الباب الأول

البرمجة الموجهة نحو الأهداف

Object Oriented Programming (OOP)

مفتتح

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

فلسفة البرمجة الموجهة نحو الأهداف (OOP)

بـــدأت فلسفة البرمجة الموجهة نحو الأهداف (Object Oriented Programming) مـــع ظهـــور لغـــة سى++ ، حيث قدمت اللغة نمط الفصـــائل (Classes) علاوة على اتجاه جديد فى البرمجة وهو التعامل مع البيانات فى صورة أهداف (Objects). وفكرة الفصائل تعتمد على محاكـــاة الطبـــيعة ، فالطبيعة تحتوى على مجموعة كبيرة من الفصائل الـــتى قد تتشابه فى خصائصها وسلوكها وقد ترث 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 فـــى بعـــض الخصائص مثل الحواس والأطراف. ومن فصيلة الإنسان تنحدر فصيلة الرجل وفصيلة المرأة. وهما ترثان من فصيلة الإنسان كل خصائصـــها وسلوكها ، علاوة على الخصائص المتفردة لكل من جنس الرجل وجنس المرأة.

وعـــندما نســـتخدم ألفـــاظ الإنسان أو الرجل أو المرأة فإننا نستخدم الـــتجريد ، بمعنى أننا لا نركز اهتمامنا على إنسان معين مثل محمد أو مـــرزوق أو زينـــب. فهذه الأســـماء المحـــددة تدل على أمـــثلـــة (Instances) مخـــتلفة (أو أهـــداف ـــ Objects) من فصيلة الإنسان. بمعنى أن الأمثلة أو الأهداف هى الصورة الواقعية للفصائل.

وفـــى مجال البرمجة فإننا فى تعاملنا مع البيانات نقسمها إلى فصائل تتشـــابه فـــى الســـلوك والخصـــائص ، وتمـــثل الصـــورة المجردة (Abstraction) للبيانات. ومن الفصائل نخلق الأمثلة (أو الأهداف) التى تمـــثل الصـــورة الواقعـــية 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) الصـــفات والسلوك فى كبسولة واحدة وهى الفصيلة. ومن الجائز أن يكتـــب الفصـــيلة مـــبرمج ما ، أما خلق الأهداف واستخدامها فيقوم به مـــبرمج آخـــر ــ ربما فى شركة أخرى. كما يمكننا أن نورث الفصيلة لنبـــتكر مـــنها 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 مـــن الصفر ، إننا فقط نضيف إلى فصيلة الإنسان صفات الموظف مثل اســـم الشركة التى يعمل بها ، وسلوك الموظف مثل قيامه بالعمل عدداً معينا من الساعات كل شهر. كما أن البرمجة الموجهة نحـــو الأهـــداف لهـــا ممـــيزات أخرى تـــوفر الـــوقت والجهد مثل تعـــــدد الأشكال (Polymorphism). وعلى سبيل المثال فإنك تستطيع فى برنامج ما أن تبـــنى دالة واحدة للرسم مثل "Draw" ، وبحسب نوع الهدف المرتبط

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

ولتلخـــيص الموضـــوع فإننا نعرض هنا المبادىء 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 هى البداية الأولى لاستخدام البرمجة الموجهـــة نحـــو الأهـــداف ولكن اللغة كانت مزيجاً من الاتجاهين فى الـــبرمجة: الاتجاه التقليدى المشابه للغة سى واللغات الأخرى القديمة ، والاتجاه الجديد فى البرمجة الموجهة نحو الأهداف. ولذلك فإنك تستطيع باستخدام لغة سى++ أن تكتب برنامجاً محكماً يغلف البيانات فى صورة كبسولة من الحقول والدوال ، كما يمكنك أن تكتب برنامجاً مهلهلاً يعتمد على المتغيرات العامة (Global Variables) أو يحتوى على مجموعة

من المؤشرات بمشاكلها التى تؤدى إلى تبديد الذاكرة وإضاعة الوقت فى صيانة البرنامج.

أمـــا لغـــة سى# (وتُنطق سى شارب) فقد تم بناؤها من الصفر كلغة موجهـــة نحـــو الأهداف. بمعنى أنه لا يمكنك أن تنشىء برنامجاً بدون وضـــعه بداخـــل كبســـولة الفصيلة ، علاوة على ما يلزم الفصيلة من خصائص تساعدك على بناء الأهداف. لا يمكنك أن تكتب برنامجاً بلغة سى # محتوياً على بعض المتغيرات المبعثرة فى البرنامج والتى تؤدى إلى المشكلات التقليدية التى دأب المبرمجون على حلها جيلاً بعد آخر. والحقـــيقة أن لغـــة ســـى# ، كتطوير للغة سى++ ، لم تقدم فقط بعض الملامح الجديدة ، بل تخلصت من الكثير من ملامح سى++ التى كانت تتسبب فى الأخطاء وتؤدى إلى المشكلات المعروفة.

وسوف نعرض مقارنة كاملة بين لغة سى# ولغة سى++ فى الفقرات القادمة.

المعمار دوت.نت (.NET)

فـــى الحقـــيقة أن لغة سى# هى إحدى اللغات الجديدة التى تقع تحت مظلـــة المعمـــار دوت.نـــت. فهـــناك لغـــات أخـــرى ابتكرتها شركة ميكروسوفت مثل .NET Visual Basic و J# وامـــتداد لغة سى++ (C++ Managed Extensions). جمـــيع هـــذه اللغات ، علاوة على لغات أخرى قدمتها شركات مختلفة ، تقوم تحت مظلة المعمار دوت.نت

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

بيئة تنفيذ البرامج (Common Language Runtime)

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

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

الـ ميتا ـ 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) وبارامتراتها. وتسمى هذه المعلومات "ميتا ــ داتا" بمعنى معلومات عن البيانات. وهذه المعلومات

لهـــا اســـتخدامات كثـــيرة منها تمكين المبرمج من إنشاء الأهداف من الفصائل بصرف النظر عن اللغة التى كتبت بها الفصائل.

المـجمّعات (Assemblies)

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

نظرة سريعة على الكود

لـــنلق نظـــرة سريعة على مثال للفصيلة بدون الدخول فى تفصيلات اللغة:

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

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

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

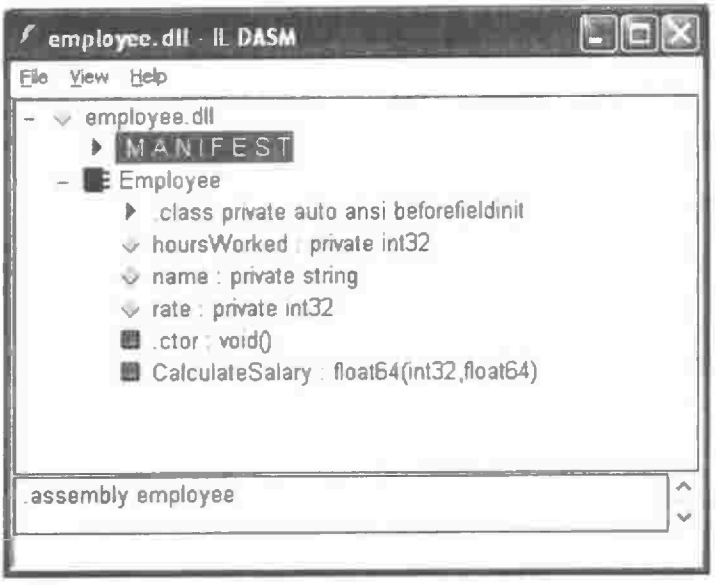
فـــى هذا المثال ، بصرف النظر عن اللغة ، نرى فصيلة الموظف التى تحتوى على خصائص الموظف وهى الحقول:

- الاسم (name)
- عدد ساعات العمل (hoursWorked)
- معدل الأجر (rate)

كما نرى سلوك الفصيلة وهو الدالة أو الأسلوب المستخدم فى حساب المرتـــب (CalculateSalary). وهـــذا الأســـلوب يســـتخدم بعـــض البارامـــترات (hoursWorked وrate) ومنها يخرج بالنتيجة النهائية وهى المرتب salary.

وبالطـــبع فـــإن الفصيلة يمكن أن تحتوى على تفصيلات 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 ــ داتا التى تحتوى عليها المكتبة (يستخدم فى ذلك الـــبرنامج 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

وكمـــا نرى بالشكل أن المنافستو يحتوى على حقول الفصيلة الثلاثة وعلـــى الأسلوب CalculateSalary() والبارامترات المستخدمة فيه. وهـــذا هـــو كـــل ما يلزم المبرمج الذى يريد استخدام هذه الفصيلة فى بـــرنامجه. بمعـــنى آخـــر فـــإن المبرمج لا يحتاج الاطلاع على الكود المصـــدر (Source code) لكى يستخدم الفصيلة. ويمكنك أن تحصل على هذا المنافستو باستخدام الأمر:

ILDasm Employee.dll

(بفرض أن اسم الملف هو Employee.dll).

وباســـتخدام الأمر ILDASM يمكنك الاطلاع على منافستو أى ملف تنفـــيذى (exe) او مكتـــبة (dll) فتحصل على الفصائل بما تحتويه من

أعضـــاء. كمـــا أنـــك لو ضغطت ضغطة مزدوجة على أحد العناصر الموجـــودة بالشـــكل السابق فإنك ترى كود اللغة الوسطى المناظر لكل عنصر.

برامج ترجمة اللغة الوسطى

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

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

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

كيف تحصل على مترجم سى# ؟

يمكنك أن تبنى برامجك المكتوبة بلغة سى# إما باستخدام المترجم فى بيـــئة الأوامـــر (Command Line Compiler) أو باســـتخدام البيئة المجمعـــة (IDE) الموجـــودة بالإستوديو المـــرئى (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)

هـذا هـو أسـهل الطرق للحصول على مترجم سى# (علاوة على مـترجم لغـة VB .NET ومترجم #J). ويمكنك إنزال هذه الحزمة من الإنترنت عند العنوان:

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

ثم اضغط على النص التالى:

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

والحـزمة عبارة عن ملف واحد يحمل الاسم dotnetfx.exe. وبتنفيذ هذا البرنامج يتم تركيب دوت.نت.

الحزمة إس ـ دى ـ كى (Microsoft .NET Framework SDK)

يمكنك إنزال هذه الحزمة من الإنترنت عند نفس العنوان:

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

ثم اضغط على النص التالى:

How to Get the Microsoft .NET Framework 1.1

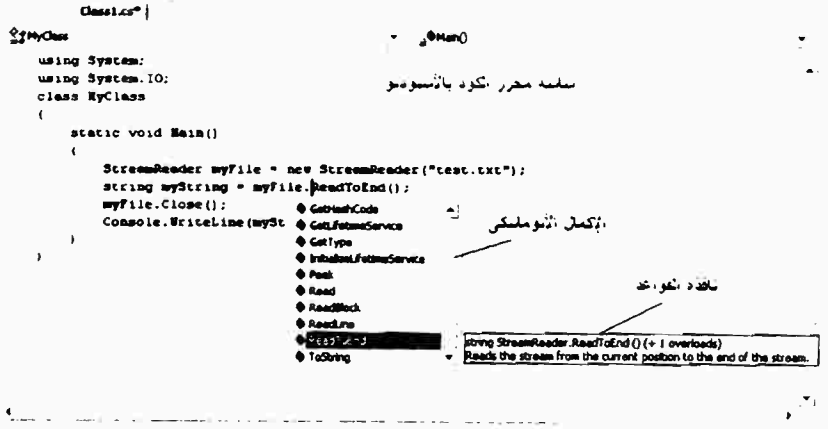
ثم اتبع الخطوات الموضحة حتى تصل إلى إنزال الحزمة:

.NET Framework 1.1 SDK

وتحتوى هذه الحزمة على المعمار دوت.نت علاوة مجموعة من الوثائق والمـنافع. ويتطلـب تركيب هذه الحزمة وجود الحزمة دوت.نت على الكومبـيوتر. والحـزمة عـبارة عـن ملف تنفيذى يحمل اسماً مثل "NDP10_SP_Q321884_En.exe" ، وبتنفـيذ هـذا الـبرنامج يتم تركيب الحزمة SDK.

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

لـــو أنك استخدمت الإستوديو فإنك سوف تحصل على عدة تسهيلات فى الـــبرامج ، حيـــث يحتوى الإستوديو على محرر خاص للبرامج علاوة علـــى أدوات لتصـــميم التطبـــيقات المخـــتلفة مثل التطبيقات النوافذية وتطبـــيقات الـــوب وقد خصصنا الباب الحادى عشر لجولة سريعة فى الإســـتوديو. وبصفة عامة فإن أهم وسائل المساعدة هى نافذة القواعد (IntelliSense) التى تصاحبك كلما حركت الفأر فوق عنصر من عناصر اللغـــة 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.



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

الحزمة السريعة سى# اكسبريس (C# Express)

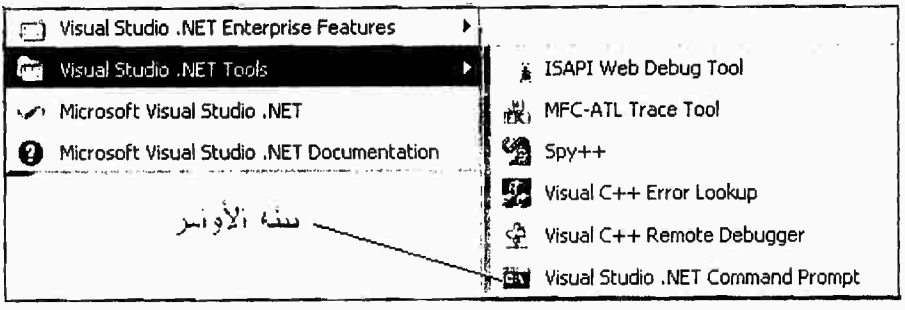
وتـــوزع هذه الحزمة مجاناً على الإنترنت ، وسوف يتم توزيعها فى بدايـــة عـــام 2005. كما ستوزع مجموعات أخرى من الحزم السريعة للغـــات الأخرى بغرض تشجيع الهواة والمبتدئين على استخدام منتجات ميكروسوفت.

تشغيل مترجم سى# فى بيئة الأوامر

نقـــدم فـــى هـــذه الفقرة طريقة إعداد وتشغيل مترجم سى# فى بيئة الأوامر.

الإعداد للتشغيل

إذا كنـــت تســـتخدم برنامج الإستوديو فإنك لا تحتاج إلى إعداد بيئة الأوامـــر. كـــل ماعلـــيك أن تســـتخدم أداة بيئة الأوامر التى تصاحب الإستوديو كما بالشكل التالى:



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

أمـــا إذا كنـــت قد انزلت دوت.نت أو الحزمة SDK من الإنـــترنت ، فعلـــيك أن تضـــــع 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:

لكـــى 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

اكتـــب الـــبرنامج بمحـــرر نصوص مثل برنامج النوتة (Notepad.exe) واحفظـــه كملف بالامتداد cs (مثل myprogram.cs) ، ثم استخدم البرنامج **csc.exe** لترجمة الملف كالأمثلة الآتية:

- للحصـــول علـــى ملف تنفيذى (myprogram.exe) استخدم الأمر التالى:

```
csc myprogram.cs
```

- للحصول على ملف مكتبة (myprogram.dll) استخدم الأمر:

```
csc /t:library myprogram.cs
```

- لـترجمة الملـف مـع ربطه بملف مكتبة dll (أو أكثر) استخدم الأمر:

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

حيـث lib1.dll, lib2.dll هـى ملفـات المكتـبة المطلوب ربطها بالبرنامج myprogram.exe.

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

أهم الفوارق بين سى# و سى++

لعلـك ـ إذا كنت قادماً من خلفية لغة سى++ ـ تنتظر مقارنة بين اللغتين لكى تلم بسرعة بما سوف تجده وما لن تجده فى لغة سى#. وفى هـذا الجـدول نعـرض ملخصاً سريعا يحتوى أهم بنود المقارنة بين اللغتيـن. ومـع ذلك فلو كنت جديداً على كل من اللغتين فلتتخطى هذه الفقرة وتنتقل إلى الباب الثانى مباشرة.

الخاصية	المقارنة
المتغـيرات العامة (global variables)	لا تسمح بها سى#.
تمرير المؤشرات إلى الدوال (Pointer parameters)	بيـنما لا تحـتوى لغة سى# على مؤشرات ، لكن هناك طريقة لتمرير البارامترات كمراجع باستخدام ref و out.
المؤشرات (Pointers)	يمكـنك استخدام المؤشرات فى لغة سى# باستخدام

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

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 البولياني (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.

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

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 المرئى
.NET	المعمار دوت.نت
Safe code	الكود المأمون
Unsafe code	الكود غير المأمون
Garbage Collector	جامع القمامة