write("myDc ={0}  \nmyFl ={1}\nmyDb ={2}",
            myDc.GetType(), myFl.GetType(    ),
myDb.GetType());
```

```
  }
}

/*
Output:
System.Decimal
System.Single
System.Double
*/
```

تدريب (3-4)

```
// Drill 3-4.cs
// Get types 3

using System;

class MyClass
{
  static void Main()
  {
    Console.WriteLine(9223372036854775808L.GetType());
    Console.WriteLine(123UL.GetType());
    Console.WriteLine(4294967296L.GetType());
     Console.WriteLine(4294967290U.GetType());
  }
}

/*
Output:
System.UInt64
System.UInt64
System.Int64
System.UInt32
*/
```

تدريب (3-5)

```
// Drill3-5.cs
// Strings

using System;

class UnicodeChars
```

```
{
  static void Main()
  {
    string a = " \u0041";
     string b = " \u0042";
    string c = " \u0043";

    Console.WriteLine("{0}, {1} and {2} are the first three
letters.", a, b, c);
  }
}

/*
Output:
A, B and C are the first three letters.
*/
```

الباب الرابع

تدريب (4-1)

```
// Drill 4-1.cs
// Character Tester

using System;

public class CharTester
{
  public static void Main()
  {
    Console.Write("Please enter a character: ");
    char yourChar = (char) Console.Read();
      if (Char.IsLower(yourChar))
        Console.WriteLine("The letter {   0} is small.", yourChar);
      else if (Char.IsUpper(yourChar))
        Console.WriteLine("The letter {0} is capital.", yourChar);
      else
      Console.WriteLine("The character {0} is not alphabetic.",
                                yourChar);
  }
}
```

```
/*
Run 1:
Please enter a character: a
The letter a is small.

Run 2:
Please enter a character: A
The letter A is capital.

Run 3:
Please enter a character: 3
The character 3 is not alphabetic.

Run 4:
Please enter a character: #
The character # is not alphabetic.
*/
```

تدريب (4-2)

```
// Drill 4-2.cs
// Two dim arrays example

using System;

class JaggedClass
{
  static void Main ()
  {
  // Two dim array:
  string[,] grades = new string[2,4]
        { {"Pass","Good", "VeryGood", "Distinct"},
{"55%","65%", "75%", "85%"} };
     for (int j = 0; j <= 3; j++)
  {
    Console.Write("Grade={0} \t",grades[0,j]);
    Console.WriteLine("Score={0}",grades[1,j]);
   }
  }
}

/*
Output:
```

```
Grade=Pass     Score=55%
Grade=Good     Score=65%
Grade=VeryGood  Score=75%
Grade=Distinct  Score=85%
*/
```

الباب الخامس

تدريب (5-2)

```
// Drill 5-2.cs
// Private constructors

public class MyClass
{
  private MyClass() {}
  public static int companyName;
  public static int employmentDate;
}

public class MyOtherClass
{
  static void Main()
  {
  MyClas s myObject = new MyClass(); // not allowed in this class
  }
}

/*
Output:
error CS0122: 'MyClass.MyClass()' is inaccessible due to its
      protection level
*/
```

الباب السادس

تدريب (6-1)

```
// Drill6-1.cs
// using override and virtual

using System;
  cla ss Citizen
```

```
  {
    string idNumber = "111  -2345-H";
    string name = "Tarek M. Ashoor";

    public virtual void GetInformation()
    {
      Console.WriteLine("Name: {0}", name);
      Console.WriteLine("ID Card Number: {0}", idNumber);
    }
  }
  class Employee: Citizen
  {
    string companyName = "Technology Group Inc.";
    string companyID = "ENG  -RES-101-C";

    public override void GetInformation()
    {
      // Calling the base class GetPersonalInfo method:
       base.GetInformation();

      Console.WriteLine("  \nJob Information:");
      Console.WriteLine("Company Name: {0}", companyName);
      Console.WriteLine("Company ID: {0}", companyID);
    }
  }

class MainClass {
  public static void Mai  n()
  {
    Employee E = new Employee();
    E.GetInformation();
  }
}
/*
Output:
Citizen's Information:
Name: Hani M. Ashoor
ID Card Number: 111 -2345-H

Job Information:
Company Name: Technology Group Inc.
Company ID: ENG-RES-101-C
```

```
*/
```

تدريب (6-2)

```
// Drill 6-2.cs
// Abstract Classes

using System;

abstract class MyBaseClass
{
  // Fields:
  protected i    number = 100;
  protected string name = "Mohamed Aly";

  // Abstract method:
  public abstract void MyMethod();

  // Abs tract properties:
  public abstract int Number
  { get; }
  public abstract string Name
  { get; }
}

// Inheriting the class:
class MyDerivedClass: MyBaseClass
{
  // Overriding properties:
  public override int Number
  {
    get { return   number; }
  }
  public override string Name
  {
    get { return name; }
  }
  // Overriding the method:
  public override void MyMethod()
  {
    Console.WriteLine("Number = {0}", Number);
    Console.WriteLine("Name = {0}", Name);
  }
```

```
}

class MySecondDerivedClass: MyDerivedClass
{
  public override void MyMethod()
  {
    // Implentation
    Console.Write("Hello...");
    Console.WriteLine("again!");
  }
}

class MainClass
{
  public static void Main()
  {
    MyDerivedCl ass myObject1 = new MyDerivedClass();
    MySecondDerivedClass myOject2 = new
MySecondDerivedClass();
    myObject1.MyMethod();
    myOject2.MyMethod();
  }
}

/*
Output:
Number = 100
Name = Mohamed Aly
Hello...again!
*/
```

تدريب (6-3)

```
// Drill 6-3.cs
// Swap strings by reference

using System;

class MyClass
{
  static void Swap(ref string s1, ref string s2)
  {
    string temp = s1;
```

```
    s1 = s2;
    s2 = temp;
    Console.WriteLine("Inside the swap metho  d: " +
              "s1 = {0}, s2 = {1}", s1, s2);
  }
  public static void Main()
  {
    string s1 = "Mohamed";
    string s2 = "Aly";
    Console.WriteLine("Before swapping: "+
              "s1 = {0}, s2 = {1}      ", s1, s2);
    Swap(ref s1, ref s2);
    Console.WriteLine("After swapping: "+
              "s1 = {0}, s2 = {1}", s1, s2);
  }
}

/*
Output:
Before swapping: s1 = Mohamed s2 = Aly
Inside the swap method: s1 = Aly, s2 = Mohamed
After swapping: s1 = Aly, s2 = Mohamed
*/
```

تدريب (6-4)

```
// Drill 6-4.cs
// Overloading Example

using System;

public class MyClass
{
  public void MyMethod(out int x, out int y, out int z)
  {
    x = 1945;
  y = 1966;
  z = 1987;
  }

  public vo id MyMethod(ref int x, ref int y)
  {
    x++;
```

```
  y++;
  }
}

class MainClass
{
  public static void Main()
  {
  int d1, d2, d3;
  int m=100 , n=200;

  MyClass mc = new MyClass();

  mc.MyMethod(out d1, out d2, out d3);
  mc.MyMethod(re f m, ref n);

  Console.Write ("My dates are: {0}, {1}, {2}  \n", d1, d2 , d3);
  Console.Write ("My numbers are : {0}, {1}", m , n);
  }
}

/*
Output:
My dates are: 1945, 1966, 1987
My numbers are : 101, 201
*/
```

تدريب (6-5)

```
// Drill 6-5.cs
// params example

using System;

public class MyClass
{
  public void MyMethod(params object[] myObjArray)
  {
    for ( int i = 0 ; i < myObjArray.Length ; i++ )
      Console.WriteLine(myObjArray[i]);
    Console.W riteLine();
  }
}
```

```
class MainClass
{
  static void Main()
  {
    MyClass mc = new MyClass();
    mc.MyMethod(11, 22, 33);
    mc.MyMethod(33.65, 'A', "My original string");
  }
}

/*
Output:
11
22
33

33.65
A
My original string
*/
```

تدريب (6-6)

```
// Drill 6-6.cs
// Overloading operators

using System;

public class CompNum
{
  public int real;
  public int imag;

  // COnstructor:
  public CompNum(int r, int i)
  {
    real = r;
    imag = i;
  }

  // The overloaded operator:
```

```
  pu blic static CompNum operator+(CompNum c1, CompNum c2)
  {
    // Return the sum as a complex number:
    return new CompNum(c1.real + c2.real, c1.imag + c2.imag);
  }

  // Override ToString():
  public override string ToString()
  {
    return  (String.Format("{0} + {1}i", real, imag));
  }

  static void Main()
  {
    CompNum n1 = new CompNum (15, 33);
    CompNum n2 = new CompNum (10, 12);

    // Add two complex numbers using the overloaded + operator:
    CompNum sum = n1 + n2;

    // Display the objects:
    Console.WriteLine("Num1 = {0}", n1);
    Console.WriteLine("Num2 = {0}", n2);
    Console.WriteLine("Sum  = {0}",sum);
  }
}

/*
Output:
Num1 = 15 + 33i
Num2 = 10 + 12i
Sum  = 25 + 45i
*/
```

الباب السابع

تدريب (7-1)

```
// Drill 7-1.cs
// Struct properties

using System;
```

```
public struct Color
{
  // Fields:
  private int r;
  private  int g;
  private int b;

  // Constructor:
  Color(int r, int b, int g)
  {
   this.r = r;
   this.b = b  ;
   this.g = g;
  }

  // Properties:
  public int R
  {
    get { return r; }
    set { r = value; }
  }
  public int B
  {
    get { return b; }
    set { b = value; }
  }
  public int G
  {
    get { return g; }
    set { g =   value; }
  }
  // Override the method ToString():
  public override string ToString()
  {
    return (String.Format("Red = {0}, Green = {1}, Blue = {2}",
R, B, G));
  }
  static void Main()
  {
    // Declare objects:
    Color c1 = new Colo  r();
```

```
    Color c2 = new Color(100, 100, 0);
    // Print objects:
    Console.WriteLine("The first object:");
    Console.WriteLine("The colors are: {0}", c1);
    Console.WriteLine("The second object:");
    Console.WriteLine("The colors are:   {0}", c2);
  }
}

/*
Output:
The first object:
The colors are: Red = 0, Green = 0, Blue = 0
The second object:
The colors are: Red = 100, Green = 100, Blue = 0
*/
```

تدريب (7-2)

```
// Drill 7-2.cs
// Passing struct & class objects

using System;

class MyClass
{
  public string classField;
}

struct MyStruct
{
  public string structField;
}

class MainClass
{
  public static void MyMethod1(MyStruct s)
  {
     s.structField = "New Value";
  }
  public static void MyMethod2(MyClass c)
  {
    c.classField = "New Value";
```

```
    }

    static void Main()
    {
      // Create class and struct objects:
      MyStruct sObj = new MyStruct();
      MyClass  cObj = new MyClass();

      // Initialize the values of struct and class objects:
      sObj.structField = "Original Value";
      cObj.classField = "Original Value";

      // Print results:
      Console.WriteLine("Results before call  ing methods:");
      Console.WriteLine("Struct member = {0}", sObj.structField);
      Console.WriteLine("Class member = {0}  \n", cObj.classField);

      // Change the values through methods:
      MyMethod1(sObj);
      MyMethod2(cObj);

       // Print results:
      Console.WriteLine("Results after calling methods:");
      Console.WriteLine("Struct member = {0}", sObj.structField);
      Console.WriteLine("Class member = {0}", cObj.classField);
    }
}

/*
Output:
Results before callin g methods:
Struct member = Original Value
Class member = Original Value

Results after calling methods:
Struct member = Original Value
Class member = New Value
*/
```

تدريب (7-3)

```
// Drill 7-3.cs
// Using PInvoke
```

```
using System;

using System.Runtime.InteropServices;

class PlatformInvokeTest
{
   [DllImport("msvcrt.dll")]
   static extern int puts(string c);

   static void Main()
   {
     string s = "This is an example of platform invoke.";
     puts(s);
   }
}

/*
Output:
This is an example of platf orm invoke.
*/
```

تدريب (7-4)

```
// Drill 7-4.cs
// Simulating unions

using System;
using System.Runtime.InteropServices;

[StructLayout(LayoutKind.Explicit)]
struct UnionStruct
{
   [FieldOffset(0)]
   public long longVar;
   [FieldOffset(0)]
   public int byte1;
    [FieldOffset(4)]
   public int byte3;
}
class MyClass
{
  static void Main()
```

```
  {
    UnionStruct u = new UnionStruct();

    u.byte1 = 15;
    u.byte3 = 15;
    Console.WriteLine("The bytes of the long number:
{0:x}",u.longVar);
  }
}

/*
Output:
The bytes of the long number: f0000000f
*/
```

الباب الثامن

تدريب (8-1)

```
// Example 8-1.cs
// Explicit interface implementation example.

using System;

public interface ITemp1
{
  double Convert(double d);
}

public interface ITemp2
{
  double Convert(double d);
}

public class TempConverter: ITemp1, ITemp2
{

  double ITemp1.Convert(double d)
  {
    // Convert to Fahrenheit:
    return (d * 1.8) + 32;
  }
```

```
  double ITemp2.Convert(double d)
  {
    // Convert to Celsius:
    return (d   - 32)/ 1.8;
  }
}

class MyClass
{
  public st atic void Main()

  {
    // Create a class instance:
    TempConverter cObj = new TempConverter();

    // Create instances of interfaces
    // Create a From  -Celsius-to-Fahrenheit object:
    ITemp1 iCF = (ITemp1) cObj;
    // Create From  -Fahrenheit-to-Celsius object:
    ITemp2 iFC = (ITemp2) cObj;

    String display = @"Please select the converter
1. From Celsius to Fahrenheit.
2. From Fahrenheit to Celsius.
:";
    Console.Write(display);

  double F=0, C = 0;
  string sele ction = Console.ReadLine();
  switch(selection)
  {
    case "1":
      Console.Write("Please enter the Celsius temperature: ");
      C = Convert.ToDouble(Console.ReadLine());
      F = iCF.Convert(C);
      Console.WriteLine("Temperatu  re in Fahrenheit: {0:F2}",F);
      break;

    case "2":
      Console.Write("Please enter the Fahrenheit temperature: ");
      F = Convert.ToDouble(Console.ReadLine());
```

```
      C = iFC.Convert(F);
      Console.WriteLine("Temperature in C  elsius: {0:F2}",C);
      break;

    default:
      Console.WriteLine("Please select a converter.");
      break;
    }
  }
}

/*
Run #1:
Please select the converter
1. From Celsius to Fahrenheit.
2. From Fahrenheit to Celsius.
:1
Please enter the Celsius temperature: 0
Temperature in Fahrenheit: 32.00

Run #2:
Please select the converter
1. From Celsius to Fahrenheit.
2. From Fahrenheit to Celsius.
:2
Please enter the Fahrenheit temperature: 32
Temperature in Celsius: 0.00
*/
```

تدريب (8-2)

```
// Drill 8-2.cs
// The as operator

using System;

public class MyClass
{
  static void TestType(object o)
  {
   if (o as string != null)
     Console.WriteLine ( "The object   \"{0}\" is a string.", o);
   else
```

```
    Console.WriteLine ( "Th  e object \"{0}\" is not a string. It is
{1}.",

o, o.GetType());
  }

  static void Main()
  {
    object o1 = "Hello World!";
    object o2 = 123;
    object o3 = 12.34;
    TestType(o1);
    TestType(o2);
    TestType(o3);
  }
}

/* Output:

The object "Hello World!" is a string.
The object "123" is not a string. It is System.Int32.
The object "12.34" is not a string. It is System.Double.
*/
```

الباب التاسع

تدريب (9-1)

```
// Drill 9-1.cs
// Ordering exceptions

using System;

class MyClass
{
  static void Main()
  {
    int x = 0;
    int y = 10;
    try
    {
     int z = y/x;
```

```
    }
    // The most expected exception:
  catch  (DivideByZeroException e)
    {
      Console.WriteLine("Arithmetic Exception Handler: {0}", e);
    }

    // The most expected exception:
    catch (ArithmeticException e)
    {
      Console.WriteLine("Arithmetic Exception Handler:    {0}", e);
    }

    // Catch the general exception:
    catch (Exception e)
    {
      Console.WriteLine("General Exception Handler: {0}", e);
    }
    // Continue the program:
    Console.WriteLine("Program Continues...");
  }
}

/*
Output:
Arithmetic Exception Handler: System.DivideByZeroException:
Attempted to divide by zero.
  at MyClass.Main()
Program Continues...
*/
```

تدريب (9-2)

```
// Drill9-2.cs
// Reading a text file

using System;
using System.IO;

class MyClass
{
  static void  Main()
  {
```

```
        StreamReader myFile = null;
       try
       {
      myFile = new StreamReader("test.txt");
      string myString = myFile.ReadToEnd();
            Console.WriteLine(myString);
            myFile.Clos    e();
       }
       catch (FileNotFoundException)
       {
           Console.WriteLine("The file you are trying to open is
not found.");
       }
    catch
    {
        Console.WriteLine("General catch statement.");
    }
  }
}
/*
Output:
The file you are trying to open is not found.
*/
```

تدريب (9-3)

```
// Drill 9-3.cs
// Processing files using finally

using System;
using System.IO;

class MyClass
{
  static void Main()
  {
    int counter = 0;
    string line;
    StreamReader file = null;
    try
    {
      file = new StreamReader("test.txt");
      while((line = file.ReadLine()) != null)
```

```
        {
          Console.WriteLine (line);
          counter++;
        }
      }
      catch (Fil  eNotFoundException)
      {
        Console.WriteLine("The file you are trying to open is not
found.");
      }
      catch
      {
        Console.WriteLine("General catch statement.");
      }
      finally
      {
   if (file != null)
        file  .Close();
      }
   }
}
/*
Output:
The file you are trying to open is not found.
*/
```

الباب العاشر

تدريب (10-1)

```
// Drill 10-1.cs
// Using gelegates

using System;

// Declare a delegate:
delegate void MyDelegate(int n, string s);

class MainClass
{
  stati c void Main()
  {
```

```
    // Instantiate the class:
    MyClass obj = new MyClass();

    // Intantiate the delegate:
    MyDelegate d = new MyDelegate(obj.MyMethod);

    // Invoke the delegate:
  d(255, "Moustafa Amin");
  }
}

class MyClass
{
// The encapsulated Method:
  public void MyMethod(int id, string name)
  {
    Console.WriteLine("ID = {0}  \nName = {1}", id, name);
  }
}

/*
Output:
ID = 425
Name = Moustafa Amin
*/
```

تدريب (10-2)

```
// Drill 10-2.cs
// Adding and removing delegates

using System;

// Declare a delegate:
delegate void MyDelegate();

class MyClass
{
  public static void MyMethod1()
  {
    Console.Write("MyMethod #1 ");
  }
```

```
   public static void  MyMethod2()
   {
      Console.Write("MyMethod #2 ");
   }

   public static void Main()
   {
   // Declare delegate object and reference MyMehod1:
      MyDelegate d1 = new MyDelegate(MyClass.MyMethod1);

   // Declare delegate object and refer  ence MyMehod2:
      MyDelegate d2 = new MyDelegate(MyClass.MyMethod2);

      // Declare delegate d3 by adding d1 and d2. This will invoke
both

MyMehod1 and MyMehod2:
      MyDelegate d3 = d1 + d2;

      // Declare delegate d4 by removing d1 fr   om d3. This will
invoke

MyMehod2 only:
      MyDelegate d4 = d3   - d1;

      Console.Write("Invoking d1, referencing ");
      d1();
      Console.Write(" \nInvoking d2, referencing ");
      d2();
      Console.Write(" \nInvoking d3, referenci ng ");
      d3();
      Console.Write(" \nInvoking d4, referencing ");
      d4();
   }
}

/*
Output:
Invoking d1, referencing MyMethod #1
Invoking d2, referencing MyMethod #2
Invoking d3, referencing MyMethod #1 MyMethod #2
Invoking d4, referencing  MyMethod #2
*/
```

فـــى لقائنا بلغة سى# قد تعرفنا بأغلب ملامحها التى تفيدنا فى بناء التطبيقات. أما الملامح التالية فلم نناقشها فى هذا الكتاب:

- الكود غير المأمون (Unsafe code)
- الخيوط (Threads)

كما تعرفنا من خلال الأمثلة بالكثير من فصائل وأساليب دوت.نت ولكن تغطيتها بالكامل يحتاج لقاء منفصل.

أما تطبيقات الاستوديو التى لم نناقشها بالتفصيل فى هذا الكتاب فهى:

- تطبيقات النوافذ (Windows Forms)
- تطبيقات الوب (Web Forms)
- خدمات الوب (Web Services)

وتحتاج هذه الموضوعات إلى لقاءات منفصلة.

كما أن الطراز الجديد من الإستوديو المرئى (Visual Studio 2005) سوف يقدم العديد من الملامح الجديدة التى نمت الحاجة إليها بناء على استخدام المبرمجين للغة وتجربتها فى التطبيقات خلال العام الأول من صدورها.

وسوف نلتقى تباعاً بهذه الموضوعات فى الكتب القادمة.

مع أطيب التمنيات.

نحن فى حاجة إلى رأيك

لموافاتنا برأيك فيما نشر أو فيما ترغب أن تراه فى القريب العاجل ، برجاء الاتصال بالناشر تليفونيا (637-9863 أو 638-9372) أو كتابةً على العنوان الموضح على الغلاف ، أو بإرسال بريد إليكترونى (إيميل) إلى العنوان: Sam@Abolrous.com