الباب الثانى

الجولة الأولى فى لغة سى#

محتويات الباب

- البرنامج الأول: أهلاً أيها العالم (Hello World!)
- ترجمة وتشغيل البرنامج (Compilation and Executing)
- التعليقات (Comments)
- إعلان الفصيلة (Class Declaration)
- الأسلوب الرئيسى Main
- استخدام أساليب دوت.نت للطباعة (WriteLine, Write)
- استخدام التوجيهات (Directives)
- استخدام المتغيرات المحلية (Local Variables)
- معمار برنامج سى#
- تأهيل الأسماء (Qualifying Names)
- العرف الشائع فى كتابة البرامج.

البرنامج الأول: أهلاً أيها العالم (Hello World!)

من الشائع ــ بعد ظهور لغة سى ــ أن يكون أول برنامج نكتبه هو البرنامج الذى يطبع عبارة "Hello World!" على الشاشة. وهذا هو نص البرنامج:

مثال (2-1)

```
// Example 2-1.cs
// The first program in C#

class HelloWorld
{
  static void Main()
  {
    System.Console.WriteLine("He llo World!");
  }
}
```

ترجمة وتشغيل البرنامج

اكتب هذا النص فى برنامج النوتة (Notepad) أو ما يماثله ، واحفظه تحت اسم ما مثل:

FirstProgram.cs

ثم ترجم البرنامج باستخدام الأمر التالى فى بيئة الأوامر:

csc FirstProgram.cs

وبهذا فإنك تحصل على الملف التنفيذى الآتى:

FirstProgram.exe

قم بتشغيل هذا البرنامج بإدخال اسمه فى بيئة الأوامر:

FirstProgram

وسوف ترى العبارة "Hello World!" مطبوعة على الشاشة.

ملاحظـــة: إذا كنت تستخدم البيئة المجمعة للإستوديو المرئى (Visual Studio) فيمكـــنك كـــتابة الـــبرنامج وترجمته كبرنامج "كونصول" (Console) كما هو موضح بالباب الحادى عشر.

وفيما يلى نقدم أهم الملاحظات على البرنامج.

ســـوف نكتـــب الكلمات المفتاحية (Keywords) للغة سى# أو مكتبة دوت.نت بالبنط الأسود (الثقيل) عند ظهورها فى النص لأول مرة.

التعليقات (Comments)

يحتوى كل من السطر الأول والسطر الثانى على تعليقات:

```
// Program 2-1.cs
// The first program in C#
```

والعلامـــتان "//" فى بداية السطر تحوّل السطر كله إلى تعليق ، فلا يلتفت إليه المترجم. وهذا يماثل التعليقات بلغة سى++.

كمـــا يجوز أن تأتى علامات التعليق فى أى مكان فى السطر فتحول ما بعدها ، وحتى نهاية السطر ، إلى تعليق ، مثل:

```
class HelloWorld      // This is a class declaration
```

أمـــا علامات التعليق التقليدية فى لغة سى "/*" و "*/" فهى تستخدم لتحويل مجموعة من السطور إلى تعليق ، مثل:

```
/* This is a comment block:
```

```
This is another comment inside the comment block.
This is a third comment. */
```

إعلان الفصيلة (Class Declaration)

الصـــفة العامة لبرنامج سى# أنه يقع بالكامل بداخل فصيلة (أو منشأ ــ كما سيلى) . والفصيلة المستخدمة هنا قد منحناها الاسم "HelloWorld" ونرى ذلك فى السطر الثانى:

```
class HelloWorld
```

ولـــك أن تستخدم ما تشاء من الأسماء ، المهم أن يكون الاسم كلمة واحـــدة لا تتخللها مسافات بيضاء. ويأتى الاسم مباشرة بعد كلمة **class** وهـــى تكتـــب بالحروف الصغيرة (small) ، وهذه قاعدة ملزمة قى لغة ســـى# لأن حالـــة الحروف لا يجوز تغييرها ، تماماً كما فى لغة سى وســـى++. ويتبع اسم الفصيلة محتوياتها التى تكتب بداخل القوسين {}. وقـــد تحتوى الفصيلة على الحقول (Fields) والأساليب (Methods) (التى يطلق عليها أيضاً اسم الدوال الأعضاء "Function members"). وفى هذا الـــبرنامج اســـتخدمنا أسلوباً واحدا بالاسم **Main** ، وهو أساسى فى كل بـــرامج ســـى#. ونلاحـــظ أن جســـم البرنامج كله مكتوب بين قوسي الفصيلة.

ويطلق على كل مـــا يأتى بين هذين القوسين اسم البلوك (Block).

أما جسم البرنامج فقد يتخلله أى عدد من البلوكات ، وسوف نعلق على ذلك بالتفصيل.

الأسلوب الرئيسى (The Main Method)

يحتوى كل برنامج على أسلوب رئيسى يسمى "Main" وهو يبدأ بالحرف M الكبير. وهذا الأسلوب يناظر الدالة الرئيسية "main" فى لغة سى++. ولا فرق فى المضمون بين الدالة فى لغة سى وبين الأسلوب فى لغة سى# ، ولكننا سوف نلتزم بالاسم الذى اختاره مصمّمو اللغة.

ولا يمكنك ترجمة أى برنامج إلى ملف تنفيذى (.exe) ما لم يحتو على الأسلوب الرئيسى Main.

ويسبق الأسلوب الرئيسى دائما كلمة static التى تحدد نوع الأسلوب، ولنا معها لقاء منفصل.

ومن الجائز أن يكون الأسلوب Main من النمط الخالى من الرصيد، أى تسبقه كلمة **void** ، كما فى هذا المثال ، أى:

```
static void Main()
{
}
```

ومن الجائز أن يكون الأسلوب من النمط الصحيح ، أى تسبقه كلمة **int** ، وفى هذه الحال فإنه يرجع قيمة عددية باستخدام الكلمة المفتاحية **return** ، أى:

```
static int Main()
{
  return 0;
}
```

كما أن الأسلوب Main يأخذ صوراً أخرى إذا رغبنا تشغيل البرنامج باستخدام بعض البارامترات. وسوف نعرض ذلك فى فقرة مستقلة.

استخدام أساليب دوت.نت للطباعة (Write, WriteLine)

كما نرى فى المثال أن وظيفة الدالة الرئيسية تنحصر فى طباعة النص "Hello world!" على الشاشة باستخدام الأسلوب **WriteLine** الذى لا ينتمى إلـــى لغة سى# ولكنه ينتمى إلى دوت.نت. ولذلك فقد لزم تعريف مكان وجـــود هـــذا الأسلوب باستخدام اسم الفصيلة "**Console**" و حيز الاسم (namespace) الـــذى يحـــتوى على هذه الفصيلة "**System**". وبهذا يصبح الاسم الكامل لهذا الأسلوب هو:

```
System.Console.WriteLine()
```

ويســـمى الاســـم الكامل لأى عنصر من عناصر اللغة بالاسم المؤهل (Qualified name).

ولكـــى تطـــبع أى حرفى فإنك تضعه بين علامتى اقتباس بداخل قوسى الأسلوب:

```
System.Console.WriteLine("Hello World!");
```

وتنـــتهى عـــبارات لغـــة سى# بعلامة الفاصلة المنقوطة حيث أنها هى الوسيلة التى يتعرف بها المترجم على نهاية العبارة.

ويمكـــنك أن تختار ما بين الأسلوب **Write** أو WriteLine فكلاهما يؤدى نفـــس العمل ، فيما عدا أن الأسلوب WriteLine ينتقل بك إلى سطر جديد بعـــد طباعة الحرفى. ولو أنك أضفت عبارة طباعة ثانية إلى البرنامج فإنها سوف تظهر على سطر مستقل.

تدريب (1-2)

أضـــف عبارة طباعة جديدة إلى المثال السابق لتطبع حرفى جديد مثل "

! Hello C# user" وتحقق من النتيجة ، أى:

```
Hello World!
Hello C# user!
```

عـندئذ اسـتبدل الأسلوب WriteLine بالأسلوب Write فى نفس البرنامج وتحقق من أن العبارتين سوف تظهران على نفس السطر ، أى:

```
Hello World!Hello C# user!
```

هـل لاحظت أن مفردات دوت.نت تختلف عن مفردات لغة سـى# فـى طريقة الكتابة؟ إن مفردات دوت.نت ، سواء كانت أسماء أسـاليب أو فصـائل فإنهـا تكتـب بالحروف الصغيرة والكبيرة (مثل WriteLine) بخـلاف مفـردات لغـة سـى# التى تكتب دائما بالحروف الصغيرة (باستثناء الأسلوب Main).

استخدام التوجيهات (Directives)

لعلك تقول لنفسك أن استخدام أسلوب من أساليب دوت.نت عملية "شاقة" تتطلـب الكثـير من الكتابة لتحقيق هدف بسيط مثل طباعة حرفى على الشاشـة ، فما بالك لو أردت استخدام أساليب أخرى؟ هناك حل. يمكنك فى بداية البرنامج أن تضع توجيهاً بالصورة الآتية:

```
using System;
```

وينـــتهى التوجـــيه بفاصلة منقوطة ، تماماً مثل عبارات اللغة. إن هذا التوجيه يخبر المترجم بأنك سوف تستخدم حيز الاسم System ولذلك فإنه عندما لا يتعرف على كلمة ما فى برنامجك فإنه يبحث عنها مباشرة فى الحيز System وفى هذه الحالة يمكنك أن تكتب عبارة الطباعة كالآتى:

```
Console.WriteLine("Hello World!");
```

ولكـــنك للأسف لا تستطيع أن تستغنى عن كلمة Console لأنها تمثل اسم الفصيلة التى تحتوى على الأسلوب.

وعندما تتقدم فى كتابة البرامج فإنك سوف تستخدم فصائل أخرى تنتمى إلى حيزات مختلفة ، وسوف تتحقق عندئذ من فائدة التوجيهات.

وهذه هى الصورة النهائية للبرنامج بعد إضافة التوجيه.

مثال (2-2)

```
// Program 2-2.cs
// The second program in C#

using System;
class HelloWorld
{
  static void Main()
  {
    Console.WriteLine("Hello World!");
  }
}
```

استخدام المتغيرات المحلية (Local Variables)

بـــدلاً من استخدام أسلوب الطباعة فى طباعة الحرفى "Hello World!" مباشـــرة ، يمكنك أن تختزن الحرفى فى متغير حرفى من النمط **string** كالآتى:

```
string myString = "Hello World!";
```

ثم تستخدم أسلوب الطباعة لطباعة المتغير كالآتى:

```
Console.WriteLine(myString);
```

ومن الجدير بالذكر أن المتغيرات لا تستخدم إلاّ بداخل الأساليب ولذلك فإنه يطلق عليها المتغيرات المحلية ، أى أن الأسلوب Main سوف يصبح كالآتى:

```
static void Main()
{
  string myString = "Hello World!";
  Console.WriteLine(myString);
}
```

كما يجوز استخدام عبارة الطباعة لطباعة رسالة معينة بجانب محتويات المتغير كالآتى:

```
Console.WriteLine("The string is: "+ myString);
```

وهذا هو النص الجديد للبرنامج.

مثال (2-3)

```
// Example 2-3.cs
// Local variables.

using System;
class MyClass
```

```
{
        static void Main()
        {
                string myString = "Hello World!";
                Console.WriteLine("The string is: "+ mySt ring);
        }
}
```

وبالطـــبع فإن اللغة تحتوى على مجموعة كبيرة من أنماط المتغيرات العدديـــة ســـوف نتعرض لها جميعاً فى حينها. وبصفة عامة يمكنك أن تعلن عن متغير عددى صحيح (integer) كالآنى:

```
int myInt;
```

ومن الجدير بالذكر أنه يمكنك الإعلان عن المتغير فى أى مكان بداخل الأســـلوب. كمـــا يمكـــنك أن تشحن المتغير بقيمة ابتدائية مع الإعلان كالآتى:

```
int myInt = 0;
```

وبصفة عامة فإن لغة سى# لن تسمح باستخدام أى متغير غير مشحون بقيمة ابتدائية.

وكما هو الحال مع سائر اللغات الأخرى فإن جمع قيم المتغيرات العددية يتم باستخدام مؤثر الجمع (+) كالمثال الآتى:

```
int sum = myInt + yourInt;
```

ولطباعة محتويات المتغير العددى استخدم نفس أسلوب الطباعة:

```
Console.WriteLine(myInt);
```

ولكـــى تطـــبع رسالة حرفية مع المتغير ، يمكنك استخدام العبارة الآتية كمثال:

```
Console.WriteLine("My integer = " + myInt.ToString());
```

أما الأسلوب **ToString()** الذى ظهر هنا فهو من أساليب مكتبة دوت.نت ، وهو يستخدم فى تحويل البيانات إلى الصورة الحرفية. كما يجوز استخدام الصورة الآتية من أسلوب الطباعة التى تحتوى ضمنياً على أسلوب التحويل ToString():

```
Console.WriteLine("My integer = {0}", myInt);
```

وفى هذه الصورة فإن الرمز {0} يستبدل بقيمة المتغير myInt. ولو أردت أن تطبع أكثر من متغير واحد فإنك تستخدم الرموز {0} و {1} وهكذا. أى:

```
Console.WriteLine("My integer = {0}, Your integer = {1}", myInt, yourInt);
```

وكما نرى أن هذه الطريقة مختصرة عن الطريقة الأولى.

تدريب (2-2)

اكتب برنامجاً بلغة سى# لجمع عددين صحيحين (int) وطباعة الناتج مع رسالة مناسبة.

معمار برنامج سى#

يتكون برنامج سى# من ملف أو أكثر. وقد يحتوى الملف الواحد على أى من العناصر الآتية (سوف يلى شرح العناصر المختلفة ونحن نتقدم فى فصول الكتاب):

- التوجيهات (Directives)

- حيز اسم (Namespace) أو أكثر ، ويجوز أن يحتوى حيز الاسم بداخله على العناصر الأخرى علاوة على حيزات أسماء أخرى ، وهو اختيارى.
- الفصائل (Classes)
- المنشئآت (Structs)
- الوصلات البينية (Interfaces)
- المناديب (Delegates)
- القوائم (Enumerations)
- الأسلوب Main بداخل فصيلة أو منشأ ، ووجوده إجبارى. وفى حالة إذا كان البرنامج مكوناً من أكثر من ملف فإنه يلزم وجوده فى ملف واحد فقط.

وفى المثال التالى نرى برنامجاً يحتوى على جميع هذه العناصر. وبالرغم من أن البرنامج فارغ من المضمون ، لكنه يمكن ترجمته وتشغيله بنجاح لأنه يتبع قواعد اللغة.

مثال (4-2)

```
// Example 2-4.cs

using System;  // توجيه

namespace Namespace1  // حيز اسم
{
  class Class1  // فصيلة
  {
```

```
  }
  struct Struct1  //  منشأ
  {
  }
  interface Interface1  //  وصلة بينية
  {
  }
  delegate int Delegate1( );  //  وفد
  enum Enum1  //  منشأ متعدد
  {
  }
  namespace Namespace2  //  حيز اسم
  {
  }
  class Class2  //  فصيلة
  {
    static void Main()  //  الأسلوب الرئيسى
    {
    }
  }
}
```

تأهيل الأسماء (Qualifying Names)

لا بأس فى مجال الحياة اليومية أن يتشابه بعض الأشخاص فى الاسم الأول مـــثل "محمـــد". ولكنك لو استخدمت الاسم الثلاثى أو اسم العائلة فســـوف يختفى اللبس بين الأسماء. وفى مجال البرمجة نطلق على هذه العملية "تأهيل الأسماء". وتأهيل الأسماء المستخدمة فى البرنامج يجعل كل عنصر من عناصر البرنامج متفرداً حتى لو تشابهت الأسماء. فعلى

سبيل المثال يحتوى البرنامج التالى على فصيلتين كل منهما تحمل الاسم MyC2 ومـــع ذلك فإن استخدام التأهيل للاسم يجنبنا الخلط بين الفصيلتين حيـــث أنـــه يحـــدد تبعية الفصيلة. فباستخدام الاسم المؤهل لكل فصيلة نحصل على الاسمين التاليين:

```
MyNS1.MyC1.MyC2
MyNS1.MyNS2.MyC2
```

وهذا هو البرنامج:

مثال (2-5)

```
// Example 2-5.cs
// Qualifying names

namespace MyNS1                  // MyNS1
{
  class MyC1                     // MyNS1.MyC1
  {
    class MyC2                   // MyNS1.MyC1.MyC2 الاسم المؤهل:
    {
    }
  }
  namespace M yNS2               // MyNS1.MyNS2
  {
    class MyC2                   // MyNS1.MyNS2.MyC2 الاسم المؤهل:
    {
    }
  }
}
namespace MyNS3                  // MyNS3
{
  class MyC3                     // MyNS3.MyC3
  {
    static void Main()
    {
    }
  }
}
```

نـــرى فـــى البرنامج السابق أمام كل عنصر من العناصر الاسم الكامل (المؤهـــل) لهـــذا العنصـــر. وتحديد تبعية أسماء العنصر فى البرنامج السابق يتم كالآتى:

- الحـــيز MyNS1 والحـــيز MyNS3 يتبعان الحيز العام للأسماء وهو بمـــثابة الجذر الذى تتفرع منه كل الفروع. لذلك فإن الاسم المؤهل لكل منهما هو نفسه اسم الحيز.
- الحـــيز MyNS2 يقـــع بداخل الحيز MyNS1 أى أن اسمه المؤهل هو MyNS1.MyNS2
- الفصـــيلة MyC1 تقع بداخل الحيز MyNS1 وبذلك فإن اسمها المؤهل هو MyNS1.MyC1
- الاســـم المؤهـــل للفصيلة MyC2 التى تقع بداخل الفصيلة MyC1 هو MyNS1.MyC1.MyC2
- الاســـم المؤهـــل للفصيلة MyC2 التى تقع بداخل الحيز MyNS2 هو MyNS1.MyNS2.MyC2

العرف الشائع فى كتابة البرامج

بعـــيداً عـــن القواعد الملزمة ، فإن هناك عرف شائع لكتابة البرامج ، والعـــرف هو مجرد توصيات تجعل برنامجك سهل القراءة. أما بالنسبة لمترجم اللغة فلا أهمية للطريقة التى تكتب بها البرنامج.

- للتفريق بين الأسماء الخاصة بلغة سى# أو التابعة لمكتبة دوت.نت وبين الأسماء المبتكرة التى تمنحها لفصائلك أو متغيراتك ابدأ الاسم بكلمة مميزة مثل my أو your كالأمثلة الآتية: myVariable ، yourAccountDetails ، MyClass.

- أسماء الأنماط (الفصائل والمنشئآت إلخ) والأساليب تبدأ أسماؤها بحرف كبير (Capital) مثل Class1. وإذا كان الاسم مكوناً من أكثر من كلمة ، استخدم الحروف الكبيرة فى بداية كل كلمة مثل: YourNumberClass ، MyMainMethod.

- من المفضل أن تبدأ أسماء الوصلات البينية بالحرف I مثل IMyInterfaceA.

- أسماء المتغيرات المحلية تبدأ بحرف صغير مثل: myVariable ، yourVariable.

- من المفضل أن يكون الاسم ذا معنى مثل MyAccountNumber ، YourFamilyTree. أما الأسماء مثل x ، y فهى أسماء لا تدل على مضمون ما ومن المفضل تجنبها (إلاّ إذا كانت تناسب المضمون كما فى برامج الرياضة).

- من الشائع أيضاً أن يبدأ القوس الأيسر على سطر مستقل تحت اسم النمط (مثل كلمة class) أو الأسلوب مباشرة ، ويأتى القوس الأيمن فى نفس المكان على سطر مستقل ، أى:

```
class HelloWorld
{
  جسم الفصيلة //
}
```

كمـــا نلاحظ أن الكتابة تبدأ على بعد مناسب مثل ثلاث مسافات (أو بياضة "tab") بعد مكان القوس الأيسر.

- اتـــرك سطرا خاليا بعد كل عملية مميزة. على سبيل المثال فإنه بعد فقـــرة الإعـــلان عن متغيراتك فى البرنامج ، عليك أن تترك سطرا خالـــياً لتمييز الإعلانات عن عمليات المعالجة. حتى بداخل عمليات المعالجـــة فإنـــه من الأفضل ترك سطر خال بعد كل عملية فرعية (حلقـــة تكرارية ، عبارة شرطية ، إلخ). والغرض من هذه القواعد أن يكـــون البرنامج سهل القراءة والتعديل إذا رجعت إليه فى وقت لاحق.

تذكر هذه المصطلحات

لغة سى# (سى شارب)	CSharp / C#
لغة سى شارب المرئية	Visual C#
البرمجة الموجهة نحو الأهداف	Object Oriented Programming
حيز الاسم	Namespace
حقل	Field

Method	أسلوب
Main method	الأسلوب الرئيسى
Variable	متغير
Local variables	المتغيرات المحلية
Directives	التوجيهات
Qualifying names	تأهيل الأسماء
Building	البناء (الترجمة والربط)
Compilation	الترجمة
Linking	الربط
Console Application	تطبيقات الكونصول
Declaration	إعلان
Comments	التعليقات
IDE (Integrated Development Environment)	البيئة المجمعة