Following the rules

As with all programming languages, C# has a number of syntax rules that must be precisely followed to ensure the code is correctly formatted for the C# compiler to clearly understand:

- **Case-sensitivity** – C# is a case-sensitive language, which means that uppercase "A" and lowercase "a" are regarded as totally different items.
- **Termination** – All statements in C# language must be terminated by a **;** semicolon character, just as all sentences in English language must be terminated by a **.** period character. For example: `Console.WriteLine( "Hello World!" ) ;`
- **Single-line comments** – Brief comments on a single line must begin with // two forward slash characters. For example: `// Output the traditional greeting.`
- **Block comments** – Extended comments on multiple lines must begin with /* forward slash and asterisk characters, and must end with the reverse */ asterisk and forward slash. For example:

```
/*
  C# Programming in easy steps.
  Getting started with the traditional greeting.
*/
```

- **White space** – Spaces, tabs, newline characters, and comments are ignored by the C# compiler, so can be used extensively to organize code without performance penalty.
- **Escape sequences** – The C# compiler recognizes **\n** as a newline character and **\t** as a tab character, so these can be used to format output. For example: `Console.WriteLine("Line One \n Line Two") ;`
- **Naming conventions** – A programmer-defined identifier name in C# code may begin with an _ underscore character or a letter in uppercase or lowercase. The name may also contain an underscore, letters, and numerals. For example: `class MyNo1_Class`
- **Keywords** – The C# language has a number of keywords (listed opposite) that have special syntactic meaning and may not be used to name programmer-defined items in code.

Hot tip

It is recommended that you comment your code to make it readily understood by others or when revisiting your own code later.

Don't forget

The **WriteLine( )** method automatically adds a newline after its output.

C# Reserved Keywords

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

Hot tip

If you absolutely must use a keyword to name a programmer-defined element, it may be prefixed by an **@** character to distinguish it from the keyword – but this is best avoided.

C# Contextual Keywords

add	alias	ascending	async
await	descending	dynamic	from
get	global	group	into
join	let	orderby	partial
remove	select	set	value
var	where	yield	

Don't forget

Contextual keywords have special significance in certain code. For example, **get** and **set** in method declarations.