الدرس السابع: switch و do while

اليوم نكمل معا المشوار في تعلم أساسيات لغة السي شارب ، و سوف نتكلم أولا عن عبـارة مفيـدة جـدا ألا و هي **switch** .

أخذنا في درس سابق **if** و كيفية استعمالها ، وقلنا أنها لا أن تحوي على شرط واحد أو قل حالة واحـدة ، و لكن تصور معي المسألة التالية ، صباغ أتي إليك و طلب منك أن تبرمج له برنامجا بحيث يدخل هو رمـز اللون و يظهر له البرنامج اسم اللون ، و أعطاك الجدول التالي للألوان :

رمز اللون	اسم اللون	
1	**Red**	أحمر
2	**Yellow**	أصفر
3	**Green**	أخضر
4	**Black**	أسود
5	**White**	أبيض
6	**Blue**	أزرق
7	**Orange**	برتقالي

لتبرمج هذا البرنامج ستحتاج إلى 7 جملة **if** ، وهذا شيء ممل ، أليس كذلك ؟؟

الخبر السار لك أن هذه سبع الجمل يمكنك أن تكتبها في جملة واحدة من جمل **switch** ، تستخدم جملة **switch** للاختيار المتعدد مثل الحالة السابقة ، و هيكل **switch** كالتالي :

```
switch (اسم المتغير)
{
        case الحالة الأولى:
                        افعل كذا و كذا;
                        break;
        case الحالة الثانية:
                        افعل كذا و كذا;
                        break;
        case الحالة الثالثة:
                        افعل كذا و كذا;
                        break;
}
```

دعني أعطيك برنامج الصباغ :

```
// برنامج الصباغ
using System;
class Pointer
{
    static void Main()
      {
            int codeNumber;
            //أدخل الرمز اللوني
            Console.Write("Enter Code Number :");
            codeNumber = Int32.Parse(Console.ReadLine());

            switch (codeNumber)
            {
                  case 1:
                        Console.WriteLine("Red");
                        break;
                  case 2:
                        Console.WriteLine("Yellow");
                        break;
                  case 3:
                        Console.WriteLine("Green");
                        break;
                  case 4:
                        Console.WriteLine("Black");
                        break;
                  case 5:
                        Console.WriteLine("White");
                        break;
                  case 6:
                        Console.WriteLine("Blue");
                        break;
                  case 7:
                        Console.WriteLine("Orange");
                        break;
            }
      }
}
```

و النتيجة التالي:

```
Enter Code Number :5
White
```

و الآن بعد هذا المثال تبين لك كيفية استعمال **switch** ، تلاحظ أننا أخذنا متغيرا و قارناه مع عدد من القيم ، فإذا كان يساوي إحداها ، فإنه ينفذ مجموعة الجمل التي تأتي مع تلك القيمة .

و لا بد مع كل حالة من حالات **switch** أن تنتهي بالجملة **break** أو ما يعادلها [سنأخذها فيما بعد] بحيث لا تتصل جملة **case** بأخرى ، حيث تقوم **break** بإنهاء جملة **switch** كليا .

فإذا نسيت أن تضع جملة **break** في نهاية جملة **case** فإن المترجم سوف يعطيك خطأ في الترجمة .

و يستثنى من هذه القاعدة إذا كانت جملة case فارغة حيث يقوم البرنامج بقـراءة جملـة case التـي تليها ، فإذا لم تكن فارغة ينهيها بغض النظر إذا كانت قيمتها مساوية للمتغير أم لا .

خذ المثال التالي : معلم أراد أن يحسب عدد الذين أخذوا A ، و الذين أخذوا B ، و الـذين أخـذوا C ، في فصل مكون من عشرة طلاب ، بحيث يقوم هو بإدخال أحد الحروف الثلاثة السابقة و البرنامج يعمل إحصائية للدرجات ، و قد كتب البرنامج التالي :

```
// الإحصاء
using System;
class Switch
{
      static void Main()
      {
            char grade;
            int aCount = 0, bCount = 0, cCount = 0;
            for (int i = 1; i <= 10; i ++)
            {
                  Console.Write("Enter a letter of grade :");
                  grade = Char.Parse(Console.ReadLine());

                  switch (grade)
                  {
                        case 'A' :  //سوف ينفذ المترجم الجملة التالية
                        case 'a' :
                              ++aCount;
                              break;
                        case 'B' :
                              goto case 'b';
                        case 'b' :
                              ++ bCount;
                              break;
                        case 'c' :
                        case 'C' :
                              ++cCount;
                              break;
                        default:
                              Console.WriteLine("Incorrect letter");
                              break;
                  }
            }
            Console.WriteLine("\n Total: A = " + aCount + "  B = "
                              +  bCount + "  C = " + cCount);
      }
}
```

و النتيجة كالتالي :

```
Enter a letter of grade :a
Enter a letter of grade :c
Enter a letter of grade :c
Enter a letter of grade :c
Enter a letter of grade :b
Enter a letter of grade :c
Enter a letter of grade :b
Enter a letter of grade :a
Enter a letter of grade :b
Enter a letter of grade :c

Total: A = 2  B = 3  C = 5
```

قد تلاحظ أننا أضفنا إلى **switch** جملة **default** ، تنفذ هذه الجملة بشكل افتراضي إذا لم توجد جملة **case** تساوي المتغير .

و أيضا أضفنا **goto** و هي تعادل جملة **break** و معناها اذهب إلى ، فمثلا **goto case4** معناها اذهب إلى جملة **case4** و **goto default** معناها اذهب إلى الجملة الافتراضية و هكذا ..

الحلقة التكرارية do ... while

تتميز الحلقة التكرارية **do ... while** عن بقية الحلقات بأن الجمل التي تحتويها يجب أن تنفذ مرة واحدة على الأقل ، و هيكلها كالتالي :

```
do {
        افعل كذا وكذا ;
   }
while (الشرط) ;
```

فمن هذه الهيكلة يتبين لك كيف أن جمل **do** يجب أن تنفذ مرة واحدة على الأقل .

ما رأيك الآن أن ترجع إلى برنامج الصباغ ؟ قم بتعديله بحيث تجعله يعمل بشكل متكرر حتى تدخل الرقم -1 و ذلك باستخدام **while** ، و تأكد من أنه يعمل بشكل جيد ...

و الآن دعني أكتب لك ذلك البرنامج بواسطة **do ... while**

```
using System;
class Class1
{
	static void Main()
	{
		int codeNumber;
		do
		{
			//أدخل الرمز اللوني
			Console.Write("Enter Code Number :");
			codeNumber = Int32.Parse(Console.ReadLine());
			switch (codeNumber)
			{
				case 1:
					Console.WriteLine("Red");
					break;
				case 2:
					Console.WriteLine("Yellow");
					break;
				case 3:
					Console.WriteLine("Green");
					break;
				case 4:
					Console.WriteLine("Black");
					break;
				case 5:
					Console.WriteLine("White");
					break;
				case 6:
					Console.WriteLine("Blue");
					break;
				case 7:
					Console.WriteLine("Orange");
					break;
				case -1:
					Console.WriteLine("Exit");
					break;
			}
		}
			while (codeNumber != -1);
	}
}
```

والنتيجة :

```
Enter Code Number :1
Red
Enter Code Number :2
Yellow
Enter Code Number :3
Green
Enter Code Number :4
Black
Enter Code Number :5
White
Enter Code Number :6
Blue
Enter Code Number :7
Orange
Enter Code Number :-1
Exit
```

أرأيت كيف سهلت علينا **while** ... **do** المهمة ؟!

جمل **break** و **continue**:

لقد استخدمنا سابقا جملة **break** لجعل جملة **switch** تنتهي ، وهي كذلك تستخدم نفس الطريقة لجعل الحلقات التكرارية تنتهي (**for**,**while**,**do**...**while**) .

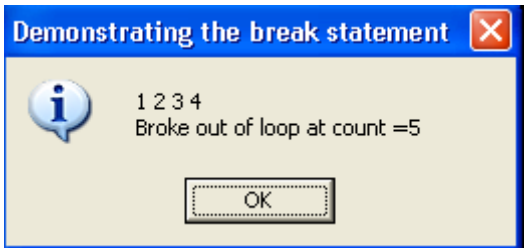
المثال التالي يوضح الكيفية التي تعمل بها **break** لجعل البرنامج يخرج عن الحلقة التكرارية **for**:

```
using System;
using System.Windows.Forms;

class BreakTest
{

      static void Main()
      {
            string output = " ";
            int count;
            for (count = 1; count <= 10; count++)
            {
                  if (count == 5)
                        break;
                  output += count + " ";
            }
            output += "\n Broke out of loop at count =" + count;
            MessageBox.Show(output, "Demonstrating the break statement",
            MessageBoxButtons.OK,MessageBoxIcon.Asterisk);
      }
}
```

و المخرجات :



ترى في هذا البرنامج أنه لما **count** وصلت إلى 5 نفذ جملة **break** فخرج الحلقة التكرارية **for** .

في الجانب الآخر توجد جملة **continue** و التي تقوم بإنهاء الدورة الحالية للحلقة التكرارية و يبدأ من جديد .

و الاستثناء الوحيد من هذه القاعدة هو في الحلقة التكرارية **for** حيث يقوم البرنامج بتخطي جملة **for** حتى يصل إلى الجملة التي تزيد أو تنقص (الجملة الثالثة راجع الدرس السابق) من قيمة المتحكم فينفذها ثم يبدأ من جديد .

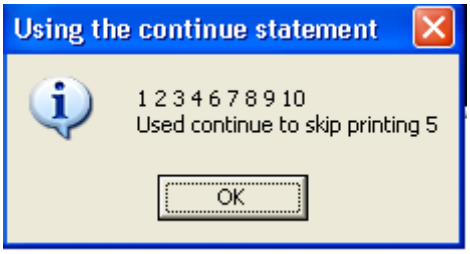
المثال التالي يوضح هذه الفكرة :

```
using System;
using System.Windows.Forms;

class continueTest
{

      static void Main()
      {
            string output = " ";
            for (int count = 1; count <= 10; count++)
            {
                  if (count == 5)
                        continue;
                  output += count + " ";
            }
            output += "\n Vsed continue to skip printing 5 ";
            MessageBox.Show(output, "Using the Continue statement",
            MessageBoxButtons.OK,MessageBoxIcon.Asterisk);
      }
}
```

و المخرجات كالتالي :



حيث استخدمنا جملة **continue** لتخطي طباعة الرقم خمسة .

العوامل المنطقية و العلاقات :

قد تكون تساءلت هل هناك إمكانية لدمج شرطين معا ، كأن تقول لو حدث كذا و كذا أو تقول لو أن حدث كذا أو كذا ؟ سأخبرك بأن هذا موجود في لغة السي شارب ، ويسمى عوامل العلاقات ، و سنأخذها واحدة واحدة .

أولا : عامل العلاقة " و " و يمثل بـ &&

و هو عامل ثنائي يأخذ حدين ، الجدول التالي يصف الاحتمالات التي يمكن أن يوجد بها ذانك الحدين :

الحد الأول	الحد الثاني	ناتج علاقة " و " &&
خطأ	خطأ	خطأ
صح	خطأ	خطأ
خطأ	صح	خطأ
صح	صح	صح

حيث نرى أن هذه العلاقة تستلزم أن يكون الحدان كلاهما صحيح حتى تكون هي صحيحة .

ثانيا :عامل العلاقة " أو " و يمثل بـ ||

و هو عامل ثنائي يستلزم حدين ، ويكون صحيح بمجرد أن يكون أحد الحدين صحيح ، و الجدول التالي يمثل ذلك :

الحد الأول	الحد الثاني	ناتج علاقة " أو " ||
خطأ	خطأ	خطأ
صح	خطأ	صح
خطأ	صح	صح
صح	صح	صح

و طريقة تنفيذ هذان العاملين تبدأ باختيار الحد الذي في اليسار قم الحد الذي في اليمين .

مثلا :

```
gender == 1     &&     age >= 65
```

في هذا الشرط عندنا حدان الحد الأول أن يكون الجنس ذكرا و أن يكون العمر أكبر من 65 ، لكي نعتبر الشخص أنه مسن .

فإذا كان الحد الذي في اليسار خطأ فإن البرنامج لا يختبر الحد الموجود في اليمين ، و يعتبر هذا الشرط خطأ لأن عامل " و " && يستلزم أن يكون كلا الحدين صحيح ، و بما أن الحد الأول خطأ فالشرط إذا خطأ .

مثال على عامل " أو "

```
cost == 50     ||     color == "Red"
```

في هذا الشرط عندنا حدان ، أن يكون السعر يساوي 50 أو اللون أحمر ، فإذا كان السعر يساوي 5 فإن البرنامج لا يختبر الحد الثاني (اللون الأحمر) ، و إنما مباشرة يضع ناتج الشرط صح ؛ لأن أو || تستلزم أن يكون أحد الحدين صحيحا ليكون الناتج صحيحا ، وبما أن السعر صحيح فلا داعي لاختبار الحد الثاني .

و هنا ترى كيف أن فهم عمل العاملين " أو " و " و " سوف يساعدك لجعل برنامجك يعمل بسرعة أكبر ، فاحرص على أن يكون الحد الذي في اليسار احتمالية خطئه أكثر من الذي في اليمين في علاقة " و " ، و احتمالية صحته أكثر في علاقة " أو " ، حتى تتفادى اختبار الحد الثاني .

و الآن نأتي إلى العوامل المنطقية ،وهي **AND** ، **OR** ، **NOT** ، **XOR**

أولا : AND ، وتمثل بـ & :

هذا العامل نفس العامل المنطقي " و " إذا استعملناه بين حدين منطقيين أي يكون نتاجهما صح أو خطأ ، يكمن الاختلاف أنه يقوم البرنامج باختبار الحدين أولا ، أي يختبر هل هما صح أم خطأ ، ثم يقوم بمقارنة بينهما ثانيا ، هذا إذا كان الحدين منطقيين [نتاجهما **bool**] أما إذا كان عدد صحيح [**int, uint , long, ulong**] فإنها تقوم بتحويلها إلى قيمتها في النظام الثنائي ثم يقوم بعملية **AND** عليهما ، ثم يحولها إلى نظام العشري ، و من الصعب علي شرح لك هذه العلميات بدون شرح نظام العد الثنائي و الذي هو علم بحد ذاته ، و هو بطبيعة الحال خارج نطاق هذه الدروس.

ثانيا : عملية OR و تمثل بـ | :

نفس الكلام الذي قلناه عن **AND** ينطبق على **OR** ، فهي مشابهة لعمل علاقة " أو " إذا كان الحدين منطقيين [من نوع **bool**] .

ثالثا : عملية NOT و تمثل بـ ! :

تقوم هذه **NOT** بقلب الصح إلى خطأ و الخطأ إلى صح و هي معامل أحادي يحتاج إلى حد واحد فقط .

رابعا : عملية XOR وتمثل بـ ^ :

هذه العملية مهمة جدا ، و سوف تواجهها كثيرا ، و هي تعطي ناتج صحيح إذا كان أحد العاملين صح و الآخر خطأ فقط ، الجدول التالي يبين لك هذه :

الحد الأول	الحد الثاني	ناتج عملية XOR
خطأ	خطأ	خطأ
صح	خطأ	صح
خطأ	صح	صح
صح	صح	خطأ

و لتقريب كيفية استعمال هذه العملية ، تصور أنك تعمل محرر نصوص ، و أردت أن تعمل خاصية الخط العريض **Bold** ، فعندما تختار النص الذي تريد أن تطبق عليه خاصية الخط العريض فهو يكون على إحدى الحالتين عريض (صح) ،غير عريض (خطأ) ، فينتج مع الاحتمالات التالية :

الخاصية الحالية	الخاصية المنشودة	ناتج عملية XOR
عريض	عريض	غير عريض
غير عريض	عريض	عريض

و هذا بالفعل هو الذي يحدث في برنامج **word** ، اذهب إليه و جربه بنفسك !

و الآن سوف أختم هذا الدرس بمثال على هذه العمليات :

```
using System;

class logicalOpertors
{
      static void Main()
      {

            Console.WriteLine("Conditional AND (&&)" +
                  "\n false && false = " + (false && false) +
                  "\n false && true = " + (false && true) +
                  "\n true && false = " + (true && false) +
                  "\n true && true = " + (true && true ));
            Console.WriteLine("Conditional OR (||)" +
                  "\n false || false = " + (false || false) +
                  "\n false || true = " + (false || true) +
                  "\n true || false = " + (true || false) +
                  "\n true || true = " + (true || true ));
            Console.WriteLine("logical AND (&)" +
                  "\n false & false = " + (false & false) +
                  "\n false & true = " + (false & true) +
```

```
                  "\n true & false = " + (true & false) +
                  "\n true & true = " + (true & true ));
            Console.WriteLine("logical OR (|)" +
                  "\n false | false = " + (false | false) +
                  "\n false | true = " + (false | true) +
                  "\n true | false = " + (true | false) +
                  "\n true | true = " + (true | true ));
            Console.WriteLine("logical XOR (^)" +
                  "\n false ^ false = " + (false ^ false) +
                  "\n false ^ true = " + (false ^ true) +
                  "\n true ^ false = " + (true ^ false) +
                  "\n true ^ true = " + (true ^ true ));
            Console.WriteLine("logical NOT (!)" +
                  "\n  ! false = " + (! false) +
                  "\n  ! true = " + (! true));

      }
}
```

والنتيجة :

```
Conditional AND(&&)
 false && false = False
 false && true = False
 true && false = False
 true && true = True
Conditional OR(||)
 false || false = False
 false || true = True
 true || false = True
 true || true = True
logical AND(&)
 false & false = False
 false & true = False
 true & false = False
 true & true = True
logical OR(|)
 false | false = False
 false | true = True
 true | false = True
 true | true = True
logical XOR(^)
 false ^ false = False
 false ^ true = True
 true ^ false = True
 true ^ true = False
logical NOT(!)
 !  false = True
 !  true = False
```

آه نسيت أن أعطيك الجدول الذي يبين لك الأولوية في تنفيذ المعاملات ، وهو كالتالي

الترتيب	المعامل	طريقة تنفيذه أو الربط	نوعه
1	( )	من يسار إلى اليمين	الأقواس
2	+ + - -	من اليمين إلى اليسار	معامل أحادي للاحق
3	++ - - + - ! (type)	من اليمين إلى اليسار	معامل أحادي سابق
4	* / %	من اليسار إلى اليمين	المضروبات
5	+ -	من اليسار إلى اليمين	المجموعات (من الجمع
6	< <= > >=	من اليسار إلى اليمين	علاقات
7	== !=	من اليسار إلى اليمين	التساوي
8	&	من اليسار إلى اليمين	العملية المنطقية **AND**
9	^	من اليسار إلى اليمين	العملية المنطقية **XOR**
10	|	من اليسار إلى اليمين	العلمية المنطقية **OR**
11	&&	من اليسار إلى اليمين	علاقة " و "
12	||	من اليسار إلى اليمين	علاقة " أو "
13	?:	من اليمين إلى اليسار	علاقات الإسناد
14	= += -= *= /= %=	من اليمين إلى اليسار	

الواجب :

1 – اكتب برنامج يظهر الإشكال التالية واحدة تلو الأخرى :

```
*           **********  **********          *
**          *********    *********         **
***         ********      ********        ***
****        *******        *******       ****
*****       ******          ******      *****
******      *****            *****     ******
*******     ****              ****    *******
********    ***                ***   ********
*********   **                  **  *********
**********  *                    * **********
```

2 – اكتب برنامجا يطبع لك المعين التالي :

```
    *
   ***
  *****
 *******
*********
 *******
  *****
   ***
    *
```

3 – قم بتعديل البرنامج رقم 2 بحيث يستقبل عدد فردي من 1 إلى 19 لكي يحدد عدد الصفوف في المعين ، ثم يقوم البرنامج بطباعة ذلك المعين ؟

4 – قم بتعديل البرنامج رقم 1 لكي يطبع تلك المثلثات بحيث تكون جنبا إلى جنب ؟

****

***

*