

الباب الثاني

الجولة الأولى في لغة سي #

## 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 — بعد ظهور لغة سي — أن يكون أول برنامج نكتبه هو البرنامج الذي 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("Hello World!");
    }
}
```

### ترجمة وتشغيل البرنامج

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

FirstProgram.cs

ثم ترجم البرنامج باستخدام الأمر التالي في بيئة الأوامر:

csc FirstProgram.cs

وبهذا فإنك تحصل على الملف التنفيذي الآتي:

FirstProgram.exe

قم بتشغيل هذا البرنامج بإدخال اسمه في بيئة الأوامر:

FirstProgram

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

**ملاحظة:** إذا كنت تستخدم البيئة المدمجة للإستوديو المرئي (Visual Studio) فيمكنك كتابة البرنامج وترجمته كبرنامج "كونصول" (Console) كما هو موضح بالباب الحادي عشر.

وفيما يلي نقدم أهم الملاحظات على البرنامج.



سوف نكتب الكلمات المفتاحية (Keywords) للغة سي# أو مكتبة دوت.نت بالبنط الأسود (الثقيل) عند ظهورها في النص لأول مرة.

### التعليقات (Comments)

يحتوي كل من السطر الأول والسطر الثاني على تعليقات:

```
// Program 2-1.cs
```

```
// The first program in C#
```

والعلامتان "//" في بداية السطر تحول السطر كله إلى تعليق ، فلا يلتفت إليه المترجم. وهذا يماثل التعليقات بلغة سي++.

كما يجوز أن تأتي علامات التعليق في أى مكان في السطر فتحول

ما بعدها ، وحتى نهاية السطر ، إلى تعليق ، مثل:

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

أما علامات التعليق التقليدية في لغة سي "/" و "\*" فهي 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)

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

```
class HelloWorld
```

ولك أن تستخدم ما تشاء من الأسماء ، المهم أن يكون الاسم كلمة واحدة لا تتخللها مسافات بيضاء. ويأتى الاسم مباشرة بعد كلمة `class` وهى تكتب بالحروف الصغيرة (small) ، وهذه قاعدة ملزمة فى لغة سى# لأن حالة الحروف لا يجوز تغييرها ، تماماً كما فى لغة سى++ . ويتبع اسم الفصيلة محتوياتها التى تكتب بداخل القوسين { } . وقد تحتوى الفصيلة على الحقول (Fields) والأساليب (Methods) (التى يطلق عليها أيضاً اسم الدوال الأعضاء "Function members"). وفى هذا البرنامج استخدمنا أسلوباً واحداً بالاسم `Main` ، وهو أساسى فى كل برامج سى#. ونلاحظ أن جسم البرنامج كله مكتوب بين قوسى الفصيلة.

ويطلق على كل ما 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 مصممو اللغة.

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

ويسبق الأسلوب الرئيسى دائماً 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 رغبتنا 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 تتحصر فى طباعة النص "Hello world!" على الشاشة باستخدام الأسلوب **WriteLine** الذى لا ينتمى إلى لغة سي# ولكنه ينتمى إلى دوت.نت. ولذلك فقد لزم تعريف مكان وجود هذا الأسلوب باستخدام اسم الفصيلة "Console" و حيز الاسم (namespace) الذى يحتوى على هذه الفصيلة "System". وبهذا يصبح الاسم الكامل لهذا الأسلوب هو:

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

ويسمى الاسم الكامل لأى عنصر من عناصر اللغة بالاسم المؤهل (Qualified name).

ولكى تطبع أى حرفى فإنك تضعه بين علامتى اقتباس بداخل قوسى الأسلوب:

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

وتنتهى عبارات لغة سي# بعلامة الفاصلة المنقوطة حيث أنها هى الوسيلة التى يتعرف بها المترجم على نهاية العبارة.

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

## تدريب (2-1)

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

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

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

عندئذ استبدل الأسلوب WriteLine بالأسلوب Write فى نفس البرنامج وتحقق من أن العبارتين سوف تظهران على نفس السطر ، أى:  
Hello World!Hello C# user!



هل لاحظت أن مفردات دوت.نت تختلف عن مفردات لغة سى# فى طريقة الكتابة؟ إن مفردات دوت.نت ، سواء كانت أسماء أساليب أو فواصل فإنها تكتب بالحروف الصغيرة والكبيرة (مثل WriteLine) بخلاف مفردات لغة سى# التى تكتب دائما بالحروف الصغيرة (باستثناء الأسلوب Main).

## استخدام التوجيهات (Directives)

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

```
using System;
```

وينتهي التوجيه 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 في برنامجك فإنه يبحث عنها مباشرة في الحيز System وفي هذه الحالة يمكنك أن تكتب عبارة الطباعة كالاتي:

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

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

وعندما تتقدم في كتابة البرامج فإنك سوف تستخدم فواصل أخرى تنتمي إلى حيزات مختلفة ، وسوف تتحقق عندئذ من فائدة التوجيهات. وهذه هي 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)

بدلاً من استخدام أسلوب الطباعة فى طباعة الحرفى "Hello World!" مباشرة ، يمكنك أن تختزن الحرفى فى متغير حرفى من النمط **string** كالتالى:

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

ثم تستخدم أسلوب الطباعة لطباعة المتغير كالتالى:

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

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

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

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

```
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: " + myString);
    }
}
```

وبالطبع فإن اللغة تحتوى على مجموعة كبيرة من أنماط المتغيرات العددية سوف نتعرض لها جميعاً فى حينها. وبصفة عامة يمكنك أن تعلن عن متغير عددى صحيح (integer) كالآتى:

```
int myInt;
```

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

```
int myInt = 0;
```

وبصفة عامة فإن لغة سي# 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 العددي استخدم نفس أسلوب الطباعة:

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

ولكى تطبع رسالة حرفية مع المتغير ، يمكنك استخدام العبارة الآتية كمثل:

```
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 ضمناً 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 اختياري.

- الفصائل (Classes)

- المنشآت (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 (2-4)

```
// 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 مثل "محمد". ولكنك لو استخدمت الاسم الثلاثى أو اسم العائلة فسوف 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 ومع ذلك فإن استخدام التأهيل للاسم يجنبنا الخلط بين الفصيلتين حيث أنه يحدد 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        // الاسم المؤهل: MyNS1.MyC1.MyC2  
        {  
        }  
    }  
}  
  
namespace MyNS2           // MyNS1.MyNS2  
{  
    class MyC2             // الاسم المؤهل: MyNS1.MyNS2.MyC2  
    {  
    }  
}  
  
namespace MyNS3           // MyNS3  
{  
    class MyC3             // MyNS3.MyC3  
    {  
        static void Main()  
        {  
        }  
    }  
}
```

نرى فى البرنامج السابق أمام كل عنصر من العناصر الاسم الكامل (المؤهل) لهذا العنصر. وتحديد تبعية أسماء العنصر فى البرنامج السابق يتم كالاتى:

- الحيز MyNS1 والحيز MyNS3 يتبعان الحيز العام للأسماء وهو بمثابة الجذر الذى تتفرع منه كل الفروع. لذلك فإن الاسم المؤهل لكل منهما هو نفسه اسم الحيز.
- الحيز MyNS2 يقع بداخل الحيز MyNS1 أى أن اسمه المؤهل هو MyNS1.MyNS2
- الفصيلة MyC1 تقع بداخل الحيز MyNS1 وبذلك فإن اسمها المؤهل هو MyNS1.MyC1
- الاسم المؤهل للفصيلة MyC2 التى تقع بداخل الفصيلة MyC1 هو MyNS1.MyC1.MyC2
- الاسم المؤهل للفصيلة MyC2 التى تقع بداخل الحيز MyNS2 هو MyNS1.MyNS2.MyC2

## العرف الشائع فى كتابة البرامج

بعيداً عن القواعد الملزمة ، فإن هناك عرف شائع لكتابة البرامج ، والعرف هو مجرد توصيات تجعل برنامجك سهل القراءة. أما بالنسبة لمتروجم اللغة فلا أهمية للطريقة التى تكتب بها البرنامج.

- للتفريق بين الأسماء الخاصة بلغة سي# أو التابعة لمكتبة دوت.نت وبين الأسماء المبتكرة التي تمنحها لفصائلك أو متغيراتك ابدأ الاسم بكلمة مميزة مثل my أو your كالأمثلة الآتية: myVariable ، MyClass ، yourAccountDetails .
- أسماء الأنماط (الفصائل والمنشآت إلخ) والأساليب تبدأ أسماؤها بحرف كبير (Capital) مثل Class1. وإذا كان الاسم مكوناً من أكثر من كلمة ، استخدم الحروف الكبيرة في 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 فهي أسماء لا تدل على مضمون ما ومن المفضل تجنبها (إلا إذا كانت تتناسب المضمون كما في برامج الرياضة).
- من الشائع أيضاً أن يبدأ القوس الأيسر على سطر مستقل تحت اسم النمط (مثل كلمة class) أو الأسلوب مباشرة ، ويأتي القوس الأيمن في نفس المكان على سطر مستقل ، أي:

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

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

- اترك سطرًا خاليًا بعد كل عملية مميزة. على سبيل المثال فإنه بعد فقرة الإعلان عن متغيرائك فى البرنامج ، عليك أن تترك سطرًا خاليًا لتمييز الإعلانات عن عمليات المعالجة. حتى بداخل عمليات المعالجة فإنه من الأفضل ترك سطر خال بعد كل عملية فرعية (حلقة تكرارية ، عبارة شرطية ، إلخ). والغرض من هذه القواعد أن يكون البرنامج سهل القراءة والتعديل إذا رجعت إليه فى وقت لاحق.

### تذكر هذه المصطلحات

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

الباب الثاني (الجولة الأولى في لغة سي#)

Method	أسلوب
Main method	الأسلوب الرئيسي
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 المدمجة

