بسم الله الرحمن الرحيم

الحمد لله رب العالمين وأفضل الصلاة وأتم التسليم على سيدنا محمد وعلى آله وصحبه أجمعين

سبحانك اللهم لا علم لنا إلا ما علمتنا إنك أنت العليم الحكيم

الأهداف التعليمية من هذا الدرس :

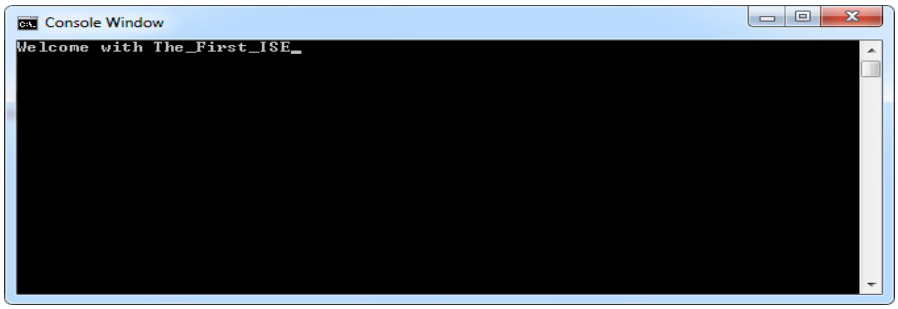
1- كتابة برامج بسيطة باستخدام الرماز (code).
2- كتابة عبارات التفاعل مع المستخدم (الدخل والخرج).
3- المتحولات (variables) وكيفية التصريح عنها واستخدامها في البرنامج.
4- التعرف على العمليات الحسابية في C#.
5- فهم ترتيب وأولوية تنفيذ العمليات الحسابية.
6- كتابة عبارات اتخاذ القرار في البرنامج (العبارات الشرطية).
7- استخدام عمليات المساواة والمقارنة في العبارات الشرطية (if statement).

1.3 : مقدمة :

في كل وقت تستخدم فيه الحاسوب ، فأنت تستخدم أنواع مختلفة من البرامج والتي تنفذ مهام مختلفة لخدمتك.

فمثلاً : برنامج البريد الالكتروني يساعدك على إرسال واستقبال الرسائل الالكترونية (E-mails) ومستعرض الويب يسمح لك باستعراض صفحات الويب من المواقع المنتشرة حول العالم ، طبعاً مبرمجون أمثالك يطورون هذه البرامج.

سوف تكون معظم البرامج التي سنكتبها في هذه السلسلة من نوع تطبيقات المترجم السطري (Console Applications) التي تسمح للمستخدم بالتفاعل معها عن طريق إدخال النصوص وعرض المعلومات المعالجة في نافذته التي تدعى أيضاً (Command Prompt).



2.3 : برنامج بسيط يعرض رسالة ترحيبية للمستخدم :

دعنا نبدأ رحلتنا الاستكشافية للغة سي شارب عبر برنامج بسيط يعرض رسالة ترحيبية للمستخدم (سنناقش فيما يأتي القواعد اللغوية المتبعة في كتابة هذا البرنامج) ، والبرنامج وخرجه مكتوب بالشكل 1.3 حيث سيوضح لنا هذا البرنامج ميزات هامة للغة سي شارب.

سي شارب تستخدم نظام تدوين (notations) ربما يبدو غريباً لغير المبرمجين ، دعنا نتفق بداية على أنه كل برنامج مكتوب في هذه السلسلة سوف يحتوي على أرقام الأسطر والتي هي ليست جزءاً من رماز ذلك البرنامج (سوف نتعلم في الدرس العملي كيفية إظهار أرقام الأسطر في بيئة العمل Visual Studio 2010).

```
1  // Text-displaying application.
2  using System;
3
4  public class Welcome1
5  {
6      // Main method begins execution of C# application
7      public static void Main()
8      {
9          Console.WriteLine("Welcome with The_First_ISE");
10     } // end Main
11 } // end class Welcome1
```

```
Welcome with The_First_ISE
```

الشكل 1.3 برنامج يعرض رسالة ترحيبية للمستخدم

سوف نستعرض الآن هذا البرنامج سطراً سطراً حيث تدعى هذه الطريقة بمراجعة الرماز (code walkthrough).

السطر 1 : `// Text-displaying application.`

يبدأ هذا السطر بـ // والتي تشير إلى أن تتمة السطر هي عبارة عن تعليق (comment) ، المبرمجون يكتبون التعليقات في برامجهم ليجعلوا من رمازهم أكثر قابلية للقراءة والفهم.

متـرجم لغة سي شارب (C# compiler) يتجاهل هذه التعليقات عند الترجمة. بشكل عام سوف نبتدأ كل البرامج التي سنكتبها في هذه السلسلة بتعليق يصف الهدف أو الغرض من هذه البرنامج .

التعليق الذي يبدأ بـ // يدعى التعليق السطري (single-line comment) وسمي بهذا الاسم لأنه ينتهي مفعوله بنهاية السطر الذي يوجد فيه ، وهذه النوع من التعليقات يمكن أن توجد في وسط السطر وتستمر حتى نهايته (كما في السطر 10 – 11).

ويوجد نوع آخر من التعليقات تدعى التعليقات المحدودة (delimited comments) يمكنها أن تمتد على أكثر من سطر ، هذا النوع من التعليقات يبدأ بالمحدد /* وينتهي بالمحدد */ والنص الذي بداخل المحددين يعتبر تعليقاً ويتجاهله المترجم كما قلنا.

```
/* النص المعلق الممتد
   على أكثر من سطر */
```

أخطاء برمجية شائعة :

نسيان أحد محددي التعليق يولد خطأ لغوياً (syntax error).

القواعد اللغوية في لغة برمجة معينة تحدد قواعد كتابة برنامج صحيح بتلك اللغة ، ويكون هناك خطأ لغوياً عندما يواجه المترجم رمازاً فيه مخالفة لقواعد لغة السي شارب ففي تلك الحالة لا يقدم المترجم ملفاً تنفيذاً بل يقوم بإظهار رسالة خطأ لمساعدتنا للتعرف على الخطأ وإصلاحه.

الأخطاء اللغوية تدعى أيضاً بـ خطأ ترجمة (compiler error) لأن المترجم يكشفها خلال عملية الترجمة ، ولن تستطيع تشغيل البرنامج حتى تصلح تلك الأخطاء اللغوية.

السطر 2 : `using System;`

هو عبارة عن توجيه (directive) يخبر المترجم عن الصنف الذي نحتاجه وسوف نستخدمه في برنامجنا.

هناك عدد ضخم من الأصناف المعرفة مسبقاً والتي يمكن إعادة استخدامها (reuse) بدلاً من إعادة بناء أصناف تؤدي نفس المهمة (reinventing the wheel) أي إعادة اختراع العجلة.

هذه الأصناف (classes) تنظم في مجموعة من الأصناف المترابطة وظيفياً والتي تدعى فضاءات أسماء (namespaces).

بشكل مجتمع فضاءات أسماء .NET. نشير إليها بـ FCL (Framework Class Library).

كـل توجيـه `using` يعـرف فضـاء الأسـماء الـذي يحتـوي علـى الأصـناف المعرفـة مسـبقاً والتي يُسمح للبرنامج باستخدامها.

الرابط التالي يعرض لك هذه الأصناف بشكل شامل :

http://msdn.microsoft.com/en-us/library/ms229335

أخطاء برمجية شائعة :

نسيان توجيه المترجم باستخدام كلمة `using` لفضاء الأسماء الذي يحتوي على الصنف (class) المستخدم في البرنامج يولد خطأ ترجمة (compiler error).

عند حدوث هذا النوع من الأخطاء ينصح بتفحص عملية التوجيه و تهجئة اسم فضاء الأسماء ومراعاة حالة الأحرف.

السطر 3 :

ببساطة هو عبارة عن سطر فارغ ، المبرمجون عادةً يستخدمون الأسطر الفارغة والفراغات بين المحارف ليجعلوا من رمازهم أكثر قابلية للقراءة والفهم ، و ندعو الأسطر الفارغة والفراغات بين المحارف وعلامات الجدولة بالمسافات البيضاء (whitespace).

المسافة البيضاء تهمل من قبل المترجم ، في هذا الدرس ودروس قادمة سوف نناقش بعض الاصطلاحات لاستخدام المسافات البيضاء لجعل الرماز أكثر قابلية للقراءة والفهم.

السطر 4 : `public class Welcome1`

يبدأ هذا السطر بالتصريح عن الصنف `Welcome1` (class declaration).

كل برنامج سي شارب يتألف من تصريح واحد على الأقل عن صنف يعرف من قبلك (كمبرمج) ، وهذا ما يعرف بالصنف المعرف من قبل المستخدم (user-defined classes).

إن الكلمة المفتاحية `class` تقدم التصريح عن الصنف وتُتبع مباشرة باسم ذلك الصنف الذي نريد تعريفه.

الكلمات المفتاحية (keywords) التي تدعى أحياناً بالكلمات المحجوزة (reserved words) تكون محجوزة للاستخدام من قبل سي شارب وتكون تهجئتها

دوماً بالأحرف الصغيرة ،و يمكنك الاطلاع على جميع الكلمات المفتاحية للغة سي شارب من هذا الرابط :

http://msdn.microsoft.com/en-us/library/x53a06bb(v=VS.100).aspx

بالاصطلاح : أسماء الأصناف تبدأ بحرف كبير ويتم كتابة الحرف الأول من الكلمات الأخرى المؤلفة لهذا الاسم بحرف كبير أيضاً مثل (SampleClassName) وهذا يعرف بتدوين باسكال (Pascal Notation).

اسم الصنف هو معرف (identifier) الصنف وهو عبارة عن سلسلة من المحارف.

يمكن أن تحتوي محارف المعرف على الحروف والأرقام و (_) ولا يجوز أن تبدأ برقم ولا يجوز أيضاً أن تحتوي على مسافة بيضاء ولا يجوز أن تكون إحدى الكلمات المفتاحية ، مثلاً (Welcome1 , identifier , _value , m_inputField) كلها معرفات صالحة ، بينما الاسم (7button)غير صالح كمعرف لأنه يبدأ برقم والاسم (input Field) ليس صالحاً كمعرف أيضاً لأنه يحتوي على مسافة بيضاء.

بشكل طبيعي المعرف الذي لا يبدأ بحرف كبير ليس اسماً لصنف.

إن السي شارب حساسة لحالة الأحرف (case sensitive) بمعنى أن المترجم يفرق بين الأحرف الكبيرة والأحرف الصغيرة ويعتبرها مختلفة فمثلاً (A1) تختلف عن (a1) وكلاهما معرفين صالحين.

المعرفات يمكن أن تسبق بالمحرف @ وهذا يشير إلى أن هذه الكلمة يجب أن تفسر على أنها معرف حتى ولو كانت كلمة مفتاحية مثل (@class).

عادة برمجية مستحسنة :

بالاصطلاح : نبدأ دائماً معرف اسم الصنف بحرف كبير ، ونبدأ كل الكلمات المؤلفة لهذا المعرف بحرف كبير أيضاً (Pascal Notation).

أخطاء برمجية شائعة :

لغة السي شارب حساسة لحالة الأحرف (case sensitive) ، وعدم مراعاة هذه الحساسية عند كتابة المعرفات سيولد أخطاء ترجمة.

أما بالنسبة للكلمة المفتاحية `public` فيمكننا أن نفهمها الآن أنها لا تفرض أي قيود على الأصناف التي تريد التعامل مع هذه الصنف وهي إحدى معرفات الوصول الأربع (access modifier) التي سنتكلم عنها في دروس قادمة.

عادة برمجية مستحسنة :

بالاصطلاح : الملف الذي يحتوي على صنف عام (`public`) وحيد من الأفضل تسميته بنفس اسم الصنف الموجود ضمنه ، طبعاً متبوعاً بلاحقة سي شارب (.cs) ، هذه التسمية تجعل من السهل على المبرمجين الآخرين وحتى عليك أن تحدد مكان وجود الأصناف التي تبحث عنها.

السطر 5 : `{`

هذا القوس المعقوف اليساري يبدأ عنده جسم الصنف المعرف والقوس المعقوف اليميني في السطر 11 ينهي التصريح عن الصنف وهو القوس التوأم للقوس المعقوف اليساري.

لاحظ أن السطر 6 والسطر 10 قد زِيدت مسافتهما البادئة ، إن هذه الزيادة في المسافة البادئة هي أحد أنواع اصطلاحات المسافات البيضاء التي تكلمنا عليها سابقاً.

عادة برمجية مستحسنة :

كلما قمت بكتابة القوس المعقوف اليساري ({) في رمازك ، اضغط على زر الإدخال في لوحة المفاتيح (Enter) ثم اكتب القوس التوأم (}) ، ثم اضغط على زر التحكم + زر الإدخال (Ctrl + Enter).

هذه العادة تساعدك على الوقاية من أخطاء الترجمة الناتجة عن نسيان أقواس إنهاء التصريح.

السطر 6 : `// Main method begins execution of C# application`

هو عبارة عن تعليق وهو يصف الغرض من كتابة الأسطر التي تليه (الوسيلة `Main()`).

السطر 7 : `public static void Main()`

هو عبارة عن تصريح عن الوسيلة Main() والتي هي نقطة البداية لكل تطبيق سي شارب والأقواس المتوسطة ( ) الموجودة بعد معرف الوسيلة تشير إلى أن الكتلة البرمجية المحصورة بين القوسين المعقوفين التوأمين ({ }) هي عبارة عن وسيلة (method) والكلمة المفتاحية static تمكن الأصناف والأغراض الأخرى من الوصول إلى هذه الوسيلة مباشرة ودون الحاجة لإنشاء غرض من الصنف الذي توجد فيه وهي ضرورية للوسيلة Main() (سنتكلم عنها بالتفصيل في دروس قادمة).

التصريح عن صنف جديد يحتوي بشكل طبيعي على واحد أو أكثر من الوسائل ، وأسماء هذه الوسائل (methods) عادةً ما يُتبع في تسمية معرفاتها على طريقة تدوين باسكال المستخدم في أسماء معرفات الأصناف.

في أي تطبيق واحدة وواحدة فقط من الوسائل يجب أن يكون اسمها Main وإلا فإن البرنامج لن يعمل ، ويجب مراعاة أن يكون الحرف الأول من معرف الوسيلة Main() هو حرف كبير.

الوسائل قادرة على تنفيذ مهام معينة وقادرة أيضاً على إرجاع معلومات عندما تنهي عملها ، الكلمة المفتاحية void تشير إلى أن الوسيلة Main() لن تعيد قيمة بعد أن تنهي مهمتها.

لاحقاً سوف نرى كيف أن العديد من الوسائل تعيد قيمة ما.

حاصل القول هنا أن الوسيلة Main() هي نقطة البداية والسطر الأول الذي ينفذ في برنامجك.

السطر 8 : `{`

هذا القوس يبدأ عنده جسم الوسيلة Main() والقوس التوأم في السطر 10 ينهي هذا الجسم.

السطر 9 : `Console.WriteLine("Welcome with The_First_ISE");`

بداية لاحظ أن هذا السطر قد تم زيادة مسافته البادئة ، إن هذا السطر يأمر الحاسوب بتنفيذ عمل ندعوه الطباعة على الشاشة حيث يطبع هذا الأمر سلسلة من المحارف (strings) وهي الرسالة الترحيبية المحتواة ضمن علامتي الاقتباس ("Welcome with The_First_ISE") والفراغات الموجودة ضمن الرسالة لن

تخضع لقاعدة المسافات البيضاء أي لن يتم تجاهلها من قبل المترجم وسيتم طباعتها كما هي.

الصنف `Console` يزودنا بعمليات الدخل والخرج القياسية التي تسمح لتطبيقاتنا (من نوع المترجم السطري) أن تقرأ وتكتب (تعرض) النصوص في نافذة المترجم السطري (command prompt) عبر الوسائل المختلفة التي تقدمها.

والنقطة (.) التي بعد الصنف `Console` تمكننا من الوصول لأعضاء (member)[1] الصنف `Console` والتي تدعى نقطة الوصول للأعضاء (member access).

والوسيلة `WriteLine()` تعرض (تطبع) على شاشة المترجم السطري سلسلة من المحارف (نصاً يمثل الرسالة الترحيبية) ندعو الرسالة التي بداخل الوسيلة `WriteLine()` بمعامل الوسيلة (argument).

بعد أن تنهي الوسيلة `WriteLine()` مهمتها تقوم بوضع مؤشر الكتابة (cursor) في المترجم السطري في بداية سطر جديد.

ندعو السطر 9 بالكامل بما في ذلك الفاصلة المنقوطة بـ عبارة (statement) ، أغلبية العبارات تنتهي بفاصلة منقوطة (;).

عندما تنفذ العبارة في السطر 9 يعرض الحاسوب الرسالة الترحيبية على نافذة المترجم السطري ، الوسيلة بشكل عام تكون مؤلفة من عبارة أو أكثر والتي تنفذ مهام الوسيلة المطلوبة.

أخطاء برمجية شائعة :

إذا كان هناك خطأ ترجمة وقامت بيئة العمل بإظهار تقرير الأخطاء في نافذة الأخطاء (Error List) التي تظهر عند وجود خطأ ما في Visual Studio 2010 ، فالخطأ ربما لن يكون في السطر الذي يشار إليه في الرسالة.

تفحص أولاً السطر الذي تشير إليه رسالة الخطأ فإذا كنت متأكداً من أنه لا يوجد خطأ فتفحص عدة أسطر سابقة وبعد تصحيح خطأ متوقع قم مباشرة بترجمة البرنامج من جديد ثم تأكد من عدم وجود أخطاء وكرر ذلك حتى التخلص من جميع الأخطاء أي لا تصلح الأخطاء كلها دفعة واحدة.

[1] أعضاء الصنف (class member) : هي العناصر المختلفة من وسائل وواصفات وخصائص و..... الموجودة داخل جسم الصنف.

أخطاء برمجية شائعة :

نسيان الفاصلة المنقوطة (;) في نهاية عبارة ما يولد خطأ لغوياً.

بعض المبرمجين يجدون أنه من الصعوبة بمكان و خاصة عندما يتعقد الرماز وتكثر أسطره تحديد توائم الأقواس التي تحدد جسم الصنف أو أجسام الوسائل والكتل البرمجية الأخرى ، لهذا السبب يقوم بعض المبرمجين بإضافة تعليق بنهاية كل تصريح عند قوس الإغلاق (}).

وهذا ما قمنا به في السطر 10 والسطر 11

فالسطر 10: `} // end Main`

يدل على نهاية التصريح عن الوسيلة `Main()`.

والسطر 11 : `} // end class Welcome1`

يدل على نهاية التصريح عن الصنف `Welcome1` .

هذه التعليقات من شأنها تأكيد إغلاق جسم الوسيلة والصنف وربط الأقواس التوائم بشكل صحيح.

عادة برمجية مستحسنة :

إدراج تعليق بنهاية التصريح عن أي كتلة برمجية يدل على نهاية هذا التصريح من شأنه زيادة قابلية قراءة وفهم الرماز المكتوب في برنامجك.

وبهذا نكون قد انتهينا من مراجعة رماز برنامجنا الأول في لغة السي شارب وسوف نتعلم في الدرس العملي كيفية كتابة هذا الرماز في بيئة العمل VS2010 وتشغيله أيضاً.

3.3 : برنامج جمع عددين صحيحين :

يقرأ هذا البرنامج عددين صحيحين من المستخدم ، ويقوم بحساب مجموع هذين العددين ويعرض النتيجة على الشاشة (شاشة المترجم السطري).

البرنامج يحتفظ بالأعداد للوصول إليها واستخدامها لاحقاً ، فالبرامج بشكل عام تحتفظ بأرقام ونصوص ومعلومات أخرى في ذاكرة الحاسوب وتصل إليها وتتعامل معها

بواسطة ما يسمى بالمتحولات (variables) والبرنامج التالي المكتوب في الشكل 2.3 يعرض لنا هذه الأفكار وغيرها.

```
1  // Displaying the sum of two numbers input from the keyboard.
2  using System;
3
4  public class Addition
5  {
6      // Main method begins execution of C# application
7      public static void Main()
8      {
9          int number1; // declare first number to add
10         int number2; // declare second number to add
11         int sum; // declare sum of number1 and number2
12
13         Console.Write("Enter first integer: "); // prompt user
14         // read first number from user
15         number1 = Convert.ToInt32(Console.ReadLine());
16
17         Console.Write("Enter second integer: "); // prompt user
18         // read second number from user
19         number2 = Convert.ToInt32(Console.ReadLine());
20
21         sum = number1 + number2; // add numbers
22
23         Console.WriteLine("Sum is {0}", sum); // display sum
24     } // end Main
25 } // end class Addition
```

```
Enter first integer: 45
Enter second integer: 72
Sum is 117
```

الشكل 2.3 برنامج يعرض مجموع عددين صحيحين مدخلين بواسطة لوحة المفاتيح

السطر 1 : `// Displaying the sum of two numbers input from the keyboard.`

تعليق يصف الهدف أو الغرض من هذا البرنامج.

السطر 4 : `public class Addition`

يبدأ بالتصريح عن صنف جديد اسمه `Addition` ، تذكر أن جسم أي صنف يبدأ بقوس معقوف يساري ({) السطر 5 وينتهي بآخر يميني (}) السطر 25 والبرنامج كما

قلنا يبدأ بالتنفيذ عند الوسيلة Main() (الممتدة في السطور 7 .... 24) والسطر 8 ({) يحدد بداية جسم الوسيلة Main() وقوسه التوأم (}) في السطر 24 يحدد نهاية الوسيلة Main() ، لاحظ هنا زيادة المسافات البادئة في البرنامج المكتوب بالشكل 2.3

السطر 9 : `int number1; // declare first number to add`

عبارة تصريح عن المتحول number1 (variable declaration statement) والتي تعرف نمط واسم المتحول number1 المستخدم في هذا البرنامج.

المتحول (variable) : هو مكان في ذاكرة الحاسوب يمكن من خلاله تخزين قيمة واستخدمها لاحقاً عند الحاجة إليها.

بشكل عام : يتم التصريح عن المتحولات بتحديد اسمها ونمطها قبل استخدامها ، فاسم المتحول يسمح للبرنامج بالوصول لقيمة هذا المتحول في الذاكرة.

يمكن أن يكون اسم المتحول أي معرف صالح (valid identifier) ، ونمط المتحول يحدد نوع المعلومات التي يمكن أن يخزنها هذا المتحول في الذاكرة.

تنتهي عبارة التصريح عن المتحول بفاصلة منقوطة (;) وكما نرى عبارة التصريح في هذا السطر تبين بأن نمط المتحول number1 هو int أي أنه سوف يقبل قيماً لأعداد صحيحة فقط مثل (0 ، 11- ، 7) ومجال الأرقام المسموح بها في هذا النمط هي من -2,147,483,648 (int.MinValue) وحتى +2,147,483,647 (int.MaxValue).

هناك أنماط معطيات عددية أخرى في لغة السي شارب تجد تفصيلها في ملحق هذا الدرس مثل :

bool,byte,sbyte,char,short,ushort,float,double,decimal……

عبارتي التصريح عن المتحولين في **السطرين 10 – 11** تصرح عن المتحولين number2,sum ليكونا من نمط int.

يمكن لعبارة التصريح أن تمتد على عدة أسطر وذلك بفصل أسماء المتحولات بفاصلة (,) ووضعها بسطر جديد ولكن بشرط أن تكون المتحولات من نمط واحد.

فمثلاً السطور 9 – 10 – 11 يمكن أن تكتب هكذا :

```
int number1, // declare first number to add
    number2, // declare second number to add
    sum; // declare sum of number1 and number2
```

عادة برمجية مستحسنة :

التصريح عن المتحول بسطر منفصل يسمح بإضافة تعليق يزيد من وضوح الرماز.

عادة برمجية مستحسنة :

اختيار أسماء ذات معنى ودلالة واضحة للمتحول يساعد الرماز على أن يشرح نفسه بنفسه (حتى بدون تعليقات **أحياناً**).

عادة برمجية مستحسنة :

بالاصطلاح : معرفات أسماء المتحولات تبدأ بحرف صغير وكل كلمة من الكلمات المؤلفة لهذا الاسم تبدأ بحرف كبير ، تدعى هذه الطريقة في التدوين (camel notation) وكمثال على ذلك (sampleVariableName).

السطر 13 : `Console.Write("Enter first integer: "); // prompt user`

يستخدم هذا السطر الوسيلة `Write()` (التي هي أحد الوسائل الأعضاء في الصنف `Console`) لعرض رسالة تطلب من المستخدم القيام بعمل معين.

تختلف هذه الوسيلة `Write()` عن أختها `WriteLine()` بأنها تبقي مؤشر الكتابة بعد الرسالة مباشرة ولا تضعه في سطر جديد.

السطر 15 : `number1 = Convert.ToInt32(Console.ReadLine());`

يعمل هذا السطر عبر مرحلتين الأولى باستدعاء الوسيلة `ReadLine()` (method call) (التي هي أيضاً أحد الوسائل الأعضاء في الصنف `Console`) حيث تنتظر هذه الوسيلة المستخدم لكتابة سلسلة من المحارف بواسطة لوحة المفاتيح وأن يضغط زر الإدخال بالنهاية.

كما ذكرنا سابقاً بعض الوسائل تعيد قيمة ما بعد الانتهاء من تنفيذ مهامها ، هذه الوسيلة `ReadLine()` تعيد السلسلة المحرفية (string) المدخلة من قبل المستخدم ومن ثم نستخدم هذا النص المدخل كمعامل للوسيلة `ToInt32()` (التي هي أحد الوسائل

الأعضاء في الصنف Convert) وذلك لتحويل النمط المدخل كنص (string) إلى نمط الأعداد الصحيحة int (Int32).

عملياً : المستخدم يستطيع أن يدخل أي قيمة يريدها والوسيلة ReadLine() تقبل منه هذه القيمة المدخلة وستمرر هذه القيمة للوسيلة ToInt32() ، ولكن هذه الأخيرة لها شرط صارم ولا تتخلى عنه وهو أن معاملها يجب أن لا يحوي في سلسلة محارفه إلا على الأرقام.

فمثلاً في هذا البرنامج إذا قام المستخدم بإدخال أحرف سوف ينتج عن ذلك خطأ منطقياً أثناء التشغيل وسوف يتوقف البرنامج عن العمل ، مثل هذه الأخطاء تسمى أخطاء وقت التشغيل (Run-Time error).

في دروس قادمة سوف نناقش كيفية جعل التطبيقات أكثر متانة وذلك بالسماح لها بمعالجة أخطاء وقت التشغيل والاستمرار بالعمل (Exception Handling).

أما في المرحلة الثانية فسوف تخزن القيمة المحولة في المتحول number1 وذلك باستخدام عملية الإسناد (=) ، حيث أن هذه العبارة تقرأ كما يلي (يحصل المتحول number1 على قيمته من القيمة المعادة من الوسيلة ToInt32()).

عملية الإسناد (=) توصف بأنها عملية ثنائية لأن لها طرفين هما هنا الطرف اليميني وهو القيمة المعادة من الوسيلة ToInt32() والطرف اليساري هو المتحول number1 ندعو هذين الطرفين بمعاملي عملية الإسناد (operands).

وندعو العبارة في السطر 15 بعبارة إسناد (assignment statement) لأنها تقوم بإسناد قيمة إلى المعامل في الطرف الأيسر.

كل العمليات والمهام تنجز في المعامل الأيمن لعملية الإسناد ثم تتم عملية الإسناد.

السطر 17 : `Console.Write("Enter second integer: "); // prompt user`

يطلب من المستخدم إدخال عدد صحيح.

السطر 19 : `number2 = Convert.ToInt32(Console.ReadLine());`

يقرأ النص المدخل من قبل المستخدم ويسند قيمته بعد التحويل إلى نمط عدد صحيح إلى المتحول number2.

السطر 21 : `sum = number1 + number2; // add numbers`

عبارة الإسناد هنا تقوم بحساب مجموع العددين الصحيحين المدخلين من قبل المستخدم وتسند القيمة الناتجة عن عملية الجمع (+) إلى المتحول sum الذي يمثل مجموع العددين (المتحولين) number1 و number2.

توصف عملية الجمع (+) أيضاً بأنها عملية ثنائية لأن لها أيضاً معاملان.

وندعو جزء العبارة الذي يحتوي على عملية حسابية بالتعبير (expression) ، بمعنى آخر التعبير هو أي جزء من العبارة له قيمة مرتبطة بقيمته المعادة بعد تنفيذه.

على سبيل المثال : التعبير number1 + number2 له قيمة مرتبطة بناتج عملية الجمع.

أيضاً : التعبير Console.ReadLine() له قيمة مرتبطة بالنص المدخل من المستخدم.

السطر 23 : `Console.WriteLine("Sum is {0}", sum); // display sum`

بعد إنجاز العمليات الحسابية يأتي هذا السطر لينسق النص الذي سوف يعرض للمستخدم ألا وهو ناتج الجمع ، يستخدم هذا السطر الوسيلة WriteLine() من الصنف Console لإنجاز المهمة.

للوسيلتين WriteLine() و Write() قدرة على تنسيق البيانات التي تمرر إليهما كمعاملات وإظهارها بالشكل المناسب ، ويتم ذلك باستخدام التنسيق النصي (format string) والذي يتألف من نصوص (fixed text) وعناصر تنسيق (format items) ، حيث يتم تمرير عناصر التنسيق والنصوص كمعاملات واستبدال عناصر التنسيق بالمعاملات الممررة على الترتيب المذكور وأذكر هنا أن سي شارب تبدأ بالعد من الصفر فعنصر التنسيق ({0}) يستبدل بأول معامل في قائمة المعاملات الممررة للوسيلة والمفصولة عن بعضها بفاصلة (,) كما هو الحال في مثالنا.

ندعو قائمة المعاملات الممررة بالقائمة المجزأة بفاصلة (comma-separated list).

النصوص هي التي سوف تظهر على الشاشة وعناصر التنسيق تستبدل بالمعاملات كما هو معلن عنه في التنسيق.

ويمكن أيضاً لعملية الجمع أن تنفذ داخل قوسي الوسيلة WriteLine() فيمكننا مثلاً دمج السطرين 21 – 23 في عبارة كهذه :

```
Console.WriteLine("Sum is {0}", (number1 + number2));
```

طبعاً القوسين حول المتحولين غير ضرورين في العبارة السابقة ، ولكن وضعا للتركيز على أن ناتج عملية الجمع سوف يستبدل بعنصر التنسيق في المعامل الأول.

4.3 : العمليات الحسابية (Arithmetic Operator) :

أغلب البرامج تنفذ عمليات حسابية ، الشكل 3.3 يعرض العمليات الحسابية المستخدمة في لغة سي شارب ، لاحظ أن عدد من الرموز لا تستخدم في الجبر فالنجمة مثلاً (*) ترمز إلى عملية الضرب والرمز (%) ترمز لعملية باقي القسمة وتدعى (remainder operator) والعمليات الموجودة بالشكل كلها توصف بأنها عمليات ثنائية.

اسم العملية	الرمز الحسابي في C#	الرمز الجبري	تعبير حسابي في C#
الجمع	+	+	F + 3
الطرح	-	-	P – 9
الضرب	*	.	B * M
القسمة	/	/ أو ÷	X / Y
باقي القسمة	%	Mod	R % 2

الشكل 3.3 العمليات الحسابية المستخدمة في سي شارب

القسمة الصحيحة تتنازل عن باقي القسمة فعلى سبيل المثال التعبير 7/4 يعيد القيمة 1 ، والتعبير 17/5 يعيد القيمة 3 ، والقسم الكسري الناتج سوف يهمل (يحذف) ولا يحدث أي عملية تقريب.

سي شارب تزودنا كما رأينا بعملية باقي القسمة والتي تعيد ذلك القسم الكسري الناتج عن عملية القسمة الصحيحة فمثلاً التعبير : 7%4 يعيد القيمة 3 و التعبير 17%5 يعيد القيمة 2 ، هذه العملية تستخدم بشكل كبير مع الأعداد الصحيحة ولكن يمكن استخدامها مع الأعداد الكسرية `float,double,decimal`.

تستخدم الأقواس الصغيرة ( ) لتجميع حدود المعاملات في التعابير الحسابية كما هي العادة في التعابير الجبرية ، فعلى سبيل المثال لإجراء عملية ضرب a بمجموع b و c نكتب : a * (b + c) ، وإذا كان التعبير يحتوي على أقواس متداخلة مثل : ((a + b) * c) فالتعبير الذي بداخل الأقواس المتداخلة سينفذ أولاً.

تفرض سي شارب على العمليات الحسابية في التعابير الحسابية أولوية في التنفيذ والتي هي بالغالب مشابهة لتلك المستخدمة في الجبر والشكل 4.3 يوضح هذه القواعد.

العمليات	ترتيب التنفيذ (التجميع)
تحسب أولاً	
* / %	إذا كان هناك أكثر من عملية من هذا النوع فإنه يتم الحساب من اليسار إلى اليمين.
تحسب ثانياً	
+ -	إذا كان هناك أكثر من عملية من هذا النوع فإنه يتم الحساب من اليسار إلى اليمين.

الشكل 4.3 قواعد أولوية تنفيذ العمليات الحسابية وقواعد تجميعها في السي شارب

دعنا الآن نأخذ أمثلة على تعابير ونسلط الضوء على قواعد التجميع والأولوية :

المثال 1 : المتوسط الحسابي لخمسة متحولات :

جبرياً نكتب :

$$m = \frac{a+b+c+d+e}{5}$$

بالسي شارب نكتب :

```
m = (a + b + c + d + e) / 5;
```

الأقواس هنا مطلوبة لأن عملية القسمة تملك الأولوية الأهم من أولوية عملية الجمع وناتج جمع ما داخل القوسين سوف يقسم على 5 وهو المطلوب.

فإذا لم تتم كتابة القوسين (عن طريق الخطأ) فسوف يكون معنى التعبير جبرياً :

$$m = a+b+c+d+\frac{e}{5}$$

المثال 2 : فلنتفحص هذه المعادلة :

جبرياً نكتب :

$$y = m.x + b$$

بالسي شارب نكتب :

```
y = m * x + b;
```

هنا الأقواس غير مطلوبة ، لأن عملية الضرب هنا تنفذ أولاً لأنها أولويتها أعلى من أولوية عملية الجمع.

المثال 3 : فلنتفحص هذه المعادلة :

جبرياً نكتب :

$$z = p.r\%q + w/x - y$$

بالسي شارب نكتب :

```
z = p * r % q + w / x - y;
```

6 1 2 4 3 5

كما في الجبر من المقبول وضع أقواس غير ضرورية في التعابير الحسابية لجعل التعابير أكثر وضوحاً وفهماً.

5.3 : صناعة القرارات المنطقية (عمليات المساواة والمقارنة) :

الشرط هو كل تعبير يعيد إحدى القيمتين المنطقيتين (صح true) أو (خطأ false).

سوف نتعرف في هذه الفقرة على عبارة التحكم الشرطية البسيطة (if statement) والتي تسمح للبرنامج باتخاذ قراره المنطقي بالاعتماد على قيمة الشرط.

فعلى سبيل المثال : الشرط التالي " العلامة أكبر أو تساوي 60 درجة " تحدد فيما إذا كان الطالب الحاصل على هذه العلامة قد نجح بالامتحان أو لا.

فإذا كان الشرط في عبارة (if) صحيحاً (محققاً) فإن جسم العبارة الشرطية سوف ينفذ وعلى العكس إذا لم يكن صحيحاً (غير محققاً) فإنه لن ينفذ.

يمكننا صياغة الشروط في عبارة (if) باستخدام عمليات المقارنة (== , != , > , >= , < , <=) والمساواة (equality and relational operator) ونذكر هنا أن كل مجموعة مستقلة لها جميعاً نفس أولوية التنفيذ ، ولعمليات المقارنة أولوية أعلى من أولوية عمليات المساواة وتجميعها جميعاً من اليسار لليمين.

رمز العملية الجبري	رمز العملية في C#	مثال عن الشرط في C#	معنى الشرط
	عمليتي المساواة (equality operator)		
=	==	x == y	x يساوي y
≠	!=	x != y	x لا يساوي y
	عمليات المقارنة (relational operator)		
>	>	x > y	x أكبر من y
<	<	x < y	x أصغر من y
≥	>=	x >= y	x أكبر أو يساوي y
≤	<=	x <= y	x أصغر أو يساوي y

الشكل 5.3 قواعد أولوية تنفيذ عمليات المساواة والمقارنة في السي شارب

أخطاء برمجية شائعة :

استخدام عملية المساواة (==) مكان عملية الإسناد (=) أو العكس يمكن أن يولد خطأ منطقياً (logical error) أو خطأ لغوياً (syntax error).

البرنامج المكتوب في الشكل 6.3 يستخدم ست عبارات (if) الشرطية البسيطة للمقارنة بين عددين صحيحين يقوم المستخدم بإدخالهما ، فإذا تحقق الشرط في إحدى تلك العبارات الشرطية فعبارة الإظهار المرتبطة بتلك العبارة سوف تنفذ.

```
1  /* Comparing integers using if statements, equality operators,
2     and relational operators.*/
3  using System;
4
5  public class Comparison
6  {
7     // Main method begins execution of C# application
8     public static void Main()
9     {
10       int number1; // declare first number to compare
11       int number2; // declare second number to compare
12
13       // prompt user and read first number
14       Console.Write("Enter first integer: ");
```

```
15          number1 = Convert.ToInt32(Console.ReadLine());
16
17          // prompt user and read second number
18          Console.Write("Enter second integer: ");
19          number2 = Convert.ToInt32(Console.ReadLine());
20
21          if (number1 == number2)
22             Console.WriteLine("{0} == {1}", number1, number2);
23
24          if (number1 != number2)
25             Console.WriteLine("{0} != {1}", number1, number2);
26
27          if (number1 < number2)
28             Console.WriteLine("{0} < {1}", number1, number2);
29
30          if (number1 > number2)
31             Console.WriteLine("{0} > {1}", number1, number2);
32
33          if (number1 <= number2)
34             Console.WriteLine("{0} <= {1}", number1, number2);
35
36          if (number1 >= number2)
37             Console.WriteLine("{0} >= {1}", number1, number2);
38       } // end Main
39    } // end class Comparison
```

التنفيذ 1

```
Enter first integer: 45
Enter second integer: 45
45 == 45
45 <= 45
45 >= 45
```

التنفيذ 2

```
Enter first integer: 15
Enter second integer: 20
15 != 20
15 < 20
15 <= 20
```

التنفيذ 3

```
Enter first integer: 20
Enter second integer: 15
20 != 15
20 > 15
20 >= 15
```

الشكل 6.3 برنامج مقارنة أعداد صحيحة باستخدام عبارة (if) وعمليات المقارنة والمساواة

البرنامج يستخدم الصنف Console لطلب إدخال وقراءة عددين صحيحين من المستخدم ، ويقوم بعد ذلك بتحويل الأعداد المدخلة إلى نمط int باستخدام الوسيلة ToInt32() من الصنف Convert ويخزنها في المتحولات number1 و number2.

السطر 21 ... 37 : يقارن البرنامج من خلال العبارات الشرطية البسيطة بين قيمتي المتحولين number1 و number2 ويطبق عليهما كل عمليات المقارنة والمساواة.

تبدأ عبارة (if) الشرطية دائماً بالكلمة المفتاحية if متبوعة بقوسي الشرط وتتوقع عبارة (if) عبارة واحدة على الأقل في جسمها.

أخطاء برمجية شائعة :

نسيان كتابة أحد قوسي الشرط في عبارة (if) يولد خطأ ترجمة.

أخطاء برمجية شائعة :

تبديل الترتيب في عمليات المقارنة أو المساواة ماعدا (==) يولد خطأ ترجمة.

أخطاء برمجية شائعة :

إذا احتوت عمليات المقارنة والمساواة على فراغ فهذا يولد خطأ ترجمة.

عادة برمجية مستحسنة :

زيادة المسافة البادئة لجسم عبارة (if) الشرطية يزيد من وضوح الرماز وقابلية القراءة والفهم له.

لاحظ أنه لا يوجد فاصلة منقوطة في نهاية السطر الأول من عبارة (if) وإن وجدت على سبيل الخطأ فسوف ينتج لدينا خطأ منطقياً وقت التشغيل.

فعلى سبيل المثال :

```
if (number1 == number2) ; //logic error
   Console.WriteLine("{0} == {1}", number1, number2);
```

سوف تفسر من قبل سي شارب على الشكل التالي :

```
if (number1 == number2)
    ; //empty statement
   Console.WriteLine("{0} == {1}", number1, number2);
```

حيث أن الفاصلة المنقوطة هي عبارة فارغة بحد ذاتها ، وهي العبارة التي ستنفذ إذا تحقق شرط العبارة الشرطية (if).

عندما تنفذ العبارة الفارغة فلن ينفذ البرنامج أي مهمة ، ثم يتابع البرنامج تنفيذ عباراته بغض النظر عن نتيجة الشرط لأنه إن تحقق الشرط فسوف تنفذ العبارة الفارغة (لن ينفذ شيء).

أخطاء برمجية شائعة :

وضع فاصلة بعد قوسي الشرط في عبارة (if) مباشرة يعتبر خطأ منطقياً.

أخطاء برمجية شائعة :

وضع مسافات بيضاء في المعرفات وسلاسل المحارف وعمليات المقارنة والمساواة يعتبر خطأ لغوياً.

عادة برمجية مستحسنة :

من الأفضل فصل العبارة الطويلة في عدة سطور مع مراعاة قيود المسافات البيضاء.

الشكل 7.3 يوضح أولوية العمليات التي عرضت في هذا الدرس بشكل مجمل ، العمليات في الشكل معروضة من الأعلى إلى الأسفل بترتيب تنازلي من حيث أولوية التنفيذ.

العمليات	التجميع	النوع
* , / , %	من اليسار لليمين	ضرب
+ , -	من اليسار لليمين	جمع
< , <= , > , >=	من اليسار لليمين	مقارنة
== , !=	من اليسار لليمين	مساواة
=	من اليمين لليسار	إسناد

الشكل 7.3 أولويات تنفيذ العمليات وطريقة تجميعها

كما تلاحظ كل هذه العمليات تجمع من اليسار لليمين ما عدا عملية الإسناد فهي تجمع من اليمين لليسار فالتعبير `x + y + z` ينفذ كما لو أنه مكتوب بالشكل `(x + y) + z` وعبارة الإسناد التالية `x = y = 0` تنفذ كما لو أنها مكتوبة بالشكل `x = (y = 0)` أي أنه يتم إسناد القيمة `0` إلى المتحول `y` ومن ثم تسند قيمة المتحول `y` إلى المتحول `x` (لأن عملية الإسناد يمينة التجميع).

عادة برمجية مستحسنة :

عند كتابة التعابير المعقدة التي يصعب معها التأكد من ترتيب تنفيذ العمليات ينصح بوضع أقواس للتأكد من ترتيب تنفيذ العمليات كما يجب.

وتذكر أن بعض العمليات تجمع من اليمين لليسار مثل عملية الإسناد.

6.3 : الملحق :

الأنماط البسيطة في سي شارب (simple types)

الكلمة المفتاحية في سي شارب	صنفه في .NET	مجاله	حجمه	معناه
bool	System.Boolean	True *or* false	8 bit	True or false
sbyte	System.SByte	-128 *to* 127	8 bit	Signed integer
byte	System.Byte	0 *to* 255	8 bit	Unsigned integer
char	System.Char	'\u0000' *to* '\uFFFF' (0 *to* 65535)	16 bit	A single Unicode character
short	System.Int16	-32,768 *to* 32,767	16 bit	Signed integer
ushort	System.UInt16	0 *to* 65535	16 bit	Unsigned integer
int	System.Int32	-2,147,483,648 *to* 2,147,483,647	32 bit	Signed integer
uint	System.UInt32	0 to 4294967295	32 bit	Unsigned integer
float	System.Single	*Approximate negative range:* 3.4028234663852886E+38 *to* 1.40129846432481707E45 *Approximate positive range:* 1.40129846432481707E45 *to* 3.4028234663852886E+38 *Other supported values:* positive and negative zero positive and negative infinity not-a-number (NaN)	32 bit	Single-precision floating point type
long	System.Int64	-9223372036854775808 *to* 9223372036854775807	32 bit	Signed integer
ulong	System.UInt64	0 *to* 18446744073709551615	32 bit	Unsigned integer

Double-precision floating point type	64 bit	*Approximate negative range:* 1.7976931348623157E+308 *to* 4.94065645841246544E324 *Approximate positive range:* 4.94065645841246544E324 *to* 1.7976931348623157E+308 *Other supported values:* positive and negative zero positive and negative infinity not-a-number (NaN)	System.Double	double
decimal numbers with 29 significant digits	128 bit	Negative range: 79,228,162,514,264,337,593,543,950,335 (7.9E+28) *to* 1.0E28 Positive range: 1.0E28 *to* 79,228,162,514,264,337,593,543,950,335 (7.9E+28)	System.Decimal	decimal

أشهر محارف الهروب في سي شارب

محرف الهروب (escape sequence)	الوصف
\'	إدراج علامة الاقتباس المفردة (') ضمن السلسلة النصية.
\"	إدراج علامة الاقتباس المفردة (") ضمن السلسلة النصية.
\\	إدراج الخط المائل الخلفي (\ ) ضمن السلسلة النصية (مفيدة عند كتابة مسار ملف أو شبكة).
\a	تشغيل صوت التحذير من النظام.
\n	إدراج سطر جديد في السلسلة النصية (في بيئة التشغيل ويندوز)
\r	تعيد مؤشر الكتابة إلى أول السطر.
\t	إدراج مسافة علامة الجدولة الأفقية (TAB) ضمن السلسلة النصية.