(T1)C#的 Type(型別)、NullableType(可空型別)
CourseGUID: 29f1196a-1950-41a4-b9c1-dd13a9e92d92
==========
(T1)C#的 Type(型別)、NullableType(可空型別)
==========

==========

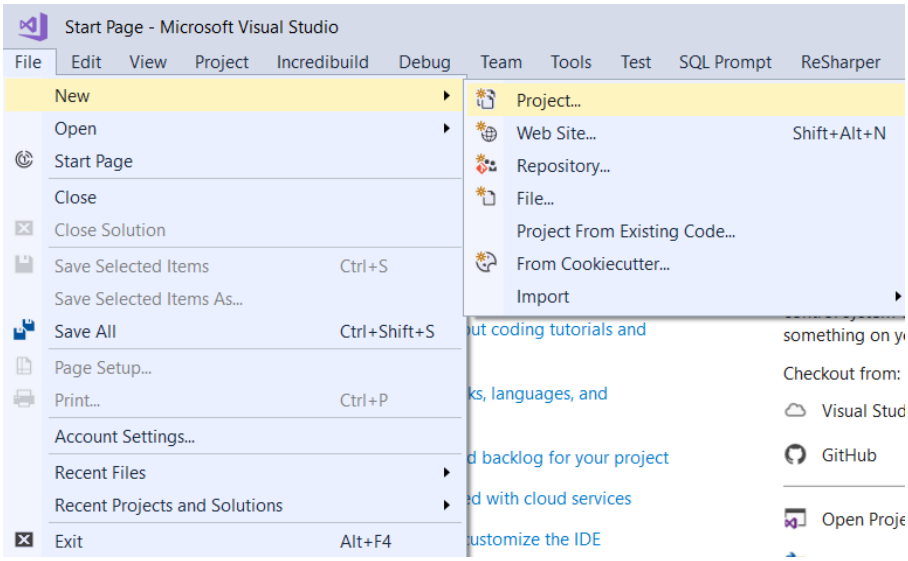
0. Summary

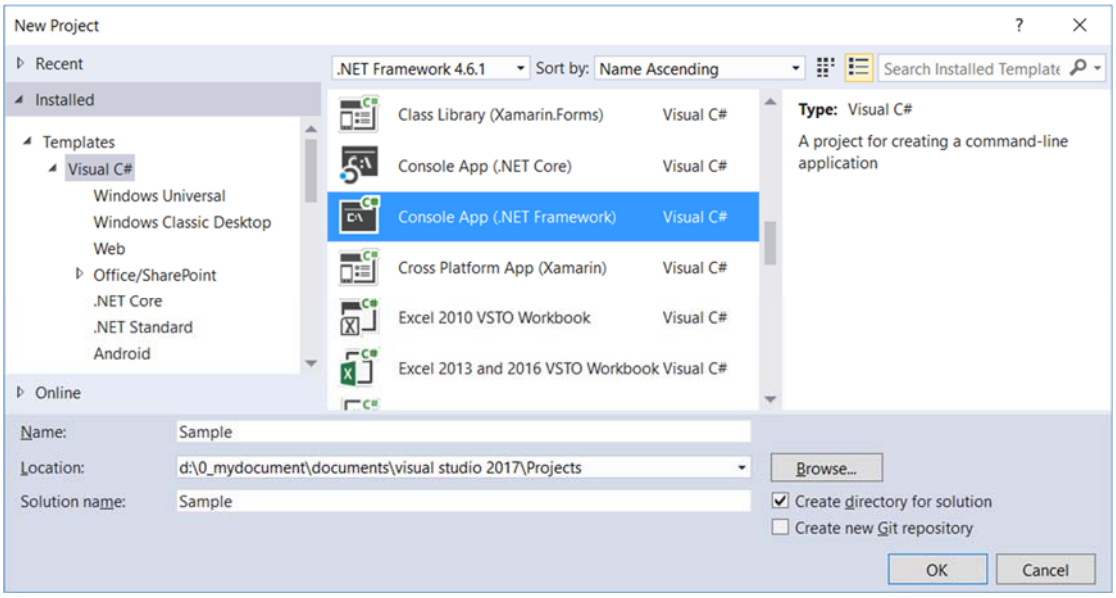
* 第 1 章: C#基礎介紹
  * Tutorial01: C#的各個型別包括介紹 Nullable，各種 loop，switch，if...else...。關於各種 operator。
  * Tutorial02: C#的物件導向，關於 Base Class，Sub Class，Interface
  * Tutorial03: 關於 Namespace，Static method 和 Instance method 比較，Ref，Out，params 關鍵字比較，MethodOverride 和 MethodHide 比較
  * Tutorial04: RAM 的 Stack 和 Heap 比較，Struct 和 Class 比較，Value Type 和 Reference Type 比較
  * Tutorial05: public，protected，private 的比較，Abstract Class 和 Interface 的比較
  * Tutorial06: public，protected，private 的比較，關於 Delegate 和 Multicast Delegates
  * Tutorial07: TryCatch 搭配常見的 Exception，蝦毀?內建的不夠用，客製化的 Exception
    * TryCatch ，Exception，FileNotFoundException，SqlException，FormatException，OverflowException，DivideByZeroException，客製化的 Exception
  * Tutorial08: 關於 Enum，Enum.GetValue 和 Enum.GetNames 的比較
  * Tutorial09: Access modifiers 大亂鬥。private V.S. public V.S. protected V.S. internal V.S. protected internal
  * Tutorial10: System.String 和 System.Text.StringBuilder 比較。Ram 的 Stack 和 Heap 比較。
  * Tutorial11: Convert.ToString 和 ToString 的比較。納尼?不夠用?那就 Override ToString。
  * Tutorial12: Contains()和 Equals()和 SequenceEqual()的 bug 的三種解決方法。Override Equals()和 GetHashCode()。IEqualityComparer。使用匿名型別。

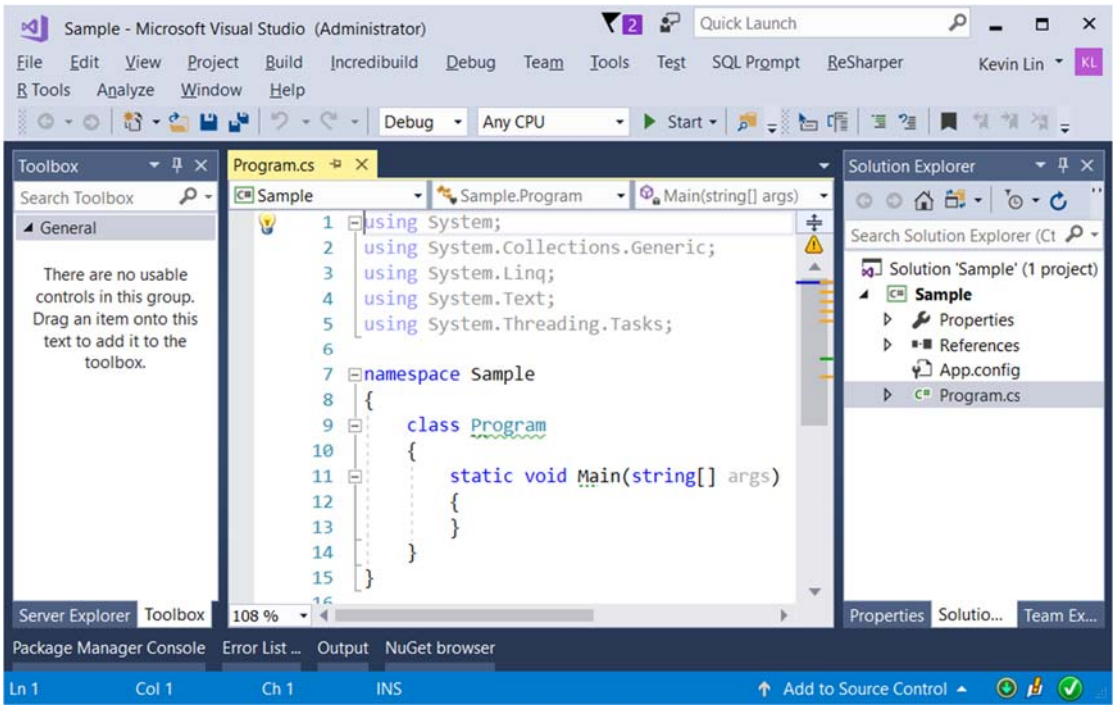
==========

1. Create New Project

File --> New --> Project... -->
Visual C# --> **Console App (.Net Framework)** -->
Name: **Sample**


Start Page - Microsoft Visual Studio
File Edit View Project Incredibuild Debug Team Tools Test SQL Prompt ReSharper
New
Open
Start Page
Close
Close Solution
Save Selected Items Ctrl+S
Save Selected Items As...
Save All Ctrl+Shift+S
Page Setup...
Print... Ctrl+P
Account Settings...
Recent Files
Recent Projects and Solutions
Exit Alt+F4
Project...
Web Site... Shift+Alt+N
Repository...
File...
Project From Existing Code...
From Cookiecutter...
Import
Checkout from:
Visual Stud
GitHub
Open Proje


New Project
Recent
Installed
Templates
Visual C#
Windows Universal
Windows Classic Desktop
Web
Office/SharePoint
.NET Core
.NET Standard
Android
Online
.NET Framework 4.6.1
Sort by: Name Ascending
Search Installed Templat
Class Library (Xamarin.Forms) Visual C#
Console App (.NET Core) Visual C#
Console App (.NET Framework) Visual C#
Cross Platform App (Xamarin) Visual C#
Excel 2010 VSTO Workbook Visual C#
Excel 2013 and 2016 VSTO Workbook Visual C#
Type: Visual C#
A project for creating a command-line application
Name: Sample
Location: d:\0_mydocument\documents\visual studio 2017\Projects
Solution name: Sample
Browse...
Create directory for solution
Create new Git repository
OK
Cancel


Sample - Microsoft Visual Studio (Administrator)
Quick Launch
File Edit View Project Build Incredibuild Debug Team Tools Test SQL Prompt ReSharper
R Tools Analyze Window Help
Kevin Lin
Debug
Any CPU
Start
Toolbox
Search Toolbox
General
There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.
Program.cs
Sample
Sample.Program
Main(string[] args)
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace Sample
{
class Program
{
static void Main(string[] args)
{
}
}
}
Solution Explorer
Search Solution Explorer (Ct
Solution 'Sample' (1 project)
Sample
Properties
References
App.config
Program.cs
Server Explorer
Toolbox
108 %
Properties
Solutio...
Team Ex...
Package Manager Console
Error List ...
Output
NuGet browser
Ln 1
Col 1
Ch 1
INS
Add to Source Control

2. Comment

// Single line comment

/*
Multi line Comments
*/

/// means XML Documentation Comments
E.g.

```
/// <summary>
/// The magic start point of Console App (.Net Framework).
/// </summary>
/// <param name="args">the main arguments.</param>
static void Main(string[] args){}
```

You may highlight the code which you want to comments, then press
Ctrl+K, Ctrl+C to comment
OR
Ctrl+K, Ctrl+U to Un-Comment

You may also use tool bar to do comments.

 means comment.

 means Un-Comment.

3. C# basic

```
using System;
using System.Linq;
namespace Sample
{
    internal class Program
    {
        /// <summary>
        /// The magic start point of Console App (.Net Framework).
        /// </summary>
        /// <param name="args">the main arguments.</param>
        private static void Main(string[] args)
        {
            //1.
            Console.WriteLine("Hello, This is main method.");
            //------------------------------------------------------------
            //2. Read the name
            Console.WriteLine("What is your name?");
            string name = Console.ReadLine();
            Console.WriteLine("Hello, " + name);
            Console.WriteLine("Have a good day. {0}", name);
```

```
//------------------------------------------------------------
//3. Escape symbol, "\"
//------------------------------------------------------
//3.1.
//Displaying double quotes
string str = "\"AAA\""; //"AAA"
Console.WriteLine(str);
//------------------------------------------------------
//3.2.
//Displaying new line character
 str = "AAA\nBBB\nCCC";
Console.WriteLine(str);
//AAA
//BBB
//CCC
//------------------------------------------------------
//3.3.
//Displaying the directory, Less Readable
 str = "c:\\AAA\\BBB\\CCC"; //c:\AAA\BBB\CCC
Console.WriteLine(str);
//------------------------------------------------------
//3.4.
//verbatim literal string displaying the directory, Better Readable
 str = @"c:\AAA\BBB\CCC\"; //c:\AAA\BBB\CCC\
Console.WriteLine(str);
//------------------------------------------------------------
//4. Operators
//------------------------------------------------------
//4.1. Assignment Operator : =
char ch = 'A';
string str2 = "AAAA";
int i = 1;
double d = 1.1;
bool b = true;
//------------------------------------------------------
//4.2. Arithmetic Operators : +,-,*,/,%
int i1 = 10;
int i2 = 2;
int i3 = 3;
Console.WriteLine("i1 = {0}, i2 = {1}, i3 = {2}", i1, i2, i3); //i1 = 10, i2 = 2, i3 = 3
Console.WriteLine("i2 + i3 = {0}", i2 + i3); //i2 + i3 = 5
Console.WriteLine("i2 - i3 = {0}", i2 - i3); //i2 - i3 = -1
Console.WriteLine("i2 * i3 = {0}", i2 * i3); //i2 * i3 = 6
Console.WriteLine("i1 / i2 = {0}", i1 / i2); //i1 / i2 = 5, Find Quotient
Console.WriteLine("i1 / i3 = {0}", i1 / i3); //i1 / i3 = 3, Find Quotient
Console.WriteLine("i1 % i2 = {0}", i1 % i2); //i1 % i2 = 0, Find Remainder
Console.WriteLine("i1 % i3 = {0}", i1 % i3); //i1 % i3 = 1, Find Remainder
//------------------------------------------------------
//4.3.
//------------------------------------------
//4.3.1.
//Comparison Operators : ==, !=,>, >=, <, <=
//Conditional Operators : &&, ||
```

```
//Ternary Operator :   Condition? True:False
//Null Coalescing Operator: Condition ??
//-----------------------------------------
//4.3.2.
// if(Condition){A}else{B} , if condition is true then A, else B.
string str3 = "";
if (i2 == 2)
{
    str3 = "i2 == 2";
}
else
{
    str3 = "i2 != 2";
}
Console.WriteLine(str3);
//i2 == 2
//-----------------------------------------
//4.3.3.
// variable = (condition) ? A : B , if condition is true then A, else B.
str3 = i2 == 2 ? "i2 == 2" : "i2 != 2";
Console.WriteLine(str3);
//i2 == 2
//-----------------------------------------
//4.3.4.
if (i2 != 5)
{
    Console.WriteLine("i2 != 5");
}
//i2 != 5
//-----------------------------------------
//4.3.5.
if (i2 == 2 && i3 == 3)
{
    Console.WriteLine("i2 == 2 is true AND i3 == 3 is true");
}
//i2 == 2 is true AND i3 == 3 is true
//-----------------------------------------
//4.3.6.
if (i2 == 2 || i3 == 5)
{
    Console.WriteLine("i2 == 2 is true OR i3 == 5 is true");
}
//i2 == 2 is true OR i3 == 5 is true

//-------------------------------------------------------------
//5. Convert, Parse, TryParse
//-------------------------------------------------------
//5.1. // Implicit convert int to float/double will not loss data.
int i4 = 100;
float f1 = i4; //Implicit convert int to float/double
Console.WriteLine("f1 == {0}", f1); // f1 == 100
//-------------------------------------------------------
//5.2. float/double to int
float f2 = 100.25F;
////5.2.1.
```

```
            ////float/double can not be implicit converted to int.
            //int i5 = f2;
            //5.2.2.
            int i6 = (int) f2;
            Console.WriteLine("i6 == {0}", i6); //i6 == 100
            //Explicit conversion will loss fractional part.
            //There is also a possiblity of overflow exception.
            //5.2.3.
            //Convert class
            int i7 = Convert.ToInt32(f2);
            Console.WriteLine("i7 == {0}", i7); //i7 == 100
            //--------------------------------------------------
            //5.3. Parse and TryParse
            //5.3.1.
            string str4 = "100TG";
            //int i8 = int.Parse(str4);   // Error! throws an exception
            //5.3.2.
            int i9;
            int.TryParse(str4, out i9);
            Console.WriteLine("i9 == {0}", i9); //i9 == 0, this means false.
            //5.3.3.
            string str5 = "100";
            int i10 = int.Parse(str5);
            int i11;
            int.TryParse(str5, out i11);
            Console.WriteLine("i10 == {0}, i11 == {1}", i10, i11); //i10 == 100, i11 == 100

        //------------------------------------------------------------
            //6. Nullable
            //--------------------------------------------------
            //6.1.  string V.S. int?
            string str6 = null; //string can be a class type, thus can be null.
            //int i12 = null; //Error! int is value type, thus can not be null.
            int? i13 = null; //int? will make it become nullable.
            //--------------------------------------------------
            //6.2.  bool?
            bool? b2 = null; //bool is value type, thus can not be null. bool? will make it become
nullable.
            if (b2 == true)
            {
                Console.WriteLine("b2 == true");
            }
            else if (b2 == false)
            {
                Console.WriteLine("b2 == false");
            }
            else
            {
                Console.WriteLine("bool? b2 == null");
            }
            //--------------------------------------------------
            //6.3.  bool?
            int i14;
            int? i15 = null;
```

```
  if (i15 == null)
  {
      i14 = 0;
  }
  else
  {
      i14 = (int) i15;
  }
  Console.WriteLine("i14 == {0}", i14);
  //------------------------------------------------------
  //6.4.  bool?
  int i16;
  int? i17 = null;
  //A = B??C , if B==null, then C, otherwise B
  i16 = i17 ?? 0;
  Console.WriteLine("i16 ={0}", i16);


//-------------------------------------------------------------
  //7. Array and loop
  //------------------------------------------------------
  //7.1.  Even Array
  //Array use index to store or retrieve items from the array.
  int[] intEvenArr1 = new int[3];
  intEvenArr1[0] = 0;
  intEvenArr1[1] = 2;
  intEvenArr1[2] = 4;
  Console.WriteLine("intEvenArr1[0] == {0}", intEvenArr1[0]);
  Console.WriteLine("intEvenArr1[1] == {0}", intEvenArr1[1]);
  Console.WriteLine("intEvenArr1[2] == {0}", intEvenArr1[2]);
  //intEvenArr1[0] == 0
  //intEvenArr1[1] == 2
  //intEvenArr1[2] == 4
  //------------------------------------------------------
  //7.2.  For loop for array
  // For loop check the condition first, if condition is ture then execute.
  // Then repeatly execute the process until the condition is false.
  // Initialize and assign values in the same line
  int[] intOddArr1 = {1, 3, 5};
  // i will start from 0, i++ means i=i+1 in each end of loop,
  // loop will stop until i >= intOddArr1.Length which is 3.
  // that means i will be 0,1,2
  for (int a = 0; a < intOddArr1.Length; a++)
  {
      Console.WriteLine("intOddArr1[{0}] == {1}", a, intOddArr1[a]);
  }
  //intOddArr1[0] == 1
  //intOddArr1[1] == 3
  //intOddArr1[2] == 5
  //------------------------------------------------------
  //7.3.  Foreach loop for array
  // For loop check the condition first, if condition is ture then execute.
  // Then repeatly execute the process until the condition is false.
  // if you use foreach, you cannot use index any more.
  // but you can use Arr.First() or Arr.Last().
  Console.Write(@"intOddArr1[] == { ");
```

```
foreach (int item in intOddArr1)
{
    Console.Write(item == intOddArr1.Last()
        ? item + " "
        : item + " , ");
}
Console.WriteLine("}");
//intOddArr1[] == { 1 , 3 , 5 }
Console.Write(@"intOddArr1[] == { ");
foreach (int item in intOddArr1)
{
    Console.Write(item == intOddArr1.First()
        ? item.ToString()
        : " , " + item);
}
Console.WriteLine(" }");
//intOddArr1[] == { 1 , 3 , 5 }
//-------------------------------------------------------
//7.4.  While loop
// While loop check the condition first, if condition is ture then execute.
// Then repeatly execute the process until the condition is false.
int i18 = 0;
Console.Write(@"intOddArr1[] == { ");
while (i18 < intOddArr1.Length)
{
    Console.Write(intOddArr1[i18] == intOddArr1.First()
        ? intOddArr1[i18].ToString()
        : " , " + intOddArr1[i18]);
    i18++; //i18 = i18+1;
}
Console.WriteLine(" }");
//intOddArr1[] == { 1 , 3 , 5 }
//-------------------------------------------------------
//7.5.  Do While loop
//Do While loop check the condition at the end of loop,
//it guaranteed to execute at least once.
int i19 = 0;
Console.Write(@"intOddArr1[] == { ");
do
{
    Console.Write(intOddArr1[i19] == intOddArr1.First()
        ? intOddArr1[i19].ToString()
        : " , " + intOddArr1[i19]);
    i19++; //i19 = i19+1;
} while (i19 < intOddArr1.Length);
//intOddArr1[] == { 1 , 3 , 5 }
Console.WriteLine(" }");
//intOddArr1[] == { 1 , 3 , 5 }

//-------------------------------------------------------------
//8. if V.S. switch
//-------------------------------------------------------
//8.1.  if
int i20 = 15;
if (i20 == 5)
```

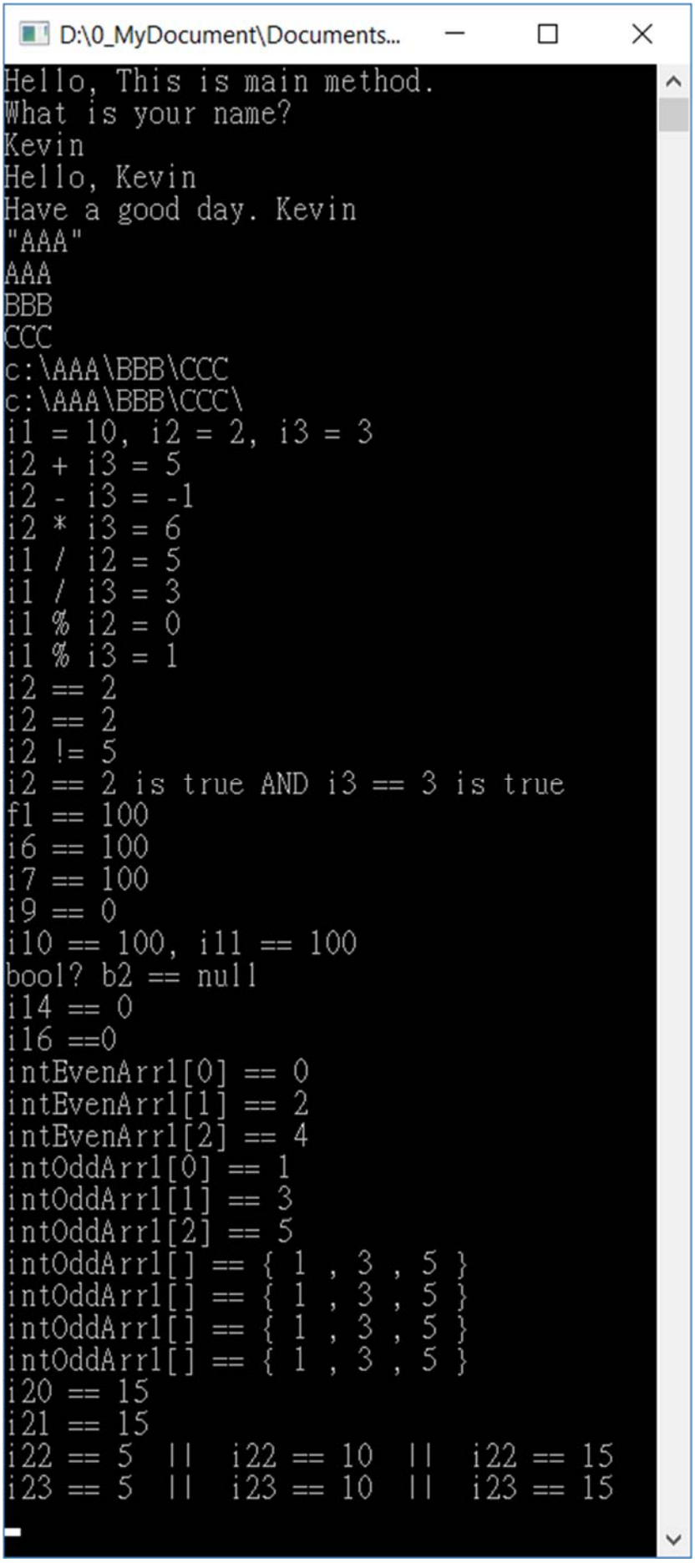
```
{
    Console.WriteLine("i20 == 5");
}
else if (i20 == 10)
{
    Console.WriteLine("i20 == 10");
}
else if (i20 == 15)
{
    Console.WriteLine("i20 == 15");
}
else
{
    Console.WriteLine("i20 != 5  &&  i20 != 10  &&  i20 != 15");
}
// i20 == 15
//-------------------------------------------------------
//8.2.  switch
int i21 = 15;
switch (i21)
{
    case 5:
        Console.WriteLine("i21 == 5");
        break; // leave the current {}
    case 10:
        Console.WriteLine("i21 == 10");
        break; // leave the current {}
    case 15:
        Console.WriteLine("i21 == 15");
        break; // leave the current {}
    default:
        Console.WriteLine("i21 != 5  &&  i21 != 10  &&  i21 != 15");
        break; // leave the current {}
}
// i21 == 15
//-------------------------------------------------------
//8.3.  switch
int i22 = 10;
switch (i22)
{
    case 5:
    case 10:
    case 15:
        Console.WriteLine("i22 == 5  ||  i22 == 10  ||  i22 == 15");
        break; // leave the current {}
    default:
        Console.WriteLine("i22 != 5  &&  i22 != 10  &&  i22 != 15");
        break; // leave the current {}
}
// i22 == 5  ||  i22 == 10  ||  i22 == 15
//-------------------------------------------------------
//8.4.  switch
int i23 = 10;
switch (i23)
{
    case 5:
```

```
                    Console.WriteLine("i23 == 5  ||  i23 == 10  ||  i23 == 15");
                    break; // leave the current {}
                case 10:
                    goto case 5; // goto can jump to other case.
                // Goto is hard to debug, you should try to avoid using goto.
                case 15:
                    goto case 5; // goto can jump to other case.
                // Goto is hard to debug, you should try to avoid using goto.
                default:
                    Console.WriteLine("i23 != 5  &&  i23 != 10  &&  i23 != 15");
                    break; // leave the current {}
            }
            // i23 == 5  ||  i23 == 10  ||  i23 == 15
            Console.ReadLine();
        }
    }
}

/*
1.
C# Built-in types
Reference:
https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/integral-types-table
https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/built-in-types-table
1.1. Boolean type : It returns true (1) or false (0)
1.2. Integer Types : sbyte < byte < char = short = ushort < int = uint < long < ulong,
1.3. Floating Types – float < double < Decimal
1.3.1.
float is 32 bits.
1.3.2.
double is 64 bits.
1.3.3.
Decimal is 128 bits, and has more precision and a smaller range.
1.4.
Char Type :
Char is a single Unicode character,
Char and int can be converted to each other.
1.5.
String Type :
string contains zero or more Unicode characters.
2.
Operators
2.1.
Assignment Operator : =
2.2.
Arithmetic Operators : +,-,*,/,%
2.3.
Comparison Operators : ==, !=,>, >=, <, <=
2.4.
Conditional Operators : &&, ||
2.5.
Ternary Operator :   Condition?True:False
2.6.
Null Coalescing Operator :   Condition??
3.
3.1.
if(Condition){A}else{B}
if condition is true then A, else B.
3.2.
if(A==1){A1}
else if(A==2) {A2}
else {A3}
if A==1 then A1, if A==2, then A2, else A3
```

```
3.3.
if(A||B){C}
if A is true OR B is true, then C
3.4.
if(A&&B){C}
if A is true AND B is true, then C
3.5.
variable = (condition) ? A : B
if condition is true then A, else B.
E.g.
str3 = i2 == 2 ? "i2 == 2" : "i2 != 2";
3.6.
A = B??C
if B==null, then C, otherwise B
E.g.
i16 = i17 ?? 0;
5.
Convert
5.1.
E.g.
//int i4 = 100;
//float f1 = i4;  //100
5.2.
//float f2 = 100.25F;
//int i6 = (int)f2;   //100
5.3.
//float f2 = 100.25F;
//int i7 = Convert.ToInt32(f2);   //100
6.
Parse V.S. TryParse
6.1.
Parse can only covert string to number.
If convertion is failed, then Parse() will throws an exception,
but TryParse() returns a bool 0 (0 means false, 1 means true.)
I suggest always use TryParse()
6.2.
E.g.
//string str4 = "100TG";
//int i8 = int.Parse(str4);   // Error! throws an exception
6.3.
E.g.
//int i9;
//int.TryParse(str4, out i9);
//Console.WriteLine("i9 == {0}",i9);  //i9 == 0, this means false.
6.4.
E.g.
//string str5 = "100";
//int i10 = int.Parse(str5);
//int i11;
//int.TryParse(str5, out i11);
//Console.WriteLine("i10 == {0}, i11 == {1}", i10, i11);  //i10 == 100, i11 == 100
7.
2 categories of data types, value type, reference type.
Value Types : int, float, double, structs, enums...etc
Reference Types : Interface, Class, delegates, arrays...etc
string can be a class type, thus can be null.
int is value type, thus can not be null.
int? will make int become nullable.
7.1.
E.g.
//string str6 = null; //string can be a class type, thus can be null.
////int i12 = null; //Error! int is value type, thus can not be null.
//int? i13 = null;    //int? will make it become nullable.
//bool? b2 = null;    //bool is value type, thus can not be null. bool? will make it become nullable.
8.
An array is strongly typed and a collection of objects.
Array use index to store or retrieve items from the array.
```

```
Arrays cannot grow in size once initialized.
9.
Loop
9.1.
For loop
For loop check the condition first,
if condition is ture then execute.
Then repeatly execute the process until the condition is false.
E.g.
//for (int a = 0; a < intOddArr1.Length; a++)
//{
//    Console.WriteLine("intOddArr1[{0}] == {1}", a, intOddArr1[a]);
//}
------------
9.2.
Foreach loop for array
For loop check the condition first, if condition is ture then execute.
Then repeatly execute the process until the condition is false.
if you use foreach, you cannot use index any more.
but you can use Arr.First() or Arr.Last().
E.g.1.
//foreach (int item in intOddArr1)
//{
//    Console.Write(item == intOddArr1.Last()
//        ? item + " "
//        : item + " , ");
//}
E.g.2.
//foreach (int item in intOddArr1)
//{
//    Console.Write(item == intOddArr1.First()
//        ? item.ToString()
//        : " , " + item);
//}
------------
9.3.
While loop
While loop check the condition first, if condition is ture then execute.
Then repeatly execute the process until the condition is false.
E.g.
//while (i18 < intOddArr1.Length)
//{
//    Console.Write(intOddArr1[i18] == intOddArr1.First()
//        ? intOddArr1[i18].ToString()
//        : " , " + intOddArr1[i18]);
//    i18++;  //i18 = i18+1;
//}
------------
9.4.
Do While loop
Do While loop check the condition at the end of loop,
it guaranteed to execute at least once.
E.g.
//do
//{
//    Console.Write(intOddArr1[i19] == intOddArr1.First()
//        ? intOddArr1[i19].ToString()
//        : " , " + intOddArr1[i19]);
//    i19++;  //i19 = i19+1;
//} while (i19 < intOddArr1.Length);
*/
```


D:\0_MyDocument\Documents...
Hello, This is main method.
What is your name?
Kevin
Hello, Kevin
Have a good day. Kevin
"AAA"
AAA
BBB
CCC
c:\AAA\BBB\CCC
c:\AAA\BBB\CCC\
i1 = 10, i2 = 2, i3 = 3
i2 + i3 = 5
i2 - i3 = -1
i2 * i3 = 6
i1 / i2 = 5
i1 / i3 = 3
i1 % i2 = 0
i1 % i3 = 1
i2 == 2
i2 == 2
i2 != 5
i2 == 2 is true AND i3 == 3 is true
f1 == 100
i6 == 100
i7 == 100
i9 == 0
i10 == 100, i11 == 100
bool? b2 == null
i14 == 0
i16 ==0
intEvenArr1[0] == 0
intEvenArr1[1] == 2
intEvenArr1[2] == 4
intOddArr1[0] == 1
intOddArr1[1] == 3
intOddArr1[2] == 5
intOddArr1[] == { 1 , 3 , 5 }
intOddArr1[] == { 1 , 3 , 5 }
intOddArr1[] == { 1 , 3 , 5 }
intOddArr1[] == { 1 , 3 , 5 }
i20 == 15
i21 == 15
i22 == 5 || i22 == 10 || i22 == 15
i23 == 5 || i23 == 10 || i23 == 15