

الباب الأول

البرمجة الموجهة نحو الأهداف

Object Oriented Programming (OOP)

مفتتح

فى هذا الباب تلتقى بمفاهيم البرمجة الموجهة نحو الأهداف (OOP) التى بدأت مع لغة سى++ فى أوائل التسعينيات ثم استخدمها المعمار دوت.نت (.NET) كقاعدة أساسية لبناء لغات البرمجة الجديدة ، وفى مقدمتها لغة سى# .

كما نقدم فى هذا الباب فكرة عن اللغات الأخرى التى تعمل تحت مظلة دوت.نت. وفى النهاية نقدم مقارنة سريعة بين لغتى سى++ وسى# باعتبار أن سى# هى التطور الطبيعى للغة سى++ .

محتويات الباب

- فلسفة البرمجة الموجهة نحو الأهداف (OOP)
- سي# لغة البرمجة الموجهة نحو الأهداف
- المعمار دوت.نت (.NET)
- نظرة سريعة على الكود
- كيف تحصل على مترجم لغة سي#
- أهم الفوارق بين سي# و سي++

فلسفة البرمجة الموجهة نحو الأهداف (OOP)

بدأت فلسفة البرمجة الموجهة نحو الأهداف (Object Oriented Programming) مع ظهور لغة سي++ ، حيث قدمت اللغة نمط الفصائل (Classes) علاوة على اتجاه جديد فى البرمجة وهو التعامل مع البيانات فى صورة أهداف (Objects). وفكرة الفصائل تعتمد على محاكاة الطبيعة ، فالطبيعة تحتوى على مجموعة كبيرة من الفصائل التى قد تتشابه فى خصائصها وسلوكها وقد تراث الخصائص والسلوك من بعضها البعض. ولو اعتبرنا فصيلة الإنسان مثلاً فإننا نرى أن جميع أعضاء هذه الفصيلة تتشابه فى السلوك مثل القدرة على الكلام ، وتتشابه فى بعض الخصائص مثل الحواس والأطراف. ومن فصيلة الإنسان تتحدر فصيلة الرجل وفصيلة المرأة. وهما تراثان من فصيلة الإنسان كل خصائصها وسلوكها ، علاوة على الخصائص المتفردة لكل من جنس الرجل و جنس المرأة.

وعندما نستخدم ألفاظ الإنسان أو الرجل أو المرأة فإننا نستخدم التجريد ، بمعنى أننا لا نركز اهتمامنا على إنسان معين مثل محمد أو مرزوق أو زينب. فهذه الأسماء المحددة تدل على أمثلة (Instances) مختلفة (أو أهداف – Objects) من فصيلة الإنسان. بمعنى أن الأمثلة أو الأهداف هى الصورة الواقعية للفصائل.

وفى مجال البرمجة فإننا فى تعاملنا مع البيانات نقسمها إلى فصائل تتشابه فى السلوك والخصائص ، وتمثل الصورة المجردة (Abstraction) للبيانات. ومن الفصائل نخلق الأمثلة (أو الأهداف) التى تمثل الصورة الواقعية للبيانات. وهذا يختلف عن الطريقة التقليدية للبرمجة وهى استخدام المتغيرات المنفردة والمؤشرات مثل $x, y, & z$. والميزة التى تمنحنا إياها الفصائل هى إمكانية كبسلة (Encapsulation) الصفات والسلوك فى كبسولة واحدة وهى الفصيلة. ومن الجائز أن يكتب الفصيلة مبرمج ما ، أما خلق الأهداف واستخدامها فيقوم به مبرمج آخر - ربما فى شركة أخرى. كما يمكننا أن نورث الفصيلة لنبتكر منها فصائل أخرى تشبهها وتضيف إليها خصائص وصفات جديدة. وعلى سبيل المثال فإن فصيلة الموظف (Employee) ترث من فصيلة الإنسان كل خصائصها وسلوكها ، لكنها تضيف إليها خصائص وسلوك الموظف المميزة ، وهكذا. ومبدأ الوراثة (Inheritance) يوفر الكثير من الوقت فى البرمجة. فنحن لا نحتاج أن نخلق فصيلة الموظف من الصفر ، إننا فقط نضيف إلى فصيلة الإنسان صفات الموظف مثل اسم الشركة التى يعمل بها ، وسلوك الموظف مثل قيامه بالعمل عدداً معيناً من الساعات كل شهر. كما أن البرمجة الموجهة نحو الأهداف لها مميزات أخرى توفر الوقت والجهد مثل تعدد الأشكال (Polymorphism). وعلى سبيل المثال فإنك تستطيع فى برنامج ما أن تبنى دالة واحدة للرسم مثل "Draw" ، وبحسب نوع الهدف المرتبط

بهذه الدالة تتغير النتيجة التي نحصل عليها. فقد تكون النتيجة مستطيلاً أو دائرة أو شبه منحرف.

ولتلخيص الموضوع فإننا نعرض هنا المبادئ الأربعة للبرمجة الموجهة نحو الأهداف:

مبادئ البرمجة الموجهة نحو الأهداف

١. التجريد (Abstraction)

٢. الكبسلة (Encapsulation)

٣. الوراثة (Inheritance)

٤. تعدد الأشكال (Polymorphism)

سى # لغة البرمجة الموجهة نحو الأهداف

بالرغم من أن لغة سى++ كانت هي البداية الأولى لاستخدام البرمجة الموجهة نحو الأهداف ولكن اللغة كانت مزيجاً من الاتجاهين في البرمجة: الاتجاه التقليدي المشابه للغة سى واللغات الأخرى القديمة ، والاتجاه الجديد في البرمجة الموجهة نحو الأهداف. ولذلك فإنك تستطيع باستخدام لغة سى++ أن تكتب برنامجاً محكماً يغلف البيانات في صورة كبسولة من الحقول والدوال ، كما يمكنك أن تكتب برنامجاً مهلهلاً يعتمد على المتغيرات العامة (Global Variables) أو يحتوى على مجموعة

من المؤشرات بمشاكلها التي تؤدي إلى تبيد الذاكرة وإضاعة الوقت في صيانة البرنامج.

أما لغة سي# (وتتطرق سي شارب) فقد تم بناؤها من الصفر كلغة موجهة نحو الأهداف. بمعنى أنه لا يمكنك أن تنشئ برنامجاً بدون وضعه بداخل كبسولة الفصيلة ، علاوة على ما يلزم الفصيلة من خصائص تساعدك على بناء الأهداف. لا يمكنك أن تكتب برنامجاً بلغة سي# محتوياً على بعض المتغيرات المبعثرة في البرنامج والتي تؤدي إلى المشكلات التقليدية التي دأب المبرمجون على حلها جيلاً بعد آخر. والحقيقة أن لغة سي# ، كتطوير للغة سي++ ، لم تقدم فقط بعض الملامح الجديدة ، بل تخلصت من الكثير من ملامح سي++ التي كانت تتسبب في الأخطاء وتؤدي إلى المشكلات المعروفة.

وسوف نعرض مقارنة كاملة بين لغة سي# ولغة سي++ في الفقرات القادمة.

المعمار دوت.نت (.NET)

في الحقيقة أن لغة سي# هي إحدى اللغات الجديدة التي تقع تحت مظلة المعمار دوت.نت. فهناك لغات أخرى ابتكرتها شركة ميكروسوفت مثل Visual Basic و #J وامتداد لغة سي++ (C++ Managed Extensions). جميع هذه اللغات ، علاوة على لغات أخرى قدمتها شركات مختلفة ، تقوم تحت مظلة المعمار دوت.نت

وتستخدم فصائله ومكتباته. وهذه هي الفائدة الرئيسية للمعمار دوت.نت: إمكانية بناء البرنامج بأكثر من لغة وإمكانية التعاون بين لغات البرمجة المختلفة. وبصفة عامة فإن العناصر التي نعرضها في الفقرة التالية هي أهم مكونات المعمار دوت.نت.

بيئة تنفيذ البرامج (Common Language Runtime)

واختصارها CLR. وهي عبارة عن بيئة متكاملة لبناء البرامج (بصرف النظر عن اللغة المستخدمة) وتحتوى على مكتبة هائلة من الفصائل التي تخدم الأغراض العامة مثل:

- فصائل التجميع (Collection classes) مثل الجداول (Hash tables) والطوابير (Queues) والخزن (Stacks).
- قواعد البيانات (Databases)
- نماذج النوافذ (Windows Forms) التي تستخدم كواجهة لتطبيقات برامج النوافذ
- الشبكات الكومبيوترية.

الـ ميتا - داتا (Metadata)

عند ترجمة برنامج ما فإن بيئة التنفيذ تنتج مجموعة من المعلومات عن الفصائل المستخدمة تتضمن اسم الفصيلة ، وأسماء الحقول وأنماطها ، وأسماء الدوال (أو الأساليب) وبارامتراتهما. وتسمى هذه المعلومات "ميتا - داتا" بمعنى معلومات عن البيانات. وهذه المعلومات

لها استخدامات كثيرة منها تمكين المبرمج من إنشاء الأهداف من الفصائل بصرف النظر عن اللغة التي كتبت بها الفصائل.

المجمعات (Assemblies)

كانت الصورة النهائية للبرامج هي الملفات التنفيذية (exe) أو المكتبات (dll أو lib) ، أما البرامج المتوافقة مع المعمار دوت.نت فإنها تشحن في صورة مجمعات. وعند بناء برنامج ما فإنه يترجم إلى لغة وسطى (Intermediate Language) وتختصر IL. ويحتوى المجمع على اللغة الوسطى ، والميتا – داتا ، وأية ملفات أخرى تساعد على التشغيل. وبذلك فإن المجمع يحتوى على خريطة كاملة للمشروع شاملاً المكتبات والمراجع المستخدمة علاوة على أية مجمعات أخرى يعتمد عليها المشروع. والمجمع قد يكون ملفاً واحداً أو مجموعة ملفات.

نظرة سريعة على الكود

لنلق نظرة سريعة على مثال للفصيلة بدون الدخول في تفاصيل

اللغة:

```
class Employee
{
    // Fields: (الخصائص) الحقول
    string name;
    int hoursWorked;
    int rate;

    // Methods: (الأساليب) (السلوك)
    double CalculateSalary(int hoursWorked, double rate)
```

```
{  
  double salary;  
  salary = rate * hoursWorked;  
  return salary;  
}  
}
```

فى هذا المثال ، بصرف النظر عن اللغة ، نرى فصيلة الموظف التى تحتوى على خصائص الموظف وهى الحقول:

- الاسم (name)

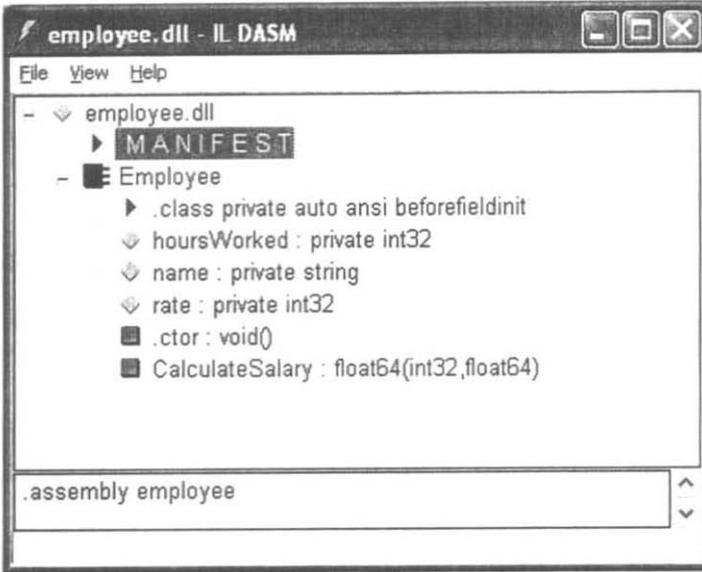
- عدد ساعات العمل (hoursWorked)

- معدل الأجر (rate)

كما نرى سلوك الفصيلة وهو الدالة أو الأسلوب المستخدم فى حساب المرتب (CalculateSalary). وهذا الأسلوب يستخدم بعض البارامترات (hoursWorked و rate) ومنها يخرج بالنتيجة النهائية وهى المرتب salary.

وبالطبع فإن الفصيلة يمكن أن تحتوى على تفاصيل أكثر ولكننا بغرض ضرب المثال قد بسطناها إلى أقصى حد.

بعد ترجمة هذا البرنامج إلى مكتبة (dll) يمكنك أن تلقى نظرة على اللغة الوسطى والميتا - داتا التى تحتوى عليها المكتبة (يستخدم فى ذلك البرنامج ILDasm.exe الذى يأتى مع دوت.نت). وفى الشكل التالى نرى "منافستو" الفصيلة الذى يحتوى على هذه البيانات.



شكل (1-1) منافستو فصيلة الموظف كما يعرضه البرنامج ILDasm

وكما نرى بالشكل أن المنافستو يحتوى على حقول الفصيلة الثلاثة وعلى الأسلوب CalculateSalary() والبارامترات المستخدمة فيه. وهذا هو كل ما يلزم المبرمج الذى يريد استخدام هذه الفصيلة فى برنامجه. بمعنى آخر فإن المبرمج لا يحتاج الاطلاع على الكود المصدر (Source code) لكى يستخدم الفصيلة. ويمكنك أن تحصل على هذا المنافستو باستخدام الأمر:

```
ILDasm Employee.dll
```

(بفرض أن اسم الملف هو Employee.dll).

وباستخدام الأمر ILDASM يمكنك الاطلاع على منافستو أى ملف تنفيذى (exe) او مكتبة (dll) فتحصل على الفصائل بما تحتويه من

أعضاء. كما أنك لو ضغطت ضغطة مزدوجة على أحد العناصر الموجودة بالشكل السابق فإنك ترى كود اللغة الوسطى المناظر لكل عنصر.



برامج ترجمة اللغة الوسطى

هناك برنامجان يأتیان مع الإستوديو المرئى أو دوت.نت وهما:

١. مترجم اللغة الوسطى (ILASM): وهو يستخدم فى ترجمة البرامج المكتوبة باللغة الوسطى IL وتحولها إلى برنامج تنفيذى أو مكتبة. واللغة تنتمى إلى اللغات منخفضة المستوى (LLL).

٢. المترجم العكسى للغة الوسطى (ILDASM): وهو يقوم بالعملية العكسية ، أى أنه يقرأ ملف البرنامج التنفيذى أو المكتبة ويستخرج منه الكود المكتوب باللغة الوسطى علاوة على المنافستو بما يحتويه من ميتا - داتا.

كيف تحصل على مترجم سي# ؟

يمكنك أن تبني برامجك المكتوبة بلغة سي# إما باستخدام المترجم فى بيئة الأوامر (Command Line Compiler) أو باستخدام البيئة المجمع (IDE) الموجودة بالإستوديو المرئى (Visual Studio .NET). ويمكنك الحصول على المترجم بأى طريقة من الطرق الآتية ، وهى جميعاً مجانية فيما عدا برنامج الإستوديو حيث أنه المنتج الأساسى لشركة ميكروسوفت:

الحزمة دوت.نت (Microsoft .NET Framework)

هذا هو أسهل الطرق للحصول على مترجم سي# (علاوة على مترجم لغة .NET VB ومترجم #J). ويمكنك إنزال هذه الحزمة من الإنترنت عند العنوان:

<http://msdn.microsoft.com/vcsharp/downloads/updates/>

ثم اضغط على النص التالي:

How to Get the Microsoft .NET Framework 1.1

ثم اتبع الخطوات الموضحة حتى تصل إلى إنزال الحزمة:

Microsoft .NET Framework Version 1.1 Redistributable Package

والحزمة عبارة عن ملف واحد يحمل الاسم dotnetfx.exe. وبتنفيذ

هذا البرنامج يتم تركيب دوت.نت.

الحزمة إس . دي . كي (Microsoft .NET Framework SDK)

يمكنك إنزال هذه الحزمة من الإنترنت عند نفس العنوان:

<http://msdn.microsoft.com/vcsharp/downloads/updates/>

ثم اضغط على النص التالي:

How to Get the Microsoft .NET Framework 1.1

ثم اتبع الخطوات الموضحة حتى تصل إلى إنزال الحزمة:

.NET Framework 1.1 SDK

وتحتوى هذه الحزمة على المعمار دوت.نت علاوة مجموعة من الوثائق

والمنافع. ويتطلب تركيب هذه الحزمة وجود الحزمة دوت.نت على

الكمبيوتر. والحزمة عبارة عن ملف تنفيذي يحمل اسماً مثل "

، وبتنفيذ هذا البرنامج يتم تركيب

الحزمة SDK.

ميكروسوفت إستوديو دوت.نت (MS Visual Studio .NET)

لو أنك استخدمت الإستوديو فإنك سوف تحصل على عدة تسهيلات في البرامج ، حيث يحتوى الإستوديو على محرر خاص للبرامج علاوة على أدوات لتصميم التطبيقات المختلفة مثل التطبيقات النوافذية وتطبيقات الوب وقد خصصنا الباب الحادى عشر لجولة سريعة فى الإستوديو. وبصفة عامة فإن أهم وسائل المساعدة هى نافذة القواعد (IntelliSense) التى تصاحبك كلما حركت الفأر فوق عنصر من عناصر اللغة حيث تمدك باستمرار بالصيغة التى تستخدم مع هذا العنصر. أما الخاصية الأخرى فإنها خاصية الإكمال الأتوماتيكى (Auto Completion) حيث تساعدك على أن تجد الدالة (أو الأسلوب) التى تبحث عنها. انظر الشكل التالى.

```

using System;
using System.IO;
class MyClass
{
    static void Main()
    {
        StreamReader myFile = new StreamReader("test.txt");
        string myString = myFile.ReadToEnd();
        myFile.Close();
        Console.WriteLine(mySt);
    }
}

```

إكمال الأتوماتيكى

نافذة القواعد

string StreamReader.ReadToEnd() (+ 1 overloads)
Reads the stream from the current position to the end of the stream.

شكل (1-2): شاشة محرر الكود بالإستوديو وبها نافذتنا القواعد والإكمال الأتوماتيكى

الحزمة السريعة سي# اكسبريس (C# Express)

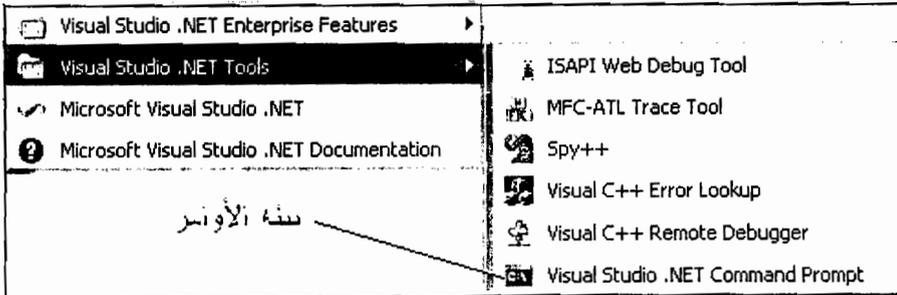
وتوزع هذه الحزمة مجاناً على الإنترنت ، وسوف يتم توزيعها في بداية عام 2005. كما ستوزع مجموعات أخرى من الحزم السريعة للغات الأخرى بغرض تشجيع الهواة والمبتدئين على استخدام منتجات ميكروسوفت.

تشغيل مترجم سي# فى بيئة الأوامر

نقدم فى هذه الفقرة طريقة إعداد وتشغيل مترجم سي# فى بيئة الأوامر.

الإعداد للتشغيل

إذا كنت تستخدم برنامج الإستوديو فإنك لا تحتاج إلى إعداد بيئة الأوامر. كل ما عليك أن تستخدم أداة بيئة الأوامر التى تصاحب الإستوديو كما بالشكل التالى:



شكل (1-3) أداة بيئة الأوامر كما تظهر فى قائمة البرامج

أما إذا كنت قد انزلت دوت.نت أو الحزمة SDK من الإنترنت ،
فعليك أن تضع اسم الدوسيه المحتوى على المترجم (csc.exe) فى
الممر (Path).

ويوجد هذا الدوسيه عند العنوان:

C:\WINDOWS\Microsoft.NET\Framework\v1.1.4322

(أو C:\WINDOWS\Microsoft.NET\Framework\v1.0.3705 إذا كنت تستخدم
الطراز 1.0).

ملاحظة:

لكى تضع اسم الدوسيه فى الممر اكتب الأمر التالى فى بيئة الأوامر (
Command Prompt) أو اكتبه فى ملف تجميع (Batch file):
path=C:\WINDOWS\Microsoft.NET\Framework\v1.1.4322;%path%
مع تغيير اسم الطراز (v1.1.4322) حسب الموقف.

ترجمة البرامج

اكتب البرنامج بمحرر نصوص مثل برنامج النوتة (Notepad.exe)
واحفظه كملف بالامتداد cs (مثل myprogram.cs) ، ثم استخدم البرنامج
csc.exe لترجمة الملف كالأمتلة الآتية:

○ للحصول على ملف تنفيذى (myprogram.exe) استخدم الأمر
التالى:

csc myprogram.cs

○ للحصول على ملف مكتبة (myprogram.dll) استخدم الأمر:
csc /t:library myprogram.cs

○ لترجمة الملف مع ربطه بملف مكتبة dll (أو أكثر) استخدم الأمر:

csc /r:lib1.dll,lib2.dll mprogram.cs

حيث lib1.dll, lib2.dll هي ملفات المكتبة المطلوب ربطها بالبرنامج myprogram.exe.

واستخدام المترجم من بيئة الأوامر يعنى أنك لن تتلقى مساعدة فى كتابة الكود كما هو الحال فى بيئة الإستويو.

أهم الفوارق بين سي# و سي++

لعلك – إذا كنت قادماً من خلفية لغة سي++ – تنتظر مقارنة بين اللغتين لكى تلم بسرعة بما سوف تجده وما لن تجده فى لغة سي#. وفى هذا الجدول نعرض ملخصاً سريعاً يحتوى أهم بنود المقارنة بين اللغتين. ومع ذلك فلو كنت جديداً على كل من اللغتين فلتتخطى هذه الفقرة وتنتقل إلى الباب الثانى مباشرة.

المقارنة	الخاصية
لا تسمح بها سي#.	المتغيرات العامة (global variables)
بينما لا تحتوى لغة سي# على مؤشرات ، لكن هناك طريقة لتميرير البارامترات كمراجع باستخدام ref و .out.	تمرير المؤشرات إلى الدوال (Pointer parameters)
يمكنك استخدام المؤشرات فى لغة سي# باستخدام	المؤشرات (Pointers)

خاصية الكود غير المأمون (Unsafe) (code).	
تختلف الحرفيات تماماً عنها في لغة سي++.	الحرفيات (Strings)
لا يمكنك استخدام متغير قبل شحنه بقيمة ما.	شحن المتغيرات المحلية) Initialization of local (variables)
لا يمكنك التحكم في هدم الأهداف كما في لغة سي++ ، حيث أن عمليات الهدم تتم أوتوماتيكياً بفضل خاصية جامع القمامة (Garbage Collector).	أساليب الهدم (Destructors) (
تمثل دوال البناء في لغة سي++ . إذا لم تكتب دالة بناء في سي# فإن المترجم يستخدم دالة بناء سابقة التعريف ، وتشحن الحقول بالقيم سابقة التعريف.	دوال البناء) (Constructors)
تعتمد لغة سي# على مكتبة المعمار دوت.نت في أداء عمليات الخرج والدخل.	الدخل والخرج (IO)
لا توجد هذه الخاصية في سي#. استخدم طريقة التحميل الزائد للأساليب (Method overloading) لتحقيق نفس الغرض.	البارامترات سابقة التعريف للأساليب) (Default parameters)
الأنماط العامة هي البديل للنماذج في لغة سي++ (ومع ذلك فإنها سوف تنشر مع الطراز VS 8.0 من الاستوديو).	الأنماط العامة (Generics)
تسمح اللغة بوراثنة فصيلة واحدة فقط ، بخلاف سي++ التي تسمح بالوراثة المتعددة.	الوراثة (Inheritance)

تطبيق الوصلات البينية (Interface implementation)	تسمح سى# للفصيلة (أو الوصلة البينية) بتطبيق أكثر من وصلة بينية فى نفس الوقت.
المصفوفات (Arrays)	تختلف سى# فى طريقة الإعلان عن ومعالجة المصفوفات.
تحويل الأنماط (Type conversion)	لا يمكن التحويل ما بين بعض الأنماط وبعضها مثل النمط البوليانى (bool) والنمط الصحيح (int). لاحظ فى لغة سى++ أن القيمة صفر تعادل القيمة false. ليس هذا هو الحال مع سى#.
النمط الطويل (long)	بينما يساوى طوله 32 بت فى سى++ فإن طوله يساوى 64 بت فى سى#.
الفصائل والمنشآت (Classes and Structs)	بينما فى لغة سى++ يمكنك الإعلان عن الفصيلة بكلمتى struct أو class ، ولكن الكلمتين فى سى# تختلفان اختلافاً جوهرياً.
المندوب (delegate)	المندوب فى سى# يناظر المؤشر إلى دالة (Function pointer) فى لغة سى++. مع الفارق الأساسى فى أن المندوب مأمون الاستخدام (Safe).
استدعاء أعضاء الفصيلة الموروثة من الفصائل المشتقة	تستخدم الكلمة base لتحقيق هذا الغرض فى سى#.
ركوب الأساليب (override)	يلزم استخدام الكلمة override فى لغة سى# لإعلان الأساليب الراكبة.

<p>لا توجد في سي# . ومع ذلك فإن توجيهات المعالج المبدئي (Preprocessor directives) تستخدم في الترجمة المشروطة (Conditional compilation) . كما يستخدم التوجيه using في الإشارة إلى الأنماط المختلفة بحيزات الاسم والاستغناء عن التأهيل الكامل للأسماء عند استخدامها.</p>	<p>ملفات العناوين (Header files)</p>
<p>علاوة على الكلمات try, catch, throw فإن سي# تستخدم كلمة مفتاحية جديدة هي finally .</p>	<p>الاستثناءات أو معالجة الأخطاء (Exception handling).</p>
<p>تستخدم لغة سي# مجموعة جديدة من المؤثرات كما أضافت استخدامات جديدة لبعض مؤثرات لغة سي++ .</p>	<p>المؤثرات (Operators)</p>
<p>غيرت سي# في مدلول بعض الكلمات المفتاحية مثل static و extern .</p>	<p>استخدام الكلمات المفتاحية (Keywords)</p>
<p>لا تستخدم هذه الطريقة في بناء الفصيلة الموروثة كما في سي++ . بدلاً من ذلك تستخدم دوال البناء.</p>	<p>الشحن بالقائمة (Initialization list)</p>
<p>يختلف اسم وطريقة إعلان الدالة الرئيسية عنه في سي++ .</p>	<p>الدالة الرئيسية Main</p>

تذكر هذه المصطلحات

Class	فصيلة
Interface	وصلة بينية
Object	هدف
Instance	مثال
Inheritance	الوراثة
Base class	فصيلة الأساس (الموروثة)
Derived class	الفصيلة المشتقة (الوارثة)
Abstraction	التجريد
Encapsulation	الكبسلة
Polymorphism	تعدد الأشكال
.NET	المعمار دوت.نت
Intermediate Language	اللغة الوسطى IL
Command Line/Prompt	بيئة الأوامر / خط الأوامر
Microsoft Visual Studio	الإستوديو المرئي
.NET	المعمار دوت.نت
Safe code	الكود المأمون
Unsafe code	الكود غير المأمون
Garbage Collector	جامع القمامة