مقدمة في البرمجة
Introduction To Programming

IPG101

الفصل الرابع
مقدمة إلى لغة البرمجة C#
Introduction To C# Programming Language

الكلمات المفتاحية

C#، .NET Framework، Visual Studio، برنامج، صنف، فضاء أسماء، كلمات مفتاحية.

ملخص الفصل

يتضمن هذا الفصل مقدمة إلى لغة البرمجة C#، إذ يبدأ بعرض تعريف موجز بإطار العمل .NET Framework. كما يقدم لمحة تاريخية حول نشوء لغة وإصداراتها، ومن ثم ينطلق ليعرف ببيئة التطوير Visual Studio كيفية البدء بكتابة برنامجنا الأول في لغة C# مع توضيح الشكل العام للبرنامج مع التعريف بأهم الكلمات المفتاحية المستخدمة في هذه اللغة.

أهداف الفصل

بنهاية هذا الفصل سيكون الطالب قادراً على:

- فهم إطار العمل Net.
- تعداد الإصدارات المختلفة للغة C# والفروقات فيما بينها.
- تشغيل بيئة التطوير Visual Studio وتمييز مكوناتها.
- فهم البنية العامة للبرنامج المكتوب بلغة C#.
- كتابة وتنفيذ البرنامج الأول بلغة C#.
- ذكر الكلمات المفتاحية في لغة C#.

محتويات الفصل

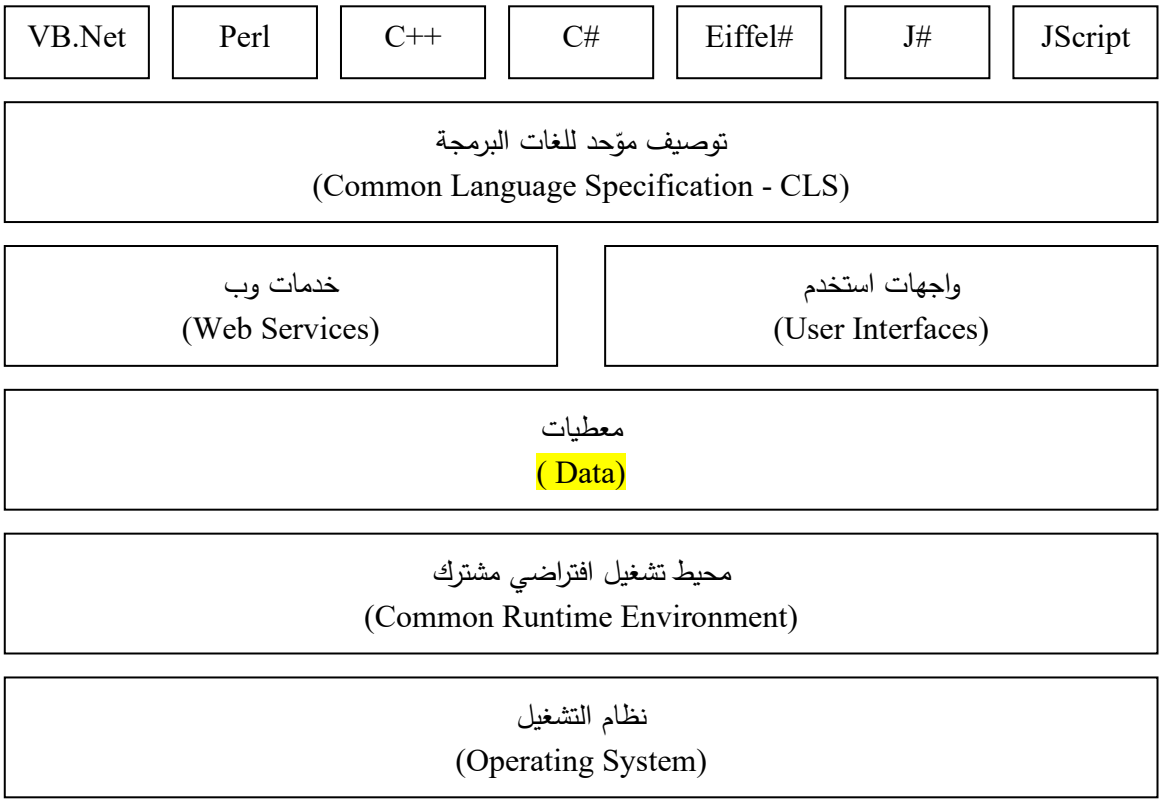
1. إطار العمل Dot NET.
2. لمحة تاريخية عن لغة C#.
3. البرنامج الأول في لغة C#.
4. تحليل النص البرمجي.
5. الكلمات المفتاحية في لغة C#.

1- إطار العمل Net Framework.

اقترحت Microsoft استراتيجية جديدة لتوزيع عملية معالجة المعطيات في إطار بنيان برمجي متكامل، تحت اسم "Dot Net". يرتكز البنيان الآنف الذكر على مجموعة من الأفكار المؤسِّسة التي تطمح للوصول إلى بيئة برمجية تتمتع بالمواصفات التالية:

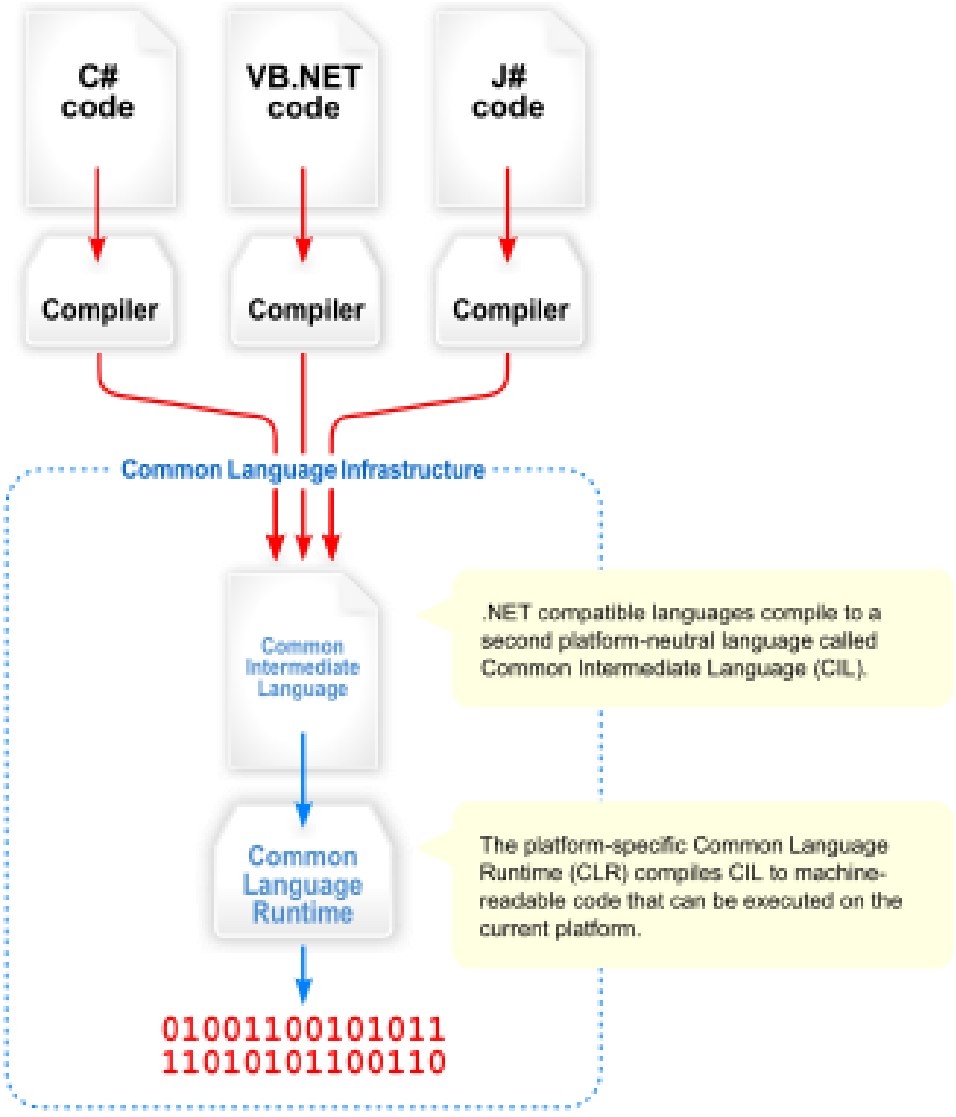
- شفافية التعامل مع التطبيق من ناحية كونه تطبيق محلي أو تطبيق إنترنت؛
- توزيع المعطيات على عدد من المخدمات عوضاً عن تركيزها ضمن مخدم واحد، وبحيث يحتوي كل مخدم على الخدمات اللازمة للتعامل مع جزئه الخاص من المعطيات؛
- تحويل عملية شراء تطبيق برمجي وتثبيته على مخدمات محلية إلى عملية استئجار خدمة برمجية تقدمها مجموعة مخدمات على الإنترنت؛
- تحويل الحاسب الشخصي إلى طرفية ذكية تساعد في البحث عن الخدمة المطلوبة وتشغيلها عن بعد؛
- تأمين مكوِّنات برمجية جاهزة يمكن لمطوري البرامج مكاملتها ضمن برامجهم دون الحاجة لإعادة برمجتها؛

نبين فيما يلي بنية إطار العمل .NET.



الشكل 1- معمارية إطار العمل .NET FRAMWORK.

الفكرة الأساسية التي عمل هذا الإطار على تحقيقها هي توفير إمكانية التشغيل التفاعلي (يمكن لكل لغة استخدام شيفرات برمجية مكتوبة بلغات أخرى) عبر العديد من لغات البرمجة. يتم تنفيذ البرامج المكتوبة لـ . NET Framework في بيئة برمجية مستقلة عن العتاد (على عكس البيئة المرتبطة بالعتاديات المادية Hardware) تسمى Common Language Runtime (CLR). CLR عبارة عن آلة افتراضية للتطبيق توفر خدمات مثل الأمان وإدارة الذاكرة ومعالجة الاستثناءات.



الشكل 2– ترجمة الشيفرة المكتوبة بلغات مختلفة إلى لغة مشتركة

بالنتيجة، تقدم Microsoft من خلال Dot Net محيطاً برمجياً يُدعى (Dot Net Framework) يساعد المُبرمج في برمجة وتشغيل تطبيقاته سواءاً كانت تطبيقات كلاسيكية تعمل ضمن محيط نظام التشغيل Windows أو تطبيقات وب تعمل ضمن محيط مخدم وب.

2- لمحة تاريخية عن لغة C#

تم إطلاق لغة C# مع إطار العمل .NET Framwork 1.0 في العام 2002 ومن ثم خضعت للعديد من عمليات التطوير، يجمل الجدول التالي إصدار هذه اللغة:

Version	.NET Framework	Visual Studio
C# 1.0	.NET Framework 1.0/1.1	Visual Studio .NET 2002
C# 2.0	.NET Framework 2.0	Visual Studio 2005
C# 3.0	.NET Framework 3.0\3.5	Visual Studio 2008
C# 4.0	.NET Framework 4.0	Visual Studio 2010
C# 5.0	.NET Framework 4.5	Visual Studio 2012/2013
C# 6.0	.NET Framework 4.6	Visual Studio 2013/2015
C# 7.0	.NET Core 2.0	Visual Studio 2017
C# 8.0	.NET Core 3.0	Visual Studio 2019
C# 9.0	.NET 5.0	Visual Studio 2019
C# 10.0	.NET 6.0	Visual Studio 2022

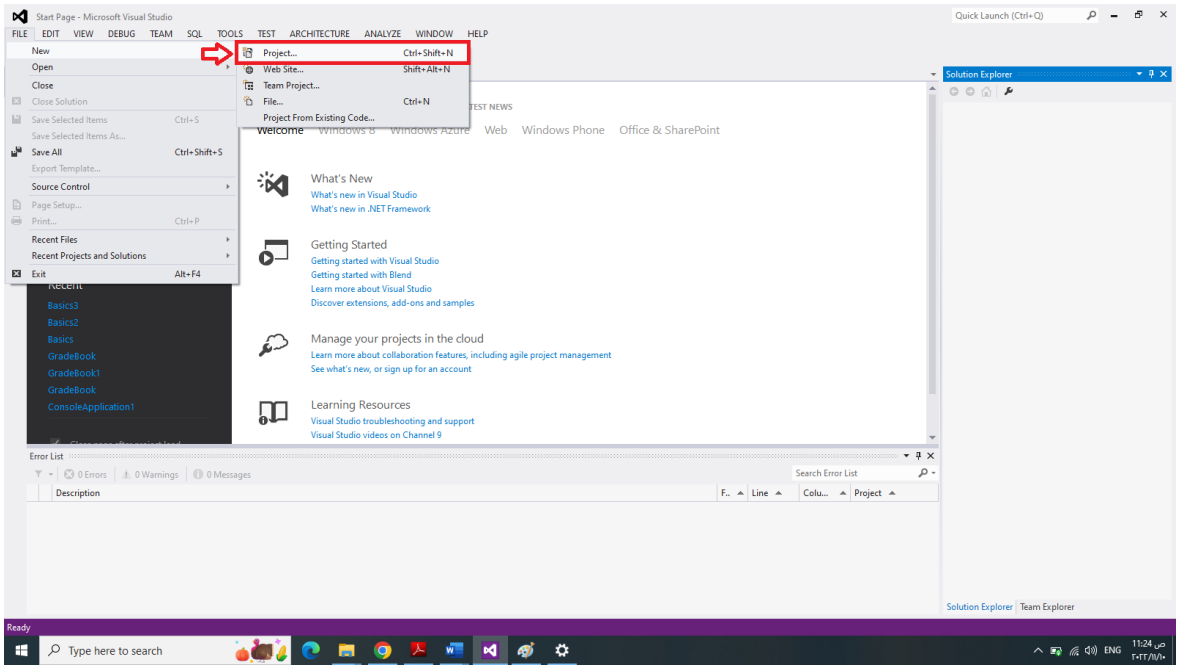
3- البرنامج الأول في لغة C#

نبدأ الآن مع لغة C#، وسنتعلم في هذه الفقرة كيف نقوم بتشغيل بيئة التطوير Visual Studio وإنشاء مشروع بلغة C# وكتابة البرنامج الأول، ونوضح مكونات واجهة بيئة التطوير والكتل الأساسية في البرنامج.

بعد القيام بتنصيب بيئة التطوير Visual Studio على جهازك (يفضل تنصيب إصدار حديث قدر الإمكان)، يمكن القيام بتشغيله عبر الذهاب إلى قائمة إبدأ والتنقل إلى مكان البرنامج ضمن قائمة البرامج.

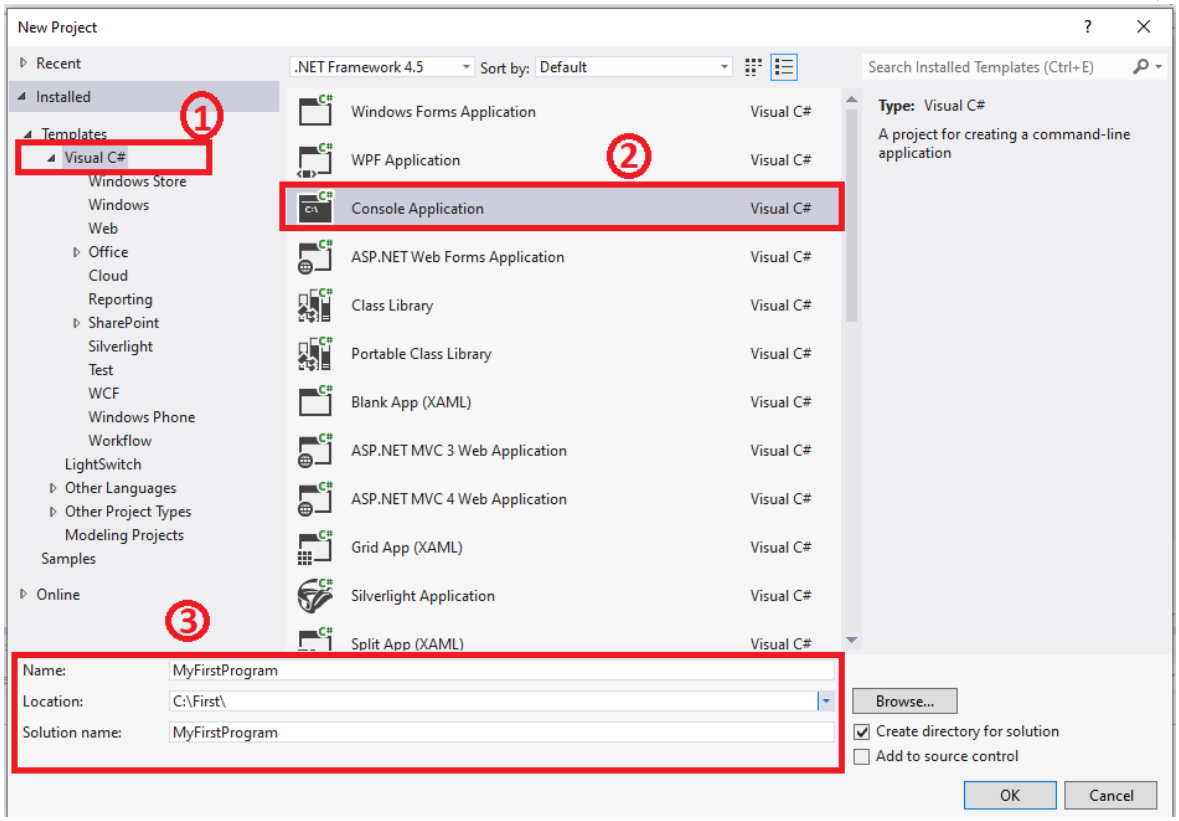
بعد إقلاع البرنامج، إذهب إلى القائمة File و اختر New ومن ثم Project (كما هو مبين في الشكل 3).

ملاحظة: يمكنك فتح مشروع سابق موجود عبر النقر على أمر القائمة Open ومن ثم الذهاب إلى مكان المشروع على القرص واختياره.



الشكل 3– إنشاء مشروع جديد في Visual Studio

في الخطوة التالية، تظهر لدينا نافذة New Project:



الشكل 4– نافذة New Project

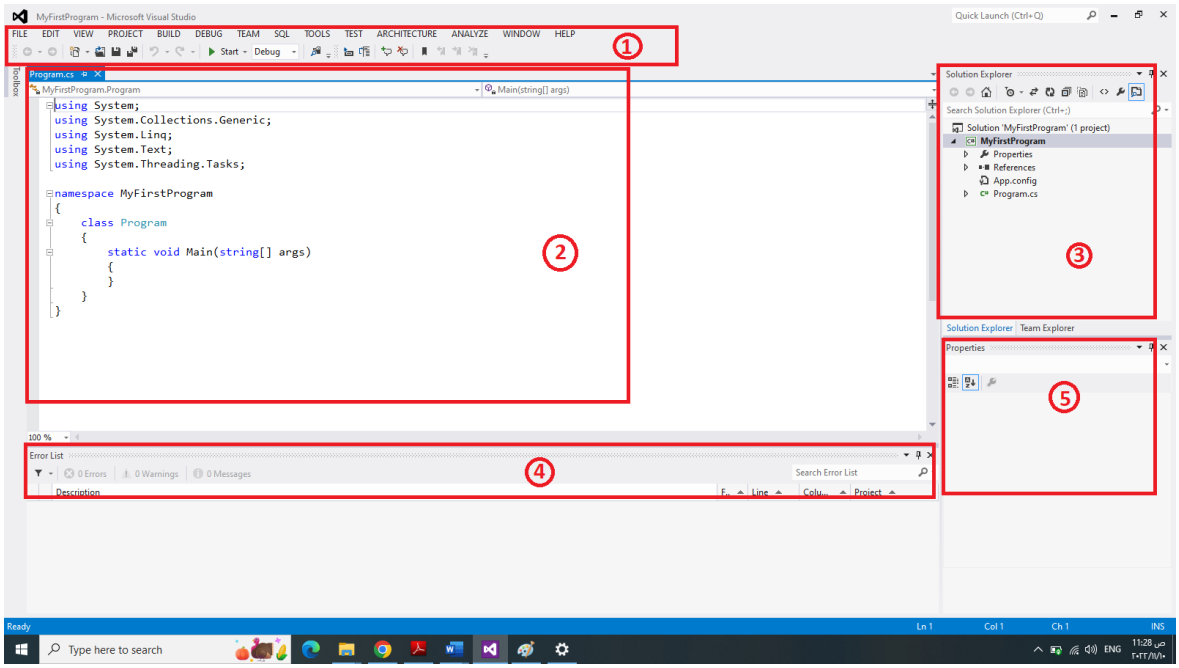
في هذه النافذة نقوم بالخطوات التالية:

1- تحديد نوع المشروع: نختار Visual C#.

2- تحديد نوع التطبيق: نختار Console Application.

3- نحدد اسم المشروع ومكان تخزينه على القرص الصلب.

بالضغط على الزر OK تظهر لدينا بيئة محرر التطوير للبرنامج:



الشكل 5- نافذة محرر التطوير

تتضمن هذه الواجهة المكونات التالية:

1- شريط القوائم والأدوات التي تتيح اختيار تنفيذ الأوامر وإجراء الإعدادات على المشروع البرمجي.

2- محرر النصوص وفيه تتم كتابة الشيفرة البرمجية وتحرير ملفات المشروع واستعراضها.

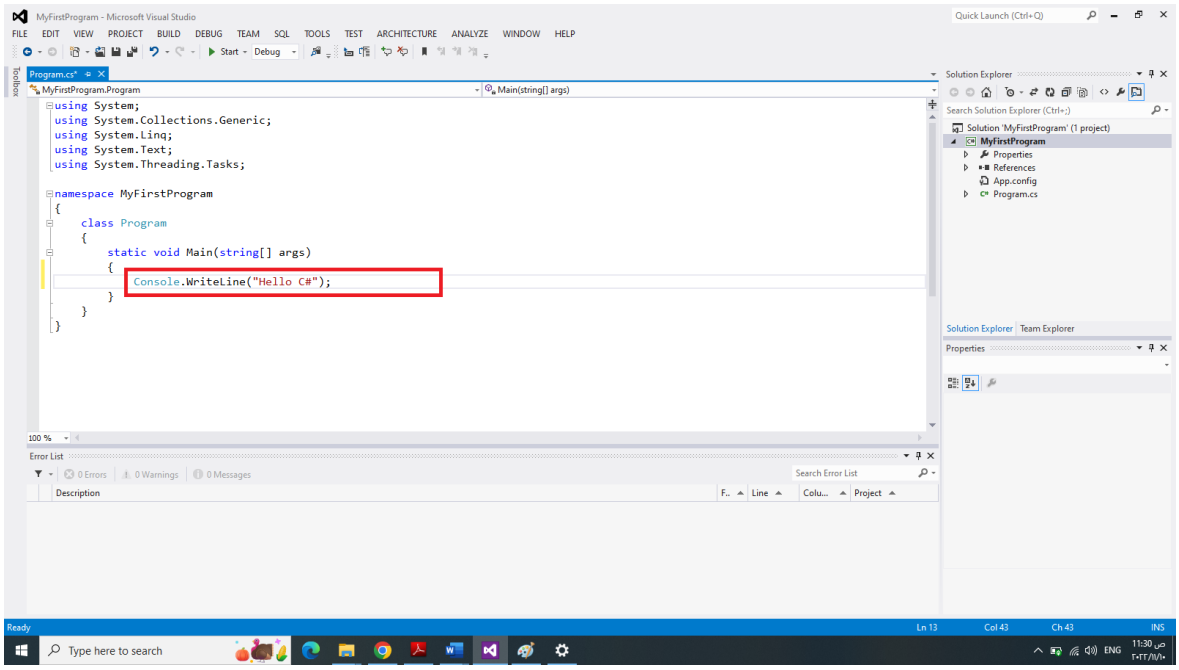
3- مستعرض المشروع ويتضمن بنية شجرية بملفات المشروع.

4- نافذة منقح المشروع ومستعرض الأخطاء.

5- نافذة الخصائص التي تظهر خصائص كل مكون من مكونات المشروع.

ملاحظة: تختلف هذه الواجهة من مستخدم لآخر، إذ يمكن إظهار نوافذ أخرى أو إخفاء بعض النوافذ من القائمة View في شريط القوائم (بحسب الحاجة).

في الجزء رقم 2 من النافذة السابقة نقوم بكتابة أول سطورنا البرمجية في لغة C# كما يلي:

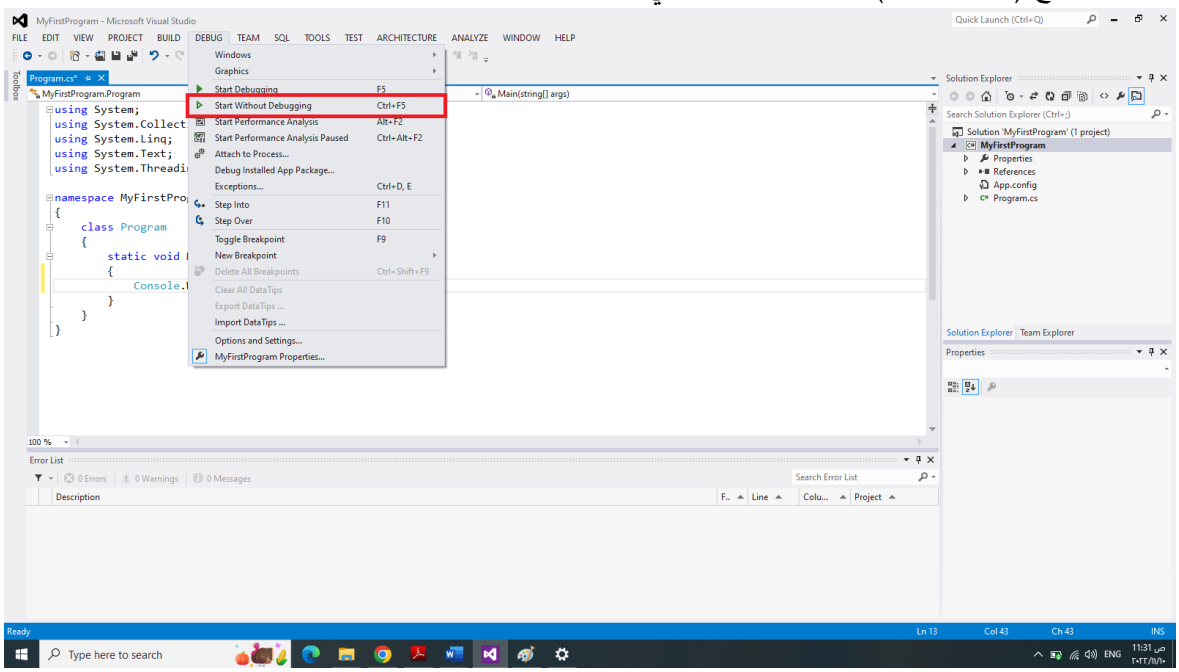


الشكل 6- كتابة الأوامر ضمن التابع Main

في حال وجود أي خطأ في كتابة البرنامج، فإن رسالة خطأ ستظهر تلقائياً في الجزء رقم 4 من الشكل رقم 5 تشير إلى الخطأ المرتكب ورقم السطر ولا يكون بالإمكان ترجمة البرنامج وتنفيذه مالم تتم معالجة الخطأ.

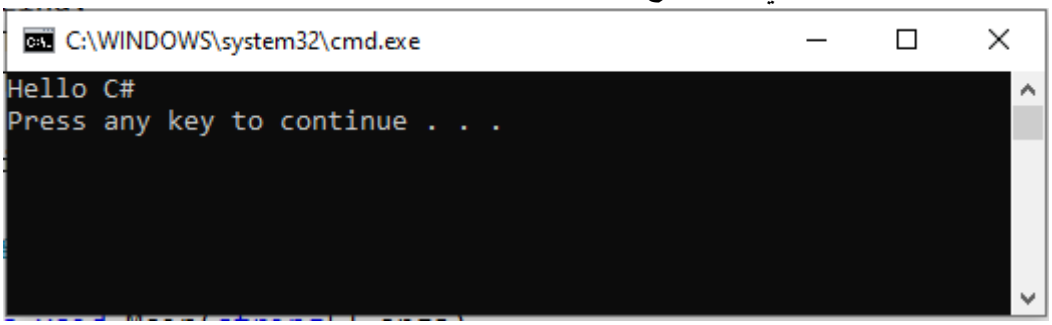
مثال: قم بحذف الفاصلة المنقوطة (;) من نهاية السطر الذي قمت بإضافته ولاحظ التغيرات على الجزء رقم 4.

لتنفيذ البرنامج، نختار القائمة DEBUG ، ثم نختار الأمر Start without debugging (ويمكن الضغط على تراكب المفاتيح (Ctrl +F5) كما هو مبين فيما يلي:



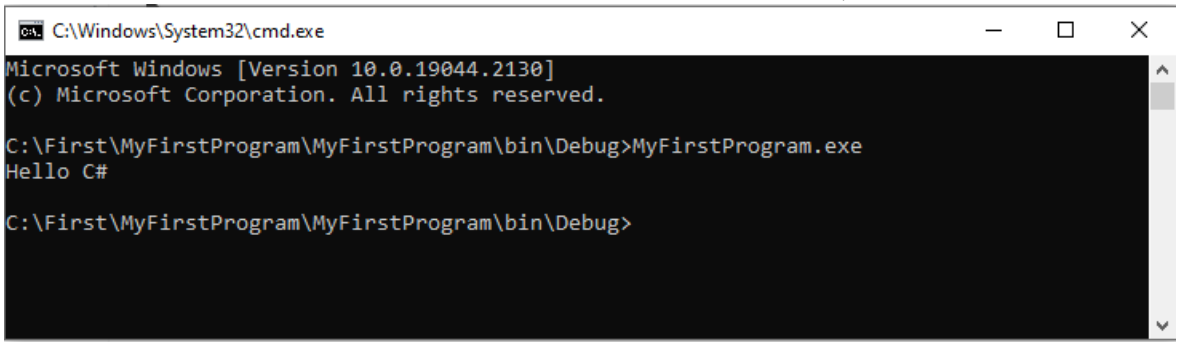
الشكل 7- تنفيذ البرنامج

تظهر لدينا عندئذ النافذة التالية التي تظهر ناتج التنفيذ:



الشكل 8- شاشة الخرج للبرنامج

ملاحظة: كان من الممكن التنفيذ عبر محث الأوامر Command Prompt الخاص بنظام DOS عبر الانتقال إلى مكان تخزين المشروع على القرص الصلب، ومن ثم المجلد Bin فالمجلد debug واستدعاء الملف التنفيذي MyFirstProgram.exe كمايلي:



الشكل 9- التنفيذ عبر محث الأوامر

4- تحليل النص البرمجي

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MyFirstProgram
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello C#");
        }
    }
}
```

- لاحظ أولاً الكلمات الملونة بالأزرق: using, namespace, class, static, void, string هي كلمات مخصصة/محجوزة للغة البرمجة C# وهي التي تساهم في التعبير عن البنية القواعدية للغة (غالباً ما تُسمّى أيضاً كلمات مفتاحية keyword).

- نتُستَخدَم العبارة "using System" للدلالة على ماندعوه فضاء الأسماء "System" الذي يُقدم مجموعة من الصفوف الجاهزة والمُعرّفة التي يمكن استخدامها ضمن التطبيق مباشرةً؛

- ينتمي الصف "Console" إلى فضاء الأسماء "System" ويُستَخدَم للتعامل مع الواجهة النصيّة (الشاشة أسود/أبيض تسمح بإظهار نصي فقط) كما لاحظنا عند تشغيل البرنامج.

- يستخدم الصف "Console" التعليمة/الإجرائية "WriteLine" لكتابة سلسلة محارف وإظهارها على الواجهة النصيّة.

- يُعرِّف المثال صفاً يُدعى "Program"، يحتوي على إجرائية "Main" يمكن اعتبارها نقطة إنطلاق لتنفيذ المثال؛

- تقوم التعليمة/الإجرائية WriteLine بإظهار عبارة "Hello C#" عند استدعاء التطبيق من واجهة التعليمات النصيّة.

ملاحظة هامة جداً:

- إن لغة C# هي لغة غرضية التوجه، وبالتالي تعتمد في تأطير البرمجة، على ما يسمى الصف class . لكننا في أساسيات البرمجة، لسنا معنيين بمعرفة آلية البرمجة في C# لبناء الصفوف بقدر ما نحن معنيون باستخداماتها.
- سنعرِّف الصف (بما يعنينا في أساسيات البرمجة) بأنه تجميع لمعطيات ووظائف، حيث أن الوظائف (هي غالباً مانسميها توابع، إجرائيات، طرائق).

من الناحية العملية البرمجية والقواعدية نطلب الوظيفة من صف بالشكل: ClassName.F()
كما في المثال:

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

- أخيراً فضاء الأسماء namespace هو تجميع لعدد من الصفوف.

5- الكلمات المفتاحية في لغة C#

تحتوي لغة C# على كلمات محجوزة يكون لها معنى خاص بالنسبة للمترجم، هذه الكلمات المحجوزة تدعى أيضاً "كلمات مفتاحية"، ولا يمكن استخدام هذه الكلمات كمعرفات (أسماء متحولات، أصناف، .... إلخ).

يتضمن الجدول حميع الكلمات المحجوزة للغة C#. ولكن في أساسيات البرمجة لا نحتاج إلا معرفة عدد محدد منها.

abstract	extern	operator	throw
as	false	out	true
base	finally	override	try
bool	fixed	params	typeof
break	float	partial	uint
byte	for	private	ulong
case	foreach	protected	unchecked
catch	get	public	unsafe
char	goto	readonly	ushort
checked	if	ref	using
class	implicit	return	value
const	in	sbyte	virtual
continue	int	sealed	void
decimal	interface	set	volatile
default	internal	short	where
delegate	is	sizeof	while
do	lock	stackalloc	yield
double	long	static	
else	namespace	string	
enum	new	struct	
event	null	switch	
explicit	object	this	