C#

Agenda

- Introduction
- Console Application
- Windows Form Application
- C# Vs. MySQL
- C# Vs. Network programming

Visual Studio Express 2013

Visual Studio Express 2013 for Windows Desktop

Microsoft Visual Studio Express 2013 for Windows Desktop

Visual Studio Express 2013 for Windows Desktop enables the creation of desktop apps in C#, Visual Basic, and C++, and supports Windows Presentation Foundation (WPF), Windows Forms, and Win32.

Sign in to Visual Studio within 30 days with your Microsoft account to synchronize your settings across multiple machines and register your product.

System requirements

Download language

English

Installation options

Microsoft Visual Studio Express 2013 for Windows Desktop - English

Install now

Microsoft Visual Studio Express 2013 for Windows Desktop - English

DVD5 ISO image (SHA-1: E61419E51F42254EE07DECF628B85C9861286250)

Microsoft Visual Studio Express 2013 for Windows Desktop Language Pack

The Visual Studio Express 2013 for Windows Desktop Language Pack is a free add-on that you can use to switch the language that's displayed in the Visual Studio user interface.

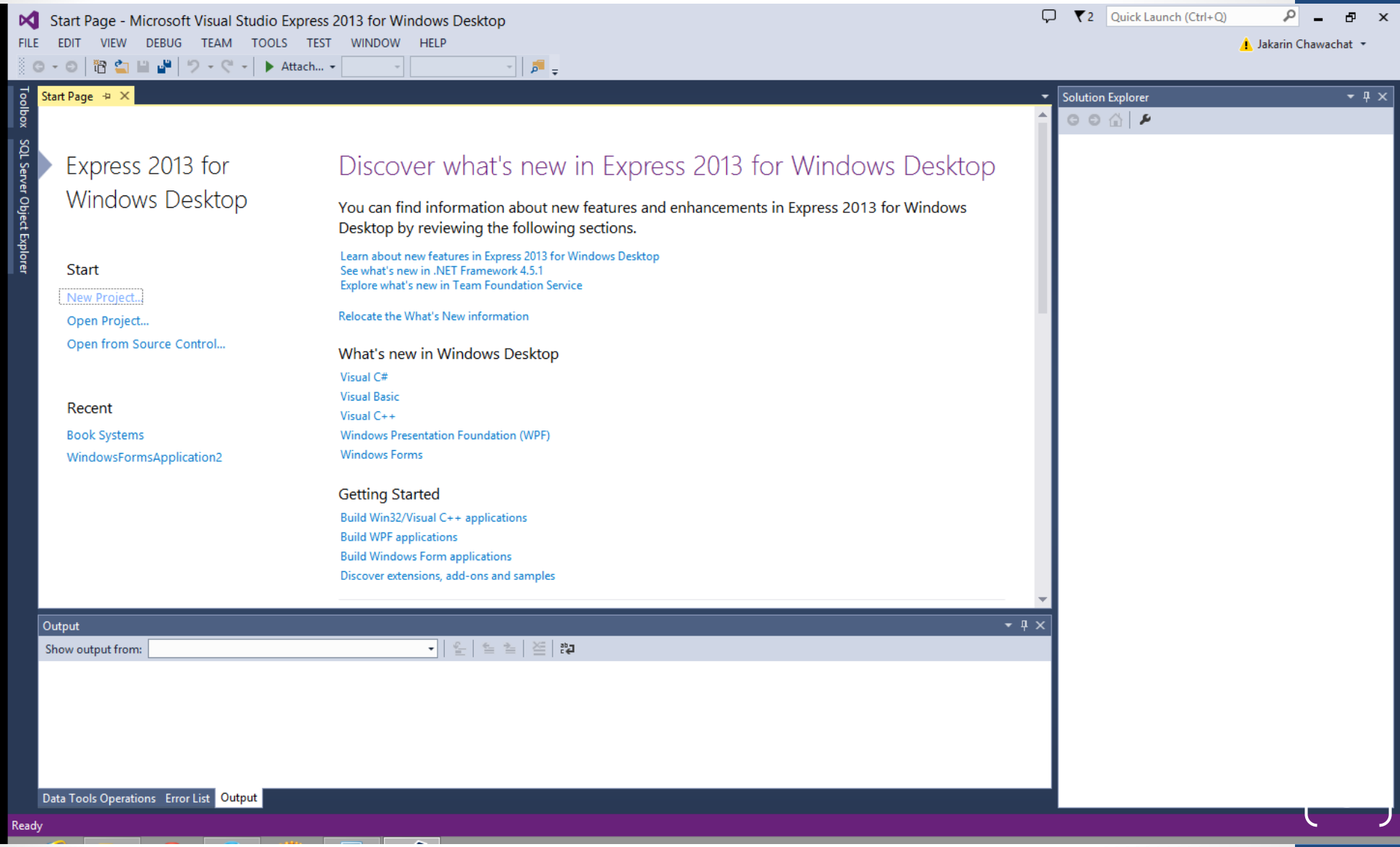
Download language

Český

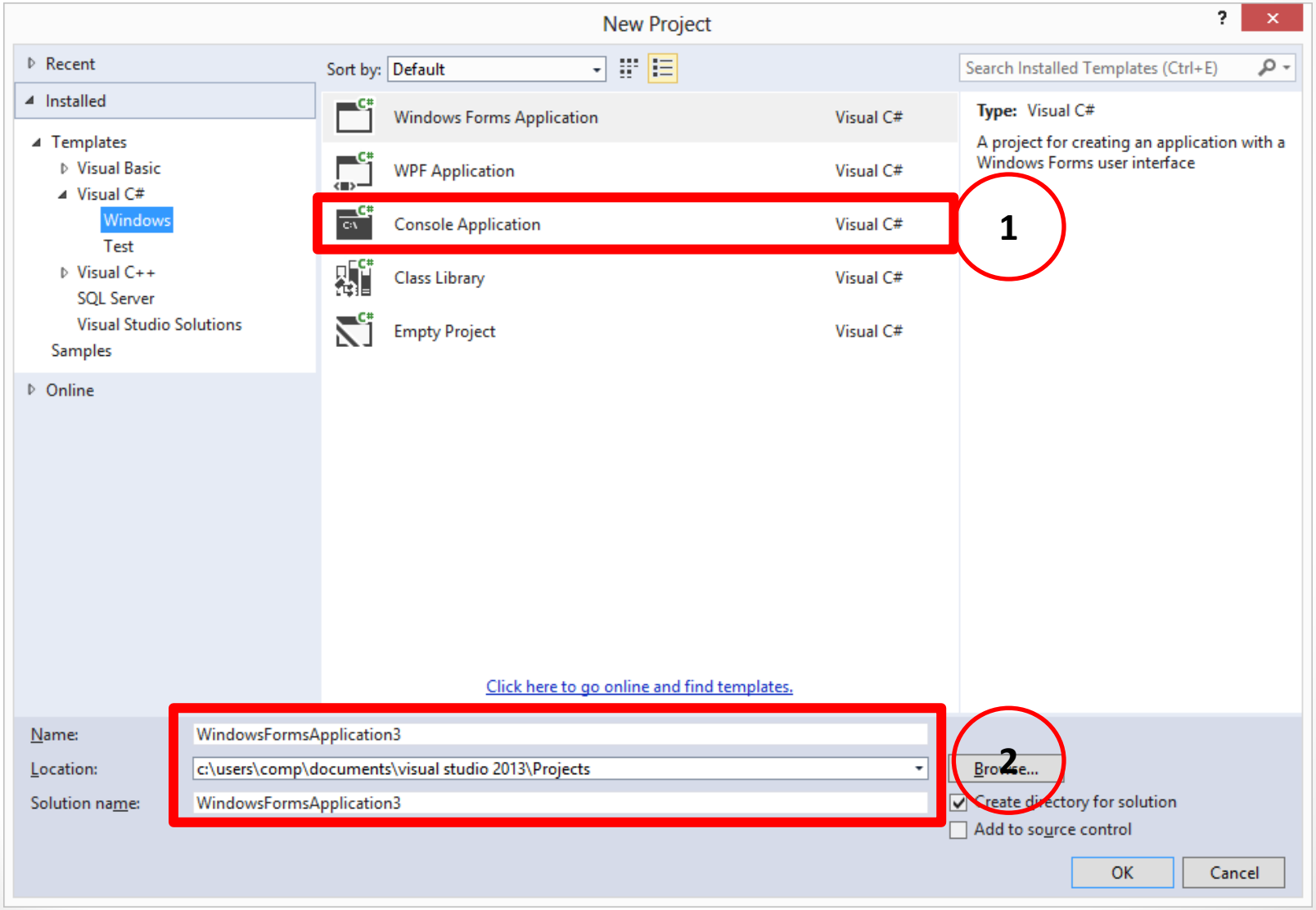
Installation options

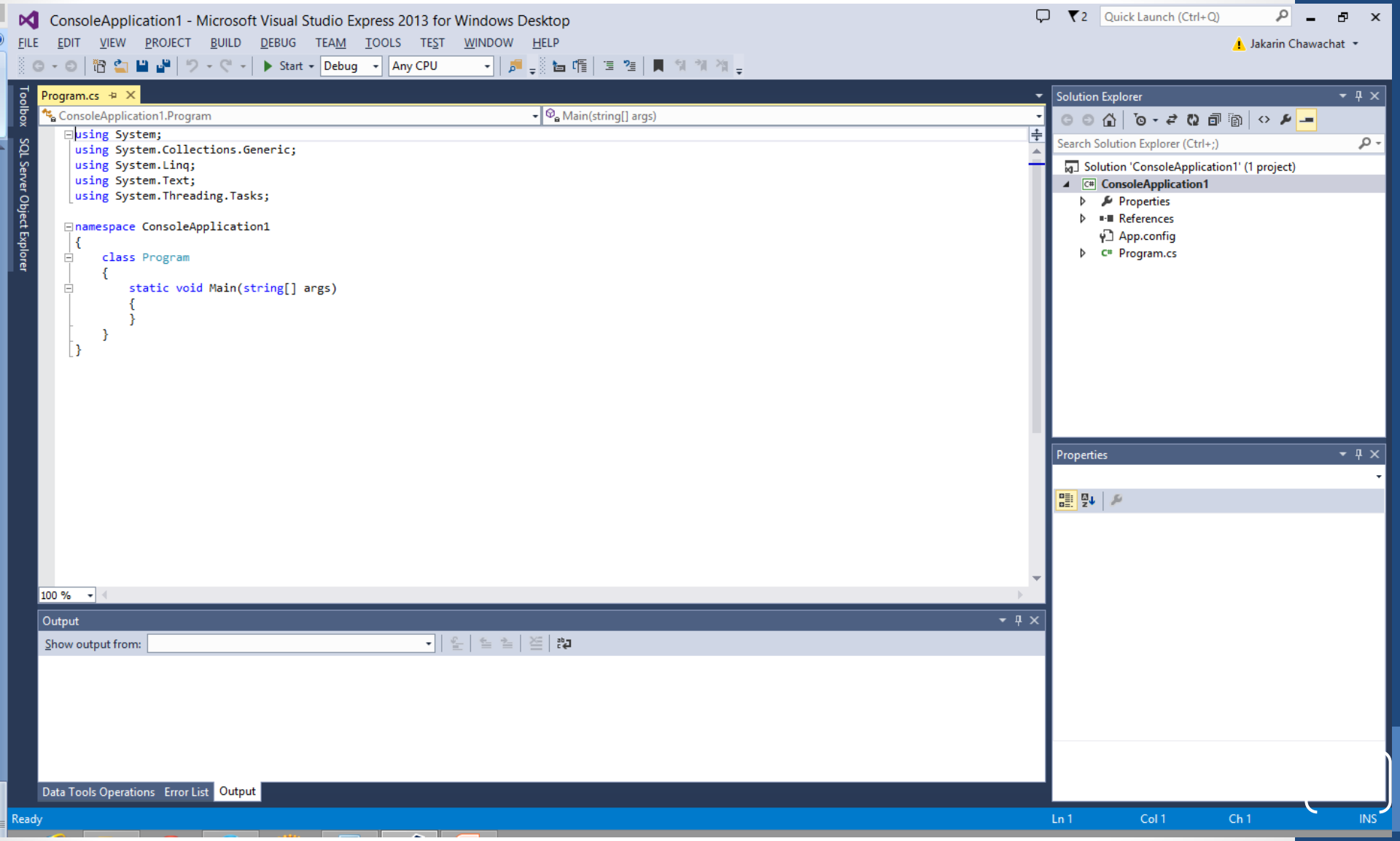
Microsoft Visual Studio Express 2013 for Windows Desktop Language Pack - Český

Download now

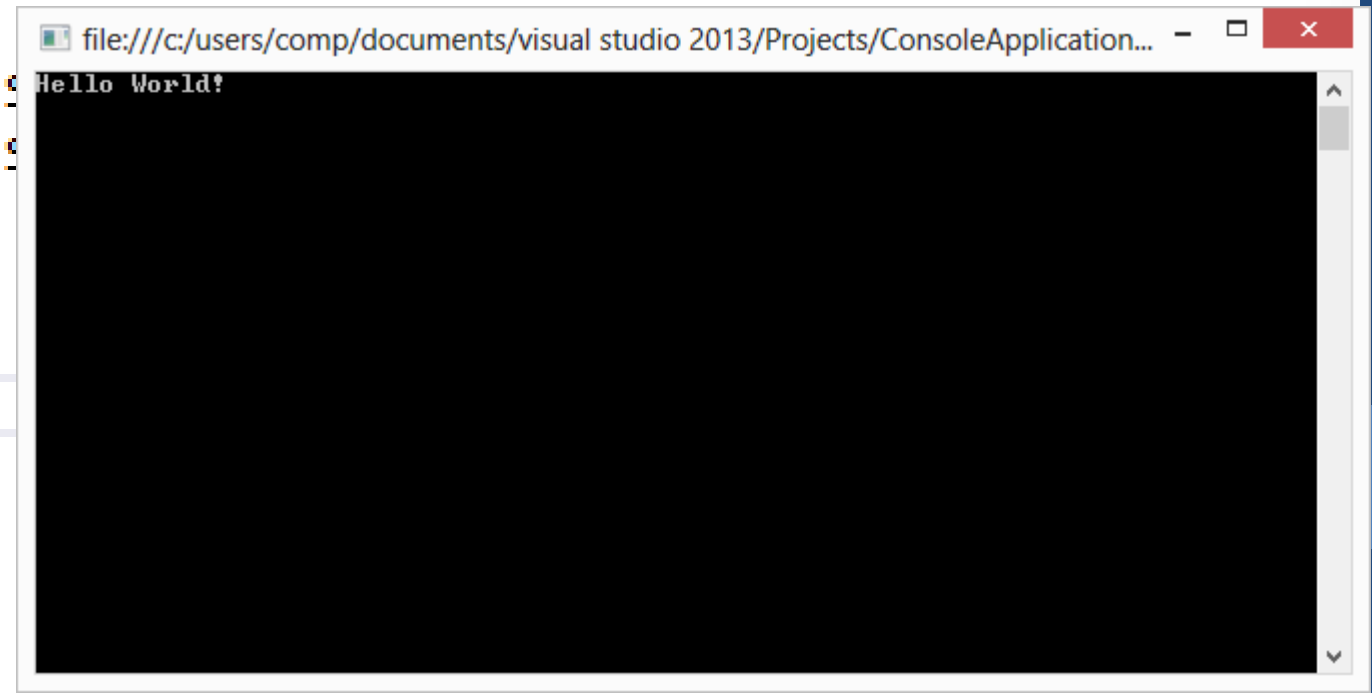

Start Page - Microsoft Visual Studio Express 2013 for Windows Desktop
FILE EDIT VIEW DEBUG TEAM TOOLS TEST WINDOW HELP
Quick Launch (Ctrl+Q)
Jakarin Chawachat
Attach...
Start Page
Toolbox
SQL Server Object Explorer
Express 2013 for Windows Desktop
Start
New Project...
Open Project...
Open from Source Control...
Recent
Book Systems
WindowsFormsApplication2
Discover what's new in Express 2013 for Windows Desktop
You can find information about new features and enhancements in Express 2013 for Windows Desktop by reviewing the following sections.
Learn about new features in Express 2013 for Windows Desktop
See what's new in .NET Framework 4.5.1
Explore what's new in Team Foundation Service
Relocate the What's New information
What's new in Windows Desktop
Visual C#
Visual Basic
Visual C++
Windows Presentation Foundation (WPF)
Windows Forms
Getting Started
Build Win32/Visual C++ applications
Build WPF applications
Build Windows Form applications
Discover extensions, add-ons and samples
Solution Explorer
Output
Show output from:
Data Tools Operations
Error List
Output
Ready

Console Application

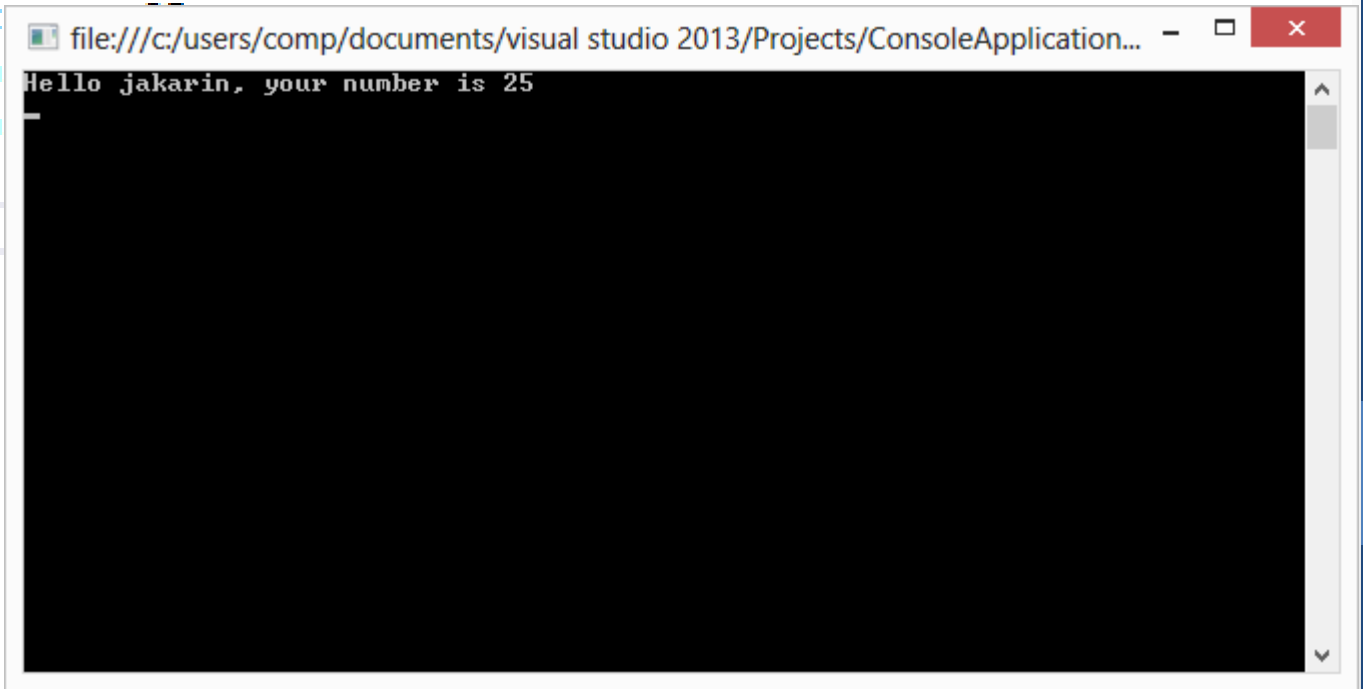



ConsoleApplication1 - Microsoft Visual Studio Express 2013 for Windows Desktop
Quick Launch (Ctrl+Q)
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST WINDOW HELP
Jakarin Chawachat
Start
Debug
Any CPU
Program.cs
ConsoleApplication1.Program
Main(string[] args)
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace ConsoleApplication1
{
class Program
{
static void Main(string[] args)
{
}
}
}
100 %
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'ConsoleApplication1' (1 project)
ConsoleApplication1
Properties
References
App.config
Program.cs
Properties
Output
Show output from:
Data Tools Operations
Error List
Output
Ready
Ln 1
Col 1
Ch 1
INS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            c
            c

        }
    }
}
```



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            string name = "jakarin";
            int
            Con
            Con
        }
    }
}
```



โครงสร้างของโปรแกรมภาษา C#

การโปรแกรมภาษา C# ขั้นพื้นฐานที่มีเฉพาะส่วนโปรแกรมหลักจะมีประกอบดังนี้

```
namespace _____(A)____
{
    class _____(B)____
    {
        static void Main()
        {
            _____(C)____
        }
    }
}
```

ชื่อของเนมสแปซ ใช้ในการกำหนดขอบเขตให้กับคลาสต่างๆ

ชื่อของ class

พื้นที่เขียนคำสั่งต่างๆ

กฎการตั้งชื่อตัวระบุ (Identifier)

- ชื่อตัวแปรต้องประกอบด้วยตัวอักษรภาษาอังกฤษ (A-Z,a-z) ตัวเลข (0-9) หรือเครื่องหมายขีดเส้นใต้ (_) เท่านั้น
- ตัวอักษรตัวแรกของชื่อต้องเป็นตัวอักษรภาษาอังกฤษหรือตัวขีดเส้นใต้
- ชื่อตัวแปรมีความยาวได้ไม่เกิน 63 ตัวอักษร
- ชื่อตัวแปรต้องไม่ซ้ำกับคำสงวน (reserved word) เช่น class, namespace, int, void, static

ชนิดข้อมูล

ชนิดข้อมูล	คำอธิบาย
char	อักขระเดี่ยว เช่น a
bool	ค่าความจริง เป็นไปได้สองค่าคือ true หรือ false
byte	จำนวนเต็มไม่มีเครื่องหมาย ตั้งแต่ 0 ถึง 255
int	จำนวนเต็มมีเครื่องหมาย ตั้งแต่ -2147483648 ถึง 2147483647
uint	จำนวนเต็มไม่มีเครื่องหมาย ตั้งแต่ 0 ถึง 4294967295
long	จำนวนเต็มมีเครื่องหมาย ตั้งแต่ -9223372036854775808 ถึง 9223372036854775807
ulong	จำนวนเต็มไม่มีเครื่องหมาย ตั้งแต่ 0 ถึง 18446744073709551615
float	จำนวนจริง (มีทศนิยมได้) เช่น 3.14159
double	จำนวนจริงที่เก็บความละเอียดมากเป็นสองเท่า
string	ข้อความ (สายอักขระ) เช่น Hello

ตัวแปร (Variable)

- ตัวแปร (Variable) เป็นตัวระบุประเภทหนึ่งที่นำมาใช้ในการอ้างถึงข้อมูล โดยค่าของมันสามารถถูกเปลี่ยนแปลงได้ตลอดเวลาที่โปรแกรมกำลังทำงานอยู่
- ในภาษา C# ตัวแปรทุกตัวต้องถูกประกาศก่อนใช้งาน

Datatype variableName;

หรือ

Datatype variableName = initialValue;

ค่าคงที่ (Constants)

- ค่าคงที่เป็นตัวระบุประเภทหนึ่งที่นำมาใช้งานเช่นเดียวกับตัวแปร โดยสิ่งที่แตกต่างคือค่าของมันไม่สามารถเปลี่ยนแปลงได้อีกหลังจากประกาศ
- วิธีการประกาศ จะมีการระบุ const นำหน้า

```
const Datatype constantName = value;
```

ตัวอย่างเช่น

```
const double myconstant = 5.127;
```

นิพจน์ทางคณิตศาสตร์ (Arithmetic Expression)

- นิพจน์ (Expression) หมายถึงส่วนของโปรแกรมที่สามารถถูกตีความเป็นค่าต่างๆ ได้ โดยนิพจน์ประกอบด้วยเทอมเพียงเทอมเดียวหรือเกิดจากการผสมกันของนิพจน์อื่นได้
- ตัวอย่างนิพจน์
  - ตัวเลขโดด เช่น 4.151, 1538
  - ข้อความ เช่น “Hello”
  - ค่าความจริง ได้แก่ true และ false
  - ตัวแปรหรือค่าคงที่เดียวๆ ที่ผ่านการกำหนดค่าแล้ว เช่น myName

ลำดับของตัวดำเนินการ

- ( )
- *, / และ %
- + และ –
- หากตัวดำเนินการมีลำดับเท่าเทียมกัน คำนวณจากซ้ายไปขวา

- หมายเหตุ
- 21/2 จะได้ 10

คำสั่งที่ใช้ในการแสดงผล

- คำสั่งหลักคือ Write และ WriteLine ซึ่งถูกนิยามไว้ในคลาสที่ชื่อว่า Console

```
using System;
class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("Hello");
        Console.ReadLine();
    }
}
```

```
class Program
{
    static void Main(string[] args)
    {
        System.Console.WriteLine("Hello");
        System.Console.ReadLine();
    }
}
```

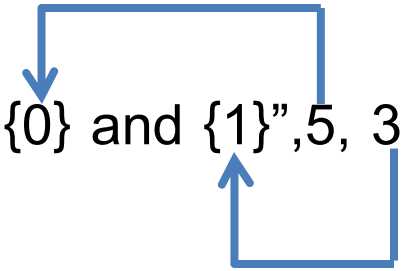
สตริงกำหนดรูปแบบ (Formatting String)

- ในการแสดงผลออกทางจอภาพด้วยคำสั่ง WriteLine หรือ Write ที่ต้องการแสดงมากกว่าหนึ่งค่า (มีพารามิเตอร์มากกว่าหนึ่งตัว) บางครั้งเราต้องการจัดรูปแบบของการแสดงผลเช่น ระบุความกว้าง จำนวนตัวอักษร ชิดซ้าย ชิดขวา โดยเราทำได้ด้วยรูปแบบดังนี้

{index [,alignment][:formatSpecifier]}

ตัวอย่างเช่น

Console.Write(“Two integers are {0} and {1}”,5, 3)

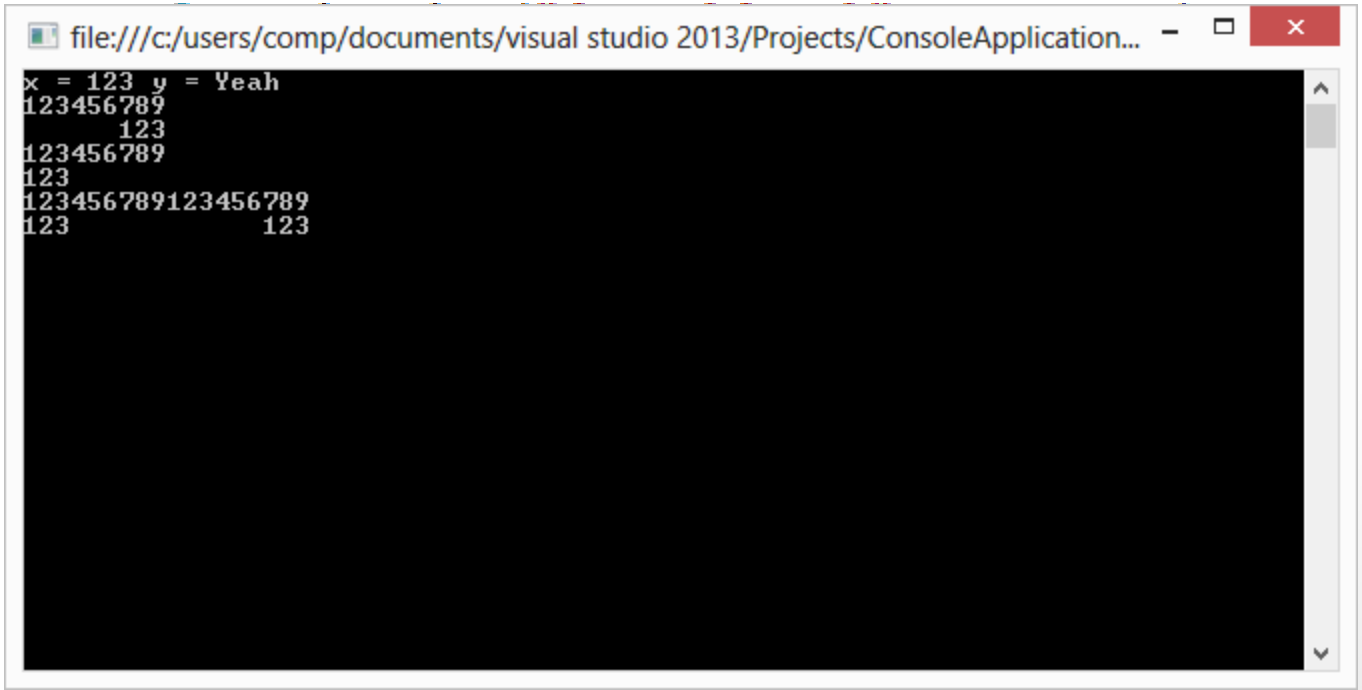


สตริงกำหนดรูปแบบ (ต่อ)

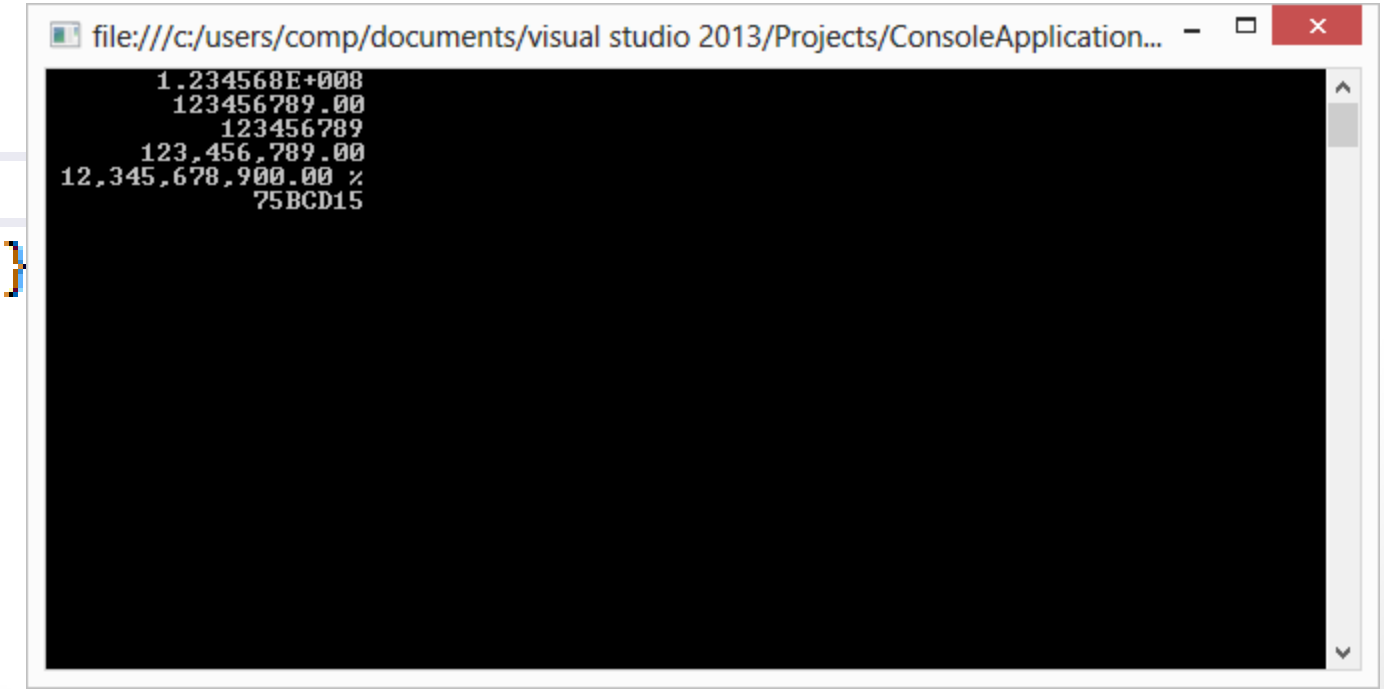
- alignment
  - เป็นจำนวนเต็มที่ใช้ระบุความกว้างหรือจำนวนตัวอักษร ถ้าเป็นเป็นลบชิดซ้าย และบวกชิดขวา

อักขระกำหนดรูปแบบ	ความหมาย
E หรือ e	Exponential (แสดงผลในรูปแบบตัวเลขทางวิทยาศาสตร์)
F หรือ f	Fixed-point (แสดงในรูปแบบทศนิยม)
G หรือ g	General (แสดงในรูปแบบทั่วไป เช่นตัวเลขจะถูกแสดงผลในรูปแบบสั้นที่สุด)
N หรือ n	Number (แสดงในรูปแบบตัวเลขเหมือน Fixed-point แต่มีเครื่องหมาย comma คั่นทุก 3 หลัก)
P หรือ p	Percentage (ตัวเลขจะถูกเปลี่ยนอยู่ในรูปของเปอร์เซนต์)
X หรือ x	Hexadecimal (แสดงในรูปแบบเลขฐานสิบหก)

```
using System;
class MainClass
{
    public static void Main(string[] args)
    {
        Console.WriteLine("x = {0} y = {1}", 123, "Yeah");
        Console.WriteLine("123456789");
        Console.WriteLine("{0,9}", 123);
        Console.WriteLine("123456789");
        Console.WriteLine("{0,-9}", 123);
        Console.WriteLine("1234567891234567 89");
    }
}
```

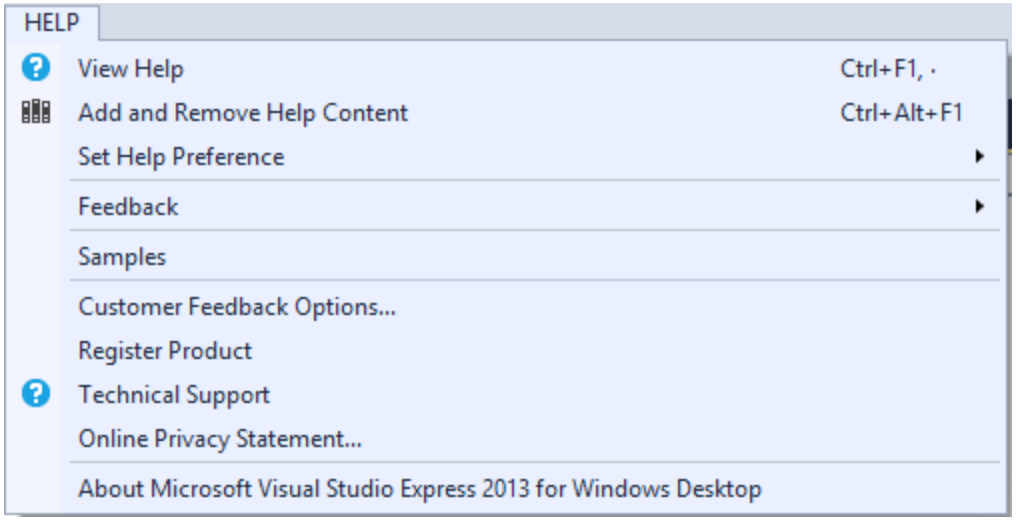


```
using System;
class MainClass
{
    public static void Main(string[] args)
    {
        int n = 123456789;
        Console.WriteLine("{0,20:E}", n);
        Console.WriteLine("{0,20:F}", n);
        Console.WriteLine("{0,20:G}", n);
        Console.WriteLine("{0,20:N}", n);
    }
}
```



HELP

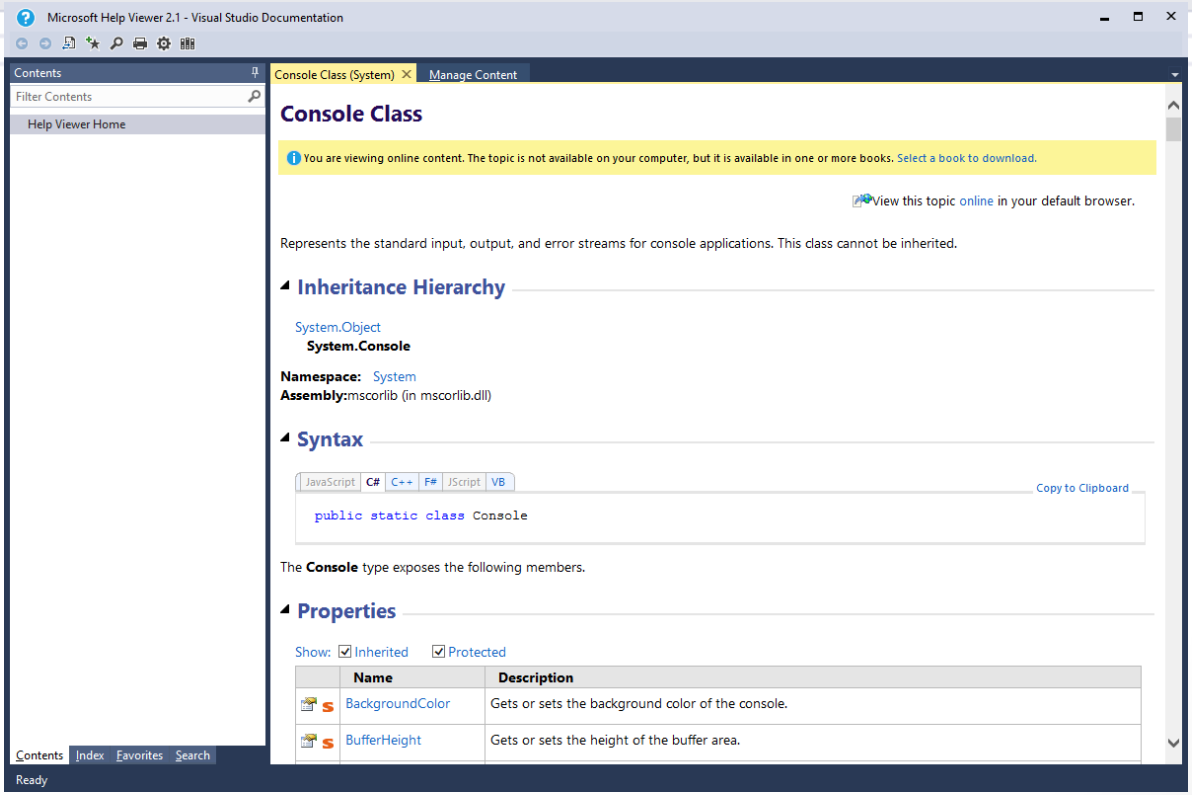
- ในโปรแกรม MS Visual Studio หรือชุดโปรแกรมอื่น ๆ ที่มีการเชื่อมต่อกับระบบไลบราลีของ Microsoft Developer Network (MSDN Library) ซึ่งรวบรวมเอกสารเชิงเทคนิคสำหรับการพัฒนาซอฟต์แวร์ บนไมโครซอฟต์วินโดวส์ไว้โดยละเอียด



```
class Program
{
    static void Main(string[] args)
    {
        System.Console.WriteLine("Hello");
        System.Console.ReadLine();
    }
}
```

```
class Program
{
    static void Main(string[] args)
    {
        System.Console.WriteLine("Hello");
        System.
    }
}
```

class System.Console
Represents the standard input, output, and error streams for console applications. This class cannot be inherited.



คำสั่งสำหรับรับข้อมูลจากผู้ใช้

- จะเรียกใช้งานด้วย Console.ReadLine โดยจะคืนค่าเป็นนิพจน์ที่มีค่าเป็นข้อความ (string) ดังนั้นจะนำค่าที่รับมาไปกำหนดให้กับตัวแปรแบบสตริงหรือผสมกับนิพจน์อื่น ๆ ที่เกี่ยวกับสตริงได้

```
string name;

name = Console.ReadLine();
```

คำสั่งสำหรับรับข้อมูลจากผู้ใช้(ต่อ)

- หากข้อมูลที่รับมาไม่ใช่ข้อความ เราต้องทำการแปลงก่อน อย่างไรก็ตาม C# ไม่มีคำสั่งที่รับข้อมูลชนิดตัวเลขโดยตรง แต่ C# มีเมธอด Parse สำหรับแปลงข้อมูลเป็นตัวเลขแต่ละชนิดให้

  <numeric_datatype>.Parse(<string_expression>)

เช่น

```
double x;

x = double.Parse(Console.ReadLine());
```

คำสั่งแบบมีเงื่อนไข

- IF
- SWITCH

นิพจน์ทางตรรกศาสตร์ (boolean expressions)

การเปรียบเทียบ	สัญลักษณ์ใน C#	ตัวอย่าง	ชนิดข้อมูลที่ใช้ได้	ความหมาย
เท่ากับ	==	x==y	ตัวเลขทุกชนิด, char, string	x เท่ากับ y
ไม่เท่ากับ	!=	x!=y	ตัวเลขทุกชนิด, char, string	x ไม่เท่ากับ y
น้อยกว่า	<	x<y	ตัวเลขทุกชนิด, char	x น้อยกว่า y
น้อยกว่าเท่ากับ	<=	x<=y	ตัวเลขทุกชนิด, char	x น้อยกว่าเท่ากับ y
มากกว่า	>	x>y	ตัวเลขทุกชนิด, char	x มากกว่า y
มากกว่าเท่ากับ	>=	x>=y	ตัวเลขทุกชนิด, char	x มากกว่าเท่ากับ y

นิพจน์ทางตรรกศาสตร์ (ต่อ)

- การนำเอานิพจน์ทางตรรกศาสตร์มาผสมกันมากกว่าหนึ่งนิพจน์จะต้องใช้ตัวเชื่อม ได้แก่
  - && เชื่อมนิพจน์ทางตรรกศาสตร์สองนิพจน์เข้าด้วยกันโดยใช้ตรรกะแบบ “และ” (AND)
  - || เชื่อมนิพจน์ทางตรรกศาสตร์สองนิพจน์เข้าด้วยกันโดยใช้ตรรกะแบบ “หรือ” (OR)
  - ! กลับค่าความจริงของนิพจน์ทางตรรกศาสตร์ เช่น !(x==1) จะเป็นจริงเมื่อตัวแปร x มีค่าไม่เท่ากับ 1

โครงสร้าง if และ if…else…

- **รูปแบบที่ 1 โครงสร้าง if**

```
if(condition)
    statement; // execute if the condition is true


if(condition){
    statement; // execute if the condition is true
    statement; // execute if the condition is true
    statement; // execute if the condition is true
}
```

โครงสร้าง if และ if…else…

- **รูปแบบที่ 2 โครงสร้าง** if…else…

```
if(condition)
    statement; // execute if the condition is true
else
    statement; // execute if the condition is false
```

โครงสร้าง if และ if…else…

- **รูปแบบที่ 3 โครงสร้าง if หลายชั้น**

```
if(condition)
    statement;
else if(condition)
    statement;
else if(condition)
    statement;
    …
else
    statement;
```

โครงสร้าง switch…case

```
switch (expression)
{
    case constant-expression-1:
            statements;
            break;
    case constant-expression-2:
            statements;
            break;
    case constant-expression-3:
            statements;
            break;
            :
    default:
            statements;
            break;
}
```

อนุญาตให้เป็นได้แค่ integer, char, string

string เช็คด้วย “xxx”

char เช็คด้วย ‘x’

คำสั่งวนซ้ำ

- while
- do…while
- for

โครงสร้าง while

```
while (condition)
   statement;

while (condition) {
   statement1;
   statement2;
   :
   statementN;
}
```

โครงสร้าง do…while

```
do statement; while (condition);
```

```
do {
   statement1;
   statement2;
   :
   statementN;
} while (condition);
```

โครงสร้าง for

```
for (init_stmt; condition; update_stmt) {
   statement1;
   statement2;
   :
   statementN;

}
```

Method

- เป็นส่วนของโปรแกรมเพื่อจัดการงานย่อยหนึ่ง ๆ โดยมองงานที่ซับซ้อนเป็นงานย่อย ๆ ที่เล็กลง ทำให้เขียนโปรแกรมแก้ปัญหาได้ง่ายขึ้น ลดการเขียนโค้ดที่ซ้ำซ้อน เพิ่มความสะดวกในการตรวจสอบและแก้ไขโปรแกรม
- แบ่งเป็น 2 ประเภท
  - แบบคืนค่า
  - แบบไม่คืนค่า

Method (ต่อ)

การส่งค่าไปยัง method สามารถทำได้โดย สร้าง method ที่มีการรับค่าจากผู้เรียกเพื่อกำหนดพฤติกรรมการทำงานของ method นั้น ๆ

ค่าที่ถูกส่งไปเรียกว่า argument ส่วน method ที่ถูกเรียกจะรับค่าเหล่านี้ผ่านทาง parameter

การประกาศและเรียกใช้ Method

ก่อนหน้านี้เราเขียนโปรแกรมทั้งหมดไว้ที่ Main แต่สำหรับงานที่ใหญ่ขึ้นเรามักจะเขียนโปรแกรมโดยแบ่งปัญหาที่ต้องการแก้ออกเป็นงานย่อย ๆ หลาย ๆ งาน ทำให้เราเขียนโปรแกรมเป็นส่วน ๆ เพื่อจัดการปัญหาย่อยเหล่านี้ได้

ส่วนของโปรแกรมเพื่อจัดการปัญหาย่อยปัญหาหนึ่ง ๆ มีหลายชื่อเรียกไม่ว่าจะเป็น subroutine, subprogram และ function

แต่ในภาษาที่สนับสนุนการเขียนโปรแกรมเชิงวัตถุเช่น C# เราจะโปรแกรมดังกล่าวว่า method

การประกาศ Method

ที่เราเคยเขียนมาเป็นแบบนี้

```
namespace (มีหรือไม่มีก็ได้)
{
   class{
         Main{
               statement;
         }
   }
}
```

```
namespace (มีหรือไม่มีก็ได้)
{
    class{
            Main{
                    statement;
            }
            MyMethod1{
                    statement;
            }
             MyMethod2{
                    statement;
            }
    }
}
```

การประกาศ method (ต่อ)

- สังเกตว่า method แต่ละอันต้องถูกประกาศอยู่ภายนอก method อื่น แต่อยู่ภายใน class
- method main ไม่จำเป็นต้องถูกประกาศเป็น method แรก

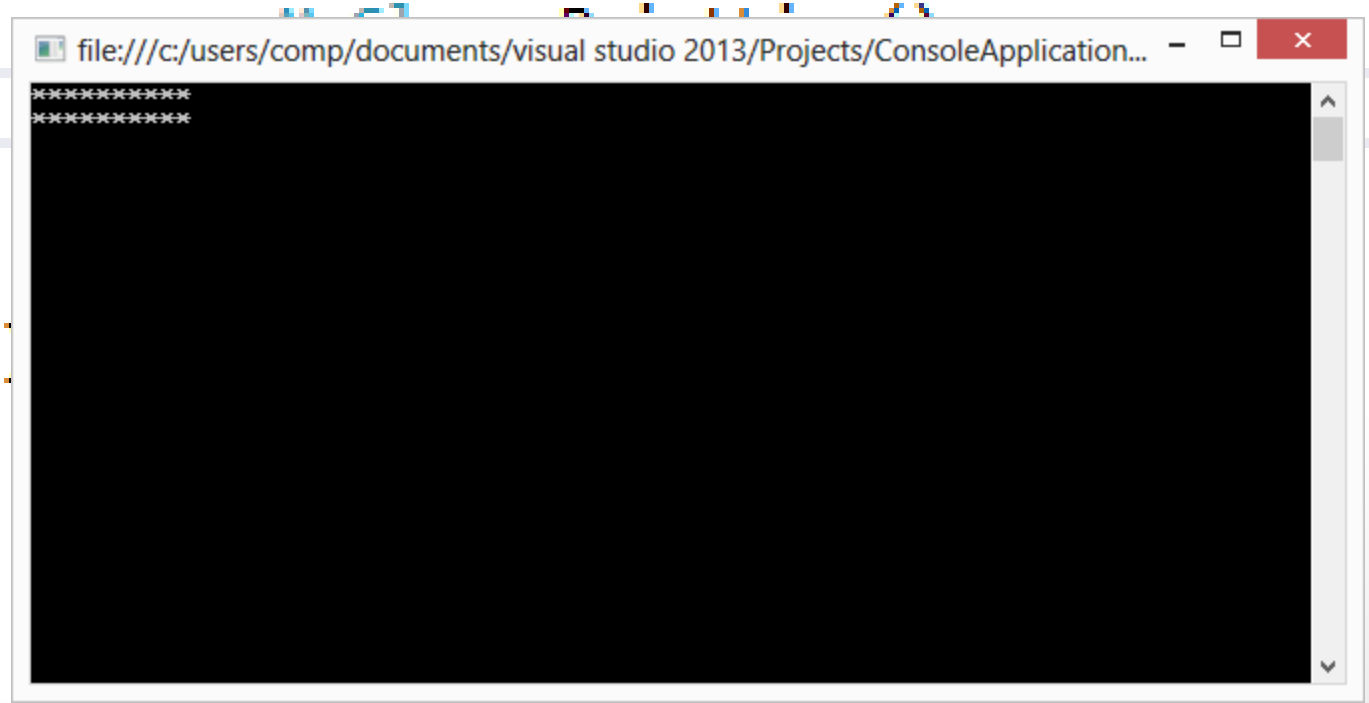
การประกาศ method มีรูปแบบดังนี้

```
static return_type method_name(parameter_list) {
   statement1;
   statement2;
   statementN;
}
```

method แบบไม่คืนค่า

- method แบบไม่คืนค่า บางครั้งเรียกว่า subroutine หรือ procedure เป็น method ที่เขียนขึ้นมาเพื่อปฏิบัติงานบางอย่างและจบงานนั้นในตัวมันเองโดยไม่ส่งค่าคืนกลับไปยังผู้เรียก
- การประกาศ method ประเภทนี้ใช้ void ในตำแหน่ง return_type และการเรียกใช้ method ประเภทนี้จะปรากฏเสมือนคำสั่งหนึ่งคำสั่ง

```
using System;
class MyClass {
    static void PrintLine() {
        for (int i = 0; i < 10; i++)
            Console.Write('*');
        Console.WriteLine();
    }

    static void Main() {
```

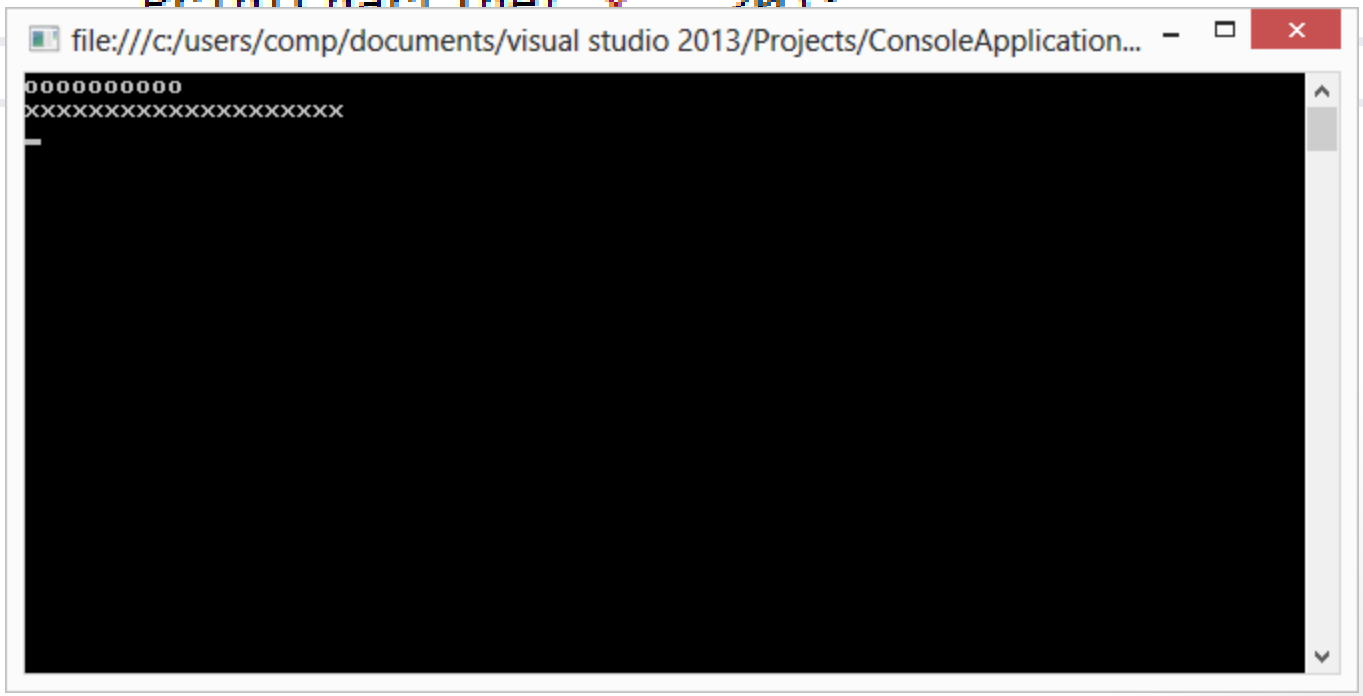


การส่งค่าไปยัง Method

เราสามารถสร้าง method ที่มีการรับค่าจากผู้เรียกเพื่อกำหนดการทำงาน ค่าที่ถูกส่งไปเรียกว่า argument ส่วน method ที่ถูกเรียกจะรับค่าเหล่านี้ผ่านทาง parameter ซึ่งถูกนิยามไว้ในส่วน parameter_list ของการประกาศ method

โดย parameter แต่ละตัวจะต้องมีรูปแบบข้อมูลกำกับไว้เสมอและ argument ที่ใช้ในขณะเรียกใช้งาน method จะต้องมีรูปแบบข้อมูลที่ตรงกัน

```
using System;
class ParamNoRet {
    static void PrintCharLine(char c, int len) {
        for (int i = 0; i < len; i++)
            Console.Write(c);
        Console.WriteLine();
    }
    static void Main() {
        PrintCharLine('o', 10);
        PrintCharLine('x', 20);
    }
}
```



Method แบบคืนค่า

เป็น method ที่ส่งผลลัพธ์กลับไปยังผู้เรียกหลังจากการทำงานใน method เสร็จสิ้น ตัวอย่างเช่น Console.ReadLine(), int.Parse

ในการสร้าง method แบบคืนค่าขึ้นมาเองนั้นต้องระบุชนิดของข้อมูลที่ method จะส่งกลับเอาไว้ในส่วนที่เป็น return_type ของการประกาศ method แทนที่จะใช้ void และภายในตัว method จะต้องมีการใช้คำสั่ง return เพื่อให้ method จบการทำงานและส่งค่ากลับไปยังผู้เรียก

return expression;

Array

Array คือชุดหรือกลุ่มของข้อมูลชนิดเดียวกัน ที่แต่ละตัวเป็นส่วนประกอบหนึ่งของกลุ่มสมาชิกที่เรียงต่อกันเป็นลำดับ โดยที่ข้อมูลแต่ละตัวจะถูกเรียกว่าสมาชิกของอาเรย์ (elements) ทำให้สามารถเก็บข้อมูลได้เป็นจำนวนมาก ๆ โดยที่ไม่ต้องประกาศตัวแปรหลายตัว

การเข้าถึงสมาชิกของอาเรย์แต่ละตัวทำได้โดยการกำหนดดัชนีซึ่งเป็นจำนวนเต็มที่เขียนอยู่ในวงเล็บก้ามปู [] เช่น x[0]

สมาชิกตัวแรกมีดัชนีเป็น 0

การประกาศ Array

- เช่นเดียวกับตัวแปรจะต้องมีการประกาศก่อนการใช้งาน โดยมีรูปแบบดังนี้

```
DataType[] ArrayName;
```

การประกาศแบบนี้สามารถนำไปใช้เพียงแค่อ้างถึง Array ชนิดนั้นๆ เท่านั้น ยังไม่มี Array ที่แท้จริงถูกสร้างขึ้น

การสร้าง Array

ในการสั่งให้คอมพิวเตอร์สร้างอาเรย์ขึ้นมาใน หน่วยความจำของเครื่อง จะใช้คำสั่ง new มีรูปแบบดังนี้

new DataType[num_elements]

DataType คือ ชนิดข้อมูล

num_elements คือนิพจน์แบบจำนวนเต็มแสดงขนาด

ตัวอย่างการประกาศ

ArrayName = new DataType[num_elements]

การสร้าง Array

- การสร้าง Array แบบกำหนดค่าเริ่มต้น

ArrayName = new DataType[num_elements]{value0, value1, …, valueN-1};

หากกำหนดค่าเริ่มต้น ไม่จำเป็นต้องระบุขนาดได้

ArrayName = new DataType[]{value0, value1, …, valueN-1};

และเขียนแบบสั้นได้ว่า

DataType[] ArrayName = {value0, value1, …, valueN-1};

การอ้างถึงข้อมูลใน Array

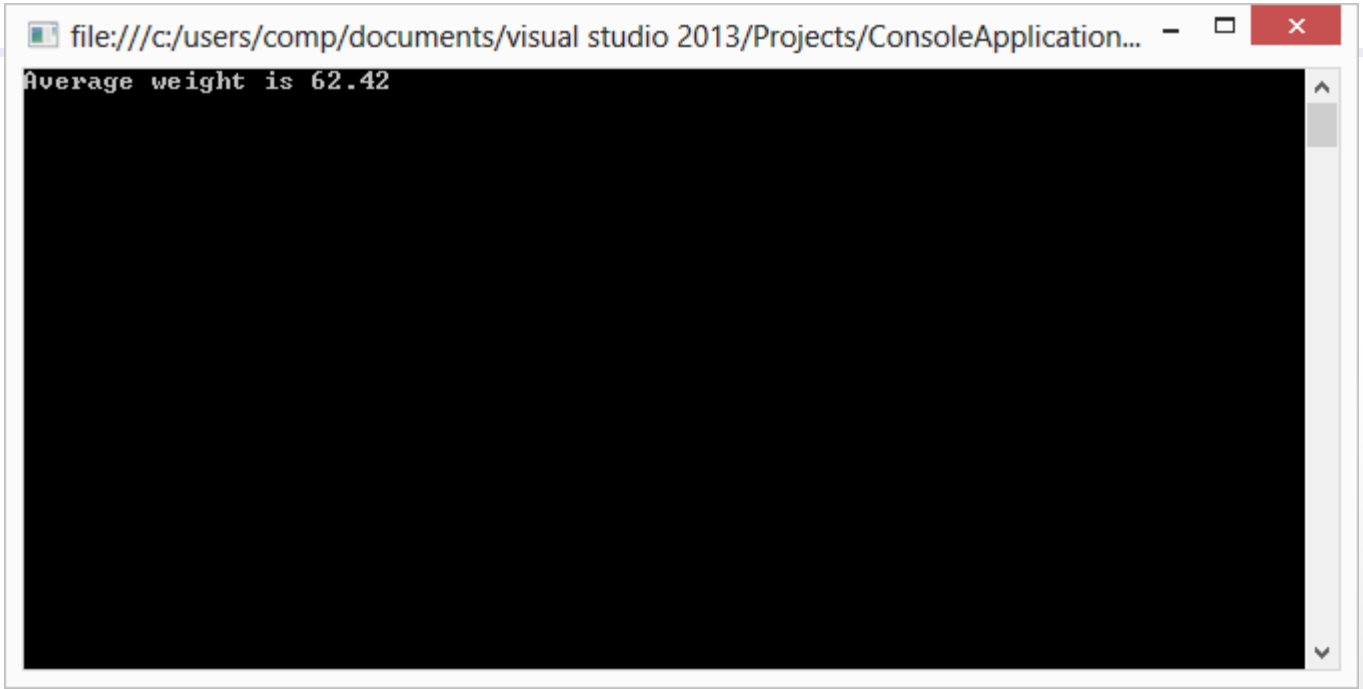
- การอ้างถึงข้อมูลใน Array
- ArrayName[idx]
- การหาขนาดของ Array
- Array.Length

คำสั่ง foreach

คำสั่ง foreach มีไว้เพื่อความสะดวกในการเข้าถึงข้อมูลแบบ Array โดยมีรูปแบบการใช้งานดังนี้

```
foreach (Datatype var in ArrayName)
  statement;
```

```
using System;
class AverageWeight {
    static void Main() {
        double[] weights = {65.5, 44.8, 70.0, 54.2, 77.6};
        double sum = 0.0;
        foreach (double x in weights)
            sum += x;
        Console.WriteLine("Average weight is {0:f2}",
            sum/weights.Length);
        Console.ReadLine();
    }
}
```



การส่ง Array ไปยัง Method

- ทำได้โดยระบุให้พารามิเตอร์ที่รับเข้ามามีชนิดข้อมูลเป็นอาเรย์

```
using System;
class ArrayTest
{
    static double ArraySum(double[] data)
    {
        double sum = 0.0;
        foreach (double x in data)
            sum += x;
        return sum;
    }

    static void Main()
    {
        double[] myData = { 1.0, 2.4, 3.6, 4.8 };
        Console.WriteLine("Sum = {0}", ArraySum(myData));
        Console.ReadLine();
    }
}
```

การอ้างถึง String ในรูป Array

- ข้อมูลแบบ string ในภาษา C# มีลักษณะเหมือนว่าข้อมูลนั้นเป็น array ของสายอักขระ ทำให้เราสามารถใช้การดำเนินการต่างๆ ที่ใช้กับ Array ได้เช่น [], foreach, .Length
- มีข้อจำกัดตรงที่เราทำได้เพียงอ่านอักขระ ณ ตำแหน่งต่างๆ ได้ แต่ไม่สามารถเปลี่ยนแปลงส่วนหนึ่งส่วนใดของข้อความได้

Array 2 มิติ

- ใน C# อนุญาตให้สามารถสร้าง array หลายมิติได้ โดยจำนวนมิติมีได้ตั้งแต่ สองมิติ สามมิติ หรือมากกว่า
- การประกาศและสร้าง array 2 มิติ

การประกาศ

```
DataType [,] ArrayName;
```

การสร้าง

```
new DataType[nrows,ncols]
```

Array 2 มิติ

```
string[,] students;

students = new String[5,3];
```

หรือ

```
string[,] students = new string [5,3];
```

การสร้าง Array 2 มิติโดยกำหนดค่าเริ่มต้น

สามารถทำได้เช่นเดียวกับ array หนึ่งมิติ

```
ArrayName = new DataType[,]{
   {value(0,0), {value(0,1),…, {value(0,ncols-1)},
   {value(1,0), {value(1,1),…, {value(1,ncols-1)},

   {value(nrows-1,0), {value(nrows-1,1),…,
   {value(nrows-1,ncols-1)}
}
```

การสร้าง Array 2 มิติโดยกำหนดค่าเริ่มต้น

```
int[,] A = {
{5, 3, 8},
{2, 6, 10},
{1, 8, 25},
{12, 3, 30}
}
```

การอ้างถึงข้อมูลใน Array

```
ArrayName[ri,ci];
```

การหาขนาดของ array

```
ArrayName.GetLength(dim_idx)
```