ព្រះរាជាណាចក្រកម្ពុជា
ជាតិ សាសនា ព្រះមហាក្សត្រ

* * * * *

មូលដ្ឋានគ្រឹះភាសា C#

BASIC OF PROGRAMMING C#

រៀបចំដោយ៖ សាកលវិទ្យាល័យ ហេង សំរិន ត្បូងឃ្មុំ

សាកលវិទ្យាល័យ ហេង សំរិន ត្បូងឃ្មុំ
UNIVERSITY OF HENG SAMRIN THBONGKHMUM

សៀវភៅ មូលដ្ឋានគ្រឹះភាសា C#

BASIC OF PROGRAMMING C#

សម្រាប់និស្សិតឆ្នាំទី២

រៀបរៀងដោយ ៖ គ្រូឧទ្ទេស ប៉ែន សារ៉ុន

គណៈកម្មការត្រួតពិនិត្យ ៖ លោក អ៊ឹង ឡាសុខ
លោក នី សៀងឡេង
លោក ខុន បណ្ឌិត

បោះពុម្ព ២០២៣

អាសយដ្ឋាន: ផ្លូវជាតិលេខ៧៣ ភូមិនិគមលើ ឃុំស្រឡប់ ស្រុកត្បូងឃ្មុំ ខេត្តត្បូងឃ្មុំ
ទូរស័ព្ទ: (៨៥៥) ១២ ៧៣ ៤២ ៣៣ (៨៥៥) ១២ ៧៩ ៧៨ ៧៩
website: www.uhst.edu.kh

មាតិកា

មាតិកា

ជំពូកទី១ សេចក្តីផ្តើម នាសា C# Language

ជំពូកទី២ សិក្សា អថេរ Variables និង Expressions

ជំពូកទី៣ សិក្សាប្រភេទទិន្នន័យ DataType C#

ជំពូកទី៤ ការសិក្សាស្វែងយល់អំពីប្រភេទ Operator ក្នុងភាសា C#

ជំពូកទី៦ ការប្រើប្រាស់ Decision Statement និង Operators

ជំពូកទី៧ សិក្សាអំពី Array

ជំពូកទី៨ Methods

ជំពូកទី៩ សិក្សា Object Oriented Programming

ជំពូកទី១០ Introduction to ADO.NET

ជំពូកទី១១ ការបង្កើត Projects

ជំពូកទី១ៈ

សេចក្តីផ្តើម ភាសា C# Language

១.និយមន័យ

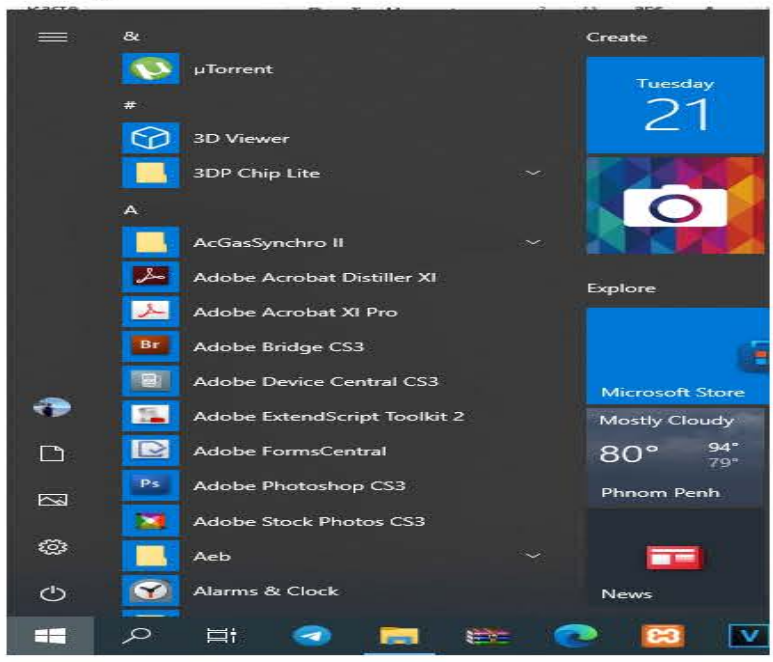
Microsoft Visual C# គឺជា component-oriented language មួយ ដែលមានសារៈសំខាន់បំផុតនៅក្នុងlanguage ផ្សេងទៀតរបស់ក្រុមហ៊ុន Microsoft ។ C# ដើរតួយ៉ាងសំខាន់នៅក្នុង architecture នៃ Microsoft .NET Framework ហើយប្រសិនបើយើងមានចំណេះដឹងនៅក្នុងភាសា C, C# ឬ Java រួចរាល់ហើយនោះ គឺយើងសិក្សា C# បានយ៉ាងងាយស្រួល។

២.របៀបចាប់ផ្តើមដំណើរការ Program ជាមួយ Visual Studio 2013 Environment:

Visual Studio 2013 គឺជា Tool ឬ Environment មួយដែលមានសមត្ថភាពយ៉ាងពេញលេញក្នុងការបង្កើត Projects C# ដែលមានទំហំតូច ឬធំ។ Visual Studio 2013 អាចឲ្យយើងប្រើប្រាស់ C# ដើម្បីបង្កើតជា Console Application ឬ Graphical User Interface។ Console Application គឺជាApplication ទាំងឡាយណាដែល run នៅក្នុង Command Prompt ចំណែក Graphical User Interface គឺ run ចេញជាទម្រង់ Form សម្រាប់ឲ្យ users ងាយស្រួលប្រើប្រាស់។ ដើម្បីបើកកម្មវិធីនោះសូមអនុវត្តតាមជំហានដូចខាងក្រោមៈ

ចុច Start Button >

All Program >

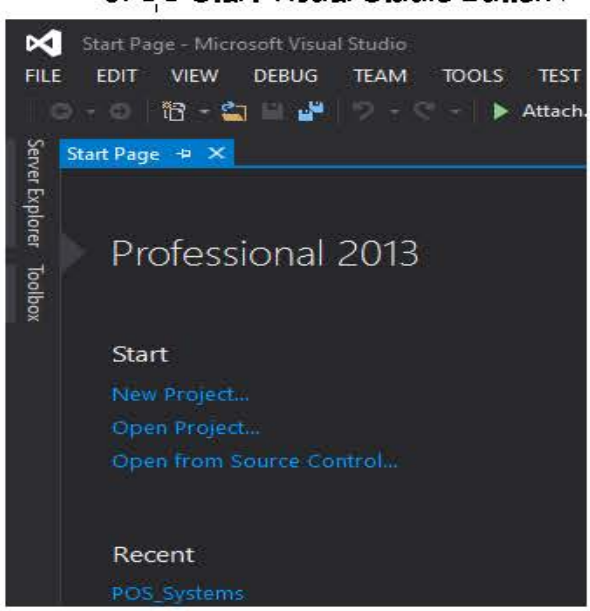


1. Microsoft Visual Studio 2013 >

2. Microsoft Visual Studio 2013 >

5. ជ្រើសរើស យកNew Project ឬ File→New → Project→

6. ចុច Start Visual Studio Button >



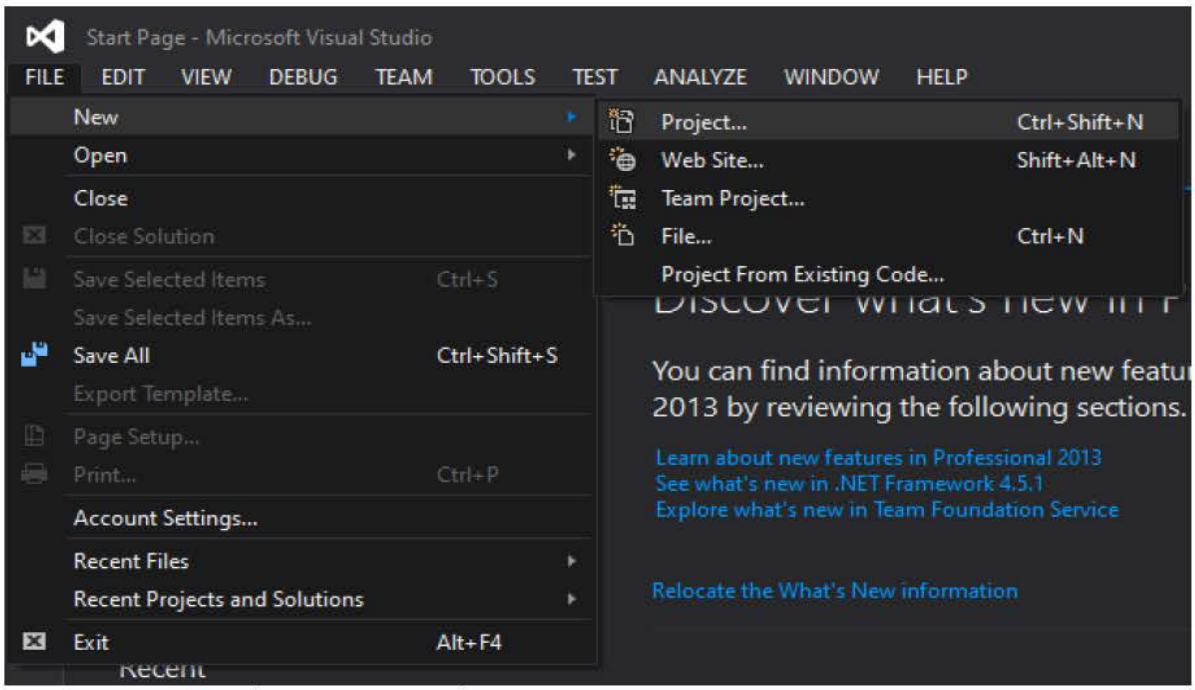
7. ចុច File Menu >

8. New >

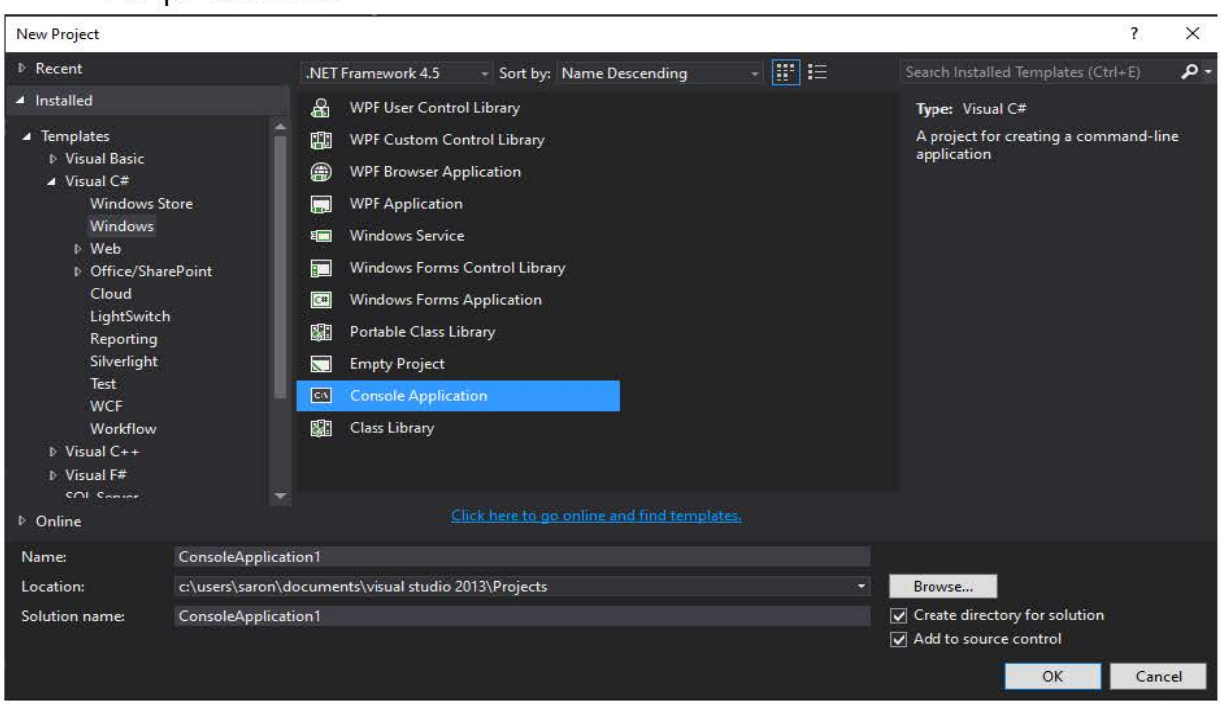
9. Project >

10. ជ្រើសរើស យក Visual C#

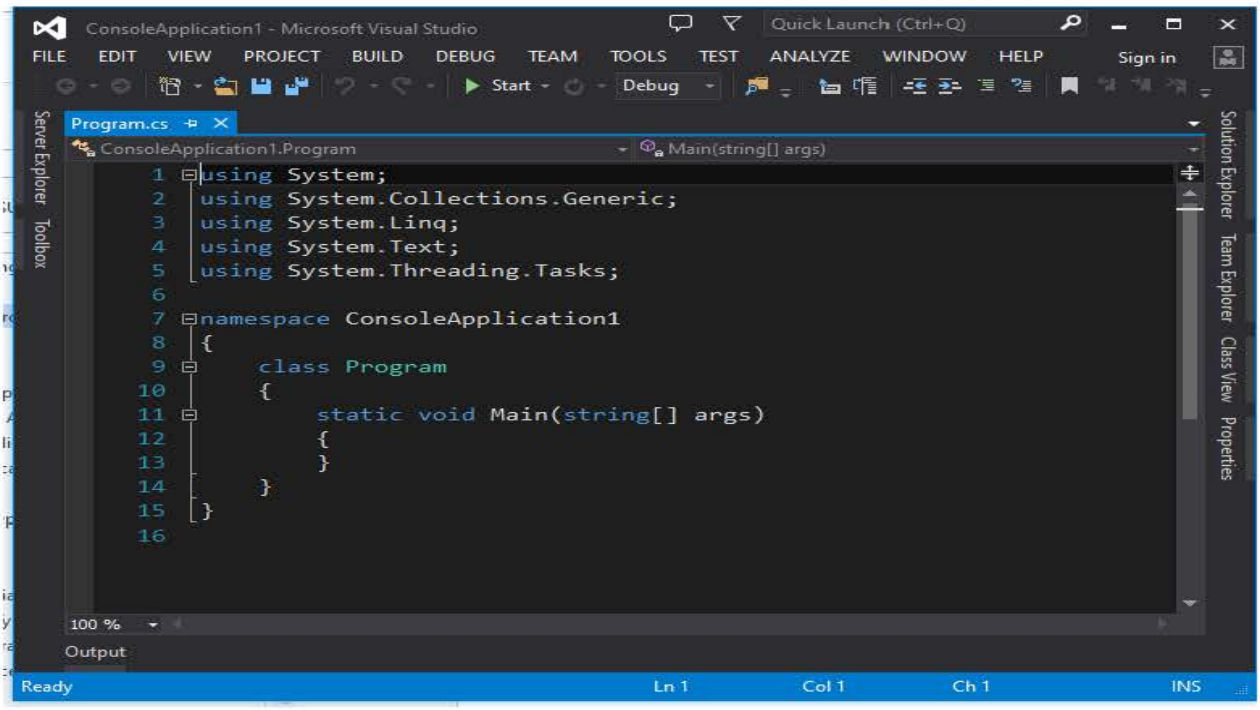
11. ជ្រើសរើស យក Console Application >



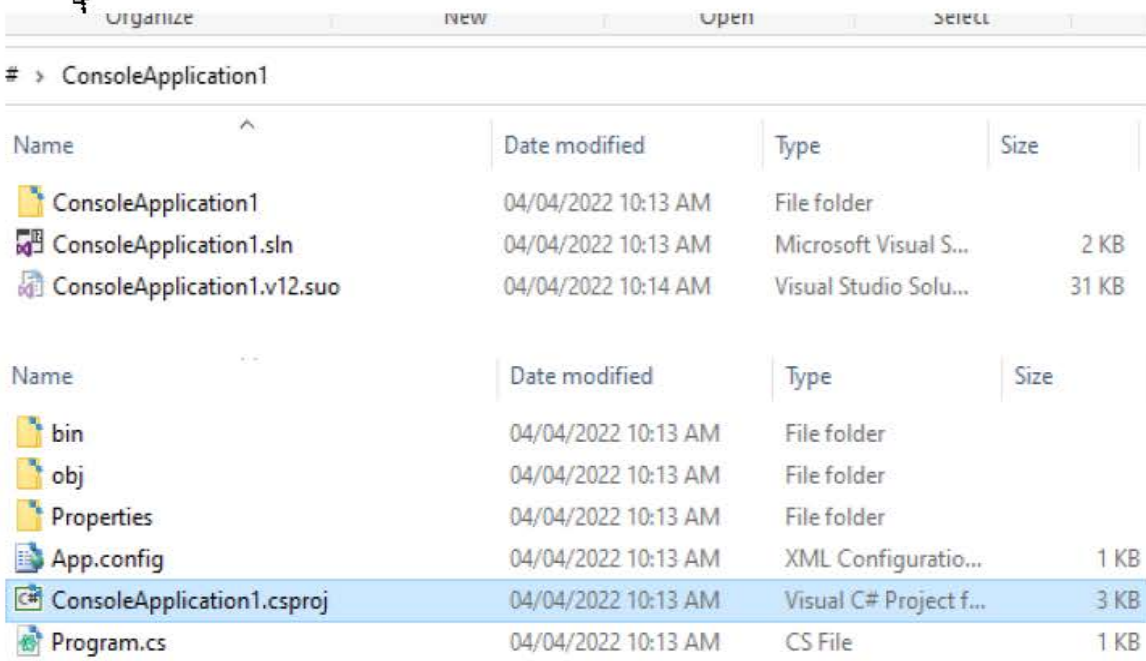
12. ក្នុងប្រអប់ Name សូមដាក់ឈ្មោះ (Ex: Console Application)-->
13. ក្នុងប្រអប់ Browse សូមជ្រើសរើសយកទីតាំងរក្សាទុក-->
14. ចុច OK Button



15. ខាងក្រោមនេះជាលទ្ធផលដែលទទួលបាន



មុននឹងធ្វើការសរសេរកូដ យើងត្រូវស្គាល់ពី Solution Explorer នៅក្នុង Visual Studio ជាមុនសិន ដែលក្នុងនោះមាន:



- ConsoleApplication1: គឺជា top-level solution file ហើយវាមានតែមួយប៉ុណ្ណោះក្នុងមួយ Application ដែលឈ្មោះពិតរបស់វាគឺមានបន្ថែម *.slh នៅខាងក្រោយ (Console Application1 .sln) ។
- ConsoleApplication1: គឺជា C# Project file ដែលនៅក្នុង Solution Folder ឈ្មោះពិតរបស់វាគឺ ConsoleApplication1.csproj ។

- Properties : គឺជា Folder នៅក្នុង ConsoleApplication1 project ដែលនៅក្នុងវាមានដូចជា File មួយ ឈ្មោះ AssemblyInfo.cs (វាគឺជា file ពិសេសដែលអនុញ្ញាត ឲ្យ យើង add ឈ្មោះអ្នកបង្កើត(author), កាលបរិច្ឆេទនៃការ បង្កើត program ។
- References : គឺជា Folder ដែលផ្ទុកនូវ references សម្រាប់ compile Code ដែលយើងសរសេរទៅជា Assembly (កាសាម៉ាស៊ីន)។
- Program.cs : គឺជា C# Source file ដែលតែងតែត្រូវបាន display នៅក្នុង Code and Text Editor Window ។ វាជាកន្លែងដែលយើងត្រូវសរសេរកូដ ដើម្បីបង្កើត Console Application ហើយក្នុងនោះ Visual Studio 2013 បានផ្តល់នូវ Code មួយចំនួនបន្ថែមដោយស្វ័យប្រវត្តិ ដើម្បីជួយឲ្យ Programmer ងាយស្រួលក្នុងការសរសេរ
  កូដ។

៣. ការចាប់ផ្តើមសរសេរកូដ

ឧទាហរណ៍ខាងក្រោមនេះបង្ហាញពីការសរសេរកូដ ដោយបង្ហាញពី Hello wold មកលើ Screen:

```
1  using System;
2  using System.Collections;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace ConsoleApplication5
9  {
10     class Program
11     {
12         static void Main(string[] args)
13         {
14
15                 Console.WriteLine("Heelo world ! ");
16                 Console.ReadKey();
17         }
18
19     }
20 }
```

បំណកស្រាយ ៖

Library file: គឺជា **file library** សម្រាប់

```
1  using System;
2  using System.Collections.Generic;
3  using System.ComponentModel;
4  using System.Data;
5  using System.Drawing;
6  using System.Linq;
7  using System.Text;
```

```
8  using System.Threading.Tasks;
9  using System.Windows.Forms;
```

namespace គឺជាឈ្មោះ **Project**

```
1  namespace ConsoleApplication5
2  {
3      //block code ;
4  }
```

Class Program គឺជាឈ្មោះ Class របស់ project work ឬជា Class មួយឈ្មោះ Program ដែលវាស្ថិតនៅក្នុង Solution Explorer ឈ្មោះ Program.cs file

```
1   class Program
2       {
3           static void Main(string[] args)
4           {
5
6                   Console.WriteLine("Heelo world ! ");
7                   Console.ReadKey();
8           }
9
10      }
```

វាគឺជា Main Function ឬ Function មួយដែលសំខាន់ជាងគេនៅក្នុងCode (គ្រប់ Code ទាំងអស់ត្រូវសរសេរនៅក្នុង Main Function ជានិច្ច)។ C# គឺជាភាសា Case-Sensitive មានន័យថា Main ខុសពី main និងខុសពី MAIN។

```
1  tatic void Main(string[] args)
2   {
3
4    Console.WriteLine("Heelo world ! ");
5    Console.ReadKey();
6   }
```

```
Console.WriteLine("Hello World");
```

គឺសម្រាប់ធ្វើការបង្ហាញពាក្យដែលនៅក្នុង Double Quote(“”) មកលើ Screen ។ Ex: "Hello World" ចំពោះ Console គឺជា Class សម្រាប់ឲ្យយើងប្រើប្រាស់នូវ Standard Input Output មួយចំនួនដូចជា WriteLine សម្រាប់បង្ហាញព័ត៌មានចេញមកលើ Screen ឬ ReadLine សម្រាប់ទទួលយកទិន្នន័យពី Keyboard ។Semicolon(;) ត្រូវបានប្រើប្រាស់នៅខាងចុងនៃ Statement ដើម្បីបញ្ចប់នូវ Statement នីមួយៗជានិច្ច។

៤.ការប្រើប្រាស់ Comment:

Comment ត្រូវបានប្រើប្រាស់នៅក្នុង source code ដើម្បីសរសេរជា statement ខ្លីៗ ការសម្រាប់ធ្វើជាការសម្គាល់ ឬពាក្យពន្យល់ផ្សេងៗ ហើយវាមិនត្រូវបាន read ដោយ compiler នោះទេ ។ ជាទូទៅ comment ត្រូវបានប្រើប្រាស់ដោយ programmer ដើម្បីសរសេរពន្យល់ ឬបញ្ជាក់ពីថ្ងៃដែលចាប់ផ្តើមសរសេរ code ក្នុង source code ។comment មានពីរប្រភេទជា Line Comment និង Block comment។

Line Comment ប្រើសម្រាប់ដាក់ Comment នៅក្នុង source code ជាជួរមួយៗដោយប្រើប្រាស់សញ្ញា double slash (//) Block comment ប្រើប្រាស់សម្រាប់ដាក់ comment នៅក្នុង Source code ជាច្រើនជួរដោយប្រើ /* comment */។

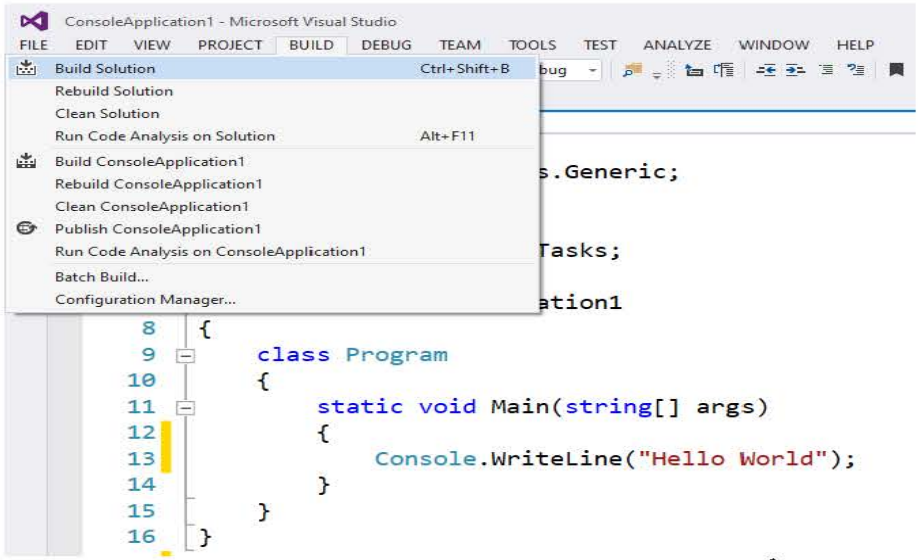
ឧទាហរណ៍ ៖

```
1  using System;
2  using System.Collections;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace ConsoleApplication5
9  {
10     class Program
11     {
12         /*
13           public class Student
14             {
15            public int Id { get; set; }
16            public string Name { get; set; }
17             }
18          */
19         static void Main(string[] args)
20         {
21
22                 Console.WriteLine("Heelo world ! ");
23                 Console.ReadKey();
24         }
25
26     }
27 }
```

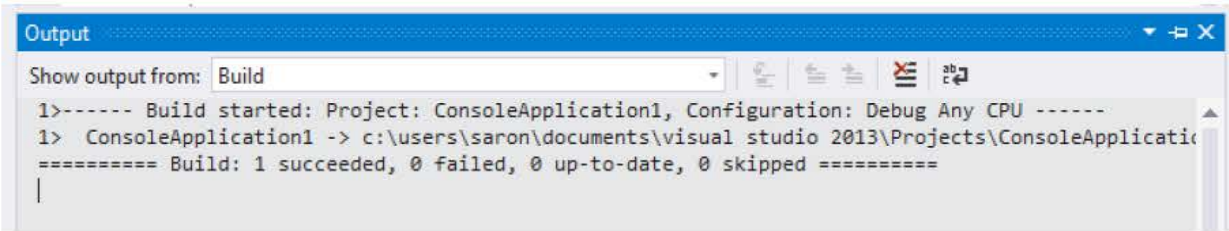
៥.របៀប Build Console Application

1. Build Menu -->

2. Build Solution (Ctrl+ Shift + B) -->



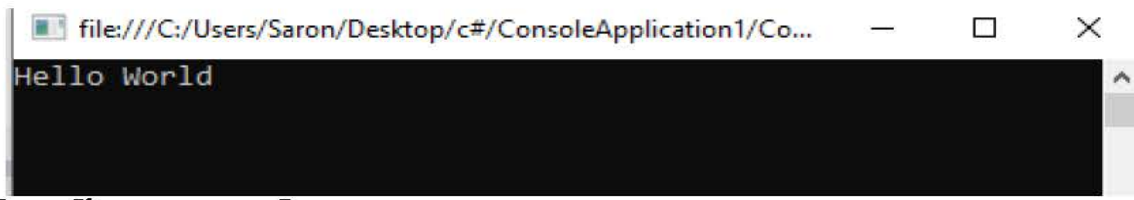
3. បន្ទាប់មកវានឹងបង្ហាញពីផ្ទាំង **Output windows** ដែលជាដំណើរការ **Compile** ទៅលើ Code ដែលបានសរសេរ



4. ចុច **Debug Menu**

5. Start Without Debugging (Ctrl+F5)

6. ខាងក្រោមនេះជាលទ្ធផលដែលទទួលបាន



៦. **សិក្សាពី namespace និង Assembly:**

Using Statement គឺត្រូវបានប្រើប្រាស់នៅខាងមុខ namespace ដើម្បីនាំយក items (Method ឬ Properties) របស់ Class មកប្រើប្រាស់ដោយសេរីនៅក្នុង Source Code។

```
1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel;
4 using System.Data;
5 using System.Drawing;
6 using System.Linq;
7 using System.Text;
8 using System.Threading.Tasks;
```

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

Class ដែលបានសរសេរគឺត្រូវបាន compiled ទៅជា Assemblies ដែលវាជាFile មួយមាន extension *.dll ឬទៅជា *.exe file។

៧.ការបង្កើត Graphical Application:

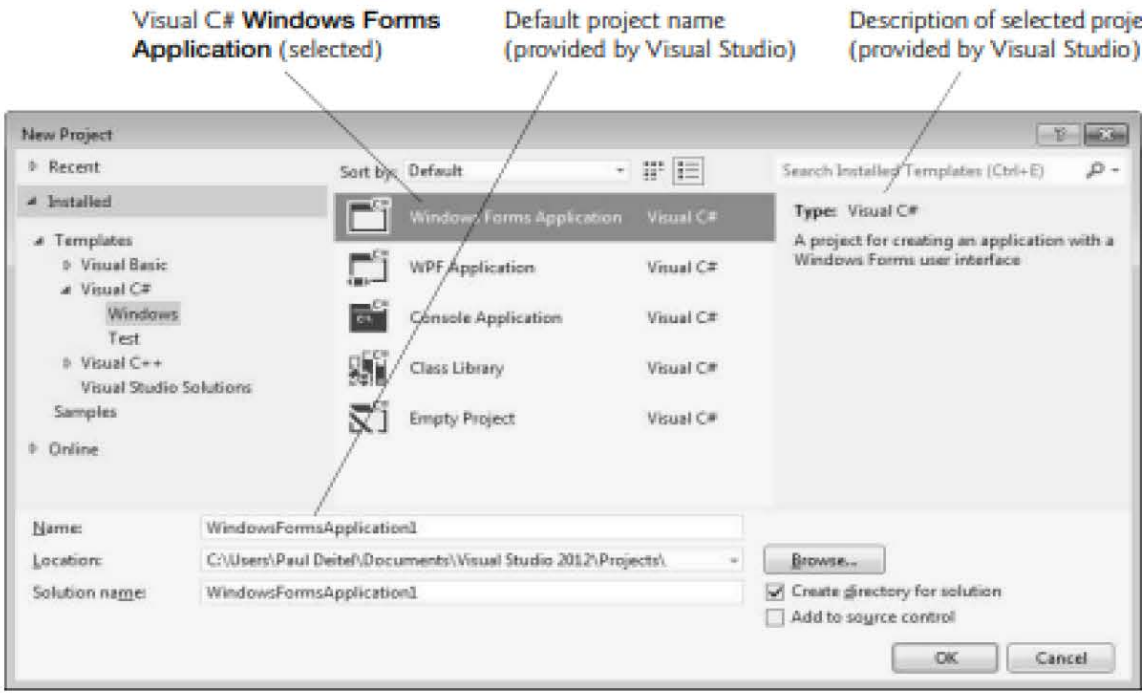
ចំពោះការបង្កើត Graphical Application, Visual Studio 2008 បានផ្តល់នូវ Views ពីសម្រាប់ឲ្យប្រើប្រាស់ដូចជា៖

- Design View : សម្រាប់រៀបចំ Layout ឬទម្រង់របស់ Form ដែលត្រូវបានបង្កើត។

- Code View : សម្រាប់កែប្រែ ឬសរសេរកូដបន្ថែមទៅឲ្យ Application។ ក្នុងនោះ Visual Studio 2013 បានផ្តល់នូវ templates ចំនួនពីរសម្រាប់ build graphical application ។

- Windows Forms Application គឺជា technology ដំបូងគេរបស់ .NET Framework version 4.0 ។

- Windows Presentation Foundation គឺជា enhanced technology ថ្មីដែលត្រូវបានបង្ហាញនៅក្នុង .NET Framework version 4.0 ដោយវាបានបន្ថែម features សំខាន់ៗមួយចំនួនទៀតលើស Windows Forms



១. ចុច File Menu -->

២. New -->

៣. Project (Ctrl+Shift+N)

៤. ត្រង់ Templates ជ្រើសរើសយក Visual C#

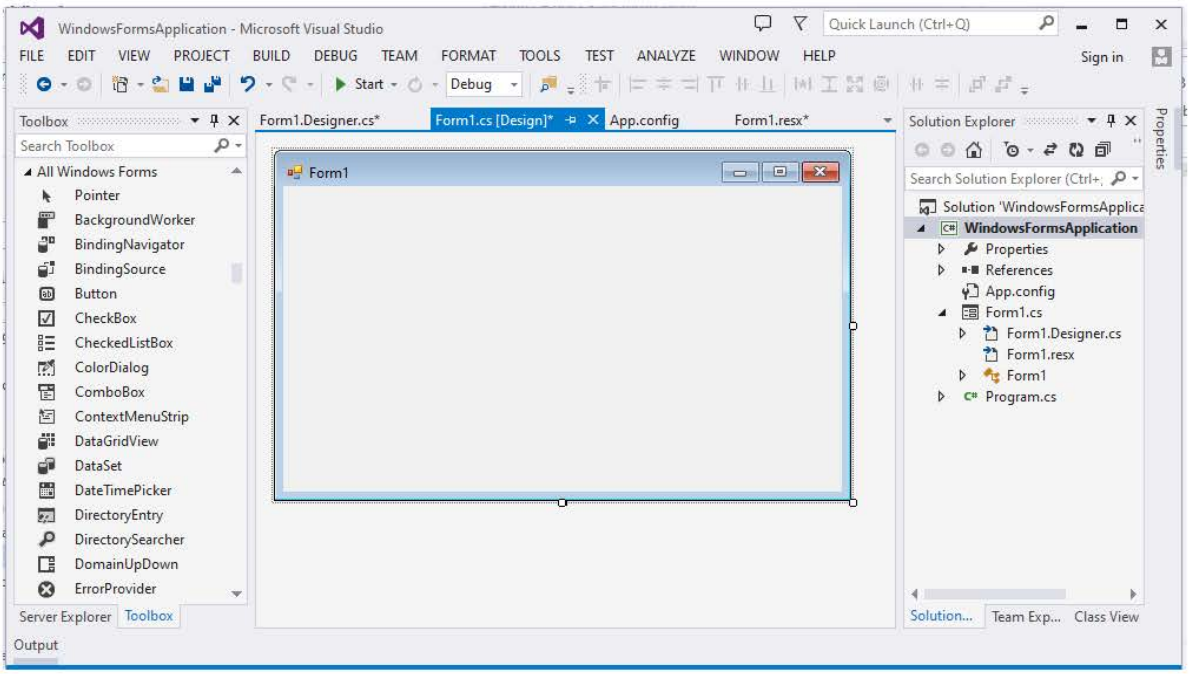
៥. Windows Forms Application

៦. ក្នុងប្រអប់ Name ដាក់ឈ្មោះ WindowsFormsApplication

៧. ក្នុងប្រអប់ Dowse ជ្រើសរើសយកទីតាំងរក្សាទុក

៨. ចុច OK Button

៩. បន្ទាប់មកវានឹងបង្ហាញ Design View Window រួចទាំង XAML Windows (extensible Application Markup Language) ដូចខាងក្រោម ៖

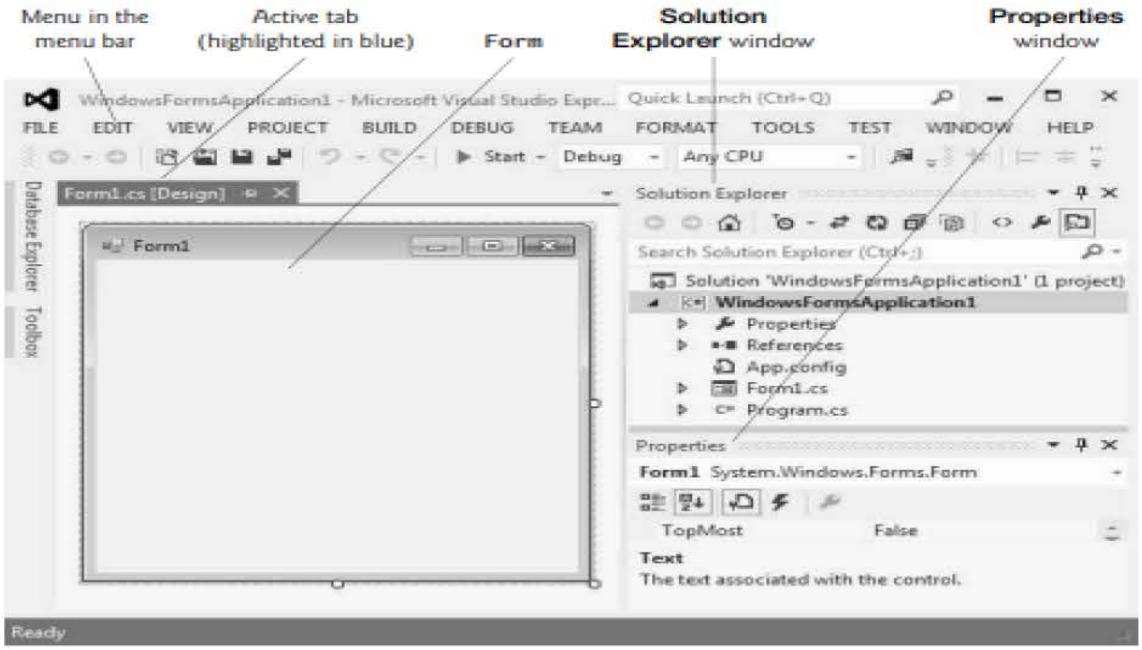


Toolbox គឺជាTool សម្រាប់ប្រើប្រាស់ក្នុងការ Design platform ដើម្បីបង្កើត ប្រព័ន្ធគ្រប់គ្រង Solution Explorer គឺជាFolder សម្រាប់ផ្ទុក file of Project Management Property គឺជា Tool សម្រាប់ធ្វើការបញ្ជូសកម្មភាពផ្សេងៗ និងមុខងារលើ Toolbox នីមួយៗ ។

មុខងារ សំខាន់ៗរបស់ Property មាន Category និង Even ៖

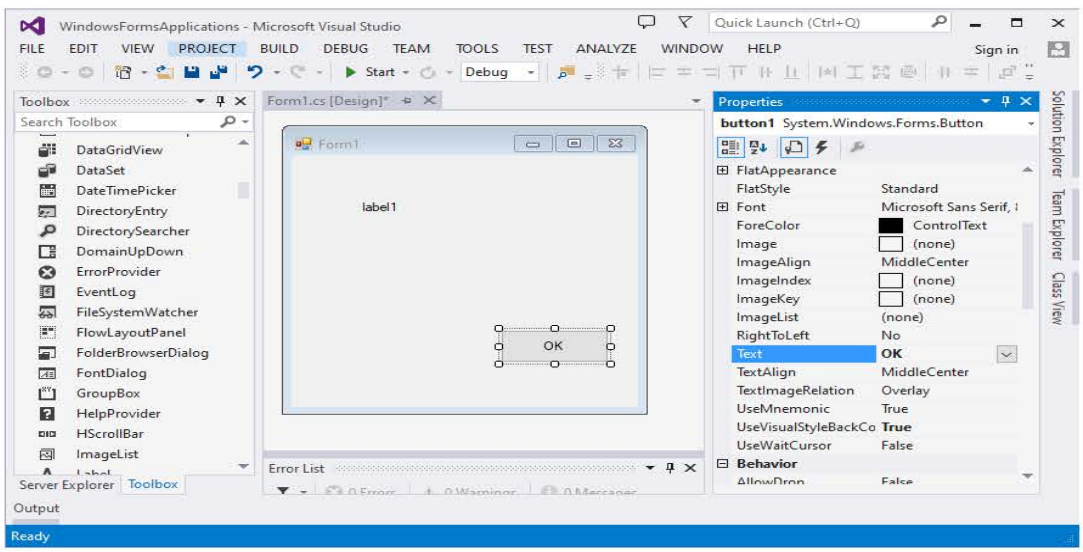
- Category Property គឺជាប្រភេទ មុខងារ ផ្សេងៗរបស់ Toolbox ក្នុងការងារDesign
- Even Properties គឺជាមានមុខងារសម្រាប់បញ្ជូទៅលើសកម្មភាព របស់ Toolbox មាន

ដូចជាៈ ចុច, កណ្តុរឆ្លាក់កាត់, កណ្តុរចាក់ចេញ, កណ្តុរចុច, កណ្តុរផ្លៀស់ប្តូរ, ។ល។



៨ **ការសរសេរសេរ** Code Graphical Application:

១. ចុច Double លើ Button OK>



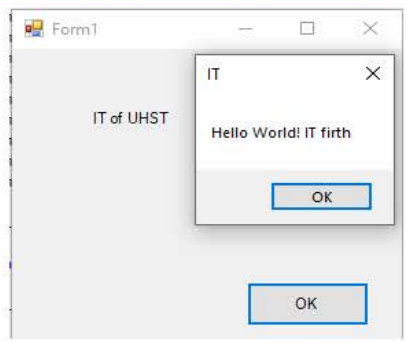
២. បន្ទាប់យើងសរសេរកូដនៅក្នុងចន្លោះ { } របស់ private void tbok_Click ដូចខាងក្រោម

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
```

```
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Windows Forms Applications
{
    public partial class Form1: Form
    {

    public Form1()
    {
      InitializeComponent();
    }

    private void btok_Click(object sender, EventArgs e)
    {
        label1.Text = "IT of UHST";
        MessageBox.Show("Hello World! IT firth");
    }

}
```

៣. ចុច Debug Menu ឬ start

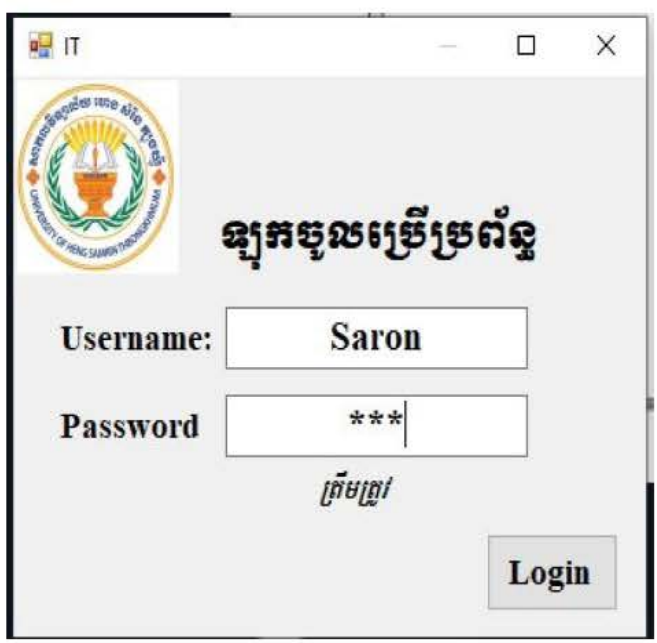
៤. Start Without Debugging



៥. លទ្ធផល

៩. លំហាត់

១. ក. ចូរសរសេរ code មួយដើម្បី display ព័ត៌មានមួយចំនួនដូចខាងក្រោម៖



១.ខ. លទ្ធផល លំហាត់ទី១ (បង្ហាញទិន្នន័យ ចាប់Data ពីForm IT មកបង្ហាញ នៅForm Dashboard_Home)

ជំពូកទី ២

សិក្សា អថេរ Variables និង Expressions

១. សំណើ Statements

Statement គឺជាការបញ្ជាដើម្បីដំណើរការការងារណាមួយនៅក្នុង Source Code ហើយ Statement នីមួយៗ ត្រូវបានបញ្ចប់ដោយសញ្ញា Semicolon (;) ។

ឧទាហរណ៍៖

```
static void Main(string[] args)
{
    Console.WriteLine("Hello World");
    Console.ReadKey();
}
```

ចំពោះ C# គឺប្រភេទ free format language ដែលមានន័យថាក្នុងការបន្ថែម space ទទេ , Tab, ឬ Enter នៅក្នុង Source Code គឺមិនធ្វើឲ្យមានភាព Error កើតឡើងនោះទេ។

២. ការប្រើប្រាស់ អថេរ Variable:

Variables គឺជាកន្លែងរក្សាទុកទិន្នន័យ memory។ គ្រប់ Variables ទាំងអស់ត្រូវតែមានឈ្មោះ និងប្រភេទទិន្នន័យដែលវាត្រូវផ្ទុកហើយក្នុងនោះ ត្រូវ declare(ប្រកាស)វាជាមុនទើបអាចប្រើប្រាស់បាននៅពេលក្រោយ។

៣. ការប្រកាស អថេរ Declaring (Creating) Variables

ដើម្បីប្រកាសអថេរយើងត្រូវដឹងប្រភេទនៃអថេរ ការកំណត់ ឈ្មោះ Variable_name ឱ្យអថេរ និងតម្លៃរបស់ Value ។

Syntax

```
type variable_Name = value;
```

៤.សិក្សាពី Identifiers:

Identifiers គឺជាការដាក់ឈ្មោះឲ្យខុសៗគ្នាទៅឲ្យ elements នៅក្នុង programs ដែលមានដូចជា Variables, names-paces, classes, ឬ methods ដែលការដាក់ឈ្មោះគឺត្រូវបាន និងទៅតាមក្បួនខ្នាតត្រឹមត្រូវដែលបានទទួលស្គាល់ដោយ ភាសាC# ។ ក្នុងការកំណត់ Identifiers ត្រូវកំណត់តាមលក្ខខណ្ឌដូចខាងក្រោម៖

ក. តួអក្សរដំបូងចាប់ផ្តើមដោយ អក្សរ ឬ underscore ប៉ុន្តែមិនមែនជាលេខ

ឧទាហរណ៍៖

Identifiers	លទ្ធផល	ហេតុផល
Enter	ត្រូវ	ចាប់ផ្តើមដោយអក្សរ
_score	ត្រូវ	ចាប់ផ្តើមដោយ underscore
3plan	ខុស	ចាប់ផ្តើមដោយលេខ
Plan3	ត្រូវ	ចាប់ផ្តើមដោយអក្សរមុនលេខ

ខ. មិនអនុញ្ញាតឲ្យប្រើប្រាស់ Space ឬសញ្ញាពិសេស (#, $, *, + ,....)

ឧទាហរណ៍៖

Identifiers	លទ្ធផល	ហេតុផល
Enter Center	ខុស	មិនអាចប្រើ Space បានទេ
Result%	ខុស	មាននិមិត្តសញ្ញ%
Football Team$	ខុស	មាននិមិត្តសញ្ញ%

គ. មិនអនុញ្ញាតឲ្យប្រើប្រាស់ មួយនឹង Reserved Identifiers ដែលមានចំនួន 77 identifiers (Keyword)

C++ Keywords	C++ Keywords	C++ Keywords
abstract	fixed	sealed
as	float	short
base	for	sizeof
bool	in	stackalloc
break	int	static
byte	interface	string
case	internal	struct
break	is	switch
case	lock	this
catch	long	throw
char	namespace	true
checked	new	try
class	null	typeof
const	object	uint
continue	operator	ulong
decimal	out	unchecked
delegate	override	unsafe
do	params	unshort
double	private	using
else	protected	virtual
enum	public	void

even	readonly	volatile
extern	ref	while
false	return	
finally	sbyte	

ចំពោះ keywords ដែលប្រើប្រាស់នៅក្នុង Code and Text Editor window គឺតែងតែបង្ហាញពណ៌ខៀវជានិច្ច។

៥.ការប្រកាស Variables (Variables Declaration):

នៅពេលដែលយើងប្រកាស Variable គឺយើងត្រូវធ្វើស្គាល់ data type (ប្រភេទទិន្នន័យ) ដែលវាត្រូវទទួលយកផងដែរ ។ Data type មានដូចជា៖ ចំនួនគត់ (integers), លេខក្បៀស (floating-point numbers), ឬអក្សរ (string) , Bool (true, Fail)....ជើម។ ហើយការប្រកាស Variable គឺ ជាការប្រាប់ទៅដល់ Compiler ឲ្យរៀបចំទីតាំង memory សម្រាប់រក្សាទុកនូវទំហំ និងប្រភេទទិន្នន័យដែល Variable នោះទទួលយក និងត្រូវតាមប្រភេទទិន្នន័យ ។

ឧទាហរណ៍៖ ខាងក្រោមនេះគឺជាការប្រកាស Variable មួយឈ្មោះ age មានប្រភេទទិន្នន័យ (Data Type) ជាចំនួនគត់ integer

បន្ទាប់ពីយើងបានធ្វើការប្រកាស Variable រួចរាល់ហើយនោះ គឺយើងអាចធ្វើការ assign តម្លៃទៅឲ្យ Variable បានផងដែរ ។ សញ្ញា (=) គឺជា assignment operator ដែលវាត្រូវបានប្រើប្រាស់ដើម្បី បោះតម្លៃដែលនៅខាងស្តាំទៅឲ្យ Variable ដែលនៅខាងឆ្វេង ។

```
static void Main(string[] args)
{
    int A;
    A = 4;
    Console.WriteLine(A);
    Console.ReadKey();
}
```

ឧទាហរណ៍៖ ខាងក្រោមនេះគឺជាតម្លៃ 30 ទៅឲ្យ Variable មួយឈ្មោះ Saron_age:

```
static void Main(string[] args)
{
    int Saron_Age;
    Saron_Age = 30;
    Console.WriteLine(Saron_Age);
    Console.ReadKey();
}
```

បន្ទាប់ពីបាន assign តម្លៃទៅឲ្យ Variable ហើយនោះគឺយើងអាចធ្វើការ display វាមកលើ Screen បានហើយក្នុងនោះ Screen គឺបង្ហាញតម្លៃរបស់ Variable មិនមែនបង្ហាញឈ្មោះ Variable នោះទេ។

៦.ការបង្ហាញ Variable (Display Variables)

ដើម្បីបង្ហាញ អថេរ យើងប្រើប្រាស់ `WriteLine()` method គឺសម្រាប់បង្ហាញ Variable values ក្នុងConsole windows ។

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   namespace Lession_2
8   {
9       class Program
10      {
11          static void Main(string[] args)
12          {
13              string name = "Hello !";
14              Console.WriteLine(name);
15              Console.ReadKey();
16          }
17      }
18  }
19  //Output: Hello !
```

៦.ការបញ្ចូលតួអក្សរ ឬតួអក្សរ និង Variable (+)

ដើម្បីបញ្ចូលអត្ថបទ និងអថេរចូលគ្នា គឺយើងប្រើប្រាស់នូវ សញ្ញា (+) Character តួអក្សរ ។

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   namespace Lession_2
8   {
9       class Program
10      {
11          static void Main(string[] args)
12          {
13              string name = "World !";
14              Console.WriteLine("Hello" + name);
15              Console.ReadKey();
16          }
17      }
18  }
19  //Output: Hello World !
```

ឧទាហរណ៍ទី២៖

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

namespace Lession_2
{
    class Program
    {
        static void Main(string[] args)
        {
            string First_Name = "PEN";
            string Last_name = "Saron";
            Console.WriteLine(First_Name + Last_name);
            Console.ReadKey();
        }
    }
}
//Output: PEN Saron
```

៨. ការប្រកាស អថេរ ច្រើនអញ្ជាន Declare Many Variables

ដើម្បីប្រកាសអថេរ ច្រើនអញ្ជាតយើងត្រូវដឹងប្រភេទនៃអថេរ ការកំណត់ ឈ្មោះ Variable_name ឱ្យអថេរ និងតម្លៃរបស់ Value ។

Syntax

```
type variable_Name1 = value , variable_Name2 = value,
     variable_Name3 = value,......,variable_NameN = value;
```

ឧទាហរណ៍ទី២៖

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

namespace Lession_2
{
    class Program
    {
        static void Main(string[] args)
        {
            int x = 5, y = 6, z = 50;
```

```
15              Console.WriteLine(x + y + z);
16              Console.ReadKey();
17          }
18      }
19  }
20   //Output: 61
```

៩. ការប្រកាស អថេរ ប្រើប្រាស់បានឯកជន

ដើម្បីប្រកាសអថេរ ច្រើនអញ្ញតយើងត្រូវដឹងប្រភេទនៃអថេរ ការកំណត់ ឈ្មោះ Variable_name ឱ្យអថេរ និងតម្លៃរបស់ Value ។ ហើយយើងហៅវាទៅប្រើប្រាស់បានច្រើនកន្លែងក្នុង Project របស់យើងបាន ។

Syntax

```
Private static data_type variable_Name1 = value
```

១០. ការប្រកាស អថេរ ប្រើប្រាស់បានច្រើនកន្លែង

ដើម្បីប្រកាសអថេរ ច្រើនអញ្ញតយើងត្រូវដឹងប្រភេទនៃអថេរ ការកំណត់ ឈ្មោះ Variable_name ឱ្យអថេរ និងតម្លៃរបស់ Value ។ ហើយយើងហៅវាទៅប្រើប្រាស់បានច្រើនកន្លែងក្នុង Project របស់យើងបាន ។

Syntax

```
public static data_type variable_Name1;
```

១១. សិក្ខាពី Primitive Data Type:

Data Type គឺត្រូវបានប្រើប្រាស់ជាមួយនឹង Variable ដើម្បីធ្វើការបញ្ជាក់ពីប្រភេទទិន្នន័យពិតប្រាកដ Variable ត្រូវទទួលយក។

Data Type	Description	Size(bits)	Range	Sample usage
int	លេខចំនួនគត់	32 bits = 4 bytes	-2^{31} ដល់ $2^{31}-1$	Int count;
long	លេខចំនួនគត់	64 bits = 8 bytes	-2^{63} ដល់ $2^{63}-1$	long wait;
float	លេខមានក្បៀស	32 bits = 4 bytes	$\pm1.5 \times 10^{45}$ ដល់ $\pm3.4 \times 10^{38}$	float away; away = 0.42F;
double	លេខមានក្បៀស	64 bits = 8 bytes	$\pm5.0 \times 10^{-324}$ ដល់ $\pm1.7 \times 10^{308}$	Double trouble; trouble = 0.42;
decimal	លេខមានក្បៀស	128 bits = 16 bytes	28 significant figures	decimal coin; coin = 0.42M;
string	តួអក្សរច្រើនតួ	16 bits ក្នុង 1 តួ	មិនកំណត់	string vest; vest ="fortytwo;
char	តួអក្សរមួយតួ	16 bits = 2 bytes	0 ដល់ $2^{16}-1$	char grill; grill = 'x';

bool	Boolean	8 bits = 1 byte	True or False	Bool teeth; teech = false;
datetime	ពេលវេលា	dd/mm/yyyy h:mm:ss		

១១.១ ការប្រកាស variable

```
Int a = 123;

double b=2345678;
```

integrated ដូច្នេះនៅពេលគណនា int គឺ Data Type មានទំហំតូចជាង double គឺត្រូវ Convert ទៅជា double មុនទើបធ្វើការគណនាតាមក្រោយ។

ចំពោះ Arithmetic Operator មួយទៀតដែលត្រូវបានប្រើប្រាស់នៅក្នុង Program ដែរ គឺ Modulus Operator (%) ការគណនារបស់វាគឺយកសំណល់នៃផលចែកណាមួយមកធ្វើជាលទ្ធផលរបស់វា។

ក្នុងភាសា C ឬ C++ មិនអនុញ្ញាតឲ្យ Modulus ប្រើប្រាស់ជាមួយនឹង Floating-Point Number នោះទេគឺអាចប្រើប្រាស់បានជាមួយនឹង Integer តែប៉ុណ្ណោះ។ ប៉ុន្តែចំពោះ C# វិញគឺអាចប្រើប្រាស់ជាមួយនឹង Integer ក៏បាន ឬ Floating-Point Number ក៏បានផងដែរ។

ឧទាហរណ៍៖

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

namespace Lession_2
{
    class Program
    {
        static void Main(string[] args)
        {
            int x = 5, y = 6, z = 50;


            Console.WriteLine(x + y + z);
            Console.ReadKey();
        }
    }
}
 //Output: 61
```

១២. កន្សោមពីជគណិត ទម្រង់បន្ទាត់ Arithmetic Expressions in Straight-Line Form

កន្សោមនព្វន្តត្រូវតែសរសេរជាទម្រង់បន្ទាត់ត្រង់ដើម្បីសម្រួលដល់ការបញ្ចូលកម្មវិធី ចូលទៅក្នុងកុំព្យូទ័រ។ ដូច្នេះកន្សោមដូចជា "a បែងចែកដោយ b" ត្រូវតែសរសេរជា a/b ដូច្នេះ

ថាអថេរ អថេរ និងសញ្ញាប្រមាណវិធីបង្ហាញក្នុងបន្ទាត់ត្រង់។ ពិជគណិតខាងក្រោម ចំណាំមិនអាចទទួលយកបានចំពោះអ្នកចងក្រង៖

$$\frac{a}{b}$$

១៣. ការដាក់ឥដ្ឋក្រចកសម្រាប់ដាក់ក្រុមកន្សោមរច សញ្ញា ( )

វង់ក្រចកត្រូវបានប្រើដើម្បីដាក់ជាក្រុមពាក្យនៅក្នុងកន្សោម C# ក្នុងលក្ខណៈដូចគ្នានឹងពិជគណិតកន្សោម។ ឧទាហរណ៍ ដើម្បីគុណនឹងបរិមាណ b+c យើងសរសេរ៖

```
a*(b+c)
```

១៤. ការសរសេរកន្សោម ពិជគណិត និងកន្សោម C#

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

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

ឥឡូវនេះសូមពិចារណាកន្សោមមួយចំនួនដោយផ្អែកលើច្បាប់នៃប្រតិបត្តិករអាទិភាព។
ឧទាហរណ៍៖

Algebra: $y = mx + b$

C#:
```
y = m * x + b;
```

ការសរសេរកន្សោមពិជគណិត និងសមមូល C# របស់វា។ ខាងក្រោមនេះគឺជាឧទាហរណ៍មួយ។ មធ្យមនព្វន្ត (មធ្យម) នៃប្រាំពាក្យ៖

Algebra: $z = pr\%q + w/x - y$

C++:
```
z = p * r % q + w / x - y;
```
6 1 2 4 3 5

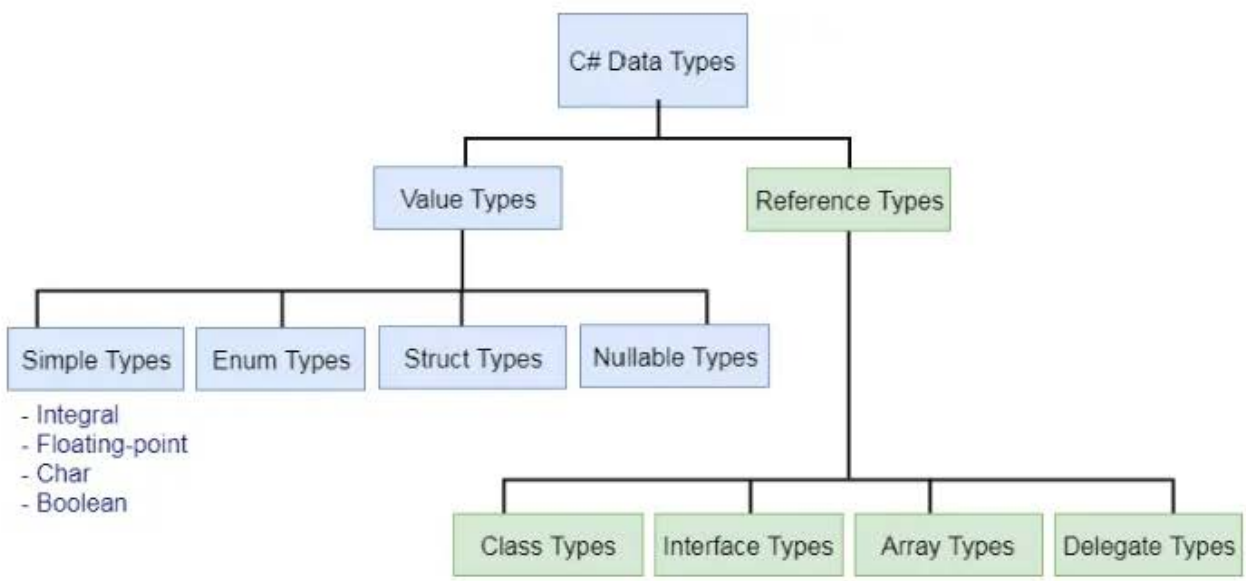
ជំពូកទី ៣

សិក្ខាប្រភេទទិន្នន័យ DataType C#

C# គឺជាភាសាវាយអក្សរstrongly-typed language មានន័យថាយើងត្រូវប្រកាសប្រភេទនៃ អថេរ ដែលបង្ហាញពីប្រភេទនៃតម្លៃដែលវានឹងរក្សាទុកដូចជា ចំនួនគត់ ទសភាគ អត្ថបទ ។ល។
ខាងក្រោមនេះប្រកាស និងចាប់ផ្តើមអថេរ នៃប្រភេទទិន្នន័យផ្សេងៗគ្នា។

Example: Variables of Different Data Types

```
string stringVar = "Hello World!!";
int intVar = 100;
float floatVar = 10.2f;
char charVar = 'A';
bool boolVar = true;
```



១. ប្រភេទទិន្នន័យដែលបានកំណត់ជាមុនរៀវភ្ជាប់ C# Predefined Data Types in C#

C# រួមបញ្ចូលប្រភេទតម្លៃដែលបានកំណត់ជាមុន value types និង reference types ប្រភេទឯកសារយោងមួយចំនួន។ តារាងខាងក្រោមរាយប្រភេទទិន្នន័យដែលបានកំណត់ជាមុន៖

Type	Description	Range	
byte	8-bit unsigned integer	0 to 255	
sbyte	8-bit signed integer	-128 to 127	

Type	Description	Range	
short	16-bit signed integer	-32,768 to 32,767	
ushort	16-bit unsigned integer	0 to 65,535	
int	32-bit signed integer	-2,147,483,648 to 2,147,483,647	
uint	32-bit unsigned integer	0 to 4,294,967,295	u
long	64-bit signed integer	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	i
ulong	64-bit unsigned integer	0 to 18,446,744,073,709,551,615	ui
float	32-bit Single-precision floating point type	-3.402823e38 to 3.402823e38	f
double	64-bit double-precision floating point type	-1.79769313486232e308 to 1.79769313486232e308	d
decimal	128-bit decimal type for financial and monetary calculations	(+ or -)1.0 x 10e-28 to 7.9 x 10e28	m
char	16-bit single Unicode character	Any valid character, e.g. a,*, \x0058 (hex), or\u0058 (Unicode)	
bool	8-bit logical true/false value	True or False	
object	Base type of all other types.		
string	A sequence of Unicode characters		
DateTime	Represents date and time	0:00:00am 1/1/01 to 11:59:59pm 12/31/9999	

ដូចដែលអ្នកអាចឃើញនៅក្នុងតារាងខាងលើដែលប្រភេទទិន្នន័យនីមួយៗ (លើកលែងតែប្រភេទតួអក្សរ និងវត្ថុ string and object) រួមបញ្ចូលជួរតម្លៃ។ compiler នឹងផ្តល់កំហុស ប្រសិនបើ តម្លៃចេញពីជួរដែលអនុញ្ញាតរបស់ប្រភេទទិន្នន័យ។ ឧទាហរណ៍ ជួរ range នៃប្រភេទទិន្នន័យ int គឺ -2,147,483,648 ដល់ 2,147,483,647។ ដូច្នេះប្រសិនបើអ្នកកំណត់តម្លៃដែលមិនមាននៅក្នុងជួរនេះ នោះ compiler នឹងផ្តល់កំហុស។

Example: Compile time error

```
// compile time error: Cannot implicitly convert type 'long' to 'int'.
int i = 21474836470;
```

តម្លៃ value មិនផ្តល់ឱ្យ integers, long, float, double, and decimal type ដែលមានក្នុងបច្ច័យ Value Suffix u,l,f,d និង m ។

Example: Value Suffix

```
uint ui = 100u;
float fl = 10.2f;
long l = 457554522222221;
ulong ul = 45755452222222ul;
double d = 11452222.555d;
decimal mon = 1000.15m;
```

ប្រភេទទិន្នន័យដែលបានកំណត់ជាមុនគឺជាឈ្មោះនៃប្រភេទ .NET (ថ្នាក់ CLR) របស់ពួកគេ។

តារាងខាងក្រោមរាយឈ្មោះជំនួសសម្រាប់ប្រភេទទិន្នន័យដែលបានកំណត់ជាមុន និងឈ្មោះថ្នាក់ .NET ដែលពាក់ព័ន្ធ។

Data Type	.NET Type	Type
byte	System.Byte	struct
sbyte	System.SByte	struct
int	System.Int32	struct
uint	System.UInt32	struct
short	System.Int16	struct
ushort	System.UInt16	struct
long	System.Int64	struct
ulong	System.UInt64	struct
float	System.Single	struct
double	System.Double	struct
char	System.Char	struct
bool	System.Boolean	struct
object	System.Object	Class
string	System.String	Class
decimal	System.Decimal	struct
DateTime	System.DateTime	struct

វាមានន័យថាមិនថាអ្នកកំណត់អថេរនៃ int ឬ Int32 ទេ ទាំងពីរគឺដូចគ្នា។

```
int i = 345;
Int32 i = 345;// same as above
```

២ លំនាំដើមតម្លៃ Default Values

គ្រប់ប្រភេទទិន្នន័យនីមួយៗ Data type មានតម្លៃលំនាំដើម default value ប្រភេទលេខគឺ 0, boolean មាន false, true ហើយ char មាន '\0' ជាតម្លៃលំនាំដើម។ ប្រើ default(typename) ដើម្បីផ្តល់តម្លៃលំនាំដើមនៃប្រភេទទិន្នន័យ ឬ C# 7.1 តទៅ សូមប្រើព្យញ្ជនៈលំនាំដើម defaul literal ។

```
int i = default(int); // 0
float f = default(float);// 0
decimal d = default(decimal);// 0
bool b = default(bool);// false
char c = default(char);// '\0'
```

```
// C# 7.1 onwards
int i = default; // 0
float f = default;// 0
decimal d = default;// 0
bool b = default;// false
char c = default;// '\0'
```

៣.បំប្លែង Conversions

តម្លៃនៃប្រភេទទិន្នន័យមួយចំនួនត្រូវបានបំប្លែងដោយស្វ័យប្រវត្តិទៅជាប្រភេទទិន្នន័យផ្សេងៗគ្នានៅក្នុង C# ។

Example: Implicit Conversion

```
int i = 345;
float f = i;
Console.WriteLine(f); //output: 345
```

នៅក្នុងឧទាហរណ៍ខាងលើ value នាំអថេរចំនួនគត់ i integer variable i ត្រូវបានបម្លែងទៅជាប្រភេទចំនួនទសភាគ float f ពីព្រោះដោយសារតែប្រតិបត្តិការបំប្លែងនេះត្រូវបានកំណត់ជាមុននៅក្នុង C# ។

ខាងក្រោមនេះគឺជាតារាងបំប្លែងប្រភេទទិន្នន័យដែលកំណត់៖

Conversion From	To
sbyte	short, int, long, float, double, decimal
byte	short, ushort, int, uint, long, ulong, float, double, decimal
short	int, long, float, double, or decimal
ushort	int, uint, long, ulong, float, double, or decimal
int	long, float, double, or decimal.
uint	long, ulong, float, double, or decimal
long	float, double, or decimal
ulong	float, double, or decimal
char	ushort, int, uint, long, ulong, float, double, or decimal
float	Double

ការបំប្លែងពី int, uint, long, ឬ ulong ទៅ float និងពី long or ulong ទៅ double អាចបណ្តាលឱ្យបាត់បង់ភាពជាក់លាក់។ គ្មានប្រភេទទិន្នន័យដែលត្រូវបានបំប្លែងទៅជាប្រភេទ char ទេ។

ទោះយ៉ាងណាក៏ដោយមិនមែនប្រភេទទិន្នន័យទាំងអស់ត្រូវបានបំប្លែងដោយចេតនាទៅជាប្រភេទទិន្នន័យផ្សេងទៀតនោះទេ។ ឧទាហរណ៍ ប្រភេទ int មិនអាចបំប្លែងទៅជា uint បានទេ។ វាត្រូវតែបញ្ជាក់យ៉ាងច្បាស់ដូចបានបង្ហាញខាងក្រោម។

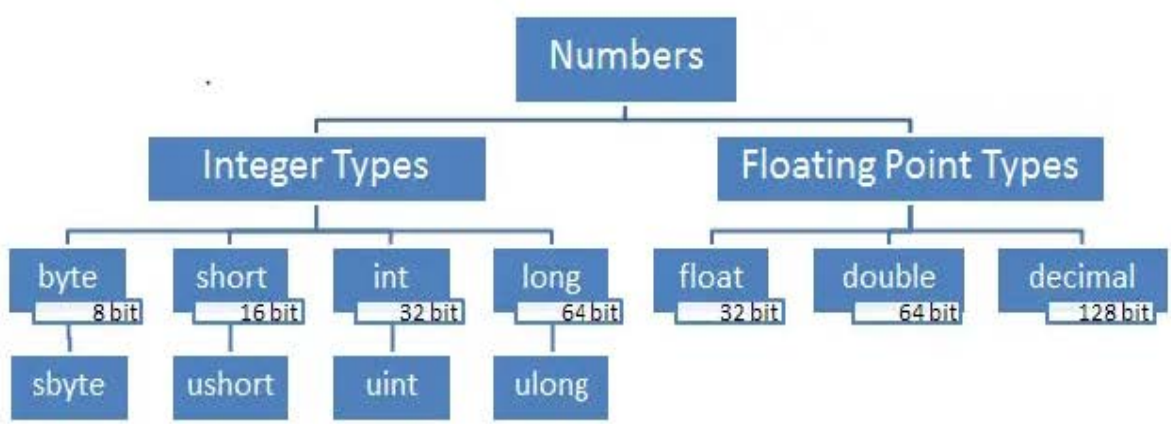
Example: Explicit Conversion

```
public static void Main()
{
    int i = 100;
    uint u = (uint) i;
    Console.Write(i);
}
```

៤. លេខ Numbers in C#

ជាទូទៅ លេខអាចចែកចេញជាពីរប្រភេទ៖ ប្រភេទចំនួនគត់ និងប្រភេទទសភាគ។ លេខប្រភេទចំនួនគត់ គឺជាលេខទាំងមូលដែលគ្មានខ្ទង់ទសភាគ។ វាអាចជាលេខអវិជ្ជមាន ឬវិជ្ជមាន។ ប្រភេទទសភាគ គឺជាលេខដែលមានចំនុចទសភាគមួយ ឬច្រើន។ វាអាចជាលេខអវិជ្ជមាន ឬវិជ្ជមាន។

C# រួមបញ្ចូលប្រភេទទិន្នន័យផ្សេងៗគ្នាសម្រាប់ប្រភេទចំនួនគត់ និងប្រភេទទសភាគ ដោយផ្អែកលើទំហំរបស់វានៅក្នុងអង្គចងចាំ និងសមត្ថភាពផ្ទុកលេខ។



Numeric Types

៤.១ ប្រភេទចំនួនគត់

លេខប្រភេទចំនួនគត់គឺជាលេខទាំងវិជ្ជមានឬអវិជ្ជមានដោយមិនមានខ្ទង់ទសភាគ។ C# រួមបញ្ចូលទិន្នន័យបួនប្រភេទសម្រាប់ចំនួនគត់៖ byte, short, int និង long ។

៤.១.១ Byte

បៃ byte data type store ចំនួនចាប់ពី 0 ដល់ 255 ។ វាតម្លៃធ្វើការ 8-bit នៅក្នុងអង្គចងចាំ memory ។

Example: byte, sbyte

```
byte b1 = 255;
byte b2 = -128;// compile-time error: Constant value '-128' cannot
be converted to a 'byte'
sbyte sb1 = -128;
sbyte sb2 = 127;
Console.WriteLine(Byte.MaxValue);//255
Console.WriteLine(Byte.MinValue);//0
Console.WriteLine(SByte.MaxValue);//127
Console.WriteLine(SByte.MinValue);//-128
```

៤.១.២ ប្រភេទទិន្នន័យខ្លី short data type

ប្រភេទទិន្នន័យខ្លី short data type គឺជាចំនួនគត់ដែលបានផ្តល់ឱ្យ ដែលអាចរក្សាទុកលេខពី -32,768 ដល់ 32,767 ។ វាត្រូវការរក្សាទុកក្នុងអង្គចងចាំ 16 ប៊ីត occupies 16-bit memory ។ ពាក្យគន្លឹះខ្លីៗគឺជាឈ្មោះសម្រាប់រចនាសម្ព័ន្ធ Int16 struct នៅក្នុង .NET។

The short data type is a signed integer that can store numbers from -32,768 to 32,767. It occupies 16-bit memory. The short keyword is an alias for Int16 struct in .NET

ប្រភេទទិន្នន័យ ushort គឺជាចំនួនគត់ដែលមិនមែនជាចំនួនគត់ unsigned integer ។ វាអាចរក្សាទុកបានតែលេខវិជ្ជមានពី 0 ដល់ 65,535 ពាក្យគន្លឹះខ្លីៗគឺជាប្រភេទទិន្នន័យសម្រាប់រចនាសម្ព័ន្ធ UInt16 នៅក្នុង .NET។

Example: short, ushort

```
short s1 = -32768;
short s2 = 32767;
short s3 = 35000;//Compile-time error: Constant value '35000'
cannot be converted to a 'short'
ushort us1 = 65535;
ushort us2 = -32000; //Compile-time error: Constant value '-32000'
cannot be converted to a 'ushort'
```

```
Console.WriteLine(Int16.MaxValue);//32767
Console.WriteLine(Int16.MinValue);//-32768
Console.WriteLine(UInt16.MaxValue);//65535
Console.WriteLine(UInt16.MinValue);//0
```

៤.១.៣ ប្រភេទចំនួនគត់វិជ្ជមាន Int

int data type គឺប្រភេទទិន្នន័យ 32-bit signed integer ។ វាអាចរក្សាទុកតម្លៃចំនួនគត់ពី -2,147,483,648 ដល់ 2,147,483,647 ។ int keyword គឺប្រភេទតម្លៃទិននន្នន័យ Int32 struct in .NET ។ uint គឺប្រភេទទិន្នន័យ 32-bit unsigned integer ។ uint keyword ឈ្មោះសម្គាល់ Uint32 struct ក្នុង.NET វាអាចរក្សាទុកតម្លៃចំនួនវិជ្ជមាន ពី 0 ដល់ 4,294,967,295 ។ ជាជម្រើសប្រើ U ឬ U បច្ច័យបន្ទាប់ពីលេខដើម្បីកំណត់វាទៅអថេរ uint ។

Example: int, uint

```
int i = -2147483648;
int j = 2147483647;
int k = 4294967295; //Compile-time error: Cannot implicitly
convert type 'uint' to 'int'.
uint ui1 = 4294967295;
uint ui2 =-1; //Compile-time error: Constant value '-1' cannot be
converted to a 'uint'
Console.WriteLine(Int32.MaxValue);//2147483647
Console.WriteLine(Int32.MinValue);//-2147483648
Console.WriteLine(UInt32.MaxValue);//4294967295
Console.WriteLine(UInt32.MinValue);//0
```

int data type គឺតែងតែត្រូវប្រើប្រាស់សម្រាប់ប្រព័ន្ធគោល ១៦ hexadecimal និង ប្រព័ន្ធគោលពីរ binary numbers ។ ប្រព័ន្ធគោល ១៦ hexadecimal ចាប់ផ្តើមពីចំនួន 0x ឬ 0X ដែលកំណត់ Prefix ក្នុង C# ជំនាន់ 7.0 ចំនួនគោលពីរ binary number ចាប់ផ្តើមពីចំនួន 0b ឬ 0B ។

Example: Hexadecimal, Binary

```
int hex = 0x2FAB;
int binary = 0b_0010_1111;
Console.WriteLine(hex);
Console.WriteLine(binary);
```

៤.១.៤ ប្រភេទចំនួនគត់វិជ្ជមាន Long

long type គឺជាប្រភេទចំនួនគត់ 64 bit ។ វាអាចរក្សាតម្លៃចំនួនចាប់ពី -9,223,372,036,854,775,808 ដល់ 9,223,372,036,854,775,807 ។ គេប្រើ l ឬ L សម្រាប់កំណត់តម្លៃ Prefix អថេរមួយចំនួនទៅជាប្រភេទ long type ។ គេកំណត់វាដោយ Int64 struct ។

ulong type គឺជាប្រភេទចំនួនគត់ ដែលមានតម្លៃវិជ្ជមាន ចាប់ពី 0 to 18,446,744,073,709,551,615 ។ ប្រសិនបើចំនួន suffixed UL, Ul, uL, ul, LU, Lu, lU, or lu វាគឺជាប្រភេទ ulong ។ ulong keyword គឺកំណត់ប្រកាសជា UInt64 struct in .NET ។

Example: long, ulong

```
long l1 = -9223372036854775808;
long l2 = 9223372036854775807;

ulong ul1 = 182233720368547758O8ul;
ulong ul2 = 182233720368547758O8UL;

Console.WriteLine(Int64.MaxValue);//9223372036854775807
Console.WriteLine(Int64.MinValue);//-9223372036854775808
Console.WriteLine(UInt64.MaxValue);//18446744073709551615
Console.WriteLine(UInt64.MinValue);//0
```

៤.១.៥ ប្រភេទចំនួនទសភាគ Floating Point Types

Floating-point គឺជាលេខវិជ្ជមាន ឬអវិជ្ជមានដែលមានចំនុចទសភាគមួយ ឬច្រើន។ C# រួមបញ្ចូលទិន្នន័យបីប្រភេទសម្រាប់លេខទសភាគ៖ float, double និង decimal។

Float data type អាចរក្សាតម្លៃជាចំនួនប្រភាគ ចាប់ពីចំនួន 3.4e−038 ដល់ 3.4e+038 ។ វារក្សាទុកជា 4 bytes ក្នុងអង្គចងចាំ ។ float keyword គឺ Single struct in .NET ។ ប្រើប្រាស់ f ឬ F suffix ជាមួយជាមួយព្យញ្ជនៈដើម្បីធ្វើឱ្យវាជាប្រភេទទសភាគ float type។

Example: float

```
float f1 = 123456.5F;
float f2 = 1.123456f;
Console.WriteLine(f1);//123456.5
Console.WriteLine(f2);//1.123456
```

៤.១.៦ ប្រភេទចំនួន Double

ប្រភេទទិន្នន័យទ្វេ double data type អាចផ្ទុកលេខប្រភាគពី 1.7e−308 ដល់ 1.7e+308 ។ វាធ្វើការ 8 បៃនៅក្នុងអង្គចងចាំ។ ពាក្យគន្លឹះទ្វេប្រកាសគឺ Double struct នៅក្នុង .NET។ ប្រើបច្ច័យ d ឬ D ជាមួយ literal ដើម្បីធ្វើឱ្យវាជាប្រភេទទ្វេdouble type ។

Example: double

```
double d1 = 12345678912345.5d;
double d2 = 1.123456789123456d;
Console.WriteLine(d1);//12345678912345.5
Console.WriteLine(d2);//1.123456789123456
```

៤.១.៧ ប្រភេទចំនួន Decimal

decimal data type អាចរក្សាទុកលេខប្រភាគចាប់ពី ±1.0 x 10-28 ដល់ ±7.9228 x 1028។ វាប្រកាសជា (16 បៃ) 16 bytes in the memory ក្នុងអង្គចងចាំ។ ទសភាគគឺជាពាក្យគន្លឹះនៃវចនាសម្ព័ន្ធ Decimal struct នៅក្នុង .NET ។

The decimal type has more precision and a smaller range than both float and double, and so it is appropriate for financial and monetary calculations.

Use m or M suffix with literal to make it decimal type.

Example: decimal

```
decimal d1 = 1234567891234567891234567789.5m;
decimal d2 = 1.1234567891345679123456789123m;
Console.WriteLine(d1);
Console.WriteLine(d2);
```

៤.១.៨ ប្រភេទចំនួន Scientific Notation

ប្រើ e ឬ E ដើម្បីបង្ហាញពីអំណាចនៃ 10 ជាផ្នែកនិទស្សន្តនៃសញ្ញាវិទ្យាសាស្ត្រជាមួយ float, double or decimal

Example:

```
double d = 0.12e2;
Console.WriteLine(d);  // 12;
float f = 123.45e-2f;
Console.WriteLine(f);  // 1.2345
decimal m = 1.2e6m;
```

```
Console.WriteLine(m);// 1200000
```

៥. ប្រភេទអក្សរ Strings

នៅក្នុង C# ខ្សែអក្សរគឺជាស៊េរីនៃតួអក្សរដែលត្រូវបានប្រើដើម្បីតំណាងឱ្យអត្ថបទ។ វាអាចជាតួអក្សរ ពាក្យឬវគ្គរវែងព័ទ្ធជុំវិញដោយសញ្ញា double quotes " "។ ខាងក្រោមនេះជាឧទាហរណ៍តួអក្សរ។

Example: String Literals

```
"S"
"String"
"This is a string."
```

C# បានផ្តល់នូវ String data type ដើម្បីរក្សាទុក store string អត្ថបទជាតួអក្សរ ។ អថេរនៃ string type អាចប្រកាស និងផ្តល់តម្លៃឱ្យ declared and assign string ដូចបង្ហាញដូចខាងក្រោម៖

Example: String Type Variables

```
string ch = "S";
string word = "String";
string text = "This is a string.";
```

ទំហំអតិបរមានៃ String object ក្នុងអង្គចងចាំគឺ 2GB ឬប្រហែល 1 ពាន់លានតួអក្សរ។ ទោះយ៉ាងណាក៏ដោយ ជាក់ស្តែងវានឹងតិចជាងអាស្រ័យលើ CPU និងអង្គចងចាំរបស់កុំព្យូទ័រ។

មានវិធីពីរយ៉ាងដើម្បីប្រកាសអថេរ string អក្សរនៅក្នុង C#។ ការប្រើប្រាស់ System.String class និងការប្រើ string keyword។ ទាំងពីរ គឺដូចគ្នា និងមិនមានភាពខុសគ្នា។ ស្វែងយល់អំពី string អក្សរ និង string អក្សរ សម្រាប់ព័ត៌មានបន្ថែម។

Example: String and string

```
string str1 = "Hello"; // uses string keyword
String str2 = "Hello"; // uses System.String class
```

ក្នុងភាសា C# string គឺជា collection ឬជា array នៃ characters ។ ដូចនេះ string អាចបង្កើតបានដោយប្រើប្រាស់ char array ឬ accessed ជា char array។

Example: String as char Array

```
char[] chars = {'H','e','l','l','o'};
string str1 = new string(chars);
```

```
String str2 = new String(chars);
foreach (char c in str1)
{
    Console.WriteLine(c);
}
```

៥.១ ប្រភេទអក្សរពិសេស Special Characters

អត្ថបទនៅក្នុង real world ពិតពពិតអាចរួមបញ្ចូលតួអក្សរណាមួយ។ នៅក្នុង C# ដោយសារ string អក្សរមួយត្រូវបានហ៊ុំព័ទ្ធដោយសញ្ញា("˝) double quotes " វាមិនអាចរួមបញ្ចូល " នៅក្នុង string អក្សរបានទេ។ ខាងក្រោមយើងមិនអាចសរសេរឬប្រកាសបានទេ compile-time error ៖

Example: Invalid String

```
string text = "This is a "string" in C#.";
```

ក្នុង C# គេបញ្ចូល escaping character \ (backslash) បន្ទាប់ពីតួអក្សរពិសេស special characters ដើម្បីម្បីបន្ថែមក្នុង string include in a string។

ការប្រើប្រាស់បេកស្លាស់ backslash \ បន្ទាប់ពីសញ្ញា " double quotes និងតួអក្សរ ឬសញ្ញាពិសេសណាមួយ special characters such as \,\n,\r,\t, ជាដើម ដើម្បី បញ្ចូលវាក្នុង string។

Example: Escape Char \

```
string text = "This is a \"string\" in C#.";
string str = "xyzdef\\rabc";
string path = "\\\\mypc\\ shared\\project";
```

៥.២ ប្រភេទ Verbatim Strings

វាធុញទ្រាន់នឹងបុព្វបទ \ ដើម្បីបញ្ចូលតួអក្សរពិសេសនីមួយៗ។ Verbatim string នៅក្នុង C# អនុញ្ញាតឱ្យមានតួអក្សរពិសេស និង line brakes ។ Verbatim string អាចត្រូវបានបង្កើតដោយការដាក់បុព្វបទ @ និមិត្តសញ្ញា មុននឹងសញ្ញាសម្រង់ទ្វេ double quotes ។

Example: Escape Sequence

```
string str = @"xyzdef\rabc";
string path = @"\\mypc\shared\project";
string email = @"test@test.com";
```

និមិត្តសញ្ញា @ ក៏អាចត្រូវបានប្រើដើម្បី multi-line string ប្រកាសអក្សរច្រើនជួរផងដែរ។

Example: Multi-line String

```
string str1 = "this is a \n" +
       "multi line \n" +
       "string";
```

```
// Verbatim string
string str2 = @"this is a
       multi line
       string";
```

សូមចំណាំថាអ្នកមិនអាចប្រើសញ្ញាបញ្ច្រាស backslash ដើម្បីអនុញ្ញាត " នៅក្នុង verbatim string ប្រសិនបើអ្នកចង់បញ្ចូល @នោះសូមប្រើសញ្ញាសម្រង់ទ្វេពីរ "" ដើម្បីរួមបញ្ចូល "នៅក្នុង verbatim string ។

```
string text = "This is a \"string\" in C#."; // valid
string text = @"This is a "string." in C#."; // error
string text = @"This is a \"string\" in C#."; // error
string text = @"This is a ""string"" in C#."; // valid
```

៥.៣ ប្រភេទ String Concatenation

តួអក្សរច្រើនយើងអាច concatenated បញ្ចូលគ្នាបានជាមួយសញ្ញ (+) ដើម្បីតភ្ជាប់គ្នា Multiple strings can be concatenated with + operator.

Example: String Concatenation

```
string name = "Mr." + "James " + "Bond" + ", Code: 007";
string firstName = "James";
string lastName = "Bond";
string code = "007";
string agent = "Mr." + firstName + " " + lastName + ", Code: " +
code;
```

៥.៤ ប្រភេទ String Interpolation

ការបញ្ចូលខ្សែអក្សរគឺជាវិធីល្អប្រសើរនៃការភ្ជាប់ខ្សែអក្សរ។ យើងប្រើសញ្ញា + ដើម្បី concatenate string variables តភ្ជាប់អថេរអក្សរជាមួយ static strings ។

C# 6 រួមបញ្ចូលតួអក្សរពិសេស $ ដើម្បីកំណត់អត្តសញ្ញាណខ្សែអក្សរដែលទាក់ទងគ្នា។ interpolated string គឺជាល្បាយបញ្ចូលគ្នានៃ static string និង string variable ដែលអថេរ string គួរតែស្ថិតនៅក្នុងតង្កៀប {} ។

Example: String Interpolation

```
string firstName = "James";
string lastName = "Bond";
string code = "007";
```

```
string fullName = $"Mr. {firstName} {lastName}, Code: {code}";
```

នៅក្នុងឧទាហរណ៍ខាងលើនៃ interpolation , $ indicates the interpolated string និង {} includes string variable ដើម្បី incorporated ជាមួយ string ។

យើងប្រើប្រាស់ ពីរ braces, "{{" or "}}" ដើម្បីបញ្ចូល { or } ក្នុង string ។

ខ ប្រភេទ Working with Date and Time in C#

នៅក្នុង C# DataTime struct ត្រូវបានបញ្ចូលដើម្បីធ្វើការជាមួយ dates កាលបរិច្ឆេទ ពេលវេលា និង times ម៉ោង ។ ដើម្បីធ្វើការជាមួយ Date និង Time នៅក្នុង C# គេត្រូវបង្កើត Object នៃ DateTime struct ដោយប្រើប្រាស់នូវ new Keyword ។ ដើម្បីបង្កើត DateTime Object ជាមួយនឹង Defaul value ដូចខាងក្រោម៖

Example: Create DateTime Object

```
DateTime dt = new DateTime(); // assigns default value 01/01/0001
00:00:00
```

defaul និង lowest value នៃ DateTime object គឺ January 1, 0001 00:00:00 (midnight)។ maximum value can be December 31, 9999 11:59:59 P.M ។ ការប្រើប្រាស់ different constructors របស់ DateTime struct ដើម្បីធ្វើការផ្តល់តម្លៃ assign an initial value ទៅឱ្យ DateTime object ។

Example: Set Date & Time

```
//assigns default value 01/01/0001 00:00:00
DateTime dt1 = new DateTime();

//assigns year, month, day
DateTime dt2 = new DateTime(2015, 12, 31);

//assigns year, month, day, hour, min, seconds
DateTime dt3 = new DateTime(2015, 12, 31, 5, 10, 20);

//assigns year, month, day, hour, min, seconds, UTC timezone
DateTime dt4 = new DateTime(2015, 12, 31, 5, 10, 20,
DateTimeKind.Utc);
```

ក្នុងឧទាហរណ៍ខាងលើ យើងបានបញ្ជាក់ ឆ្នាំ ខែ និងថ្ងៃ ទី1 ដល់ 31 នៅក្នុង constructor

អ្នកសាងសង់។ ឆ្នាំអាចចាប់ពី 0001 ដល់ 9999 ហើយខែអាចចាប់ពី 1 ដល់ 12 ហើយថ្ងៃអាចចាប់ពី 1 ដល់ 31។ ការកំណត់តម្លៃផ្សេងទៀតចេញតាមលំដាប់លំដោយ ទាំងនេះនឹង result នៅក្នុងពេលពេលដំណើរការ។

Example: Invalid Date

```
DateTime dt = new DateTime(2015, 12, 32); //throws exception: day out of range
```

ការប្រើប្រាស់dateTime constructor ដើម្បីដំឡើង date, time, time zone, calendar និង culture ។

៦.១ ប្រភេទ Ticks

Ticks គឺជាកាលបរិច្ឆេទ និងពេលវេលាដែលបង្ហាញក្នុងចំនួនចន្លោះពេល 100-nanosecond ដែលបានកន្លងផុតទៅចាប់តាំងពីថ្ងៃទី 1 ខែមករា ឆ្នាំ 0001 នៅម៉ោង 00:00:00.000 នៅក្នុងប្រតិទិន ។ ខាងក្រោមចាប់ផ្តើមវត្ត DateTime ជាមួយនឹងចំនួន ticks ៖

Example: Ticks

```
DateTime dt = new DateTime(636370000000000000);
DateTime.MinValue.Ticks;  //min value of ticks
DateTime.MaxValue.Ticks; // max value of ticks
```

៦.២ ប្រភេទ DateTime Static Fields

DateTime struct បញ្ចូលបន្ថែមនូវ static fields, properties និង methods ។ ឧទាហរណ៍ខាងក្រោមបង្ហាញពីលក្ខណៈសម្បត្តិសំខាន់ៗ static fields និង properties ៖

Example: Static Fields

```
DateTime currentDateTime = DateTime.Now;  //returns current date and time
DateTime todaysDate = DateTime.Today; // returns today's date
DateTime currentDateTimeUTC = DateTime.UtcNow;// returns current UTC date and time
DateTime maxDateTimeValue = DateTime.MaxValue; // returns max value of DateTime
DateTime minDateTimeValue = DateTime.MinValue; // returns min value of DateTime
```

៦.៣ ប្រភេទ TimeSpan

TimeSpan គឺជារចនាសម្ព័ន្ធ struct ដែលប្រើប្រាស់តំណាង Time នៅក្នុងថ្ងៃ ម៉ោង នាទី វិនាទី និងមីលីវិនាទី (days, hour, minutes, seconds, and milliseconds) ។

Example: TimeSpan

```
DateTime dt = new DateTime(2015, 12, 31);
```

```
TimeSpan ts = new TimeSpan(25,20,55);
DateTime newDate = dt.Add(ts);
Console.WriteLine(newDate);//1/1/2016 1:20:55 AM
```

៦.៤ ប្រភេទ Subtraction of two dates results in TimeSpan.

Example: Subtract Dates

```
DateTime dt1 = new DateTime(2015, 12, 31);
DateTime dt2 = new DateTime(2016, 2, 2);
TimeSpan result = dt2.Subtract(dt1);//33.00:00:00
```

៦.៥ ប្រភេទ Operators

DateTime struct overloads +, -, ==, !=, >, <, <=, >= operators គេប្រើដើម្បី បន្ថែម ដក និងការប្រៀបធៀប នៃ ពេលវេលា (addition, subtraction, and comparison of dates) ។ ទាំងនេះជាការបង្កើតការងារងាយៗជាមួយ Dates ៖

Example: Operators

```
DateTime dt1 = new DateTime(2015, 12, 20);
DateTime dt2 = new DateTime(2016, 12, 31, 5, 10, 20);
TimeSpan time = new TimeSpan(10, 5, 25, 50);

Console.WriteLine(dt2 + time); // 1/10/2017 10:36:10 AM
Console.WriteLine(dt2 - dt1); //377.05:10:20
Console.WriteLine(dt1 == dt2); //False
Console.WriteLine(dt1 != dt2); //True
Console.WriteLine(dt1 > dt2); //False
Console.WriteLine(dt1 < dt2); //True
Console.WriteLine(dt1 >= dt2); //False
Console.WriteLine(dt1 <= dt2);//True
```

៦.៦ ប្រភេទ Convert String to DateTime

កាលបរិច្ឆេទ និងពេលវេលា ជាអក្សរ ត្រឹមត្រូវអាចត្រូវបានបំប្លែងទៅជា DateTime object ដោយប្រើ Parse(), ParseExact(), TryParse() និង TryParseExact() ។ វិធីសាស្ត្រ methods Parse() និង ParseExact() ។ Parse() និង ParseExact() methods នឹងបោះតម្លៃឱ្យ exception ប្រសិនបើ specified string តំណាងមិនបានត្រឹមត្រូវ នៃ Date និង

Time ។ ដូច្នេះ វាត្រូវបានណែនាំឱ្យប្រើវិធីសាស្ត្រ recommended ដើម្បីប្រើ TryParse() ឬ TryParseExact() method ពីព្រោះបោះតម្លៃមកវិញ មិនត្រឹមត្រូវ return false បើសិន string គឺមិនត្រឹមត្រូវ ។

Example:

```
var str = "5/12/2020";
DateTime dt;

var isValidDate = DateTime.TryParse(str, out dt);

if(isValidDate)
    Console.WriteLine(dt);
else
    Console.WriteLine($"{str} is not a valid date string");
```

៧. Struct

ក្នុង C#, struct គឺជាប្រភេទតម្លៃ value type របស់ data type ដែលបង្ហាញពីរចនាសម្ព័ន្ធទន្នន័យ represents data structures ។ វាអាចជា contain parameterized constructor, static constructor, constants, fields, methods, properties, indexers, operators, events និង nested types ។

struct អាចត្រូវបានប្រើដើម្បីក្បាតម្លៃទិន្នន័យតូចដែលមិនតម្រូវឱ្យមាន inheritance ចំណុចសម្រាប្រមួល key-value pairs និង complex data structure។

៧.១. Structure Declaration

រចនាសម្ព័ន្ធត្រូវបានប្រកាសដោយប្រើពាក្យគន្លឹះ struct ។ modifier កែប្រែលំនាំដើមគឺខាងក្នុងសម្រាប់រចនាសម្ព័ន្ធ struct និង members សមាជិករបស់វា។

ខាងក្រោមនេះឧទហរណ៍ ប្រកាស structure Coordinate សម្រាប់ក្រាប៖

Example: Structure

```
struct Coordinate
{
    public int x;
    public int y;
}
```

`struct` object អាចបង្កើតដោយមាន ឬគ្មាន `new` operator ដូចគ្នានឹងអថេរប្រភេទ primitive type variables ។

Example: Create Structure

```
struct Coordinate
{
    public int x;
    public int y;
}
Coordinate point = new Coordinate();
Console.WriteLine(point.x); //output: 0
Console.WriteLine(point.y); //output: 0
```

Example: Create Structure Without new Keyword

```
struct Coordinate
{
    public int x;
    public int y;
}

Coordinate point;
Console.Write(point.x); // Compile time error

point.x = 10;
point.y = 20;
Console.Write(point.x); //output: 10
Console.Write(point.y); //output: 20
```

៧.២ Constructors in Structure

Struct មិនមែន contain parameterless constructor ទេ ។ វាអាចជា contain តែមួយ parameterized constructors ឬ static constructor ។ `struct` cannot contain a parameterless constructor. It can only contain parameterized constructors or a static constructor.

Example: Parameterized Constructor in Struct

```
struct Coordinate
{
    public int x;
    public int y;

    public Coordinate(int x, int y)
```

```
    {
        this.x = x;
        this.y = y;
    }
}
Coordinate point = new Coordinate(10, 20);

Console.WriteLine(point.x); //output: 10
Console.WriteLine(point.y); //output: 20
```

៧.៣ Methods and Properties in Structure

struct អាចជា contain properties, auto-implemented properties, methods របស់ classes ដូចគ្នា ។

Example: Methods and Properties in Struct

```
struct Coordinate
{
    public int x { get; set; }
    public int y { get; set; }

    public void SetOrigin()
    {
        this.x = 0;
        this.y = 0;
    }
}

Coordinate point = Coordinate();
point.SetOrigin();

Console.WriteLine(point.x); //output: 0
Console.WriteLine(point.y); //output: 0
```

រចនាសម្ព័ន្ធ struct ខាងក្រោមរួមបញ្ចូល static method ៖

Example: Static Constructor in Struct

```
struct Coordinate
{
    public int x;
    public int y;
    public Coordinate(int x, int y)
    {
        this.x = x;
        this.y = y;
    }

    public static Coordinate GetOrigin()
    {
        return new Coordinate();
    }
}
Coordinate point = Coordinate.GetOrigin();
Console.WriteLine(point.x); //output: 0
Console.WriteLine(point.y); //output: 0
```

៧.៤ Events in Structure

struct អាចមាន contain events ដើម្បីជូនដំណឹងអំពីសកម្មភាពមួយចំនួន។ ខាងក្រោមមាន `struct` contains event ព្រឹត្តិការណ៍មួយ៖

Example: Event in Structure

```
struct Coordinate
{
    private int _x, _y;

    public int x
    {
        get{
            return _x;
        }
        set{
            _x = value;
```

```
            CoordinatesChanged(_x);
        }
    }
    public int y
    {
        get{
            return _y;
        }

        set{
            _y = value;
            CoordinatesChanged(_y);
        }
    }

    public event Action<int> CoordinatesChanged;
}
```

ចេនាសម្ព័ន្ធខាងលើមានព្រឹត្តិការណ៍ structur contains CoordinatesChanged event ដែលនឹងត្រូវបានលើកឡើងនៅពេលដែល x ឬ y coordinate ផ្លាស់ប្តូរ។

ឧទាហរណ៍ខាងក្រោមបង្ហាញពីការដោះស្រាយ CoordinatesChanged event ៖

Example: Handle Structure Events

```
class Program
{
    static void Main(string[] args)
    {

        Coordinate point = new Coordinate();

        point.CoordinatesChanged += StructEventHandler;
        point.x = 10;
        point.y = 20;
    }

    static void StructEventHandler(int point)
    {
        Console.WriteLine("Coordinate changed to {0}", point);
    }
}
```

struct គឺជាប្រភេទតម្លៃ value type, ដូច្នេះវាលឿនជាង Class object ។ ប្រើ struct នៅពេលណាដែលអ្នកចង់ក្បាទុកទិន្នន័យ។ ជាទូទៅ structs រចនាសម្ព័ន្ធ គឺល្អសម្រាប់ការសរសេរកម្មវិធីហ្គេម ។ ទោះយ៉ាងណាក៏ដោយ វាងាយស្រួលក្នុងការផ្ទេរ class object ជាងរចនាសម្ព័ន្ធ។ ដូច្នេះកុំប្រើ struct នៅពេលអ្នកកំពុងបញ្ជូនទិន្នន័យឆ្លងកាត់ខ្សែ ឬទៅ class ផ្សេងទៀត។

៧.៥ Summary

•struct អាចបញ្ចូលបន្ថែម constructors, constants, fields, methods, properties, indexers, operators, events & nested types.

•struct មិនអាចបញ្ចូល parameterless constructor or a destructor.

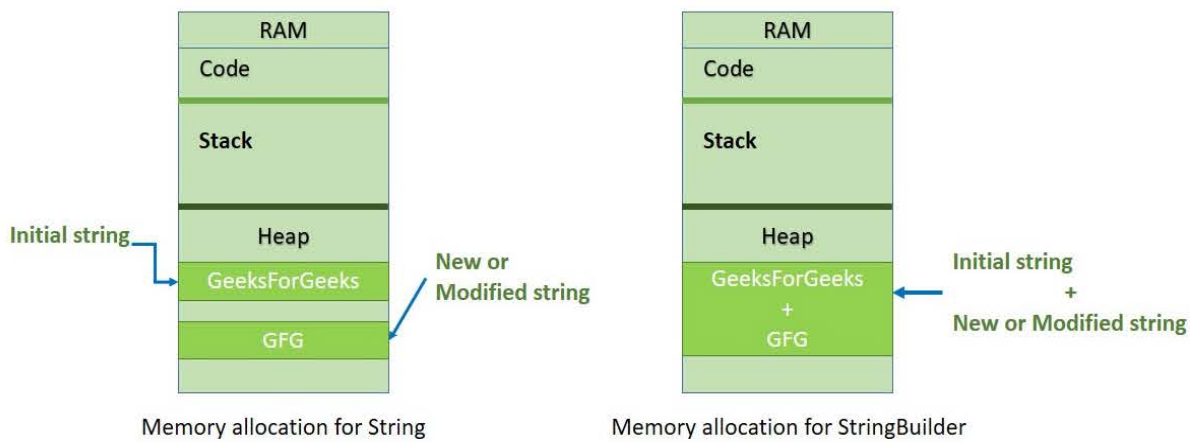
•struct អាច implement interfaces, same as class.

•struct cannot inherit another structure or class, and it cannot be the base of a class.

•struct members cannot be specified as abstract, sealed, virtual, or protected.

៨. StringBuilder

នៅក្នុង C# string type គឺមិនអាចផ្លាស់ប្តូរបាន។ វាមានន័យថាអក្សរមិនអាចត្រូវបានផ្លាស់ប្តូរនៅពេលដែលបានបង្កើត។ ឧទាហរណ៍ new string "ជំរាបសួរពិភពលោក!" នឹងកាន់កាប់កន្លែងចងចាំនៅលើគំនរ។ ឥឡូវនេះ ដោយផ្លាស់ប្តូរអក្សរដំបូង "Hello World!" ទៅ "Hello World! from Tutorials Teacher" នឹងបង្កើតវ new string object អក្សរថ្មីនៅលើគំនរអង្គចងចាំដំនួសឱ្យការកែប្រែអក្សរដើមនៅអាសយដ្ឋានអង្គចងចាំដូចគ្នា។



Memory allocation for String Memory allocation for StringBuilder

៨.១ Creating a StringBuilder Object

យើងអាចបង្កើតនៃ StringBuilder class ប្រើប្រាស់ new keywprd និង passing អក្សរចាប់ផ្តើមដំបូង initial string ។ ខាងក្រោមនេះបង្ហាញពីការបង្កើត stringBuilder objrct ៖

Example: StringBuilder

```
using System.Text; // include at the top
StringBuilder sb = new StringBuilder(); //string will be appended
later
//or
StringBuilder sb = new StringBuilder("Hello World!");
```

ជាជម្រើសអ្នកក៏អាចបញ្ជាក់សមត្ថភាពអតិបរិមានៃ object StringBuilder ដោយប្រើឧបករណ៍ overloaded constructors ដូចដែលបានបង្ហាញខាងក្រោម។

Example: StringBuilder

```
StringBuilder sb = new StringBuilder(50); //string will be appended later
//or
StringBuilder sb = new StringBuilder("Hello World!", 50);
```

ខាងលើ C# បែងចែកចន្លោះអតិបរមាចំនួន 50 តាមលំដាប់លំដោយនៅលើអង្គចងចាំ។ សមត្ថភាពនេះនឹងត្រូវបានកើនឡើងទ្វេដងដោយស្វ័យប្រវត្តិនៅពេលដែលវាឈានដល់សមត្ថភាពដែលបានបញ្ជាក់ specified capacity ។ អ្នកក៏អាចប្រើសមត្ថភាព `capacity` ឬ `length` property ប្រវែង ទ្រព្យសម្បត្តិ ដើម្បីកំណត់ ឬទាញយកសមត្ថភាពរបស់ object StringBuilder capacity ។ អ្នកអាចធ្វើការប្រើសម្រាប់រង្វិលជុំ for loop ដើម្បី get ឬ set character ទទួល កំណត់តួអក្សរនៅ specified index សន្ទស្សន៍ ដែលបានបញ្ជាក់។

Example: StringBuilder Iteration

```
1  using System;
2  using System.Text;
3
4  public class Program
5  {
6      public static void Main()
7      {
8          StringBuilder sb = new StringBuilder("Hello World!!");
9
10         for(int i=0; i< sb.Length; i++)
11         Console.Write(sb[i]);
12     }
13 }
```

៨.៣ Retrieve String from StringBuilder

StringBuilder គឺមិនមែនជា String ទេ ។ ប្រើប្រាស់ `ToString()` method ដើម្បីយក string មកពី StringBuilder objects ។

Example: Retrieve String from StringBuilder

```
StringBuilder sb = new StringBuilder("Hello World!");
var greet = sb.ToString(); //returns "Hello World!"
```

៨.៤ Add/Append String to StringBuilder

ប្រើAppend() Method ដើម្បីបន្ថែមអក្សរនៅចុងបញ្ចប់នៃcurrent StringBuilder object បច្ចុប្បន្ន។ ប្រសិនបើ StringBuilder មិនទាន់មានខ្សែអក្សរណាមួយនៅឡើយទេ វានឹងបន្ថែមវា។ វិធីសាស្ត្រ AppendLine() method បន្ថែមអក្សរដែលមានតួអក្សរ បន្ទាត់ថ្មីនៅចុងបញ្ចប់ ។

Example: Adding or Appending Strings in StringBuilder

```
1  using System;
2  using System.Text;
3
4  public class Program
5  {
6          public static void Main()
7          {
8             StringBuilder sb = new StringBuilder("Hello",50);
9
10             sb.Append("World!!");
11             sb.AppendLine("Hello C#!");
12             sb.AppendLine("This is new line.");
13
14             Console.WriteLine(sb);
15         }
16 }
```

//ouput

```
Hello World!!Hello C#!
This is new line.
```

៨.៥ Append Formated String to StringBuilder

ប្រើ AppendFormat() Method ដើម្បីធ្វើទ្រង់ទ្រាយ Format បញ្ចូល input string into អក្សរទៅក្នុងទម្រង់ដែលបាន specified format and append បញ្ចាក់ ហើយបញ្ចូលវាបន្ថែម។ Use the AppendFormat() method to format an input string into the specified format and append it.

Example: AppendFormat()

```
using System;
using System.Text;

public class Program
{
   public static void Main()
   {
    StringBuilder amountMsg = new StringBuilder("Your total amount is ");

     amountMsg.AppendFormat("{0:C} ", 25);

     Console.WriteLine(amountMsg);
   }
}
```

៨.៦ Insert String into StringBuilder

គេប្រើប្រាស់ Insert() method ដើម្បី insert srting ឱ្យ Index តាមការកំណត់ នៅក្នុង StringBuilder object ។

```
using System;
using System.Text;

public class Program
{
   public static void Main()
   {
       StringBuilder sb = new StringBuilder("Hello World!!",50);
       sb.Insert(5," C#");

       Console.WriteLine(sb);
   }
}
//output
Hello C# World!!
```

៨.៦ Remove String in StringBuilder

គេប្រើប្រាស់ Remove() Method ដើម្បីយកចេញនូវអក្សរ remove string ពី index និងរហូតដល់ប្រវែង length ជាក់លាក់ តាមរយៈការកំណត់ដោយ index ។

```
using System;
using System.Text;

public class Program
{
   public static void Main()
   {
       StringBuilder sb = new StringBuilder("Hello World!!",50);
      sb.Remove(6, 7);

      Console.WriteLine(sb);
   }
```

```
13 }
14 //output
15 Hello
```

៨.៨ Replace String in StringBuilder

គេប្រើប្រាស់ Replace () method ដើម្បី ជំនួសទាំងអស់នៃអក្សរដែលបានកំណត់ជាមួយនឹងការជំនួស string ។ Use the Replace() method to replace all the specified string occurrences with the specified replacement string.

```
1  using System;
2  using System.Text;
3
4  public class Program
5  {
6     public static void Main()
7     {
8         StringBuilder sb = new StringBuilder("Hello World!!",50);
9         sb.Replace("World", "C#");
10
11        Console.WriteLine(sb);
12    }
13 }
14 //output
15 Hello C#!!
```

Points to Remember :

1. StringBuilder is mutable.
2. StringBuilder performs faster than string when appending multiple string values.
3. Use StringBuilder when you need to append more than three or four strings.
4. Use the Append( ) method to add or append strings to the StringBuilder object.
5. Use the ToString( ) method to retrieve a string from the StringBuilder object.

៩ ប្រភេទ Anonymous Type

នៅក្នុង C# anonymous type គឺជាប្រភេទ Class (ថ្នាក់) ដោយគ្មានឈ្មោះណាមួយ ដែលអាចមានលក្ខណៈសម្បត្តិបានតែអាន contain public read-only properties ជាសាធារណៈប៉ុណ្ណោះ។ វាមិនអាចមានសមាជិកផ្សេងទៀត ដូចជា fields, methods, events ជាដើម។

អ្នកបង្កើតប្រភេទ nonymoustype ដោយប្រើ new operator ប្រតិបត្តិការថ្មីជាមួយនឹងប្រភេទ object initializer។ implicitly typed variable-var គឺត្រូវប្រើដើម្បីប្រើដើម្បីរក្សាឯកសារយោងនៃប្រភេទ anonymous types។

ឧទាហរណ៍ខាងក្រោមបង្ហាញពីការបង្កើត anonymous type មានឈ្មោះអថេរ student ដែលមានលក្ខណៈសម្បត្តិបីដែលមានឈ្មោះថា Id, FirstName និង LastName ។

Example: Anonymous Type

```
var student = new { Id = 1, FirstName = "James", LastName = "Bond" };
```

Example: Access Anonymous Type

```
1  using System;
2
3  public class Program
4  {
5     public static void Main()
6     {
7        var student = new {Id = 1, FirstName ="James", LastName = "Bond" };
8
9       Console.WriteLine(student.Id); //output: 1
10      Console.WriteLine(student.FirstName); //output: James
11      Console.WriteLine(student.LastName); //output: Bond
12    }
13 }
```

anonymous type គឺជា property ដែលអាចបញ្ចូល anonymous type ផ្សេងទៀតបាន ។

Example: Nested Anonymous Type

```
var student = new {
                    Id = 1,
                    FirstName = "James",
                    LastName = "Bond",
                    Address = new { Id = 1,City ="London",Country = "UK" }
                };
```

យើងអាចបង្កើត Array នៃ anonymous types

Example: Array of Anonymous Types

```
var students = new[] {
            new { Id = 1, FirstName = "James", LastName = "Bond" },
            new { Id = 2, FirstName = "Steve", LastName = "Jobs" },
            new { Id = 3, FirstName = "Bill", LastName = "Gates" }
    };
```

Example: LINQ Query returns an Anonymous Type

```
class Program
{
```

```
    static void Main(string[] args)
    {
        IList<Student> studentList = new List<Student>() {
        new Student() { StudentID = 1, StudentName = "John", age = 18 },
        new Student() { StudentID = 2, StudentName = "Steve", age = 21 },
        new Student() { StudentID = 3, StudentName = "Bill",  age = 18 },
        new Student() { StudentID = 4, StudentName = "Ram" , age = 20  },
        new Student() { StudentID = 5, StudentName = "Ron" , age = 21 }
     };

      var students = from s in studentList
                  select new { Id = s.StudentID, Name = s.StudentName };

        foreach(var stud in students)
         Console.WriteLine(stud.Id + "-" + stud.Name);
    }
}

public class Student
{
    public int StudentID { get; set; }
    public string StudentName { get; set; }
    public int age { get; set; }
}
```

Output:

```
1-John
2-Steve
3-Bill
4-Ram
5-Ron
```

10. Dynamic Types

C# 4.0 (.NET 4.5) បានណែនាំប្រភេទថ្មីដែលហៅថា dynamic ដែលជៀសវាងការត្រួតពិនិត្យប្រភេទពេលវេលា compile-time type។ dynamic type ប្រភេទ

លើកលែងចេញពីការត្រួតពិនិត្យប្រភេទ compile-time ។ ជំនួសមកវិញ វាដោះស្រាយប្រភេទនៅពេលដំណើរការ run time ។ អថេរ dynamic type ត្រូវបានកំណត់ដោយប្រើពាក្យគន្លឹះ dynamic ។

Example: dynamic Variable

```
dynamic MyDynamicVar = 1;
```

compiler compiles dynamic types ត្រូវបានបញ្ចូលក្នុង ប្រភេទ object ជាច្រើនករណី ។ ទោះយ៉ាងណាក៏ដោយ ប្រភេទពិតប្រាកដនៃអថេរប្រភេទ dynamic type នឹងត្រូវបានដោះស្រាយនៅពេលដំណើរការ run-time ។

Example: dynamic Type at run-time

```
dynamic MyDynamicVar = 1;
Console.WriteLine(MyDynamicVar.GetType());
```

Output:

```
System.Int32
```

ប្រភេទ Dynamic types ផ្លាស់ប្តូរប្រភេទនៅពេល run-time ដំណើរការដោយផ្អែកលើតម្លៃដែលបានកំណត់ assigned value ។ ឧទាហរណ៍ខាងក្រោមបង្ហាញពីរបៀបដែលប្រភេទការផ្លាស់ប្តូរអថេរDynamic types ដោយផ្អែកលើតម្លៃដែលបានកំណត់ assigned value ។

Example: dynamic

```
static void Main(string[] args)
{
    dynamic MyDynamicVar = 100;
    Console.WriteLine("Value: {0}, Type: {1}", MyDynamicVar,
                                                MyDynamicVar.GetType());

    MyDynamicVar = "Hello World!!";
    Console.WriteLine("Value: {0}, Type: {1}", MyDynamicVar,
                                                MyDynamicVar.GetType());

    MyDynamicVar = true;
    Console.WriteLine("Value: {0}, Type: {1}", MyDynamicVar,
MyDynamicVar.GetType());

    MyDynamicVar = DateTime.Now;
```

```
    Console.WriteLine("Value: {0}, Type: {1}", MyDynamicVar,
MyDynamicVar.GetType());
}
```

Output:

```
Value: 100, Type: System.Int32
Value: Hello World!!, Type: System.String
Value: True, Type: System.Boolean
Value: 01-01-2014, Type: System.DateTime
```

ប្រភេទអថេរ dynamic គឺ converted បំប្លែងទៅជាប្រភេទប្រយោលផ្សេងៗទៀត ។

Example: dynamic Type Conversion

```
dynamic d1 = 100;
int i = d1;
d1 = "Hello";
string greet = d1;

d1 = DateTime.Now;
DateTime dt = d1;
```

១០.១ Methods and Parameters

ប្រសិនបើអ្នកបន្ថែមថ្នាក់វត្ថុមួយ assign class object ទៅប្រភេទ dynamic នោះ compiler នឹងមិនពិនិត្យមើលវិធីសាស្ត្រ methods ត្រឹមត្រូវ និង properties name ឈ្មោះ លក្ខណៈសម្បត្តិ នៃប្រភេទ dynamic ដែលផ្ទុក custom class object ផ្ទាល់ខ្លួនទេ។ សូមពិចារណាឧទាហរណ៍ខាងក្រោម៖

Example: Calling Methods

```
 1 using System;
 2
 3 public class Program
 4 {
 5    public static void Main()
 6    {
 7       var student = new {Id = 1,FirstName ="James",LastName= "Bond" };
 8
 9       Console.WriteLine(student.Id); //output: 1
10       Console.WriteLine(student.FirstName); //output: James
11       Console.WriteLine(student.LastName); //output: Bond
12    }
13 }
```

```
Run-time exception (line 8): No overload for method 'DisplayStudentInfo' takes '2' ar

Stack Trace:

[Microsoft.CSharp.RuntimeBinder.RuntimeBinderException: No overload for method 'Displ
   at CallSite.Target(Closure , CallSite , Object , Int32 , String )
   at System.Dynamic.UpdateDelegates.UpdateAndExecuteVoid3[T0,T1,T2](CallSite site,
   at Program.Main() :line 8
```

១១. ប្រភេទទិន្នន័យទទេ Nullable Types

អ្នកប្រហែលជាបានស្គាល់ប្រភេទ មិនតម្លៃ (null) ។ សម្រាប់ឧទាហរណ៍ *int i = null* នឹងផ្តល់ឱ្យអ្នកនូវកំហុសពេលវេលាដែល *compile* ចាប់ផ្តើម *compile time error* ។ ការចាប់ផ្តើមជាមួយប្រភេទ *nullable* ដែលបានអនុញ្ញាត ដើម្បីផ្តល់នូវតម្លៃ *null* ទៅប្រភេទតម្លៃអថេរ ។ អ្នកអាចប្រកាសប្រភេទ *nullable* ដោយប្រើប្រាស់ *nullable<s>* ដែល *s* គឺជាប្រភេទមួយ ។

Example: Nullable type

```
Nullable<int> i = null;
```

ប្រភេទ nullable ដែលអាចចាត់ទុកជាតម្លៃទទេអាចតំណាងឱ្យជួរតម្លៃត្រឹមត្រូវសម្រាប់ប្រភេទតម្លៃមូលដ្ឋានរបស់វា បូកនឹងតម្លៃទទេបន្ថែម។ ឧទាហរណ៍ *Nullable<int>* អាចត្រូវបានផ្តល់តម្លៃណាមួយពី *-2147483648* ដល់ *2147483647* ឬតម្លៃ *null* ។ ប្រភេទ *Nullable* គឺ*Nullable types are instances* នៃ *System.Nullable<s> struct* ។ គិតថាវាជាអ្វីមួយដូចជារចនាសម្ព័ន្ធខាងក្រោម។

Example: Nullable struct

```
[Serializable]
public struct Nullable<T> where T : struct
{
    public bool HasValue { get; }

    public T Value { get; }
    // other implementation
}
```

nullable នៃប្រភេទចំនួនគត់ *int* គឺជាម្មតាបូក *int* ដូចគ្នានឹងដែលនិយាយថាថាតើ *int* មានតម្លៃ ឬអត់ *(*គឺ *null* ទទេ ឬ *not* អត់*)*។ នៅសល់ទាំងអស់គឺជា *compiler* ដែលចាត់ទុក *"ទទេ" "null"* ជាតម្លៃត្រឹមត្រូវ។

Example: HasValue

```
static void Main(string[] args)
```

```
{
    Nullable<int> i = null;
    if (i.HasValue)
        Console.WriteLine(i.Value); // or Console.WriteLine(i)
    else
        Console.WriteLine("Null");
}
```

Output:

```
Null
```

ប្រើប្រាស់ GetValueOrDefault() method ដើម្បីទទួលបានតម្លៃពិតប្រាកដប្រសិនបើវាមិនមែនជា *null* និងតម្លៃលំនាំដើម default value ប្រសិនបើវា គឺជា *null* ។ ឧទាហរណ៍:

Example: GetValueOrDefault()

```
static void Main(string[] args)
{
    Nullable<int> i = null;

    Console.WriteLine(i.GetValueOrDefault());
}
```

```
Cached Result
0
```

១១.១ Shorthand Syntax for Nullable Types

អ្នកប្រើប្រាស់នៅ សញ្ញាឧទាន '?' *operator* ជាអក្សរកាត់ shorthand syntax ដូចជា int?, long? ជំនួសឱ្យការប្រើ Nullable <s>។

Example: Shorthand syntax for Nullable types

```
int? i = null;
double? D = null;
```

១១.២ ការប្រើប្រាស់សញ្ញា ឧទាន៣ ? ? Operator

សញ្ញាឧទាន *'??' operator* ដើម្បី កំណត់ប្រភេទ *nullable* ទៅជាប្រភេទ *non-nullable (assign nullable type to non-nullable type)* ។

Example: ?? operator with Nullable Type

```
int? i = null;

int j = i ?? 0;

Console.WriteLine(j);
```

Output:

```
0
```

ក្នុងឧទាហរណ៍ខាងលើ *i* គឺជា *int* ដែលមិនអាចចាត់ទុកជាមោឃៈ ហើយប្រសិនបើអ្នកកំណត់វាទៅ *int j* ដែលមិនអាចចាត់ទុកជាមោឃៈ នោះវានឹងបោះចោលការលើកលែងពេលវាត់ ប្រសិនបើ *i* ជាមោឃៈ។ ដូច្នេះ ដើម្បីកាត់បន្ថយហានិភ័យនៃករណីលើកលែង យើងបានប្រើ *'??'* ប្រតិបត្តិករដើម្បីបញ្ជាក់ថាប្រសិនបើខ្ញុំជា *null* បន្ទាប់មកកំណត់ *0* ទៅ *j* ។

១១.៣ Assignment Rules

nullable type មានច្បាប់ rules ចាត់ចែងដូចគ្នានឹងប្រភេទតម្លៃ value type ។ វាត្រូវតែត្រូវបានផ្តល់តម្លៃមួយមុនពេលប្រើវា ប្រសិនបើ nullable types ត្រូវបានប្រកាសនៅក្នុងអនុគមន៍ជាអថេរមូលដ្ឋាន local variables ។ ប្រសិនបើវាជា field នៃ class ថ្នាក់ណាមួយ នោះវានឹងមានតម្លៃ null តាម default លំនាំដើម។ ជាឧទាហរណ៍ nullable នៃប្រភេទចំនួនគត់ int ខាងក្រោមត្រូវបានប្រកាស និងប្រើប្រាស់ដោយមិនកំណត់តម្លៃណាមួយឡើយ។ compiler នឹងផ្តល់ឱ្យ "ការប្រើប្រាស់អថេរមូលដ្ឋានដែលមិនបានកំណត់ 'i'" កំហុស**"Use of unassigned local variable 'i'"** error៖

```
static void Main(string[] args)
{
    Nullable<int> i;

    Console.WriteLine(i);
}
```

Use of unassigned local variable 'i'

Unassigned nullable type-error

ក្នុងឧទាហរណ៍ខាងក្រោមនេះ nullable of int type គឺជា fieldនៃ class ដូច្នេះវានឹងមិន

ផ្តល់នូវកំហុសណាមួយឡើយ។

Example: Nullable type as Class Field

```
class MyClass
{
    public Nullable<int> i;
}
class Program
{
    static void Main(string[] args)
    {
        MyClass mycls = new MyClass();

        if(mycls.i == null)
            Console.WriteLine("Null");
    }
}
```

Output:

```
Null
```

១១.៤ Nullable Helper Class

Null ត្រូវបានចាត់ទុកថាតិចជាងតម្លៃណាមួយ។ ដូច្នេះ ប្រតិបត្តិករប្រៀបធៀបនឹងមិនដំណើរការទល់នឹង null ទេ។ ពិចារណាឧទាហរណ៍ខាងក្រោម ដែល I មិនតូចជាង j ធំជាង j ឬស្មើ j៖

Example: Nullable Type Comparison

```
static void Main(string[] args)
{
    int? i = null;
    int j = 10;
    if (i < j)
        Console.WriteLine("i < j");
    else if( i > 10)
        Console.WriteLine("i > j");
    else if( i == 10)
```

```
        Console.WriteLine("i == j");
    else
        Console.WriteLine("Could not compare");
}
```

Output:

```
Could not compare
```

Nullable static class គឺជា class ជំនួយសម្រាប់ប្រភេទ Nullable ។ វាផ្តល់នូវវិធីសាស្ត្រ compare method ប្រៀបធៀបដើម្បីប្រៀបធៀប nullable types ។ វាក៏មាន method GetUnderlyingType ដែល returns ត្រឡប់ប្រភេទមូលដ្ឋាននៃអាគុយម៉ង់នៃប្រភេទ nullable ។

Example: Helper Class

```
static void Main(string[] args)
{
    int? i = null;
    int j = 10;

    if (Nullable.Compare<int>(i, j) < 0)
        Console.WriteLine("i < j");
    else if (Nullable.Compare<int>(i, j) > 0)
        Console.WriteLine("i > j");
    else
        Console.WriteLine("i = j");
}
```

Output:

```
i < j
```

១១.៥ Characteristics of Nullable Types

Nullable types can only be used with value types.

1.The Value property will throw an InvalidOperationException if value is null; otherwise it will return the value.

2.The HasValue property returns true if the variable contains a value, or false if it is null.

3.You can only use == and != operators with a nullable type. For other comparison use the Nullable static class.

4.Nested nullable types are not allowed. Nullable<Nullable<int>> i; will give a compile time error.

១២. សិក្សាអំពី Enums

នៅក្នុងភាសា C# enum (ហៅថា enumeration) គឺជាប្រភេទតម្លៃ កំណត់ដោយអ្នកប្រើប្រាស់ដែលប្រើដើម្បីតំណាងឱ្យបញ្ជី list នៃnamed integer ចំនួនថេរដែលមានឈ្មោះ។ វាត្រូវបាន បង្កើតដោយប្រើពាក្យគន្លឹះ enum នៅក្នុង class, structure ឬ namespace ។ វាធ្វើឱ្យប្រសើរឡើងនូវការអាន ការរក្សាបាននូវកម្មវិធី និងកាត់បន្ថយភាពស្មុគស្មាញ។

enum គឺClass ពិសេស "class" ដែលតំណាង group of constants (unchangeable/read-only variables) ។ To create an enum, use the enum keyword (instead of class or interface), and separate the enum items with a comma ។

មូលដ្ឋាននៃការប្រកាស enum គឺ៖

```
enum enum_name
{
   enumeration list
};
```

enum_name គឺជាឈ្មោះដែលអ្នកចង់ផ្តល់ឱ្យទៅបញ្ជី enum របស់អ្នក។ យើងប្រើសញ្ញាក្បៀស (,) ដើម្បីបំបែក items ផ្សេងនៅក្នុង List បញ្ជីរាប់បញ្ចូល។

ឧទាហរណ៍៖ January, February, March, May, and June are months of the year ។ ដូច្នេះ នេះក្លាយជាការរាប់បញ្ចូលជាមួយនឹងឈ្មោះឆ្នាំ និងខែមករា ខែកុម្ភៈ មីនា ឧសភា និងមិថុនា ជាធាតុរបស់វា។

```
enum year
{
    // items of the enum
    January,
    February,
    March,
    April,
    May,
    June
}
```

១២.១ តម្លៃ Enum Values

ប្រសិនបើយើងមិនកំណត់តម្លៃទៅធាតុ enum items ទេធាតុត្រូវបានផ្តល់តម្លៃចំនួនគត់ដោយអ្នកចងក្រងតាមលំនាំដើមដោយចាប់ផ្តើមពី 0 ។ ធាតុទីមួយនឹងត្រូវបានកំណត់ 0 និងបង្កើនម្តងរាល់ពេលដែលយើងបន្ថែមធាតុ។

```
enum year
{
    // items of the enum
    January,    //0
    February,   //1
    March,      //2
    April,      //3
    May,        //4
    June        //5
}
```

១២.២ ផ្តល់តម្លៃ Assigning our values

កំណត់តម្លៃរបស់យើងអាចផ្លាស់ប្តូរតម្លៃនៃធាតុ enum ។ ប្រសិនបើយើងប្តូរតម្លៃមួយទៅមួយខែឧទាហរណ៍ ខែមីនា ដល់ 10។ បន្ទាប់មកអ្នកចងក្រងនឹងកំណត់តម្លៃផ្សេងទៀតតាមលំដាប់លំដោយពោលគឺ វានឹងកើនឡើងមួយពី 10៖

```
enum year
{
    // items of the enum
    January,    //0
    February,   //1
    March,      //10
    April,      //11
    May,        //12
    June        //13
}
```

១២.៣ ដំណើរការAccess an Enum

យើងដំមណើរការ enum item ដោយប្រើ សញ្ញា dot (.)

```
enum year
{
    // items of the enum
    January,
    February,
    March,
    April,
    May,
    June
```

```
10  }
11
12  Console.WriteLine(year.January); // January
13  Console.WriteLine(WeekDays.February); // February
14  Console.WriteLine(WeekDays.March); // March
15  Console.WriteLine(WeekDays.April); // April
16  Console.WriteLine(WeekDays.May); // May
17  Console.WriteLine(WeekDays.June); // June
```

១២.៤ ការបម្លែងទៅជាចំនួនគត់ Convert an enum to an Integer

យើងត្រូវបំប្លែងយ៉ាងច្បាស់ជាតុនៅក្នុង enum ដើម្បីទទួលបានតម្លៃចំនួនគត់ ។ តោះមើលកម្មវិធីដើម្បីបង្ហាញពីរបៀបបំប្លែងធាតុទៅជាតម្លៃ integer value ។

```
1  using System;
2  namespace EnumerationExample {
3
4  enum year
5  {
6
7      // items of the enum
8      January,
9      February,
10     March,
11     April,
12     May,
13     June
14
15 }
16
17 class Program {
18
19     static void Main(string[] args)
20     {
21
22    // Printing out the integer values of the items
23    Console.WriteLine("The value of January in year " + "enum is
24
25                        (int)year.January);
26    Console.WriteLine("The value of February in year " + "enum is
27
28                       (int)year.February);
29    Console.WriteLine("The value of March in year " + "enum is "
30
31                       (int)year.March);
32    Console.WriteLine("The value of April in year " + "enum is "
33
34                        (int)year.April);
35    Console.WriteLine("The value of May in year " + "enum is " +
```

```
                        (int)year.May);
    Console.WriteLine("The value of June in year " + "enum is " +
36                      (int)year.June);
37      }
    }
    }
```

```
The value of January in year enum is 0
The value of February in year enum is 1
The value of March in year enum is 2
The value of April in year enum is 3
The value of May in year enum is 4
The value of June in year enum is 5
```

១២.៣ បម្លែង enum ទៅតួអក្សរ Convert enum to String

យើងប្រើប្រាស់ `ToString()` method ដើម្បី បម្លែង enum ទៅជា តួអក្សរ ។

ឧទាហរណ៍៖ តើរបៀបបម្លែង enum ទៅតួអក្សរដូចម្តេច ? ៖

```
//Convert an enum item to string
Console.WriteLine(year.January.ToString());
Console.WriteLine(year.February.ToString());
Console.WriteLine(year.March.ToString());
```

ជំពូកទី៤

ការសិក្សាស្វែងយល់អំពីប្រភេទ Operator ក្នុងភាសា C#

នៅក្នុងភាសា C# Operator គឺជានិមិត្តសញ្ញា(symbol) ដែលគេប្រើសម្រាប់ធ្វើគណនាទៅលើតម្លៃអ្វីមួយ ឬប្រើសម្រាប់ប្រាប់ Compiler ឱ្យគណនាអ្វីមួយដូចជាសញ្ញា (-, +, x, /, % , = , ...)។

១.តើគេប្រើ Operator ដើម្បីអ្វី?

គេប្រើ Operator សម្រាប់ធ្វើគណនាឬប្រៀបធៀបតម្លៃមួយឬច្រើនដូចជា

តើ Operator មានអ្វីខ្លះ?

Operator គឺសញ្ញា (Symbol) ដែលប្រើសម្រាប់គណនា ឬប្រៀបធៀបអ្វីមួយ ។

ប្រភេទ Operator ត្រូវបានបែងចែកជាប្រភេទដូចជា៖

ប្រភេទ Operator	បរិយាយ
Arithmetic operator	ប្រមាណវិធី ពីជគណិត
Assignment operator	ប្រមាណវិធី បញ្ចូលតម្លៃ
Comparison Operators	ប្រមាណវិធី ប្រៀបធៀប
Logical Operators	ប្រមាណវិធី តក្កវិទ្យា

២.សិក្សា ពី Arithmetic Operator:

Arithmetic Operator គឺជាសញ្ញាគណនាជាមួយនឹង ផ្នែកគណិតវិទ្យា ដើម្បីរកតម្លៃលទ្ធផលនៃការគណនាណាមួយ។ វាមានដូចជា + - * / ។ ចំពោះតម្លៃ ឬ Variable ដែលត្រូវប្រើប្រាស់ជាមួយនឹង Operator ដើម្បីធ្វើការគណនាត្រូវបានហៅថា Operand ។

Arithmetic operator គឺតួនាទីរបស់វាសម្រាប់ធ្វើគណនាតម្លៃ ដូចគណិតវិទ្យាដែរ ហើយសម្រាប់ Operator ផ្សេងទៀត។

យើងអាចប្រើប្រាស់ Arithmetic Operator ទាំងអស់ជាមួយនឹងតម្លៃ char, int, long, float, double, ឬ decimal ?ក្នុងនោះសញ្ញា + ក៏អាចប្រើប្រាស់បានជាមួយនឹង string បានផងដែរ ។

Arithmetic Operators

ប្រមាណវិធី	ឈ្មោះ ប្រមាណវិធី	បរិយាយ	ឧទាហរណ៍
+	បូក	បូកបន្ថែម តម្លៃពីរ	x+y
-	ដក	ដកតម្លៃអវិជ្ជមាន និងតម្លៃ មួយផ្សេងទៀត	x-y

*	គុណ		x*y
/	ចែក	យកតំណាងចែក ចែកគួចែក	X /y
++	បូកកើនឡើង	យកតម្លៃខ្លួនឯង បូកនឹង ១	x++
--	ដកថយចុះ	យកតម្លៃខ្លួនឯង ដកនឹង ១	x--
%	ភាគរយ	រកចំនួនភាគរយ អថេរ ចែកយកសំណល់	x%

២.១ ប្រមាណវិធីបូក (+)

ប្រមាណវិធី បូក គឺគណនាផលបូក តម្លៃរបស់អថេរ ពីរ ឬច្រើន A និង B ណាដែលមាន ប្រភេទទិន្នន័យដូចគ្នា របស់សំណើ Statements ។ សម្រាប់ធ្វើការគណនាប្រមាណវិធីបូករវាង **a** និង **b** ។ នៅពេលដែលធ្វើការគណនារួចរាល់ហើយ នោះកម្មវិធីនឹងធ្វើការចាកចេញ ។

```
1
2   using System;
3   using System.Collections;
4   using System.Collections.Generic;
5   using System.Linq;
6   using System.Text;
7   using System.Threading.Tasks;
8
9   namespace ConsoleApplication5
10  {
11      class Program
12      {
13          static void Main(string[] args)
14          {
15
16              int A, B, C;
17              A = 2;
18              B = 4;
19              C = A + B;
20
21              Console.WriteLine("Result = "+C);
22              Console.ReadKey();
23          }
24
25      }
26  }
```

២.២ ប្រមាណវិធីដក (-)

ប្រមាណវិធី បូក គឺគណនាលើប្រមាណវិធីដក តម្លៃរបស់អថេរ ណាដែលមាន ប្រភេទទិន្នន័យដូចគ្នា របស់សំណើ Statements ។ សម្រាប់ធ្វើការគណនាប្រមាណវិធីដករវាង **a** និង **b** ។ នៅពេលដែលធ្វើការគណនារួចរាល់ហើយ នោះកម្មវិធីនឹងធ្វើការចាកចេញ ។

```
1  using System;
2  using System.Collections;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace ConsoleApplication5
9  {
10     class Program
11     {
12
13         static void Main(string[] args)
14         {
15
16             int A, B, C;
17             A = 2;
18             B = 4;
19             C = A - B;
20
21             Console.WriteLine("Result = "+C);
22             Console.ReadKey();
23         }
24
25     }
26 }
```

```
file:///C:/Users/PEN SARON/documents/visual studio 2013/Projects/Co
Result = -2
```

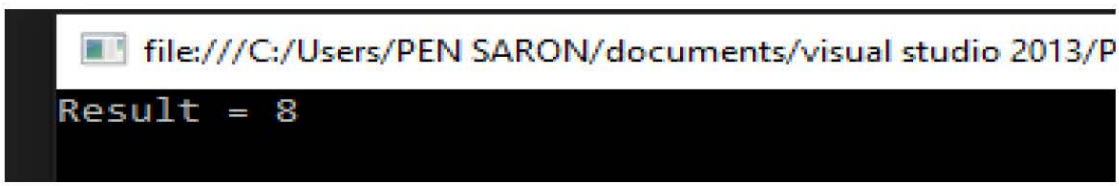
២.៣ ប្រមាណវិធីគុណ (*)

ចំពោះប្រមាណវិធីគុណ គឺយើងគុណ Field ណាដែលមាន ប្រភេទទិន្នន័យដូចគ្នា ។

សម្រាប់ធ្វើការគណនាប្រមាណវិធីគុណរវាង **a** និង **b** ។ នៅពេលដែលធ្វើការគណនារួចរាល់ហើយ នោះកម្មវិធីនឹងធ្វើការចាកចេញ ។

```
1  using System;
2  using System.Collections;
```

```
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace ConsoleApplication5
9  {
10     class Program
11     {
12
13         static void Main(string[] args)
14         {
15
16             int A, B, C;
17             A = 2;
18             B = 4;
19             C = A * B;
20
21             Console.WriteLine("Result = "+C);
22             Console.ReadKey();
23         }
24
25     }
26 }
```



២.៤ ប្រមាណវិធីចែក (/)

ចំពោះប្រមាណវិធីចែក គឺការគណនាប្រមាណវិធីចែករវាង តម្លៃនៃអថេរ ពីរ ឬច្រើន ហើយប្រភេទទិន្នន័យរបស់អថេរដូចគ្នា ។ សម្រាប់ធ្វើការគណនាប្រមាណវិធីចែករវាង **a** និង **b** ។ នៅពេលដែលធ្វើការគណនារួចរាល់ហើយ នោះកម្មវិធីនឹងធ្វើការចាកចេញ ។

```
1  using System;
2  using System.Collections;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace ConsoleApplication5
9  {
10     class Program
11     {
```

```
12
13          static void Main(string[] args)
14          {
15
16              int A, B, C;
17              A = 4;
18              B = 2;
19              C = A / B;
20
21              Console.WriteLine("Result = "+C);
22              Console.ReadKey();
23          }
24
25      }
26  }
```

```
file:///C:/Users/PEN SARON/documents/visual studio 2013/Projects/ConsoleApplication5/Cc
Result = 2
```

២.៤ ប្រមាណវិធីបន្ថែម (++)

ចំពោះប្រមាណវិធី បន្ថែម គឺការគណនាប្រមាណវិធីបន្ថែមកើនឡើងនឹង ១ តម្លៃនៃអថេរ ១ ។ សម្រាប់ធ្វើការគណនាប្រមាណវិធីចែករវាង **a** និង **b** ។ នៅពេលដែលធ្វើការគណនារួចរាល់ហើយ នោះកម្មវិធីនឹងធ្វើការចាកចេញ ។

```
 1 using System;
 2 using System.Collections;
 3 using System.Collections.Generic;
 4 using System.Linq;
 5 using System.Text;
 6 using System.Threading.Tasks;
 7
 8 namespace ConsoleApplication5
 9 {
10     class Program
11     {
12         static void Main(string[] args)
13         {
14             int A;
15             A = 4;
16
17             Console.WriteLine(A);    // output: 4
18             Console.WriteLine(A++); // output: 4
19             Console.WriteLine(A);    // output: 5
20
21             Console.ReadKey();
22         }
```

```
23
24
25      }
26
27
```

```
file:///C:/Users/PEN SARON/documents/visual studio 2013/Projects/ConsoleApplicat
4
4
5
```

២.៤ ប្រមាណវិធីបន្ថែម (--)

ចំពោះប្រមាណវិធី បន្ថែម គឺការគណនាប្រមាណវិធីថយម្តង១ តម្លៃរបស់វា១ ។ ខាងក្រោមនេះជាឧទាហរណ៍ពីភាពខុសគ្នានៃប្រើប្រាស់សញ្ញា +:

ចំពោះ Operands ដែលត្រូវបានប្រើប្រាស់ជាមួយនឹង Arithmetic Operator អាចផ្តល់លទ្ធផលខុសគ្នាទៅតាមប្រភេទតម្លៃដែលត្រូវបានសរសេរ។

```
 1
 2using System;
 3using System.Collections;
 4using System.Collections.Generic;
 5using System.Linq;
 6using System.Text;
 7using System.Threading.Tasks;
 8
 9namespace ConsoleApplication5
10{
11    class Program
12    {
13        static void Main(string[] args)
14        {
15            int A;
16            A = 4;
17            Console.WriteLine(A);    // output: 4
18            Console.WriteLine(--A); // output: 4
19            Console.WriteLine(A);    // output: 5
20
21            Console.ReadKey();
22        }
23    }
```

```
file:///C:/Users/PEN SARON/documents/visual studio 2013/Projects/ConsoleApplication
4
3
3
```

៣.សិក្ខាពី Assignment operator

Assignment operator ក្នុងភាសា C# ហើយនៅក្នុងអត្ថបទមុនក៏បានពន្យល់រួចមកហើយអំពី **Arithmetic operator**។ **AssignmentOperator** គឺជាប្រភេទសញ្ញា(Symbol)ដែលគេប្រើសម្រាប់ធ្វើការបញ្ជូនតម្លៃពីអង្គខាងឆ្វេងមកឱ្យអង្គខាងស្តាំ ។ **Assignment Operator (=)** គឺត្រូវបានប្រើប្រាស់ដើម្បីបោះតម្លៃ ឬ **Variable** ដែលនៅខាងស្តាំទៅ **Variable** ដែលនៅខាងឆ្វេងវា ។

ហើយខាងក្រោមជា ប្រភេទ Assignment operator៖

Operator	**Example**	**Same As**
=	x = 5	x = 5
+=	x += 3	x = x + 3
- =	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

៤. សិក្ខាពី Comparison Operator

Comparison Operator គេប្រើដើម្បីប្រៀបធៀបនូវ Operators ទាំងឡាយដោយផ្តល់នូវតម្លៃជា Boolean (true or false) ។

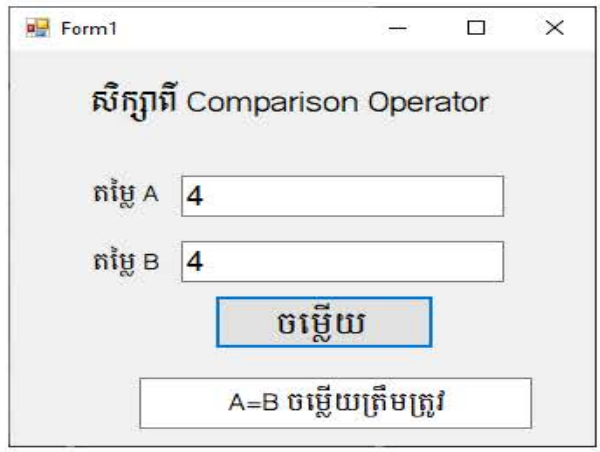
ខាងក្រោមនេះជា Comparison Operators:

Operator	Name	Example
==	ស្មើ	x == y
!=	មិនស្មើ	x != y
>	ធំជាង	x > y
<	តូចជាង	x < y
>=	ធំជាង ឬស្មើ	x >= y
<=	តូចជាង ឬស្មើ	x <= y

៤.១ ការប្រើប្រាស់ ប្រមាណវិធី ស្មើ(==)

ប្រមាណវិធី ស្មើ សញ្ញា(==) គឺត្រូវបានគេប្រើដើម្បីប្រៀបធៀប លក្ខខណ្ឌ តម្លៃណា ដែលស្មើ និង តម្លៃលក្ខខណ្ឌ ។

ឧទាហរណ៍ទី១៖



source code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void btResult_Click(object sender, EventArgs e)
        {
            double a, b;
            string Result;
            a =double.Parse(textBox1.Text);
            b =double.Parse(textBox2.Text);

           if (a == b)
            {
                Result = "A=B ចម្លើយត្រឹមត្រូវ";
```

```
                textBox3.Text = Result.ToString();
            }
            else
            {
                Result = "A!=B ចម្លើយមិនត្រឹមត្រូវ";
                textBox3.Text = Result.ToString();
            }
        }
    }
}
```

៤.២ ការប្រើប្រាស់ ប្រមាណវិធី តូចជាង (<)

Operator (<) គឺប្រើសម្រាប់ត្រួតពិនិត្យតម្លៃដែលតូចជាង ទៅនឹងសំណើ Statement ជាមួយ លក្ខខណ្ឌ Conditional ។ ប្រសិនបើ សំណើ ត្រួតពិនិត្យតម្លៃពីរផ្សេងគ្នា តូចជាង វាផ្តល់តម្លៃពិត បើសិនវា ត្រួតពិនិត្យលក្ខខណ្ឌធំជាង វាបង្ហាញតម្លៃ មិនពិត ។

```
1   class Program
2     {
3         static void Main(string[] args)
4         {
5             int x = 8 ;
6             int y = 10;
7
8             if (x < y)
9             {
10                Console.WriteLine("Your works is true! ");
11                Console.ReadKey();
12            }
13
14        }
15    }
```

៤.៣ ការប្រើប្រាស់ តូចជាងឬស្មើ Operator (<=)

Operator (<=) គឺប្រើសម្រាប់ត្រួតពិនិត្យតម្លៃដែលតូចជាងឬស្មើ ទៅនឹងសំណើ Statement ជាមួយលក្ខខណ្ឌ Conditional ។ ប្រសិនបើ សំណើ ត្រួតពិនិត្យតម្លៃពីរផ្សេងគ្នា តូចជាង វាផ្តល់តម្លៃពិត បើសិនវាត្រួតពិនិត្យលក្ខខណ្ឌធំជាង វាបង្ហាញតម្លៃ មិនពិត ។

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace ConsoleApplication2
8 {
```

```
 9     class Program
10     {
11         static void Main(string[] args)
12         {
13             int x = 8 ;
14             int y = 10;
15 
16             if (x <= y)
17             {
18                 Console.WriteLine("Your works is true! ");
19             }
20             else
21             {
22                 Console.WriteLine("Your works is False! ");
23             }
24             Console.ReadKey();
25         }
26     }
27 }
```

៤.៤ ការប្រើប្រាស់ ប្រមាណវិធី ធំជាង (>)

ប្រមាណវិធីធំជាង សញ្ញា(>) គឺត្រូវបានគេប្រើដើម្បីប្រៀបធៀប លក្ខខណ្ឌ តម្លៃណាដែលធំជាង ជាមួយតម្លៃនៃលក្ខខណ្ឌដទៃទៀត ។

```
 1     class Program
 2       {
 3           static void Main(string[] args)
 4           {
 5               int x=18 ;
 6               int y = 10;
 7 
 8               if (x > y)
 9               {
10                   Console.WriteLine("Your works is true! ");
11                   Console.ReadKey();
12               }
13 
14           }
15       }
```

៤.៥ ការប្រើប្រាស់ ប្រមាណវិធី ធំជាង ឬស្មើ (>=)

ប្រមាណវិធីធំជាង ឬស្មើ សញ្ញា(>=) គឺត្រូវបានគេប្រើដើម្បីប្រៀបធៀប លក្ខខណ្ឌ តម្លៃណាដែលធំជាង ឬស្មើ ជាមួយតម្លៃនៃលក្ខខណ្ឌដទៃទៀត ។

ឧទាហរណ៍៖

```
1   class Program
2     {
3         static void Main(string[] args)
4         {
5             int x=18 ;
6             int y = 10;
7
8             if (x >= y)
9             {
10                Console.WriteLine("Your works is true! ");
11                Console.ReadKey();
12            }
13
14        }
15    }
```

៤.៦ ការប្រើប្រាស់ ប្រមាណវិធី ស្មើ(= =)

ប្រមាណវិធី ស្មើ សញ្ញា(= =) គឺត្រូវបានគេប្រើដើម្បីប្រៀបធៀប លក្ខខណ្ឌ តម្លៃណា ដែលស្មើ និង តម្លៃលក្ខខណ្ឌ ។

ឧទាហរណ៍ៈ

```
1   class Program
2     {
3         static void Main(string[] args)
4         {
5             int x=10 ;
6             int y = 10;
7
8             if (x == y)
9             {
10                Console.WriteLine("Your works is true! ");
11                Console.ReadKey();
12            }
13
14        }
15    }
```

៤.៧ ការប្រើប្រាស់ ប្រមាណវិធី មិនស្មើ ឬផ្ទុយ(!=)

ប្រមាណវិធី មិនស្មើ ឬផ្ទុយ សញ្ញា(!=) គឺត្រូវបានគេប្រើដើម្បីប្រៀបធៀប លក្ខខណ្ឌ តម្លៃណា ដែល មិនស្មើ ឬផ្ទុយ និងតម្លៃលក្ខខណ្ឌ ។

ឧទាហរណ៍ៈ

```
1   class Program
2     {
3         static void Main(string[] args)
```

```
4           {
5               int x = 10 ;
6               int y = 10;
7
8               if (x != y)
9               {
10                  Console.WriteLine("Your works is true! ");
11              }
12              else
13              {
14                  Console.WriteLine("Your works is False! ");
15              }
16              Console.ReadKey();
17
18          }
19      }
20  //Output: Your works is False!
21
```

៥. សិក្សាពី Logical Operator

Logical Operators គេប្រើ Operator ដើម្បីបម្លែង Operator ឱ្យទៅជាតម្លៃ Boolean រួចធ្វើការប្រៀបធៀបតម្លៃអនុវត្តទាំងនោះ Or Operator ឬ (||) ផ្តល់តម្លៃ True ប្រសិនបើ Operator ខាងឆ្វេង ឬ ខាងស្តាំណាមួយមានតម្លៃ true ។ ឧទាហរណ៍ ១៖ true || false លទ្ធផលគឺ true ។ And operator នឹង (&&) ផ្តល់តម្លៃ true នៅពេលដែល Operator ទាំងពីរមានតម្លៃ true ។

logical Boolean operators អនុវត្តប្រមាណវិធី logical operations ជាមួយ bool operands ។ operators include ដែលមានដូចជា unary logical negation (!), binary logical AND (&), OR (|), និង exclusive OR (^), ហើយនឹង binary conditional logical AND (&&) and OR (||).

- Unary ! (logical negation) operator.
- Binary & (logical AND), | (logical OR), and ^ (logical exclusive OR) operators ត្រូវបានជ្រើសរើស operators តែងតែវាយតម្លៃប្រតិបត្តិការទាំងពីរ ។
- Binary && (conditional logical AND) and || (conditional logical OR) operators.

Operator	Name	Example
||	ឈ្នាប់ឬ	x || y
Or	ឈ្នាប់ឬ	x Or y
&&	ឈ្នាប់នឹង	x && y
And	ឈ្នាប់នឹង	x And y
!	ឈ្នាប់មិន	X ! y

៥.១ ឈ្មោះLogical negation operator !

បុព្វបទ **unary ! Operator** គណនាការ logical negation អវិជ្ជមានឡូជីខលនៃ **operand** របស់វា។ នោះគឺវាបង្កើតពិត ប្រសិនបើប្រតិបត្តិការវាយតម្លៃទៅមិនពិត និងមិនពិត ប្រសិនបើប្រតិបត្តិការវាយតម្លៃទៅពិត៖

```
1   static void Main(string[] args)
2      {
3
4          bool passed = false;
5          Console.WriteLine(!passed); // output: True
6          Console.WriteLine(!true); // output: False
7
8          Console.ReadKey();
9       }
```

unary postfix ! operator គឺជាnull-forgiving operator ។

៥.២ ឈ្មោះនឹង Logical AND operator &

ឈ្មោះនឹង & operator គណនា logical AND នៃ operands ប្រតិបត្តិការរបស់វា។ លទ្ធផលនៃ x & y គឺពិត true ប្រសិនបើ x និង y វាយតម្លៃទៅពិតទាំងពីរ ។ បើមិនដូច្នោះទេលទ្ធផលគឺមិនពិត false។

ឈ្មោះនឹង & operator ប្រតិបត្តិការវាយតម្លៃ operand ទាំងពីរ បើទោះបីជា operand ខាងឆ្វេងដៃវាយតម្លៃថាមិនពិត false ដូច្នេះលទ្ធផលប្រតិបត្តិការ គឺមិនពិត false ដោយមិនគិតពីតម្លៃនៃ operand ស្តាំដៃ ។ ក្នុងឧទាហរណ៍ខាងក្រោម ប្រតិបត្តិការដៃស្តាំរបស់ & ប្រតិបត្តិការគឺជាការហៅតាមវិធីសាស្ត្រដែលត្រូវបានអនុវត្តដោយមិនគិតពីតម្លៃនៃ operand ខាងឆ្វេង៖

```
1   static void Main(string[] args)
2      {
3
4          int y=4, x = 5;
5          Console.WriteLine(x > 3 & y < 10);
6          // returns True because 5 is greater than 3 AND 5 is
7   less than 10
8          Console.Read();
9
10     }
```

៥.៣ ឈ្មោះឬ Logical exclusive OR operator ^

ឈ្មោះឬ ^ គណនាឡូជីខលផ្តាច់មុខ OR ដែលត្រូវបានគេស្គាល់ផងដែរថាជា XOR ឡូជីខលនៃប្រតិបត្តិការរបស់វា។ លទ្ធផលនៃ x^y គឺពិត ប្រសិនបើ x វាយតម្លៃទៅ ពិត ហើយ y វាយតម្លៃទៅ មិនពិត ឬ x វាយតម្លៃទៅ មិនពិត ហើយ y វាយតម្លៃទៅ ពិត។ បើមិនដូច្នោះទេ លទ្ធផលគឺមិនពិត។ នោះគឺសម្រាប់ bool operands ប្រតិបត្តិការ ^ គណនាលទ្ធផលដូចគ្នានឹង inequality operator != ។

```
static void Main(string[] args)
  {

      Console.WriteLine(true ^ true);     // output: False
      Console.WriteLine(true ^ false);    // output: True
      Console.WriteLine(false ^ true);    // output: True
      Console.WriteLine(false ^ false);   // output: False
      Console.Read();

  }
```

៥.៤ ឈ្នាប់ឬ Logical OR operator |

ឈ្នាប់ឬ | ប្រតិបត្តិការគណនា logical OR ឡូជីខល OR នៃ operands របស់វា។ លទ្ធផលនៃ x | y គឺពិត ប្រសិនបើ x ឬ y វាយតម្លៃថា ពិត។ បើមិនដូច្នោះទេ លទ្ធផល គឺមិនពិត។ The | operator computes the logical OR of its operands. The result of x | y is true if either x or y evaluates to true. Otherwise, the result is false.

```
static void Main(string[] args)
   {

       int y=10, x = 5;
       Console.WriteLine(x > 3 | y < 10);
       // returns True because 5 is greater than 3 AND 5 is
less than 10
       Console.Read();

   }
```

៥.៥ លក្ខខណ្ឌឈ្នាប់នឹង Conditional logical AND operator &&

លក្ខខណ្ឌឈ្នាប់នឹង conditional logical AND operator && ដែលត្រូវបានគេស្គាល់ថាជា "short-circuiting" logical AND operator គណនា logical AND នៃ operands ប្រតិបត្តិការរបស់វា។ លទ្ធផលនៃ x && y គឺពិត ប្រសិនបើទាំងពីរ x និង y វាយតម្លៃទៅ ពិត ដូចគ្នា។ បើមិនដូច្នោះទេ លទ្ធផលគឺមិនពិត។ ប្រសិនបើ x វាយតម្លៃទៅ មិនពិត y មិនត្រូវបានគេវាយតម្លៃទេ។ logical AND operator & នៃប្រតិបត្តិការរបស់វាផងដែរ ប៉ុន្តែតែងតែវាយតម្លៃប្រតិបត្តិការទាំងពីរ។

```
static void Main(string[] args)
  {

      int y=4, x = 5;
      Console.WriteLine(x > 6 && y < 10);
      // returns True because 5 is greater than 3 AND 5 is less than

```

```
8         Console.Read();
9
10    }
```

៥.៦ លក្ខខណ្ឌឈ្នាប់ឆឺខ Conditional logical OR operator ||

តាមលក្ខខណ្ឌ ||conditional logical OR operator || ដែលត្រូវបានគេស្គាល់ថាជា "short-circuiting" ប្រតិបត្តិការឡូជីខល OR គណនាឡូជីខល OR នៃប្រតិបត្តិការរបស់វា។ លទ្ធផលនៃ x || y គឺពិត ប្រសិនបើ x ឬ y វាយតម្លៃថា ពិត។ បើមិនដូច្នោះទេ លទ្ធផល គឺមិនពិត ។ ប្រសិនបើ x វាយតម្លៃទៅ ពិត y មិនត្រូវបានគេវាយតម្លៃទេ។

```
1
2     static void Main(string[] args)
3       {
4
5           int y=4, x = 5;
6           Console.WriteLine(x > 6 || y < 10);
7     // returns True because 5,10 is greater than 4 AND 5 is less than 10
8           Console.Read();
9
10      }
```

៥.៦ លក្ខខណ្ឌ Nullable Boolean logical operators

សម្រាប់bool? operands, & (logical AND) និង | (logical OR) គាំទ្រ support តក្កវិជ្ជាដែលមានតម្លៃបីដូចខាងក្រោម៖

- operator & produces true ពិត លុះត្រាតែ operator ទាំងពីររបស់វាវាយតម្លៃថាពិត ទាំងពីរ ។ ប្រសិនបើ x ឬ y វាយតម្លៃថា មិនពិត x & y បង្កើត មិនពិត (ទោះបីជា operand ផ្សេងទៀតវាយតម្លៃទៅ ទទេ null)។ បើមិនដូច្នោះទេ លទ្ធផលនៃ x & y គឺទទេ null ។
- | operator produces false មិនពិត លុះត្រាតែប្រតិបត្តិករ operator ទាំងពីរ របស់វាវាយតម្លៃទៅមិនពិត។ ប្រសិនបើ x ឬ y វាយតម្លៃថា ពិត x | y ផលិត ពិត (ទោះបីជាប្រតិបត្តិករផ្សេងទៀតវាយតម្លៃទៅជាមោឃៈក៏ដោយ)។ បើមិនដូច្នោះទេ លទ្ធផលនៃ x | y គឺទទេnull ។

តារាងខាងក្រោមបង្ហាញពីអត្ថន័យ៖

x	y	x&y	x|y
true	true	true	true
true	false	false	true
true	null	null	true
false	true	false	true
false	false	false	false
false	null	false	null
null	true	null	true

x	y	x&y	x|y
null	false	false	null
null	null	null	null

៦. Compound assignment

In C# language, there are five compound assignment operators:

ប្រមាណវិធី	បរិយាយ គណនានឹង តម្លៃខ្លួនឯង
+=	ប្រមាណវិធី បូក
-=	ប្រមាណវិធី ដក
*=	ប្រមាណវិធី គុណ
/=	ប្រមាណវិធី ចែក
%=	ប្រមាណវិធី ចែកយកសំណល់

៦.១ += operator

ខាងក្រោមនេះជា operator ពីរ ក្នុងលំដាបនៃ Operator:

1.Add operation.

2.Assignment នៃលទ្ធផលរបស់ add operation.

សិក្សាពីរ += operator ជាមួយ code

```
int i=2;          //initializing i to 2
i+=2;             //equals to, i = i+2;
```

Statement i+=2 គឺស្មើនឹង i=i+2, ដូច្នេះ 2 នឹងត្រូវបានបន្ថែមទៅតម្លៃនៃ i ដែលផ្តល់ឱ្យយើងនូវ 4 ។ ទីបំផុត លទ្ធផលនៃការបន្ថែម 4 ត្រូវបានប្រគល់ទៅ i ដោយធ្វើបច្ចុប្បន្នភាពតម្លៃដើមរបស់វាពី 2 ទៅ 4 ។

៦.២ ប្រមាណវិធីដក -= operator

This operator performs two operations in sequence

1. Subtraction operation.

2. Assignment of the result of subtract operation

សិក្សាពីរ -= operator ជាមួយ code

```
int i=2;
i-=2;
```

Statement i-=2 គឺស្មើនឹង i=i-2 ដូច្នេះហើយ 2 នឹងត្រូវបានដកចេញពីតម្លៃនៃ i ដែលផ្តល់ឱ្យយើងនូវ 0 ។ ទីបំផុត លទ្ធផលនៃការដក 0 ត្រូវបាន assigned back ប្រគល់ឱ្យ i វិញ ដោយធ្វើបច្ចុប្បន្នភាព តម្លៃរបស់វាទៅ 0 ។

Example with -= operator

```
//C# Example of compound assignment operator -=
using System;
class A
{
public static void Main()
{
        char c='k';
        Console.WriteLine("Original character : " + c);
        c-=(char)10; // c = c+10 by using the required casting of int to char
        Console.WriteLine("Post -=10 operation on our character : " + c);

        short s=20;
        Console.WriteLine("Original short : " + s);
        s-=5*10; // s = s+(5*10);
        Console.WriteLine("Post -=5*10 operation on our short: " + s);

        int i=12;
        Console.WriteLine("Original int : " + i);
        i-=10; // i = i+10;
        Console.WriteLine("Post -=10 operation on our int: " + i);

        float f=5.5f;
        Console.WriteLine("Original float : " + f);
        f-=20; // f = f+20;
        Console.WriteLine("Post -=20 operation on our flot: " + f);

        double d=20.5;
        Console.WriteLine("Original double : " + d);
        d-=30; // d = d+30;
        Console.WriteLine("Post -=30 operation on our double: " + d);
}
}
```

Output is

```
 Original character : k
Post -=10 operation on our character : a
Original short : 20
Post -=5*10 operation on our short: -30
Original int : 12
Post -=10 operation on our int: 2
Original float : 5.5
Post -=20 operation on our flot: -14.5
Original double : 20.5
Post -=30 operation on our double: -9.5
```

៦.៣ ប្រមាណវិធីគុណ * = operator

This operator performs two operations in sequence

1.Multiplication operation.

2.Assignment of the result of multiplication operation.

សិក្សាពីរ *= operator ជាមួយ code

```
int i=2;        //initializing i to 2
i*=2;
```

Statement i*=2 គឺស្មើនឹង i=i*2 ដូច្នេះ 2 នឹងត្រូវបានគុណនឹងតម្លៃនៃ i ដែលផ្តល់ឱ្យយើងនូវ 4 ។ ចុងក្រោយ លទ្ធផលនៃគុណ 4 ត្រូវបានផ្តល់ត្រឡប់ទៅ i ដោយធ្វើបច្ចុប្បន្នភាពតម្លៃរបស់វាទៅជា 4។

```
//C# Example of compound assignment operator *=

using System;

class A
{
public static void Main()
{
     char c='d';
     Console.WriteLine("Original character : " + c);
     c*=(char)2; // c = c*2 by using the required casting of int to char
     Console.WriteLine("Post *=10 operation on our character: " + c);

     short s=20;
     Console.WriteLine("Original short : " + s);
     s*=5*10; // s = s*(5*10);
     Console.WriteLine("Post *=5*10 operation on our short : " + s);

     int i=12;
     Console.WriteLine("Original int : " + i);
     i*=10; // i = i*10;
     Console.WriteLine("Post *=10 operation on our int : " + i);

     float f=5.5f;
     Console.WriteLine("Original float : " + f);
     f*=20; // f = f*20;
     Console.WriteLine("Post *=20 operation on our float : " + f);

     double d=20.5;
     Console.WriteLine("Original double : " + d);
     d*=30; // d = d*30;
     Console.WriteLine("Post *=30 operation on our double : " + d);
}
}
```

Output

```
Original character : d
Post *=10 operation on our character: E
Original short : 20
Post *=5*10 operation on our short : 1000
Original int : 12
Post *=10 operation on our int : 120
Original float : 5.5
Post *=20 operation on our float : 110
Original double : 20.5
Post *=30 operation on our double : 615
```

៦.៤ ប្រមាណវិធីចែក /= operator

This operator performs two operations in sequence

1. Division operation
2. Assignment of the result of division operation

សិក្សាពីរ /= operator ជាមួយ code

```
int i=4;     //initializing i to 4
i/=2;
```

Statement i/=2 គឺស្មើនឹង i=i/2 ដូច្នេះ 4 នឹងត្រូវបានបែងចែកដោយតម្លៃនៃ i ដែលផ្តល់ឱ្យយើង 2 ។ ទីបំផុត លទ្ធផលនៃការបែងចែក 2 ត្រូវបានប្រគល់ត្រឡប់ទៅ i ដោយធ្វើបច្ចុប្បន្នភាពតម្លៃរបស់វាពី 4 ទៅ 2 ។

Example with /= operator

```
//C# Example of compound assignment operator /=

using System;

class A
{
public static void Main()
{
     char c='j';
     Console.WriteLine("Original character : " + c);
     c/=(char)2; // c = c/2 by using the required casting of int to char
     Console.WriteLine("Post /=2 operation on our character: " + c);

     short s=20;
     Console.WriteLine("Original short : " + s);
     s/=5*10; // s = s/(5*10);
     Console.WriteLine("Post /=5*10 operation on our short : " + s);
```

```
    int i=12;
    Console.WriteLine("Original int : " + i);
    i/=10; // i = i/10;
    Console.WriteLine("Post /=10 operation on our int : " + i);

    float f=5.5f;
    Console.WriteLine("Original float : " + f);
    f/=20; // f = f/20;
    Console.WriteLine("Post /=20 operation on our float : " + f);

    double d=20.5;
    Console.WriteLine("Original double : " + d);
    d/=30; // d = d/30;
    Console.WriteLine("Post /=30 operation on our double : " + d);
}
}
```

Output-

```
Original character : j
Post /=2 operation on our character: 5
Original short : 20
Post /=5*10 operation on our short : 0
Original int : 12
Post /=10 operation on our int : 1
Original float : 5.5
Post /=20 operation on our float : 0.275
Original double : 20.5
Post /=30 operation on our double : 0.683333333333333
```

៦.៥ ប្រមាណវិធីចែកយកសំណល់ %= operator

This operator performs two operations in sequence -

1. Modulus operation, which finds the remainder of a division operation.
2. Assignment of the result of modulus operation

សិក្សាពីរ %= operator ជាមួយ code

```
    int i=4;     //initializing i to 4
    i%=2;
```

Statement i%=2 ស្មើនឹង i=i%2 ដូច្នេះ 4 នឹងត្រូវបានបែងចែកដោយតម្លៃនៃ i ហើយនៅសល់របស់វាផ្តល់ឱ្យយើងនូវ 0 ។

ជាចុងក្រោយ លទ្ធផលនៃ modulus operation ម៉ូឌុលនេះ 0 ត្រូវបានកំណត់ត្រឡប់ទៅ i ដោយធ្វើបច្ចុប្បន្នភាពតម្លៃរបស់វាពី 4 ទៅ 0។

Example with %= operator

```
//C# Example of compound assignment operator %=

using System;

class A
{
public static void Main()
{
     char c='a';
     Console.WriteLine("Original character : " + c);
     c%=(char)2; // c = c%2 by using the required casting of int to char
     Console.WriteLine("Post %=2 operation on our character: " + c);


     short s=20;
     Console.WriteLine("Original short : " + s);
     s%=5*10; // s = s%(5*10);
     Console.WriteLine("Post %=5*10 operation on our short : " + s);


     int i=12;
     Console.WriteLine("Original int : " + i);
     i%=10; // i = i%10;
     Console.WriteLine("Post %=10 operation on our int : " + i);


     float f=5.5f;
     Console.WriteLine("Original float : " + f);
     f%=20; // f = f%20;
     Console.WriteLine("Post %=20 operation on our float : " + f);


     double d=20.5;
     Console.WriteLine("Original double : " + d);
     d%=30; // d = d%30;
     Console.WriteLine("Post %=30 operation on our double : " + d);
}
}
```

Output

```
Original character : a
Post %=2 operation on our character:
Original short : 20
Post %=5*10 operation on our short : 20
Original int : 12
Post %=10 operation on our int : 2
Original float : 5.5
```

```
Post %=20 operation on our float : 5.5
Original double : 20.5
Post %=30 operation on our double : 20.5
```

៧.សិក្ខាពី Controlling Precedence:

ក្នុង C# ចំពោះ Operator សញ្ញាមួយចំនួនដូចជា (*, /, និង %) គឺធ្វើការគណនាមុន សញ្ញា (+ និង -) ។ ដូចច្នេះ 2 + 3 * 4 លទ្ធផលដែលទទួលបានគឺ

```
int i=12;
int i=12+2*3;
int i=2*4-3;
```

ដើម្បីទទួលបានលទ្ធផលខុសយើងអាចប្រើប្រាស់ parentheses ( ) ហ៊ុំព័ទ្ធ Expression ទាំងឡាយណាដែលត្រូវគណនាមុន ដូច្នេះមានន័យថា សញ្ញា ( ) គឺធ្វើការគណនាមុនគេបង្អស់។

```
int i=12;
int i=(12+2)*3;
int i=2*(4-3);
```

ចំពោះ Operator ដែលមាន Precedence ដូចគ្នាគឺវាដំណើរការគណនាពីឆ្វេងទៅតាមធម្មតា ។

```
int i=12/6*2;
int j=12+2-3;
```

ចំពោះ Associativity គឺជាការលំដាប់នៃការគណនាដែលក្នុងនោះ Operator ដែលមាន Precedence ដូចគ្នានោះ Associativity របស់គឺជា left-associative (12/6*2) មានន័យថាលំដាប់នៃការគណនាគិតចាប់ពីឆ្វេងទៅស្តាំ។

៨. Incrementing and Decrementing Variables:

៨.១ Increment/Decrement Operators

Increment/decrement operators juis v ugwain run ប្រើដើម្បីបង្កើន បន្តយតំលៃ របស់ variable ដែលជា Integer ហើយជាទូទៅប្រើដើម្បីរាប់ Iteration របស់ loop ។

```
1 $x = $x + 1; // $x is incremented.
2 $x += 1; // $x is incremented.
3 $x++; // $x is incremented
4 $x = $x - 1; // $x is decremented.
5 $x--; // $x is decremented.
6 $x = 1; // $x is decremented.
```

ប្រសិនបើយើងត្រូវការបន្ថែម តម្លៃ 1 ទៅឲ្យ Variable នោះយើងអាចប្រើសញ្ញា+ Operator:

```
count= count+1;
```

ភាសាC# បានផ្តល់នូវ Operator មួយសម្រាប់បន្ថែមតម្លៃ 1 ទៅឲ្យ Variable ខ្លួនវាដោយយើងត្រូវប្រើប្រាស់ សញ្ញា ++ នៅខាងក្រោយ Variable នោះ ។

```
count++;
```

ក្នុងនោះយើងក៏អាចប្រើប្រាស់សញ្ញា -- ដើម្បីបន្ថយតម្លៃ 1 ចេញពី Variable បានផងដែរ។

```
count--;
```

ចំពោះ -- Operator និង ++ Operator ដែលហៅថា Unary Operator ។

៩. Prefix and Postfix:

Increment++ និង decrement – Operator គឺនឹងផ្តល់តម្លៃខុសនៅពេលដែលយើងដាក់នៅខាងមុន ឬខាងក្រោយ Variable ។ ក្នុងការដាក់សញ្ញានៅខាងមុខ Variable ត្រូវបានហៅថា prefix form ចំណែកការដាក់សញ្ញានៅខាងក្រោយ Variable ត្រូវបានហៅថា postfix form ។

```
count++; //postfix increment
++count; //prefix increment
count--; //postfix decrement
--count; //prefix increment
```

ខាងក្រោមនេះជាលទ្ធផលខុសគ្នានៃការប្រើប្រាស់ ++x ជាមួយនឹង x++

int x;

```
x = 42;
Console.WriteLine (x++); //x is now 43, 42 written out
Console.WriteLine (++x); //x is now 43, 43 written out
```

តាមរយៈឧទាហរណ៍ខាងលើបង្ហាញថា៖

x++ គឺវាធ្វើការមុននឹងបន្ថែមតម្លៃមានន័យថាវាផ្តល់ទៅឲ្យ Console.WriteLine តម្លៃចាស់ 42 រួចទើបបន្ថែមតម្លៃ 1 ទៅឲ្យខ្លួនវាស្មើ 43។

++x គឺវាបន្ថែមតម្លៃមុននឹងវាធ្វើការមានន័យថាបន្ថែមតម្លៃ 1 ទៅឲ្យខ្លួនវាស្មើ 43 រួចទើបផ្តល់ទៅឲ្យ Console.WriteLine តម្លៃថ្មី 43 ដែរ។

១០. Declaring Implicitly Typed Local Variables:

យើងអាចធ្វើការ initialize Variable នៅលើត្រង់ Statement តែមួយក៏បាន ដូចឧទាហរណ៍ខាងក្រោម ៖

```
int s=100;
```

ឧទាហរណ៍ខាងលើមានន័យថា គឺជាការបង្កើត Variable មួយឈ្មោះ myInt មាន Data Type int ហើយក្នុងនោះយើងបានផ្តល់តម្លៃ 99 ភ្លាមៗទៅឲ្យវា ។ សូមចងចាំថាយើងត្រូវផ្តល់តម្លៃឲ្យ Variable ទៅតាមប្រភេទទិន្នន័យដែលវាត្រូវទទួលយកផងដែរ។

លក្ខណៈពិសេសរបស់ C# គឺយើងអាចឲ្យវាជ្រើសរើសយក Date Type ណាមួយដែលសាកសមទៅនឹង Variable ដែលយើងបង្កើតបានផងដែរដោយវាធ្វើការត្រួតពិនិត្យទៅលើតម្លៃដែលបោះទៅឲ្យ Variable នោះ។

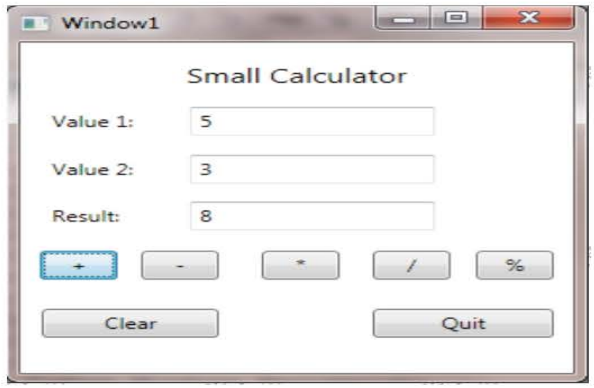
```
var myVariable =100;
var  myOtherVariable= " Hello IT ";
```

តាមឧទាហរណ៍ខាងលើ Variable myVariable និង myOtherVariable គឺត្រូវបានហៅថា implicitly typed variables។ ចំពោះ var Keyword គឺត្រូវបានប្រើប្រាស់ដើម្បីប្រាប់ទៅឲ្យ compiler ឲ្យជ្រើសរើសយក Data Type ដ៏ត្រឹមត្រូវមួយសម្រាប់ Variable ទៅតាមតម្លៃដែលផ្ទុក។ដូច្នេះយើងបាន myVariable គឺជា int ចំណែក myOtherVariable គឺ String ។ ហើយនៅពេលក្រោយទៀត យើងមិនអាចធ្វើការផ្តល់តម្លៃផ្សេងៗទៀតដូចជា float, double, ឬ string ទៅឲ្យ myVariable បានទៀតឡើយ។

ចំពោះ Variable ទាំងឡាយណាដែលប្រើប្រាសមួយ var Keyword ជាចំបាច់ត្រូវតែ assign តម្លៃឲ្យវាភ្លាមៗបើមិនដូច្នេះទេ នឹងមាន Error កើតឡើង។

១១. លំហាត់:

ចូរសរសេរ code មួយដើម្បី display ព័ត៌មានមួយចំនួនដូចខាងក្រោម៖



ជំពូកទី៥

ការសិក្សាស្វែងយល់អំពីប្រភេទ Math ក្នុងភាសា C#

នៅក្នុងភាសា C# Math គឺជាប្រមាណវិធី ដែលគេប្រើសម្រាប់ធ្វើគណនាទៅលើតម្លៃអ្វីមួយ ឬប្រើសម្រាប់ប្រាប់ Compiler ឱ្យគណនាអ្វីមួយដូចជា (Min, Max, Sqrt, Round, Abs,.....) ។

តើគេប្រើ Math ដើម្បីអ្វី?

Method	Description
Abs()	Returns the absolute value of a specified number.
Acos()	Returns the angle whose cosine is the specified number.
Acosh()	Returns the Inverse hyperbolic cosine of the specified number.
Asin()	Returns the angle whose sine is the specified number.
Asinh()	Returns the Inverse hyperbolic sine of the specified number.
Atan()	Returns the angle whose tangent is the specified number.
Atan2()	Returns the angle whose tangent is the quotient of two specified numbers.
Atanh()	Returns the Inverse hyperbolic tangent of the specified number.
BigMul()	Produces the full product of two 32-bit numbers.
Cbrt()	Returns the cube root of a specified value.
Ceiling()	Returns the smallest integral value greater than or equal to the specified number.
Clamp()	It is used to restrict a value to a given range.
Cos()	Returns the cosine of the specified angle.
Cosh()	Returns the hyperbolic cosine of the specified angle.
DivRem()	Calculates the quotient of two numbers and also returns the remainder in an output parameter.
Exp()	Returns e raised to the specified power.
Floor()	Returns the largest integral value less than or equal to the specified number.
IEEERemainder()	Returns the remainder resulting from the division of a specified number by another specified number.
Log()	Returns the logarithm of a specified number.
Log10()	Returns the base 10 logarithm of a specified number.
Max()	Returns the larger of two specified numbers.
Min()	Returns the smaller of two numbers.
Pow()	Returns a specified number raised to the specified power.
Round()	Rounds a value to the nearest integer or to the specified number of fractional digits.
Sign()	Returns an integer that indicates the sign of a number.

Method	Description
Sin()	Returns the sine of the specified angle.
Sinh()	Returns the hyperbolic sine of the specified angle.
Sqrt()	Returns the square root of a specified number.
Tan()	Returns the tangent of the specified angle.
Tanh()	Returns the hyperbolic tangent of the specified angle.
Truncate()	Calculates the integral part of a number.

១ អនុគមន៍ ABS ( )

ABS ( ) គឺជាអនុគមន៍ ប្រើសម្រាប់ប្តូរមួយចំនួន ទៅជាតម្លៃដាច់ខាត ។

Abs() is a Math class method which is used to return the absolute value of a specified number.

1. Math.Abs(Decimal)
2. Math.Abs(Double)
3. Math.Abs(Int16)
4. Math.Abs(Int32)
5. Math.Abs(Int64)
6. Math.Abs(SByte)
7. Math.Abs(Single)
8. Math.Abs(Decimal)

method is used to return the absolute value of a Decimal number.

Syntax:

```
public static decimal Abs (decimal val);
```

Parameter:

val៖ វាគឺជាចំនួនដែលត្រូវការដែលធំជាង ឬស្មើនឹង *Decimal.MinValue* ប៉ុន្តែតិចជាង ឬស្មើ នឹង *Decimal.MaxValue* នៃប្រភេទ *System.Decimal*។

Return Type៖ វាreturns លេខទសភាគនិយាយថា *r* ដូចជា $0 \le r \le Decimal.MaxValue$។

ឧទាហរណ៍៖

```
1 // C# Program to illlustrate the
2 // Math.Abs(Decimal) Method
3 using System;
4
5 class Geeks {
6
```

```
    // Main Method
    public static void Main()
    {

        // Taking decimal values
        decimal[] deci = {Decimal.MinValue, 45.14M, 0M,
                          -17.47M, Decimal.MaxValue};

        // using foreach loop
        foreach(decimal value in deci)

            // Displaying the result
            Console.WriteLine("Absolute value of {0} = {1}",
                                    value, Math.Abs(value));
    }
}
```

Output:

```
Absolute value of -79228162514264337593543950335 =
79228162514264337593543950335

Absolute value of 45.14 = 45.14

Absolute value of 0 = 0

Absolute value of -17.47 = 17.47

Absolute value of 79228162514264337593543950335 =
79228162514264337593543950335
```

១.១ *Math.Abs(Double)*

Method នេះត្រូវបានប្រើដើម្បី *return* ត្រឡប់តម្លៃដាច់ខាតនៃចំនួន *double-precision floating-point number* ។

Syntax:

```
public static double Abs (double val);
```

ឧទាហរណ៍៖

```
/// C# Program to illlustrate the
// Math.Abs(Double) Method
using System;

class Geeks {

    // Main Method
    public static void Main()
    {
```

```
10
11          // Taking a NaN
12          Double nan = Double.NaN;
13
14          // Taking double values
15          double[] doub = {Double.MinValue, 27.58, 0.0,
16                           56.48e10, nan, Double.MaxValue};
17
18          // using foreach loop
19          foreach(double value in doub)
20
21              // Displaying the result
22              Console.WriteLine("Absolute value of {0} = {1}",
23                                          value, Math.Abs(value));
24      }
25  }
26
```

លទ្ធផល៖

```
Absolute value of -1.79769313486232E+308 = 1.79769313486232E+308

Absolute value of 27.58 = 27.58

Absolute value of 0 = 0

Absolute value of 564800000000 = 564800000000

Absolute value of NaN = NaN

Absolute value of 1.79769313486232E+308 = 1.79769313486232E+308
```

១.២ Math.Abs(Int16)

method គឺគេប្រើដើម្បី return តម្លៃជាច់ខាត absolute value នៃ 16-bit signed integer ។

Syntax:

```
public static short Abs (short val);
```

Parameter:

1. val៖ វាគឺជាលេខដែលត្រូវការដែលធំជាង Int16.MinValue ប៉ុន្តែតូចជាង ឬស្មើនឹង Int16 MaxValue នៃ ប្រភេទ System.Int16។
2. Return Type៖ វាត្រឡប់ចំនួនគត់ដែលបានចុះហត្ថលេខា 16 ប៊ីតនិយាយថា r ដូចជា 0 ≤ r ≤ Int16.MaxValue។
3. .Exception៖ method នេះនឹងផ្តល់ឱ្យ OverflowException ប្រសិនបើតម្លៃ val ស្មើនឹង Int16.Min Value

Example:

```
1  // C# Program to illustrate the
2  // Math.Abs(Int16) Method
```

```
using System;

class Geeks {

    // Main Method
    public static void Main()
    {

        // Taking short values
        short[] sh = {Int16.MaxValue, 1482, -142, 0 };

        // using foreach loop
        foreach(short value in sh)

            // Displaying the result
           Console.WriteLine("Absolute value of {0} = {1}",
                                        value, Math.Abs(value));
    }
}
```

Output:

```
Absolute value of 32767 = 32767

Absolute value of 1482 = 1482

Absolute value of -142 = 142

Absolute value of 0 = 0
```

១.៣. Math.Abs(Int32)

method ប្រើប្រាស់ដើម្បី return តម្លៃជាច់ខាត absolute value នៃចំនួនគត់ 32-bit signed integer.

Syntax:

```
public static int Abs (int val);
```

Parameter:

val៖ វាគឺជាលេខដែលត្រូវការដែលធំជាង Int16.MinValue ប៉ុន្តែតូចជាង ឬស្មើនឹង Int16 MaxValue នៃប្រភេទ System.Int32។

Return Type៖ វា return ចំនួនគត់ signed integer 32 ប៊ីតនិយាយថា r ដូចជា 0 ≤ r ≤ Int16.MaxValue។

Exception៖ method នេះនឹងផ្តល់ឱ្យ OverflowException ប្រសិនបើតម្លៃ val ស្មើនឹង Int32.MinValue

Example:

```
// C# Program to illlustrate the
// Math.Abs(Int32) Method
using System;
```

```
class Geeks {

    // Main Method
    public static void Main()
    {

        // Taking int values
        int[] int_val = {Int32.MaxValue, 13482, -65525, 0};

        // using foreach loop
        foreach(int value in int_val)

            // Displaying the result
            Console.WriteLine("Absolute value of {0} = {1}",
                                        value, Math.Abs(value));
    }
}
```

Output:

```
Absolute value of 2147483647 = 2147483647

Absolute value of 13482 = 13482

Absolute value of -65525 = 65525

Absolute value of 0 = 0
```

២. អនុគមន៍ cos()

math.Cos() គឺជាអនុគមន៍ ប្រើសម្រាប់គណនាតម្លៃកេមុំកូស៊ីនីស Cos។

Syntax:

```
math.Cos(number)
```

ឧទាហរណ៍៖

```
using System;

namespace ConsoleApplication7
{
    class Program
    {
        static void Main(string[] args)
        {

            double num1 = 60.0, num2 = 0.0, num3 = 1.0;

            // It returns the hyperbolic cosine of
            // specified angle in radian
            double cosvalue = Math.Cosh(num1);
            Console.WriteLine("The cosh of num1 = " + cosvalue);
```

```
16
17              cosvalue = Math.Cosh(num2);
18              Console.WriteLine("The cosh of num2 = " + cosvalue);
19
20              cosvalue = Math.Cosh(num3);
21              Console.WriteLine("The cosh of num3 = " + cosvalue);
22              Console.Read();
23          }
24      }
25 }
```

លទ្ធផល៖

```
The cosh of num1 = -0.952412980415156
The cosh of num2 = 1
The cosh of num3 = 0.54030230586814
```

៣ អនុគមន៍ Math.BigMul()

BigMul( ) គឺជាវិធីសាស្ត្រថ្នាក់ method class method នេះត្រូវបានប្រើដើម្បីគណនាull product ផលិតផលពេញលេញនៃtwo 32-bit numbers ។

Syntax:

```
public static long BigMul(int a, int b);
```

ឧទាហរណ៍៖

```
1  using System;
2
3  namespace ConsoleApplication7
4  {
5      class Program
6      {
7          static void Main(string[] args)
8          {
9
10             Int32 x1 = 233232322;
11             Int32 x2 = 189222338;
12
13             // Using BigMul( ) method and storing
14             // result into a long(Int64) variable
15             long product = Math.BigMul(x1, x2);
16
17             // Getting the output
18           Console.WriteLine("The product of the two numbers is " + product);
19           Console.Read();
20         }
21     }
22 }
```

លទ្ធផល៖

```
The product of the two numbers is 44132765266008836
```

៤. អនុគមន៍ Math.Sin()

Math.SIN() គឺជាអនុគមន៍ ប្រើសម្រាប់ប្តូរមួយចំនួន ទៅជាតម្លៃស៊ីនីស Sin។

Syntax:

```
Math.Sin (number)
```

ឧទាហរណ៍៖

```
using System;

namespace ConsoleApplication7
{
    class Program
    {
        static void Main(string[] args)
        {

            double num1 = 60.0, num2 = 0.0, num3 = 1.0;

            // It returns the hyperbolic cosine of
            // specified angle in radian
            double Sinvalue = Math.Sin(num1);
            Console.WriteLine("The sin of num1 = " + Sinvalue);

            Sinvalue = Math.Sin(num2);
            Console.WriteLine("The sin of num2 = " + Sinvalue);

            Sinvalue = Math.Sin(num3);
            Console.WriteLine("The sin of num3 = " + Sinvalue);
            Console.Read();
        }
    }
}
```

លទ្ធផល៖

```
The Sin of num1 = -0.304810621102217
The Sin of num2 = 0
The Sin of num3 = 0.841470984807897
```

៥. អនុគមន៍ Math.Asinh()

Math.Asinh() គឺជាអនុគមន៍ ប្រើសម្រាប់គណនាតម្លៃមុំស៊ីនីស ដោយប្តូរពីរមួយចំនួន ទៅជាតម្លៃស៊ីនីស Sin។

Syntax:

```
Math.Asin (number)
```

ឧទាហរណ៍៖

```
using System;

namespace ConsoleApplication9
{
  class Program
  {
    // Main Method
   static void Main(string[] args)
    {
     double val1 = -0.0;
     double val2 = Double.PositiveInfinity;
     double val3 = Double.NaN;
    Console.WriteLine("Return value of {0} :{1}", val1, Math.Asin(val1));
    Console.WriteLine("Return value of {0} :{1}", val2, Math.Asin(val2));
     Console.WriteLine("Return value of {0} :{1}", val2,Math.Asin(val3));
     Console.Read();

        }
    }
}
```

៦. អនុគមន៍ Math.Sinh ( )

Math.Sinh ( ) គឺជាអនុគមន៍ ប្រើសម្រាប់ប្តូរមួយចំនួន ទៅជាតម្លៃស៊ីនីស Sin។

Syntax:

```
    Math.Sinh (number)
```

ឧទាហរណ៍៖

```
using System;

namespace ConsoleApplication7
{
    class Program
    {
        static void Main(string[] args)
        {

            double num1 = 60.0, num2 = 0.0, num3 = 1.0;

            // It returns the hyperbolic cosine of
            // specified angle in radian
            double Sinhvalue = Math.Sinh(num1);
            Console.WriteLine("The sin of num1 = " + Sinhvalue);
```

```
            Sinhvalue = Math.Sinh(num2);
            Console.WriteLine("The sin of num2 = " + Sinhvalue);

            Sinhvalue = Math.Sinh(num3);
            Console.WriteLine("The sin of num3 = " + Sinhvalue);
            Console.Read();
        }
    }
}
```

លទ្ធផល៖

```
The Sinh of num1 = 5.71003694907842E+25
The Sinh of num2 = 0
The Sinh of num3 = 1.1752011936438
```

៧. អនុគមន៍ Atan()

Math.Atan() គឺជាអនុគមន៍ ប្រើសម្រាប់គណនាមុំតង់សង់ ដោយប្តូរមួយពីរចំនួន ឬពីរ ទៅជាតម្លៃ tangent ។

Syntax:

Math.Atan (*number*)

or

Math.Atan (*number1, number2*)

```
public static double Atan (double d);
```

ឧទាហរណ៍៖

```
using System;

namespace ConsoleApplication9
{
  class Program
   {
     // Main Method
    static void Main(string[] args)
     {
       double val1 = -0.0;
      double val2 = Double.PositiveInfinity;
      double val3 = Double.NaN;
      Console.WriteLine("Return value of {0}:{1}",val1,Math.Atan(val1));
      Console.WriteLine("Return value of {0}:{1}",val2,Math.Atan(val2));
      Console.WriteLine("Return value of {0}:{1}",val2,Math.Atan(val3));

       Console.Read();

     }
  }
}
```

លទ្ធផល៖

```
Return value of 0 : 0
```

```
Return value of ∞ : 1.5707963267949
Return value of ∞ : NaN
```

៨. អនុគមន៍ Tan ( )

Math.Tan ( ) គឺជាអនុគមន៍ ប្រើសម្រាប់គណនាមុំតង់សង់ ដោយប្តូរមួយពីរចំនួន ឬពីរ ទៅជាតម្លៃ tangent ។

Syntax:

Math.Tan (*number*)

or

Math.Tan (*number1, number2*)

ឧទាហរណ៍៖

```
using System;

namespace ConsoleApplication7
{
    class Program
    {
        static void Main(string[] args)
        {

            double num1 = 60.0, num2 = 0.0, num3 = 1.0;

            // It returns the hyperbolic cosine of
            // specified angle in radian
            double Tanvalue = Math.Tan(num1);
            Console.WriteLine("The Tan of num1 = " + Tanvalue);

            Tanvalue = Math.Sinh(num2);
            Console.WriteLine("The Tan of num2 = " + Tanvalue);

           Tanvalue = Math.Tan(num3);
            Console.WriteLine("The Tan of num3 = " + Tanvalue);
            Console.Read();
        }
    }
}
```

លទ្ធផល៖

```
The Tan of num1 = 0.320040389379563
The Tan of num2 = 0
The Tan of num3 = 1.5574077246549
```

៩. អនុគមន៍ *Tanh()*

*Math.Tanh()*គឺជាវិធីសាស្ត្រថ្នាក់គណិតវិទ្យាដែលបានបង្កើតឡើងដែល returns គណនា តម្លៃ

អ៊ីពែរបូលhyperbolic tan នៃអាគុយម៉ង់តម្លៃទ្វេdouble value argument ដែលបានផ្តល់។

Syntax:

```
public static double Tanh(double num);
```

ឧទាហរណ៍៖

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

namespace ConsoleApplication7
{
    class Program
    {
        static void Main(string[] args)
        {

            double num1 = 60.0, num2 = 0.1, num3 = 1.0;

            // It returns the hyperbolic tan of
            // specified  angle in radian
            double tanhvalue = Math.Tanh(num1);
            Console.WriteLine("The tanh of num1 = " + tanhvalue);

            tanhvalue = Math.Tanh(num2);
            Console.WriteLine("The tanh of num2 = " + tanhvalue);

            tanhvalue = Math.Tanh(num3);
            Console.WriteLine("The tanh of num3 = " + tanhvalue);
            Console.Read();
        }
    }
}
```

លទ្ធផល៖

```
The tanh of num1 = 1
The tanh of num2 = 0.0996679946249558
The tanh of num3 = 0.761594155955765
```

១0. អនុគមន៍ Cot ( )

Cot ( ) គឺជាអនុគមន៍ ប្រើសម្រាប់ប្តូរមួយចំនួន ឬពីរ ទៅជាតម្លៃ Cotangent ។

```
Syntax:
    Cot (number)
or
    Cot (number1, number2)
```

១១. អនុគមន៍ Math.Acos()

method Math.Acos() ក្នុង C# return មុំដែលកូស៊ីនុសជាលេខដែលបាន specified number ។ លេខនេះគឺជាតម្លៃ double អាគុយម៉ង់ ។

Syntax:

```
public static double Cosh(double num);
```

ឧទាហរណ៍៖

```
using System;

namespace ConsoleApplication9
{
  class Program
   {

   // Main Method
   static void Main(string[] args)
   {
     double val1 = -0.0;
     double val2 = Double.PositiveInfinity;
     double val3 = Double.NaN;
     Console.WriteLine("Return value of {0} :{1}",val1,Math.Acos(val1));
     Console.WriteLine("Return value of {0}:{1}", val2,Math.Acos(val2));
     Console.WriteLine("Return value of {0}:{1}",val2,Math.Acos(val3));

      Console.Read();

    }
  }
}
```

លទ្ធផល៖

```
Return value of 0 : 1.5707963267949
Return value of ∞ : NaN
Return value of ∞ : NaN
```

១២. អនុគមន៍ Math.Cosh()

Math.Cosh() គឺជា Math class method វិធីសាស្ត្រថ្នាក់គណិតវិទ្យាដែលបានបង្កើត ដែល returns ផ្តល់លទ្ធផល hyperbolic cosine កូស៊ីនុសអ៊ីពែរបូលនៃអាគុយម៉ង់តម្លៃទ្វេ double value argument ដែលបានផ្តល់ឱ្យ។

Syntax:

```
public static double Cosh(double num);
```

Parameters:

num លេខ៖ វាគឺជាលេខដែល cos អ៊ីពែរបូល ត្រូវត្រលប់មកវិញហើយប្រភេទនៃប៉ារ៉ាម៉ែត្រនេះគឺ *System.Double*។

ឧទាហរណ៍៖

```
using System;

namespace ConsoleApplication7
{
    class Program
    {
        static void Main(string[] args)
        {

            double num1 = 60.0, num2 = 0.0, num3 = 1.0;

            // It returns the hyperbolic cosine of
            // specified angle in radian
            double coshvalue = Math.Cosh(num1);
            Console.WriteLine("The cosh of num1 = " + coshvalue);

            coshvalue = Math.Cosh(num2);
            Console.WriteLine("The cosh of num2 = " + coshvalue);

            coshvalue = Math.Cosh(num3);
            Console.WriteLine("The cosh of num3 = " + coshvalue);
            Console.Read();
        }
    }
}
```

លទ្ធផល៖

```
The cosh of num1 = 5.71003694907842E+25
The cosh of num2 = 1
The cosh of num3 = 1.54308063481524
```

១២. អនុគមន៍ Ceiling()

Math.Ceiling() គឺជាអនុគមន៍ ប្រើសម្រាប់រកតម្លៃលក្ខខណ្ឌលំអៀងដែលធំជាង ឬស្មើ ។

```
Math.Celling (number1, number2)
```

ឧទាហរណ៍៖

```
using System;

namespace ConsoleApplication7
{
    class Program
    {
        static void Main(string[] args)
        {
```

```
 9
10              decimal val1 = 9.99M;
11              decimal val2 = -5.10M;
12              Console.WriteLine("Result = " + Math.Ceiling(val1));
13              Console.WriteLine("Result = " + Math.Ceiling(val2));
14              Console.Read();
15          }
16      }
17 }
18
```

លទ្ធផល៖

```
Result = 10
Result = -5
```

ឧទាហរណ៍៖

```
 1
 2 using System;
 3
 4 namespace ConsoleApplication7
 5 {
 6     class Program
 7     {
 8         static void Main(string[] args)
 9         {
10             double val1 = 3.1;
11             double val2 = -55.99;
12             Console.WriteLine("Result = " + Math.Ceiling(val1));
13             Console.WriteLine("Result = " + Math.Ceiling(val2));
14             Console.Read();
15         }
16     }
17 }
18
```

លទ្ធផល៖

```
Result = 4
Result = -55
```

១៣. អនុគមន៍ `Math.Min()`

`Math.Min()` គឺជាអនុគមន៍ ប្រើសម្រាប់គណនារកតម្លៃណាតូចជាងគេបំផុត ។

```
Syntax:
        Math.Min() ( Number1, Number2 );
```

ឧទាហរណ៍៖

```
1 using System;
2
3 namespace ConsoleApplication8
```

```
 4 {
 5     class Program
 6     {
 7         static void Main(string[] args)
 8         {
 9
10             double P_Min = Math.Min(6, 56);
11
12             // Print the result
13             Console.WriteLine(P_Min);
14             Console.Read();
15         }
16     }
17 }
```

លទ្ធផល៖

```
6
```

១៤. អនុគមន៍ Math.Max()

Math.Max() គឺជាអនុគមន៍ ប្រើសម្រាប់គណនារកតម្លៃណាធំជាងគេបំផុត ។

Syntax:

Math.Min() (*Number1, Number2*);

ឧទាហរណ៍៖

```
1 using System;
2
3 namespace ConsoleApplication8
4 {
5     class Program
6     {
7         static void Main(string[] args)
8         {
9
10             double P_Max = Math.Max(6, 56);
11
12             // Print the result
13             Console.WriteLine(P_Max);
14             Console.Read();
15         }
16     }
17 }
```

លទ្ធផល៖

```
56
```

១៥. អនុគមន៍អិចស្ប៉ូណង់ហ្សែល EXP()

EXP() គឺជាអនុគមន៍ ប្រើសម្រាប់គណនា អិចស្ប៉ូណង់ហ្សែល e ឡើកជាស្វ័យគុណ នៃមួយចំនួន ។

```
Syntax:
    Math.Exp ( Number )
```

ឧទាហរណ៍៖

```
using System;

namespace ConsoleApplication7
{
    class Program
    {
        static void Main(string[] args)
        {

            Console.WriteLine(Math.Exp(10.0));
            Console.WriteLine(Math.Exp(15.57));
            Console.WriteLine(Math.Exp(529.548));
            Console.WriteLine(Math.Exp(0.00));
            Console.Read();
        }
    }
}
```

លទ្ធផល៖

```
22026.4657948067
5780495.71030692
9.54496417945595E+229
1
```

១៦. អនុគមន៍ FLOOR()

Math.FLOOR() គឺជាអនុគមន៍ ប្រើសម្រាប់គណនា បង្គត់តម្លៃចំនួនគត់ធំបំផុត ដែលតូចជាង ឬ ស្មើ នៃមួយ ចំនួន ។

```
Syntax:

    Math.FLOOR ( Number )
    Math.Floor(Decimal)
    Math.Floor(Double)
```

ឧទាហរណ៍៖

```
// C# program to illustrate the
// Math.Floor(Decimal) function
using System;
public class GFG {
```

```
5
6       // Main method
7       static public void Main()
8       {
9           // Different numbers list to find
10          // its floor values
11          Console.WriteLine(Math.Floor(0.2018));
12          Console.WriteLine(Math.Floor(123.123));
13          Console.WriteLine(Math.Floor(-0.2));
14          Console.WriteLine(Math.Floor(0.0));
15          Console.WriteLine(Math.Floor(34.67M));
16          Console.read();
17      }
    }
```

លទ្ធផល៖

```
0
123
-1
0
34
```

ឧទាហរណ៍ទី២៖

```
1  using System;
2
3  namespace ConsoleApplication7
4  {
5      class Program
6      {
7          static void Main(string[] args)
8          {
9
10             // Two values.
11             double n1 = 0.2018;
12             double n2 = 123.123;
13             double n3 = -2.2;
14             double n4 = -123.123;
15
16             // Take floors of these values.
17             double floor1 = Math.Floor(n1);
18             double floor2 = Math.Floor(n2);
19             double floor3 = Math.Floor(n3);
20             double floor4 = Math.Floor(n4);
21
22             // Print First values and floor
23
24             Console.WriteLine("value n1 = " + n1);
25             Console.WriteLine("Floor1 values is = " + floor1);
```

```
26
27              // Print 2nd values and floor
28
29              Console.WriteLine("value n2 = " + n2);
30              Console.WriteLine("Floor2 values is = " + floor2);
31
32              // Print 3rd values and floor
33
34              Console.WriteLine("value n3 = " + n3);
35              Console.WriteLine("Floor3 values is = " + floor3);
36
37              // Print 4th values and floor
38
39              Console.WriteLine("value n4 = " + n4);
40              Console.WriteLine("Floor4 values is = " + floor4);
41              Console.Read();
42          }
43      }
44  }
```

លទ្ធផល៖

```
value n1 = 0.2018
Floor1 values is = 0
value n2 = 123.123
Floor2 values is = 123
value n3 = -2.2
Floor3 values is = -3
value n4 = -123.123
Floor4 values is = -124
```

១៧. អនុគមន៍ LOG()

LOG() គឺជាអនុគមន៍ ប្រើសម្រាប់គណនា ឡូការីត Log នៃមួយចំនួន និងចំនួនគោលបានកំណត់។

Syntax:

```
      Math.Log(Double);
      Math.Log(Double, Double);
      Math.Log ( Number,Base );
public static double Log2 (double x);
```

ឧទាហរណ៍៖

```
using System;
namespace ConsoleApplication9
{
    class Program
    {
        // Main Method
        static void Main(string[] args)
        {
```

```
            double[] numbers = {-1, 0, .105, .5, .798, 1, 4, 6.9, 10, 50,
                                100, 500, 1000, Double.MaxValue};
            foreach (double number in numbers)
                Console.WriteLine("The base 10 log of {0} is {1}.",
                                number, Math.Log(number));
            Console.Read();

        }
    }
}
```

លទ្ធផល៖

```
The base 10 log of -1 is NaN.
The base 10 log of 0 is -∞.
The base 10 log of 0.105 is -2.25379492882461.
The base 10 log of 0.5 is -0.693147180559945.
The base 10 log of 0.798 is -0.225646681532328.
The base 10 log of 1 is 0.
The base 10 log of 4 is 1.38629436111989.
The base 10 log of 6.9 is 1.93152141160321.
The base 10 log of 10 is 2.30258509299405.
The base 10 log of 50 is 3.91202300542815.
The base 10 log of 100 is 4.60517018598809.
The base 10 log of 500 is 6.21460809842219.
The base 10 log of 1000 is 6.90775527898214.
The base 10 log of 1.79769313486232E+308 is 709.782712893384.
```

១៨. អនុគមន៍ LOG10()

LOG10() គឺជាអនុគមន៍ ប្រើសម្រាប់គណនា ឡូការីត Log គោល 10 នៃមួយចំនួន។

```
Syntax:

Math.Log10( Number,Base )
```

ឧទាហរណ៍៖

```
1 using System;
2
3 namespace ConsoleApplication7
4 {
5     class Program
6     {
7         static void Main(string[] args)
8         {
9             // to be calculated
10            double a = 4.55;
11            double b = 0;
12            double c = -2.45;
13            double nan = Double.NaN;
14            double positiveInfinity = Double.PositiveInfinity;
```

```
15              double negativeInfinity = Double.NegativeInfinity;
16
17              // Input is positive number so output
18              // will be logarithm of number
19              Console.WriteLine(Math.Log(a));
20
21              // positive zero as argument, so output
22              // will be -Infinity
23              Console.WriteLine(Math.Log(b));
24
25              // Input is negative number so output
26              // will be NaN
27              Console.WriteLine(Math.Log(c));
28
29              // Input is NaN so output
30              // will be NaN
31              Console.WriteLine(Math.Log(nan));
32
33              // Input is PositiveInfinity so output
34              // will be Infinity
35              Console.WriteLine(Math.Log(positiveInfinity));
36
37              // Input is NegativeInfinity so output
38              // will be NaN
39              Console.WriteLine(Math.Log(negativeInfinity));
40              Console.Read();
41          }
42      }
43 }
```

លទ្ធផល៖

```
1.51512723296286
-∞
NaN
NaN
∞
NaN
```

១៩. អនុគមន៍ PI()

Math.Pi() គឺជាអនុគមន៍ ប្រើសម្រាប់គណនា PI ។

Syntax:

```
Math.Pi()
```

ឧទាហរណ៍៖

```
1 using System;
2
3 namespace ConsoleApplication7
```

```
 4 {
 5     class Program
 6     {
 7         static void Main(string[] args)
 8         {
 9             Console.WriteLine(2 * Math.PI);
10             Console.WriteLine(Math.PI);
11             Console.Read();
12         }
13     }
14 }
```

លទ្ធផល៖

```
6.28318530717959
3.14159265358979
```

២០. អនុគមន៍ស្វ័យគុណ POWER()

Math.POWER() គឺជាអនុគមន៍ ប្រើសម្រាប់គណនា ស្វ័យគុណ នៃមួយចំនួន ឬ ស្វ័យគុណជាមួយចំនួនផ្សេងទៀត ។

Syntax:

```
    Math.POWER(Number)
or
   Math.POWER (number1, number2 )
```

ឧទាហរណ៍៖

```
1  // C# program to illustrate the
2  // Math.Pow() function
3  using System;
4  class GFG {
5
6
7      // Main Method
8      static public void Main()
9      {
10
11         // Find power using Math.Pow
12         // 6 is base and 2 is power or
13         // index or exponent of a number
14         double pow_ab = Math.Pow(6, 2);
15
16         // Print the result
17         Console.WriteLine(pow_ab);
18
19         // 3.5 is base and 3 is power or
20         // index or exponent of a number
21         double pow_tt = Math.Pow(3.5, 3);
22
23
24
```

```
            // Print the result
            Console.WriteLine(pow_tt);

            // 202 is base and 4 is power or
            // index or exponent of a number
            double pow_t = Math.Pow(202, 4);

            // Print the result
            Console.WriteLine(pow_t);
        }
    }
```

Output:

```
36
42.875
1664966416
```

២១. អនុគមន៍ RADIANS()

RADIANS() គឺជាអនុគមន៍ ប្រើសម្រាប់បម្លែង ដឺក្រេ Degrees ទៅជា រ៉ាដ្យង់ Radians ។

Syntax:

```
Math.Radians ( Number)
```

ឧទាហរណ៍៖

```
using System;

namespace ConsoleApplication7
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine(Math.Cos(45));

            double res = Math.Cos(Math.PI * 45 / 180.0);
            Console.WriteLine(res);
            Console.Read();
        }
    }
}
```

លទ្ធផល៖

```
0.525321988817 73
0.707106781186548
```

២២. អនុគមន៍ Round ( )

Round( ) Method ក្នុង C# The Math។ method Round( ) ក្នុង C# សម្រាប់បង្គត់តម្លៃមួយទៅចំនួនគត់ជិតបំផុត ឬដល់លេខដែលបានបញ្ជាក់នៃលេខប្រភាគ។ Round( ) Method in C# The Math. Round( ) method in C# rounds a value to the nearest integer or to the specified number of fractional digits

```
Syntax:

        Math.Round ( Number )
```

ឧទាហរណ៍៖

```
using System;

namespace ConsoleApplication9
{
    class Program
    {

        // Main Method
        static void Main(string[] args)
        {
            double[] numbers = {-1, 0, .105, .5, .798, 1, 4, 6.9, 10, 50,
                                100, 500, 1000, Double.MaxValue};
            foreach (double number in numbers)
                Console.WriteLine("The base 10 Round of {0} is {1}.",
                                number, Math.Round(number));

            Console.Read();

        }
    }
}
```

លទ្ធផល៖

```
The base 10 Round of -1 is -1.
The base 10 Round of 0 is 0.
The base 10 Round of 0.105 is 0.
The base 10 Round of 0.5 is 0.
The base 10 Round of 0.798 is 1.
The base 10 Round of 1 is 1.
The base 10 Round of 4 is 4.
The base 10 Round of 6.9 is 7.
The base 10 Round of 10 is 10.
The base 10 Round of 50 is 50.
The base 10 Round of 100 is 100.
The base 10 Round of 500 is 500.
```

```
The base 10 Round of 1000 is 1000.
The base 10 Round of 1.79769313486232E+308 is
1.79769313486232E+308.
```

២៣. អនុគមន៍ SIGN()

SIGN ()គឺជាអនុគមន៍ ប្រើសម្រាប់គណនាតម្លៃតាមសញ្ញ Sign នៃចំនួន តាមលក្ខខណ្ឌ ០ និង ១ ឬ -១។

Syntax:

Math.Sign(*number*)

ដំណើរការរបស់អនុគមន៍មានដូចជា៖

- If number > 0, it returns 1
- If number = 0, it returns 0
- If number < 0, it returns -1

ឧទាហរណ៍៖

```
1  using System;
 2
 3 namespace ConsoleApplication9
 4 {
 5class Program
 6{
 7
 8    // Main Method
 9    static void Main(string[] args)
10    {
11
12        // Decimal data type
13        Decimal de = 150M;
14
15        // Double data type
16        Double d = 34.5432d;
17
18        // Int16 data type
19        short sh = 0;
20
21        // Int32 data type
22        int i = -5678;
23
24        // Int64 data type
25        long l = 598964564;
26
27        // SByte data type
28        sbyte sb = -34;
29
30        // float data type
31        float f = 56.89f;
32
```

```
33
34          // displaying result
35          Console.WriteLine("Decimal: " + de + " " +
eck(Math.Sign(de)));
36          Console.WriteLine("Double: " + d + " " + check(Math.Sign(d)));
37          Console.WriteLine("Int16: " + sh + " " + check(Math.Sign(sh)));
38          Console.WriteLine("Int32: " + i + " " + check(Math.Sign(i)));
39          Console.WriteLine("Int64: " + l + " " + check(Math.Sign(l)));
40          Console.WriteLine("SByte: " + sb + " " + check(Math.Sign(sb)));
41          Console.WriteLine("Single: " + f + " " + check(Math.Sign(f)));
42          Console.Read();
43
44      }
45
46      // function to check whether the input
47      // number is greater than zero or not
48      public static String check(int compare)
49      {
50          if (compare == 0)
51              return "equal to zero";
52
53          else if (compare < 0)
54              return "less than zero";
55
56          else
57              return "greater than zero";
58      }
59  }
60  }
```

លទ្ធផល៖

```
Decimal: 150 greater than zero
Double: 34.5432 greater than zero
Int16: 0 equal to zero
Int32: -5678 less than zero
Int64: 598964564 greater than zero
SByte: -34 less than zero
Single: 56.89 greater than zero
```

២៤. អនុគមន៍ឫសការេ SQRT()

Math.Sqrt() គឺជាអនុគមន៍ ប្រើសម្រាប់ គណនារកតម្លៃ ឫសការេ ។

Syntax:

```
public static double Sqrt(double d);
```

ឧទាហរណ៍៖

```
1 // C# program to illustrate the
2 // Math.Sqrt() method
3 using System;
4
```

```
class GFG {

    // Main Method
    public static void Main()
    {
        double x = 81;

        // Input positive value, Output square root of x
        Console.Write(Math.Sqrt(x));
    }
```

លទ្ធផល៖

```
9
```

ជំពូកទី ៦

ការប្រើប្រាស់ Decision Statement និង Operators

១ . Boolean Variable

Boolean Variable គឺ Variable ដែល store តម្លៃតែពីរប៉ុណ្ណោះគឺ true និង false ។ តម្លៃ boolean ជា Logical មានពីរ ដូចជា៖

ពិត ឬមិនពិត

True ឬ False

Yes ឬ No

On ឬ Off

1 ឬ 0

syntax:

```
if( Conditional 1 >=Conditional 2 )
{
        //statement 1 =Ture; //  Statement1=False;
}
```

ដ្យាក្រាម៖

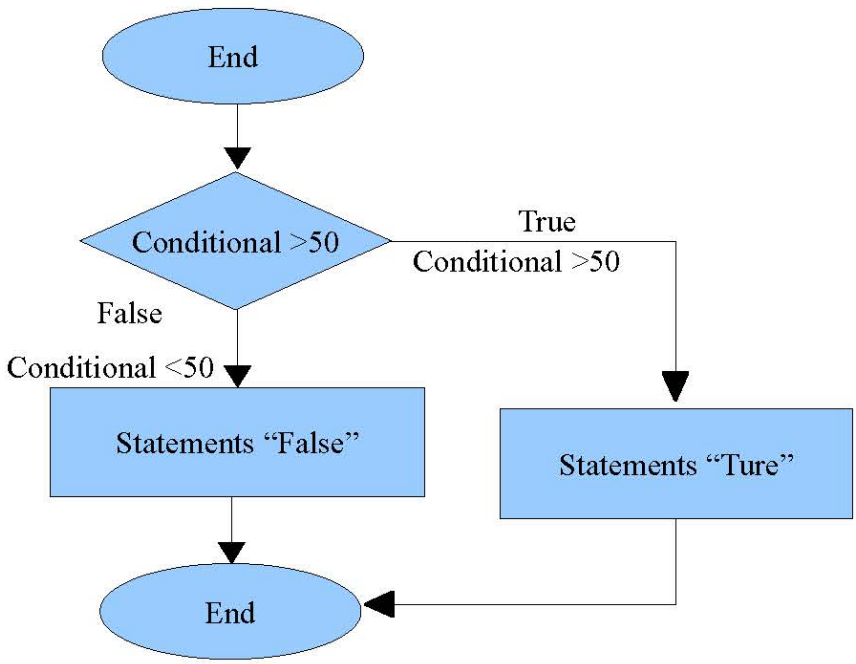


Figure: Boolean Statement

ឧទាហរណ៍៖

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   namespace ConsoleApplication1
8   {
9       class Program
10      {
11          static void Main(string[] args)
12          {
13              bool A=true;
14              bool B=false;
15                  Console.WriteLine(A);//A is true.
16                  Console.WriteLine(B);//B is False.
17                  Console.ReadKey();
18
19               // Outputs A is true.
20               // Outputs B is false.
21          }
22      }
23  }
```

ក្នុងនោះយើងអាចបន្ថែមការប្រើប្រាស់ជាមួយនឹង Boolean Operator ដើម្បីធ្វើការគណនារកមើលតម្លៃ true ឬ false បន្ថែមទៀតបានផងដែរ។

២ Operator Precedence and Associativity:

Table ខាងក្រោមនេះបង្ហាញពីលំដាប់នៃដំណើរការការងាររបស់ Operator នីមួយៗដែលក្នុងនោះ លំដាប់ដំណើរការការងាររបស់វាគឺមានអាទិភាព ចាប់ពីលើចុះក្រោម ហើយ Operators ទាំងឡាយណាដែលនៅក្នុងក្រុមជាមួយគ្នា គឺដំណើរការពីឆ្វេងទៅស្តាំ។

Category	Operators	Description	associativity
Primary	() ++ --	Precedence override	Left
Unary	! + - ++ --	Logical NOT Addition Subtraction Pre-increment Pre-decrement	Left
Multiplicative	* / %	Multiply Divide Division remainder (modulus)	Left

Additive	+ -	Addition Subtraction	Left
Relational	< <= > >=	Less than Less than or equal to Greater than Greater than or equal to	Left
Equality	= = !=	Equal to Not equal to	Left
Conditional AND Conditional OR	&& II	Logical AND Logical OR	Left Left
Assignment	=		Right

២.១ . Equality Operators:

យើងអាចប្រើប្រាស់ Equality Operators ចំនួនពីរដែលមានដូចជា equality (==) និង inequality (!=) ដើម្បីត្រួតពិនិត្យមើល ថាតើ Variable ឬ Expression ទាំងពីរជាមានតម្លៃដូចគ្នាដែរឬទេ។

Operator	Meaning	Example	Outcome if age is 42
==	Equal to	age == 100	False
!=	Not equal to	age !=0	True

២.១.១ . Equality Operators (= =)

យើងអាចប្រើប្រាស់ Equality Operators ស្មើ (= =) ដើម្បីត្រួតពិនិត្យមើល ថាតើ Variable ឬ Expression ទាំងពីរជាមានតម្លៃដូចគ្នាដែរឬទេ។

syntax:

```
if( Conditional 1 ==Conditional 2 )
{
        //statement 1 =True; // Statement1=False;
}
```

ឧទាហរណ៍៖ យើងធ្វើការប្រៀបធៀបគ្នារវាងតម្លៃ A និង B ដែលតម្លៃA និង B មានតម្លៃស្មើគ្នា ។

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   namespace ConsoleApplication2
8   {
9       class Program
10      {
```

```
11            static void Main(string[] args)
12            {
13                int A=10 ;
14                int B = 10;
15
16                if (A == B)
17                {
18                    Console.WriteLine("works is true! ");
19                    Console.ReadKey();
20                }
21
22            }
23        }
24    }
```

២.២. Equality Operators (! =)

យើងអាចប្រើប្រាស់ Equality Operators ផ្ទុយគ្នា ឬមិនស្មើ inequality (!=) ដើម្បីត្រួតពិនិត្យមើល ថាតើ Variable ឬ Expression ទាំងពីរជាមានតម្លៃដូចគ្នាដែរឬទេ។

syntax:

```
if( Conditional 1 !=Conditional 2 )
{
        // statement 1 =True;
        // Statement1=False;
}
```

ឧទាហរណ៍៖ យើងធ្វើការប្រៀបធៀបគ្នារវាងតម្លៃ A និង B ដែលតម្លៃA និង B មានតម្លៃមិនស្មើគ្នា ។

```
 1 using System;
 2 using System.Collections.Generic;
 3 using System.Linq;
 4 using System.Text;
 5 using System.Threading.Tasks;
 6
 7 namespace ConsoleApplication2
 8 {
 9     class Program
10     {
11         static void Main(string[] args)
12         {
13             int A=9 ;
14             int B = 10;
15
16             if (A != B)
17             {
18                 Console.WriteLine("works is true! ");
19                 Console.ReadKey();
```

```
20                  }
21              }
22          }
23      }
24  }
```

៣ . Relational Operators:

Relational Operators គឺប្រើសម្រាប់ធ្វើការត្រួតពិនិត្យរកមើល ថាតើ Variable ឬExpression ទាំងពីរជាមានតម្លៃដូចគ្នាដែរ ឬទេ។ ប្រសិនបើវាត្រួតពិនិត្យលក្ខខណ្ឌរបស់សំណើ ធ្វើការពិនិត្យរហូត ដួបតម្លៃពិត ឬមិនពិត ត្រឹមត្រូវតាមលក្ខខណ្ឌ ទើបវាឈប់ធ្វើការ។

សញ្ញាប្រមាណវិធី	**អត្ថាធិប្បាយ**
==	ប្រើសម្រាប់ត្រួតពិនិត្យ ឬ ប្រៀបធៀបលក្ខខណ្ឌស្មើ
<>	ប្រើសម្រាប់ត្រួតពិនិត្យ ឬ ប្រៀបធៀបលក្ខខណ្ឌខុសពី
>	ប្រើសម្រាប់ត្រួតពិនិត្យ ឬ ប្រៀបធៀបលក្ខខណ្ឌធំជាង
>=	ប្រើសម្រាប់ត្រួតពិនិត្យ ឬ ប្រៀបធៀបលក្ខខណ្ឌធំជាង ឬស្មើ
<	ប្រើសម្រាប់ត្រួតពិនិត្យ ឬ ប្រៀបធៀបលក្ខខណ្ឌតូចជាង
<=	ប្រើសម្រាប់ត្រួតពិនិត្យ ឬ ប្រៀបធៀបលក្ខខណ្ឌតូចជាង ឬ ស្មើ
!=	ប្រើសម្រាប់ប្រៀបធៀបលក្ខខណ្ឌស្មើ ឬ មិនស្មើ

ចំពោះ Relational Operators វិញគឺអាចធ្វើការត្រួតពិនិត្យរកមើល ថាតើ Variable ឬExpression ទាំងពីរជាមានតម្លៃដូចគ្នាដែរ ឬទេ ដោយផ្អែកទៅលើ Operators ចំនួន ដូចខាងក្រោម៖

Operator	Meaning	Example	Outcome if age is 42
<	តូចជាង	age < 21	False
<=	តូចជាង ឬស្មើ	age <=18	False
>	ធំជាង	age > 16	True
>=	ធំជាង ឬស្មើ	age >=30	True
!=	មិនស្មើ	10 != 20	TRUE

លក្ខណៈខុសគ្នារវាង សញ្ញ = ជាមួយនឹង សញ្ញ ==

- សញ្ញ = មានន័យថាគឺជាការ assign ខាងស្តាំតម្លៃទៅឲ្យ Variable នៅខាងឆ្វេង

Ex: x = 5 មានន័យថាបោះតម្លៃ 5 ចូលទៅក្នុង Variable x

- សញ្ញ == មានន័យថាគឺជាការប្រៀបធៀបតម្លៃរបស់ Variable ដែលនៅខាងឆ្វេងជាមួយនឹង

Variable ខាងស្តាំដើម្បីស្វែងរកលទ្ធផល True ឬ False ។

ឧទាហរណ៍៖ x==5 មានន័យថាប្រៀបធៀប x ជាមួយនឹង 5 ថាតើ x មានតម្លៃលេខ 5 ពិតមែនដែរឬទេ?

ឧទាហរណ៍៖ យើងធ្វើការកំណត់ចំណាត់ថ្នាក់របស់សិស្សដែលមានមធ្យមភាគ ធំជាង ៥០ គឺជាប់ និងអ្នកមានមធ្យមភាគតូចជាង ៥០ គឺធ្លាក់ ។

៣.១ ការប្រើប្រាស់ ប្រមាណវិធី តូចជាង (<)

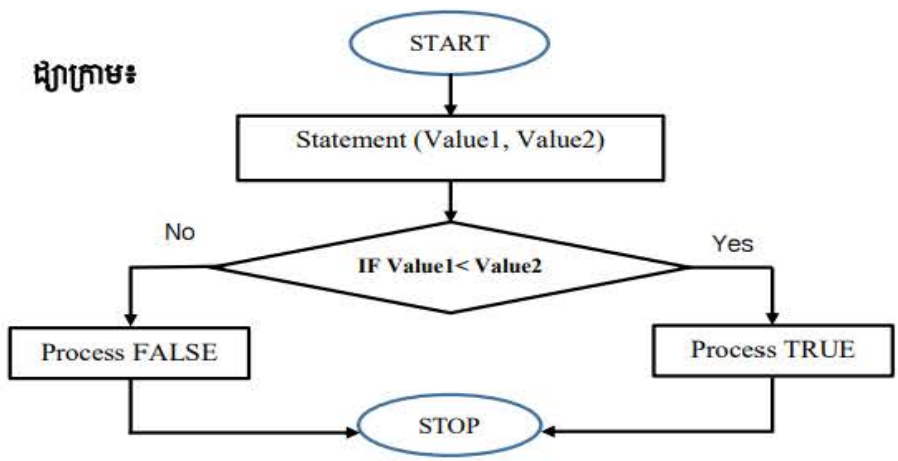
Operator (<) គឺប្រើសម្រាប់ត្រួតពិនិត្យតម្លៃដែលតូចជាង ទៅនឹងសំណើ Statement ជាមួយលក្ខខណ្ឌ Conditional ។ ប្រសិនបើ សំណើ ត្រួតពិនិត្យតម្លៃពីរផ្សេងគ្នា តូចជាង វាផ្តល់តម្លៃពិត បើសិនវាត្រួតពិនិត្យលក្ខខណ្ឌធំជាង វាបង្ហាញតម្លៃ មិនពិត ។

ប្រមាណវិធីតូចជាង សញ្ញា(<) គឺត្រូវបានគេប្រើដើម្បីប្រៀបធៀប លក្ខខណ្ឌ តម្លៃណាដែលតូចជាងជាមួយតម្លៃនៃលក្ខខណ្ឌដទៃទៀត ។

syntax:

```
if(Conditional 1 < Conditional 2)
{
     // statement 1 =True;
     // Statement1=False;
}
```

ដ្យាក្រាម៖




```
1   class Program
2     {
3         static void Main(string[] args)
4         {
5             int A=8 ;
6             int B = 10;
7
8             if (A < B)
9             {
10                Console.WriteLine("Your works is true! ");
11                Console.ReadKey();
12            }
```

```
13
14
15
```

៣.២ ការប្រើប្រាស់ តូចជាងឬស្មើ Operator (<=)

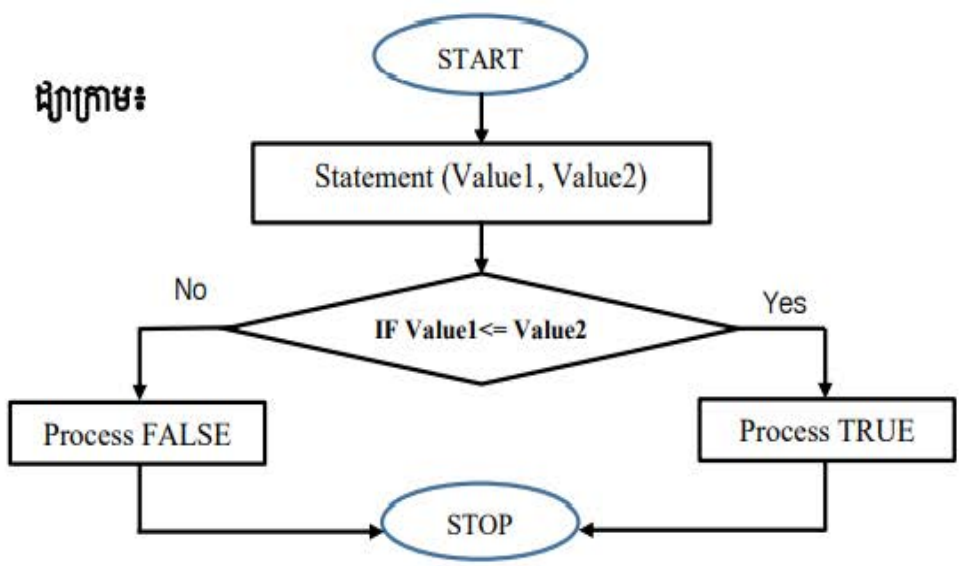
Operator (<=) គឺប្រើសម្រាប់ត្រួតពិនិត្យតម្លៃដែលតូចជាងឬស្មើ ទៅនឹងសំណើ Statement ជាមួយលក្ខខណ្ឌ Conditional ។ ប្រសិនបើ សំណើ ត្រួតពិនិត្យតម្លៃពីរផ្សេងគ្នា តូចជាង វាផ្តល់តម្លៃពិត បើសិនវាត្រួតពិនិត្យលក្ខខណ្ឌធំជាង វាបង្ហាញតម្លៃ មិនពិត ។

ប្រមាណវិធីតូចជាង សញ្ញា(<=) គឺត្រូវបានគេប្រើដើម្បីប្រៀបធៀប លក្ខខណ្ឌ តម្លៃណាដែលតូចជាងឬស្មើ ជាមួយតម្លៃនៃលក្ខខណ្ឌដទៃទៀត ។

syntax:

```
if(Conditional 1 <= Conditional 2)
{
    //statement 1 =True;
    // Statement1=False;
}
```

ដ្យាក្រាម៖




```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace ConsoleApplication2
8 {
9     class Program
10    {
11        static void Main(string[] args)
12        {
```

```
13              int A=8 ;
14              int B = 10;
15
16              if (A <= B)
17              {
18                  Console.WriteLine("Your works is true! ");
19              }
20              else
21              {
22                  Console.WriteLine("Your works is False! ");
23              }
24              Console.ReadKey();
25          }
26      }
27  }
```

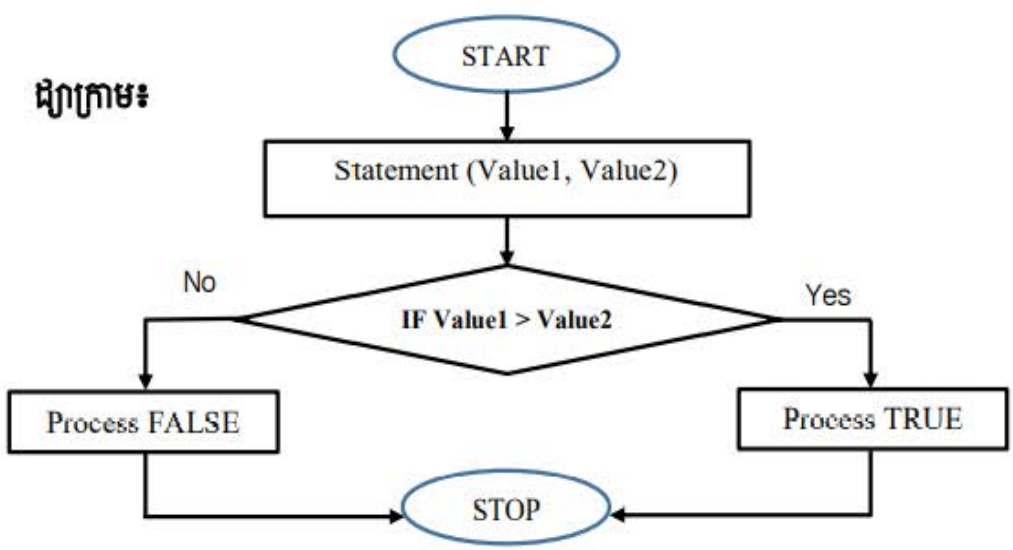
៣.៣ ការប្រើប្រាស់ ប្រមាណវិធី ធំជាង (>)

ប្រមាណវិធីធំជាង សញ្ញា(>) គឺត្រូវបានគេប្រើដើម្បីប្រៀបធៀប លក្ខខណ្ឌ តម្លៃណាដែលធំជាង ជាមួយតម្លៃនៃលក្ខខណ្ឌដទៃទៀត ។

syntax:

```
if(Conditional 1 > Conditional 2)
{
    //statement 1 =True;
    // Statement1=False;
}
```

ដ្យាក្រាម៖




```
1   class Program
2     {
3         static void Main(string[] args)
```

```
            {
                int A=18 ;
                int B = 10;

                if (A > B)
                {
                    Console.WriteLine("Your works is true! ");
                    Console.ReadKey();
                }

            }
        }
```

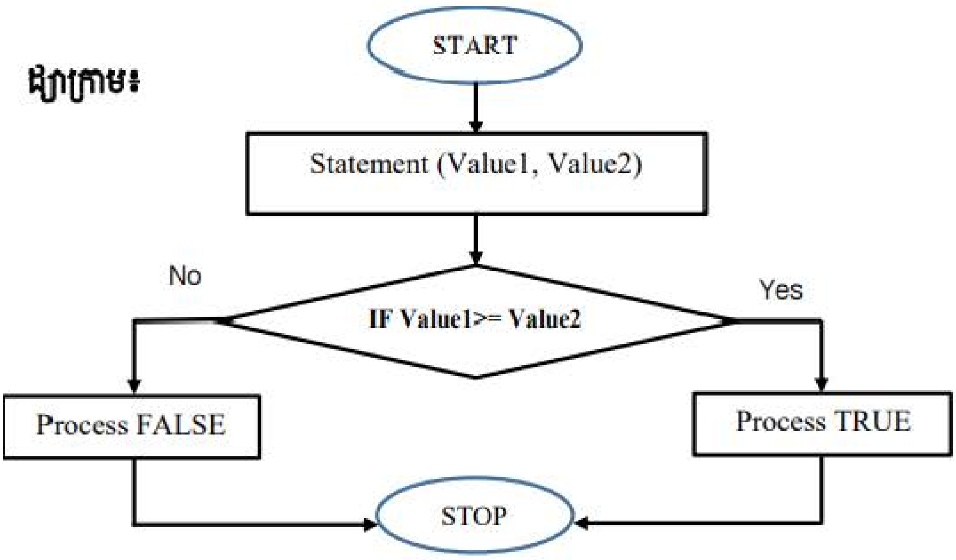
៣.៤ ការប្រើប្រាស់ ប្រមាណវិធី ធំជាង ឬស្មើ (>=)

ប្រមាណវិធីធំជាង ឬស្មើ សញ្ញា(>=) គឺត្រូវបានគេប្រើដើម្បីប្រៀបធៀប លក្ខខណ្ឌ តម្លៃណាដែលធំជាង ឬស្មើ ជាមួយតម្លៃនៃលក្ខខណ្ឌដទៃទៀត ។

syntax:

```
if(Conditional 1 >= Conditional 2)
{
    //statement 1 =True;
    // Statement1=False;
}
```

ដ្យាក្រាម៖



ឧទាហរណ៍៖

```
class Program
  {
      static void Main(string[] args)
      {
```

```
 5              int A=18 ;
 6              int B = 10;
 7
 8              if (A >= B)
 9              {
10                  Console.WriteLine("Your works is true! ");
11                  Console.ReadKey();
12              }
13
14          }
15      }
```

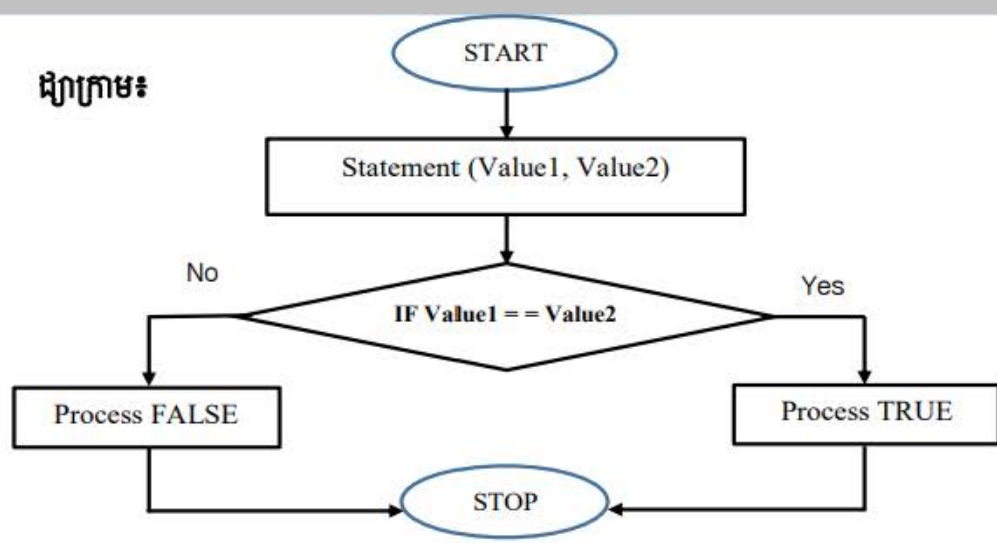
៣.៥ ការប្រើប្រាស់ ប្រមាណវិធី ស្មើ(==)

ប្រមាណវិធី ស្មើ សញ្ញា(==) គឺត្រូវបានគេប្រើដើម្បីប្រៀបធៀប លក្ខខណ្ឌ តម្លៃណា ដែលស្មើ និង តម្លៃលក្ខខណ្ឌ ។

syntax:

```
    if(Conditional 1 == Conditional 2)
    {
        //statement 1 =True;
        // Statement1=False;
    }
```

ដ្យាក្រាម៖



ឧទាហរណ៍៖

```
1    class Program
2      {
3          static void Main(string[] args)
4          {
5              int A=10 ;
6              int B = 10;
7
8              if (A == B)
```

```
 9              {
10                  Console.WriteLine("Your works is true! ");
11                  Console.ReadKey();
12              }
13
14          }
15      }
```

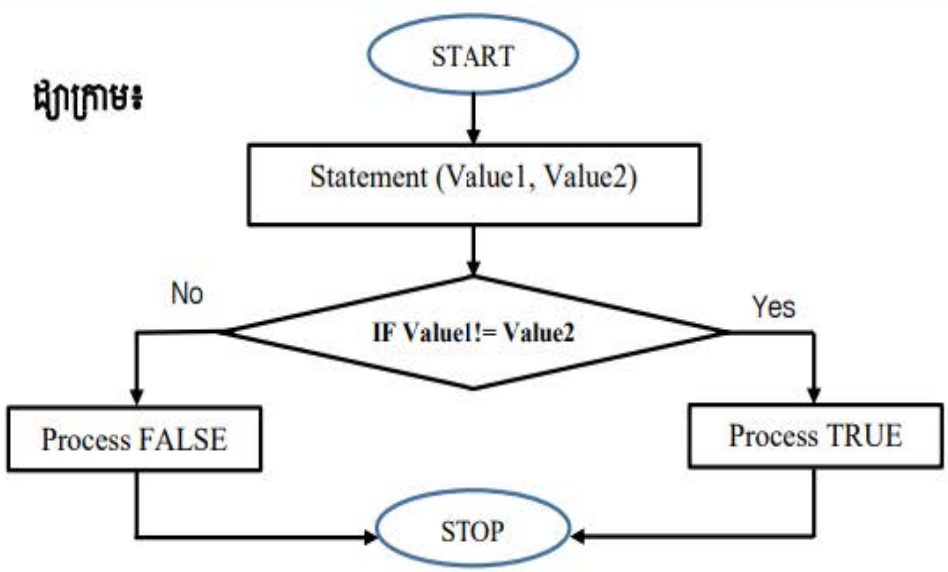
៣.៦ ការប្រើប្រាស់ ប្រមាណវិធី មិនស្មើ ឬផ្ទុយ(!=)

ប្រមាណវិធី មិនស្មើ ឬផ្ទុយ សញ្ញា(!=) គឺត្រូវបានគេប្រើដើម្បីប្រៀបធៀប លក្ខខណ្ឌ តម្លៃណា ដែលមិនស្មើ ឬផ្ទុយ និងតម្លៃលក្ខខណ្ឌ ។

syntax:

```
    if(Conditional 1 != Conditional 2)
    {
        //statement 1 =True;
        // Statement1=False;
    }
```

ដ្យាក្រាម៖



ឧទាហរណ៍៖

```
1     class Program
2       {
3           static void Main(string[] args)
4           {
5               int A=10 ;
6              int B = 10;
7
8              if (A != B)
9              {
```

```
10                  Console.WriteLine("Your works is true! ");
11              }
12              else
13              {
14                  Console.WriteLine("Your works is False! ");
15              }
16              Console.ReadKey();
17
18          }
19      }
20  //Output: Your works is False!
21
```

៤ . ការប្រើលក្ខន័ណ្ឌក្នុងប្រមាណវិធីតក្កវិទ្យា Condition Logical Operators

Condition Logical Operator ដែលមានដូចជា And Operator (&&) និង Or Operator (||) ដែលវាត្រូវបានប្រើប្រាស់ដើម្បីភ្ជាប់ជាមួយនឹង Comparison Operator ឬប្រើក្នុងលក្ខណៈដទៃទៀត ដើម្បីធ្វើការ ស្វែងរកលទ្ធផល ឬមិន ពិត។

Operator	Meaning
&	និង
|	ឬ
||	ឈ្នាប់ឬ
^	ប្រសព្វ
&&	ឈ្នាប់និង
!	ទទេ
&=	និងស្មើ
|=	ឬស្មើ
^=	ប្រសព្វស្មើ
==	ស្មើនឹង
!=	មិនស្មើ
?:	ប្រសិនបើ ពិត ឬមិនពិត if ...else

៤.១ ការប្រៀបធៀបអំពី && (And):

តម្លៃទី 1	ឈ្នាប់	តម្លៃទី 2	លទ្ធផល
True (ពិត)	**&&**	**True (ពិត)**	**True (ពិត)**
True (ពិត)	&&	False (មិនពិត)	False (មិនពិត)

False (មិនពិត)	&&	True (ពិត)	False (មិនពិត)
False (មិនពិត)	&&	False (មិនពិត)	False (មិនពិត)

Conditional AND (&&) Operator

Operatorតាមលក្ខខណ្ឌ និង (&&)

ឧបមាថាយើងចង់ធានាថានៅចំណុចណាមួយនៅក្នុងកម្មវិធីមួយថាលក្ខខណ្ឌពីរគឺជាការពិតមុនពេលយើងជ្រើសរើសផ្លូវជាក់លាក់នៃការប្រតិបត្តិ។ ក្នុងករណីនេះ យើងអាចប្រើ && (លក្ខខណ្ឌ AND) ដូចខាងក្រោម៖

```
if ( gender == 'F' && age >= 65 )
        ++seniorFemales ;
```

ឧទាហរណ៍៖

```
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)
        {
           int Score = 50;
            if (Score >= 0 && Score <=100)
              {
                  Console.WriteLine(Score);// Score 50 is true.
                  Console.ReadKey();
              }

            // Output is Score 50
        }
    }
}
```

៤.២ ការប្រៀបធៀបអំពី || (Or):

តម្លៃទី 1	ឈ្នាប់	តម្លៃទី 2	លទ្ធផល
True (ពិត)	**||**	**True (ពិត)**	**True (ពិត)**
True (ពិត)	||	False (មិនពិត)	True (ពិត)
False (មិនពិត)	||	True (ពិត)	True (ពិត)
False (មិនពិត)	||	False (មិនពិត)	False (មិនពិត)

ឧទាហរណ៍៖

```
static void Main(string[] args)
 {
    int Score = 50;
     if (Score < 0 && Score < 100)
       {
           Console.WriteLine(Score);// Score 50 is ture.
           Console.ReadKey();
       }

     // Output is Score 50
 }
```

៥. ការប្រើប្រាស់ Arithmetic Operator

Arithmetic Operator គឺជាសញ្ញាគណនាជាមួយនឹង ផ្នែកគណិតវិទ្យា ដើម្បីរកតម្លៃលទ្ធផលនៃការគណនាណាមួយ។ វាមានដូចជា + - * / ។ ចំពោះតម្លៃ ឬ Variable ដែលត្រូវប្រើប្រាស់ជាមួយនឹង Operator ដើម្បីធ្វើការគណនាត្រូវបានហៅថា Operand ។ យើងអាចប្រើប្រាស់ Arithmetic Operator ទាំងអស់ជាមួយនឹងតម្លៃ char, int, long, float, double, ឬ decimal? ក្នុងនោះសញ្ញា + ក៏អាចប្រើប្រាស់បានជាមួយនឹង string បានផងដែរ ។

ប្រមាណវិធី	ឈ្មោះ ប្រមាណវិធី	ឧទាហរណ៍
+	បូក	x+y
-	ដក	x-y
*	គុណ	x*y
/	ចែក	X /y
%	ចែកយកសំណល់	

៦. If Statement

If Statement គឺត្រូវបានប្រើប្រាស់ដើម្បីធ្វើការឬ execute នូវ block នៅពេលដែលលក្ខខណ្ឌរបស់វាពិត។

```
if(booleanExpression)
   {
      Statement_1;
   }
else
   {
     Statement_2;
   {
```

- if គឺជា Keyword សម្រាប់ប្រើប្រាស់ដើម្បីដាក់លក្ខខណ្ឌនៅក្នុង Source Code
- boolean-Expression គឺជាតម្លៃ ឬ expression ដែលប្រើភ្ជាប់ជាមួយនឹង Comparison Operator ដើម្បីស្វែងរកលទ្ធផល True
- statement-1 គឺជាបណ្តុំនៃ Code ដែលត្រូវធ្វើការនៅពេលដែល condition ទទួលបានតម្លៃ True ប៉ុន្តែបណ្តុំនៃ Code នឹងត្រូវបានរំលងចោលប្រសិនបើ Boolean Expression នៅក្នុង if ផ្តល់លទ្ធផល False វិញ។
- else គឺជា Keyword សម្រាប់ប្រើប្រាស់ដើម្បីដាក់លក្ខខណ្ឌនៅក្នុង Source Code បំបន្ទាប់ពី if ដើម្បីធ្វើការ នៅពេលដែល if ទទួលបាន Boolean Expression ជាតម្លៃ False ។
- statement-2 គឺជាបណ្តុំនៃ Code ដែលត្រូវធ្វើការនៅពេលដែល Boolean Expression ទទួលបានតម្លៃ False ប៉ុន្តែបណ្តុំនៃ Code នឹងត្រូវបានរំលងចោលប្រសិនបើ Boolean Expression នៅក្នុង if ផ្តល់លទ្ធផល True វិញ។ ចំពោះ else Keyword គឺ Optional មានន័យថា ពុំចាំបាច់ដាក់បន្ទាប់ពី if ក៏បាន។

```
static void Main(string[] args)
      {
          int Score = 50;
          if (Score == 0)
            {
                Console.WriteLine(Score);// Score =0 is false.
                Console.ReadKey();
            }
          else
          {
              Console.WriteLine(Score);// Score = 50 is ture.
              Console.ReadKey();
          }

          //Score 50
      }
```

ប្រសិនបើយើងប្រើ Boolean Variable ធ្វើជា Boolean-Expression វិញនោះ យើងអាចប្រើទម្រង់កាត់ដូចខាងក្រោម៖

```
bool hello;
if(hello == true)//ok, but is not commonly used;
   {
       Statement_1;
    }
if(hello) //better;
    {
      Statement_2;
    {
```

ក្នុងករណីដែល statement ក្នុង if ត្រូវបានសរសេរចាប់ពី 2 ជួរឡើងគឺយើងអាចប្រើប្រាស់ braces { } ដើម្បីកំណត់ Block នៃ Code ដែលត្រូវធ្វើការ។

៧ . Control Statements

Control Statements ត្រូវគេប្រើប្រាស់សម្រាប់ត្រួតពិនិត្យ ទៅលើដំណើរការរបស់Statements នៅពេលដែល Program កំពុងដំណើរការ ។ ជាទូទៅការពិនិត្យលក្ខខណ្ឌនៅក្នុង Statements មួយគឺយើងចង់ដឹងពីសកម្មភាពដំណើរការរបស់វា ដែលបានអនុវត្តន៍ និងធ្វើការទៅលើលក្ខខណ្ឌ ។ នៅក្នុងភាសា c# ការប្រើប្រាស់លក្ខខណ្ឌដើម្បីត្រួតពិនិត្យមានពីរប្រភេទពីរ គឺលក្ខខណ្ឌពិត (True) និងមិនពិត (False) ។

៧.១ Selection/Conditional Statement

Conditional Statement គឺជា Structure ដែលត្រូវបានគេប្រើប្រាស់សម្រាប់ធ្វើការបញ្ជូឱ្យប្រព័ន្ធដំណើរការ និងធ្វើការទៅតាមលក្ខខ័ណ្ឌ ។ Structure សម្រាប់ Conditional Statement មាន If statement , If else Statement និង Switch Statement ។

Statements	Description
If statement	If Statement គឺជាពិនិត្យសំណើកន្សោមលក្ខខណ្ឌ នៃតម្លៃ Boolean ចាប់ពីមួយ ឬច្រើនលក្ខខ័ណ្ឌ ។
If else Statement	An if statements can be followed by an optional else statements, Which executes when the boolean expression is false.
Switch statements	Switch Statements គឺវាធ្វើដំណើរការត្រួតពិនិត្យទៅតាម តម្លៃអថេរ(Values) ដើម្បីធ្វើតេស្តសម្រាប់ស្វែងរកតម្លៃ របស់សំណើ នៃ statement វាធ្វើប្រៀបធៀបម្តងមួយៗ រហូតទល់អស់នៃ សំណើលក្ខខណ្ឌរបស់ statements ទើបវាបញ្ចប់ដំណើរការ។

៧.២ If Condition

If គឺវាធ្វើការត្រួតពិនិត្យលក្ខខណ្ឌរបស់ សំណើ(Statement) បើសិនពិនិត្យសំណើពិតវាអនុវត្តធ្វើការ Statements នោះរហូតពិនិត្យដល់អស់សំណើវា នៅពេលដែលវាពិនិត្យទៅឃើញលក្ខខណ្ឌមិនពិត វាមិនធ្វើការ Statements បន្តទៀតទេ ហើយវានឹងចាក់ចេញ (output) ។

syntax:

```
If( Boolean_expression )
{
```

```
        statements; // will execute if the Boolean expression in true
    }
```

If statement (Boolean_expression) វាត្រួតពិនិត្យតម្លៃសំណើលក្ខខណ្ឌ របស់ statements ឃើញ ពិត (True) វានឹងបញ្ចប់ធ្វើការ ពិនិត្យមើល ហើយវាបង្ហាញតម្លៃ (Output) ។ បើសិន វាត្រួតពិនិត្យឃើញខុសលក្ខខណ្ឌ មិនពិត (False) វាធ្វើការអនុវត្តពិនិត្យលក្ខខណ្ឌ ម្តងមួយៗរហូត ទល់អស់ នៃសំណើ statement ទើបវាបញ្ចប់ដំណើរការរបស់វា។

ដ្យាក្រាម

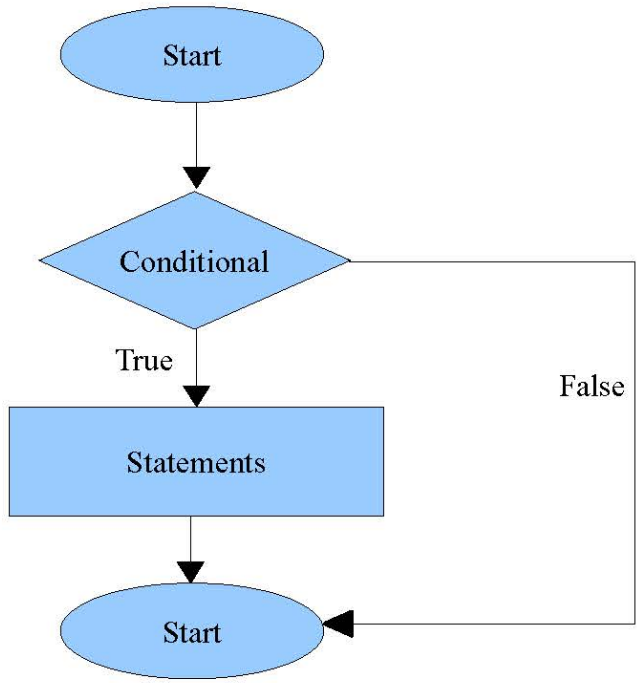


Figure: If Statement

ឧទាហរណ៍៖ if Statement

```
using System;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            int x = 20;
            int y = 18;
            if (x > y)
            {
                Console.WriteLine("x is greater than y");
                Console.ReadKey();
            }
```

```
        }
    }
}
```

៧.៣ If... else Statement

យើងប្រើប្រាស់ លក្ខខ័ណ្ឌ else statement ចំពោះ block of code ដើម្បីត្រួតពិនិត្យលក្ខខណ្ឌ របស់ សំណើ statements ប្រសិនបើ វាពិនិត្យខុសពីលក្ខខណ្ឌ មិនពិត condition is False។

syntax:

```
if (condition)
  {
    // block of code to be executed if the condition is True
  }
    else
 {
   // block of code to be executed if the condition is False
 }
```

ដំណើរការ របស់ វា គឺត្រួតពិនិត្យលក្ខខណ្ឌរបស់សំណើ statements ពិតកាលណាវាពិនិត្យត្រូវ លក្ខខណ្ឌបង្ហាញពិត ប្រសិនបើវាពិនិត្យលក្ខខណ្ឌខុសពីលក្ខខណ្ឌ វាអនុវត្តន៍លក្ខខណ្ឌ Else statement ។

ដ្យាក្រាម

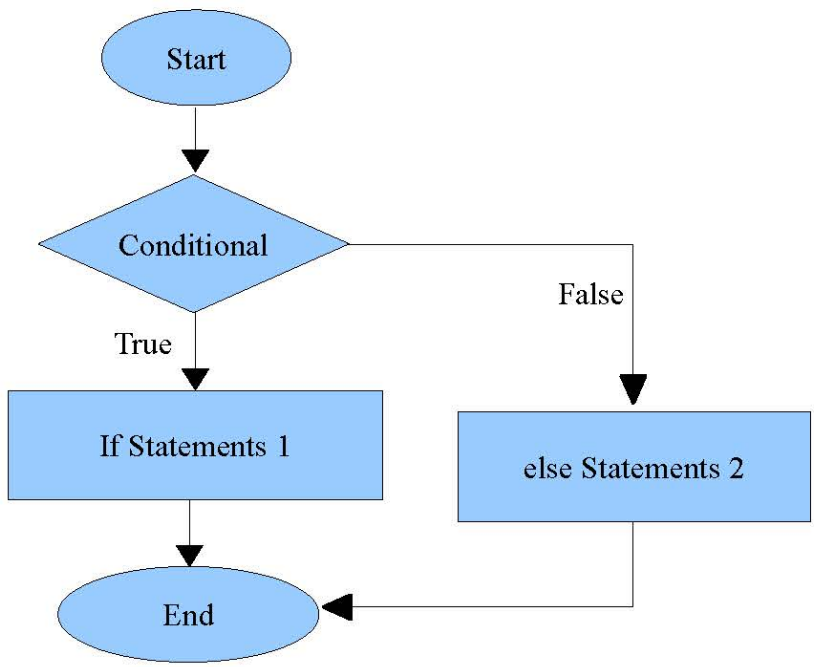


Figure: else Statement

ឧទាហរណ៍៖ else statement

```
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)
        {
            int A = 20;
            if (B < 18)
            {
                Console.WriteLine("Good day.");
            }
            else
            {
                Console.WriteLine("Good evening.");
                Console.ReadKey();
            }
            // Outputs "Good evening."
        }
    }
}
```

លទ្ធផល៖

```
Good evening.
```

៧.៤ If....else if....else Statement

If គឺត្រួតពិនិត្យលក្ខខណ្ឌរបស់សំណើ statement បើសិន statement ពិតនោះវាអនុវត្តធ្វើរបស់ statement របស់វា រួចចេញពីលក្ខខណ្ឌ If statement ។ ប្រសិនបើ វាត្រួតពិនិត្យទៅមិនពិតទេ គឺវាទៅពិនិត្យលក្ខខណ្ឌផ្សេងទៀតរបស់សំណើនៃ else if statement បន្ទាប់ហូតអស់លក្ខខណ្ឌ ។ បើសិន វាពិនិត្យនូវលក្ខខណ្ឌទៅខុសពីលក្ខខណ្ឌ នៃ Statement គឺអនុវត្តន៍ លក្ខខណ្ឌ else បន្ទាប់មកវាចាក់ចេញ ពីលក្ខខណ្ឌ ។

ដ្យាក្រាម

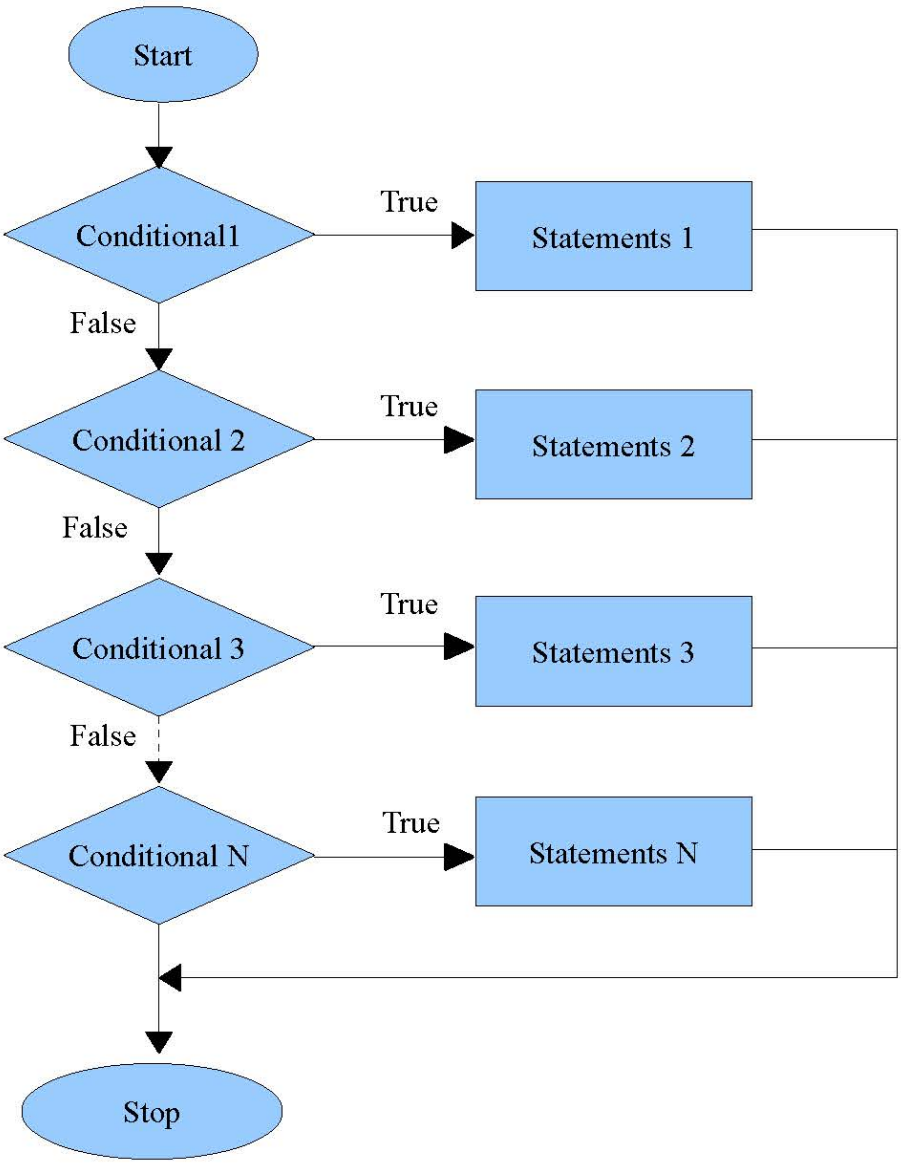


Figure: If ….else if Statement

ឧទាហរណ៍៖ if...else if statement

syntax:

```
if (condition1)
{
  // statement 1 to be executed if condition1 is True
}
else if (condition2)
{
  // statement 2 to be executed if the condition1 is false and
     condition2 is True
}
```

```
else
{
     //statement 3 to be executed if the condition1 is false and
     condition2 is False
}
```

ឧទាហរណ៍៖ If ....else if statement

```
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)
        {
            int A = 22;
            if (A < 10)
            {
                Console.WriteLine("Good morning.");
                Console.ReadKey();
            }
            else if (A < 20)
            {
                Console.WriteLine("Good day.");
                Console.ReadKey();
            }
            else
            {
                Console.WriteLine("Good evening.");
                Console.ReadKey();
            }
            // Outputs "Good evening."
        }
    }
}
```

៧.៥ Switch statement

`Switch statement` ដំណើរការត្រួតពិនិត្យគ្រប់តម្លៃរបស់អថេរ ទាំងអស់ របស់ `Case` ណាមួយពិតនោះវា `select statement` នោះបង្ហាញ ។

នេះជារបៀបដែលវាដំណើរការ :

- The `switch` expression is evaluated once
- The value of the expression is compared with the values of each `case`

•If there is a match, the associated block of code is executed
•The break and default keywords will be described later in this chapter

Syntax:

```
switch(expression)
{
  case x:
    // code block "Statement 1"
    break;
  case y:
    // code block "Statement 2"
    break;
  default:
    // code block "Statement 3"
    break;
}
```

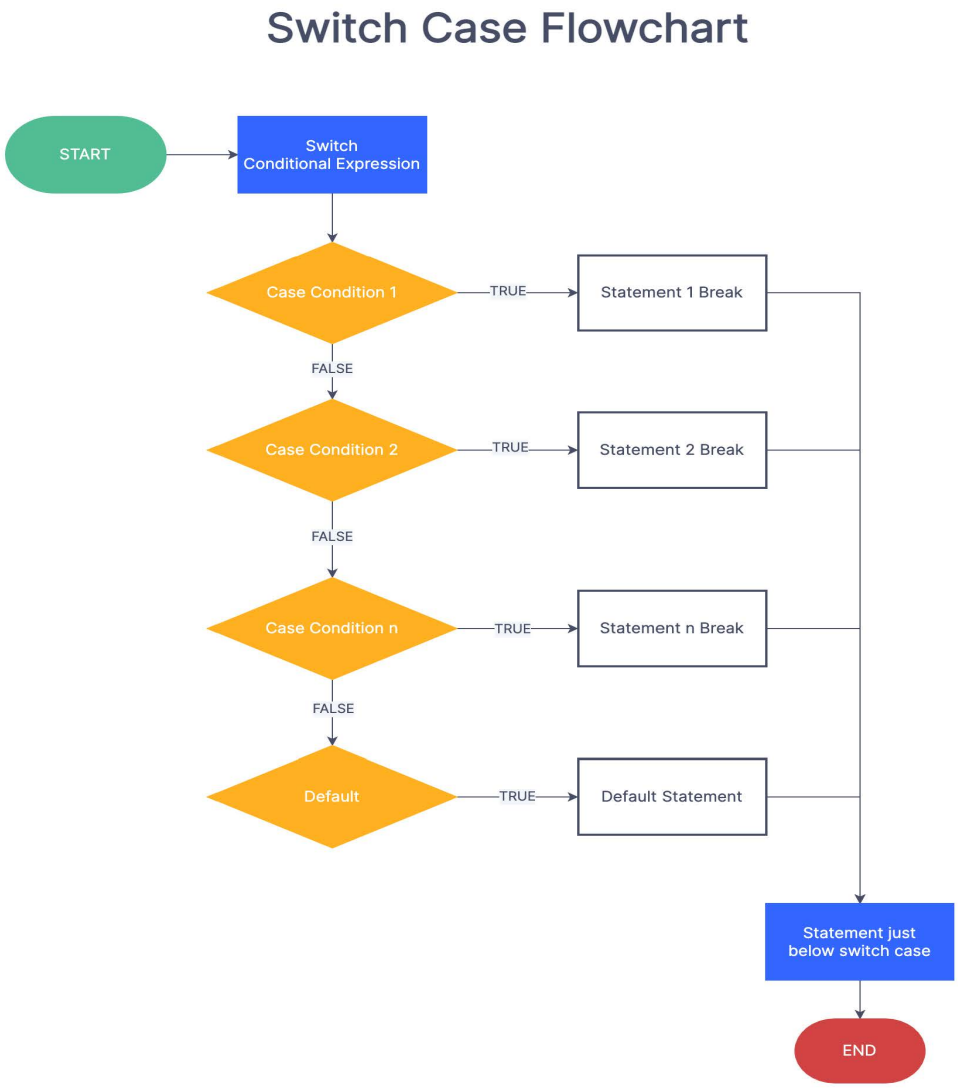


Figure : Switch statement

ឧទាហរណ៍:

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   namespace ConsoleApplication1
8   {
9       class Program
10      {
11          static void Main(string[] args)
12          {
13          int Number=2;
14             switch(Number)
15              {
16                 case 1:
17                      Console.WriteLine("One");
18                     break;
19                 case 2:
20                     Console.WriteLine("Two");
21                     break;
22                 case 3:
23                     Console.WriteLine("Three");
24                     break;
25                 case 4:
26                     Console.WriteLine("Four");
27                     break;
28              }
29             Console.ReadKey();
30          }
31      }
32  }
33  //Output Two
```

៧.៦ goto statement

goto statement អនុញ្ញាតឱ្យដំណើរការនៃ statement ទៅកាន់ទីតាំងណាមួយដោយគ្មានលក្ខខណ្ឌ តាមរយៈឈ្មោះ Label ឬតាមរយៈការកំណត់ លក្ខខណ្ឌជាក់លាក់ របស់ Variable។

Syntax:

```
label:
 if (Conditional)
 {
    //statement
    goto label;
  }
```

ដ្យាក្រាម:

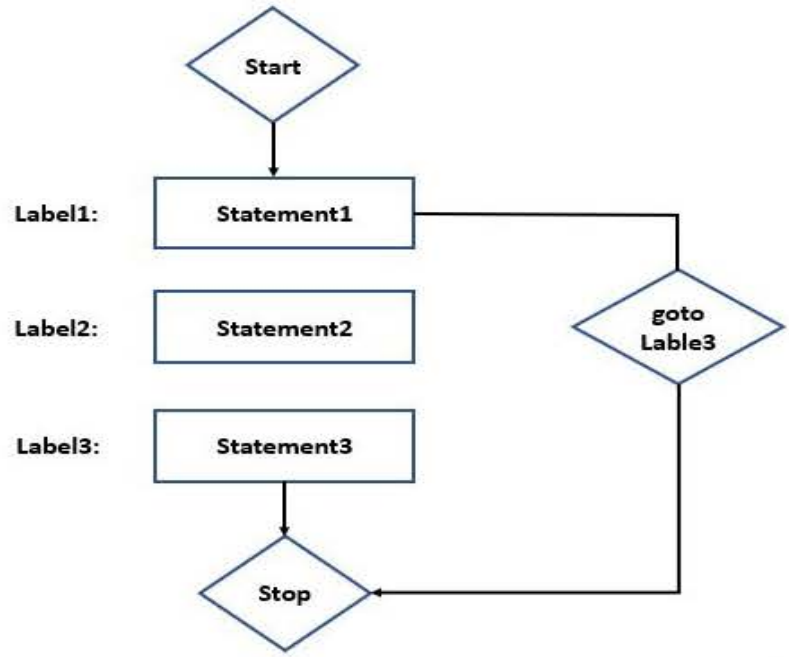


Figure: Goto Statement

ឧទាហរណ៍:

```
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)
        {
        Saron:
            int age = 18;
            if (age < 18)
            {
                goto Saron;

            }
            Console.WriteLine(age);
            Console.ReadKey();
        }
        //Output 18;
    }
}
```

៧.៧ Loop statement

នៅភាសា C# ការប្រើប្រាស់loop ក្នុងការសរសេរកម្មវិធីដោះស្រាយបញ្ហាអ្វីមួយ ឬធ្វើការអ្វីមួយ មានលក្ខណៈដូចគ្នាដដែល ច្រើនដង ហៅថាLoop ។ Loop គឺជាការធ្វើត្រួតពិនិត្យ លក្ខខណ្ឌដដែលៗ រហូតជួបលក្ខខណ្ឌ ពិត ឬជាក់លាក់ របស់សំណើ របស់ Statement ទើបវាបញ្ចប់ ដំណើរការរបស់វា។ ភាសា C# Program Loop មានបីគឺ៖

- while loop
- do loop
- for loop

៧.៧.១ while loop

while loop គឺវាធ្វើអនុវត្តន៍ដំណើរការ ធ្វើការត្រួតពិនិត្យ លក្ខខណ្ឌ ដដែលៗ រហូតវា ជួបលក្ខខណ្ឌមិនពិត ទើបវាបញ្ចប់ដំណើរការរបស់វា ។ បើវាពិនិត្យលក្ខខណ្ឌ ជួបនូវលក្ខខណ្ឌពិត រហូតវានៅតែ Loop នៃលក្ខខណ្ឌ ដដែលៗ រហូតគ្មានទីបញ្ចប់ ។

Syntax:

```
while (condition)
{
   // statement;
}
```

ដ្យាក្រាម៖

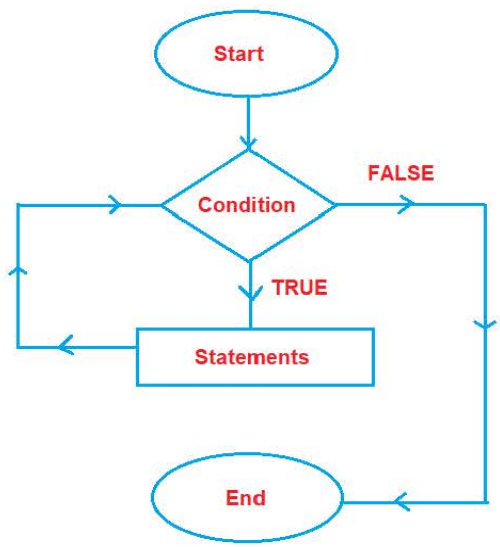


Figure: While Loop diagram

ឧទាហរណ៍៖

```
int i = 0;
while (i < 5)
{
  Console.WriteLine(i);
  i++;
}
```

លទ្ធផល៖

```
i=0
i=1
i=2
i=3
i=4
```

៧.៧.២ do loop statement

do loop គឺវាដំណើរការ ចាប់ផ្តើមដំបូងត្រួតពិនិត្យលក្ខខណ្ឌ របស់ Statement ប្រសិនបើលក្ខខណ្ឌនោះពិត វានឹងអនុវត្ត Process លក្ខខណ្ឌរហូតដល់លក្ខខណ្ឌមិនពិត ទើបវាបញ្ចប់ដំណើរការ។

```
Syntax:

    do
     {
          //code block statement;

     } while(condition);
```

ដ្យាក្រាម៖

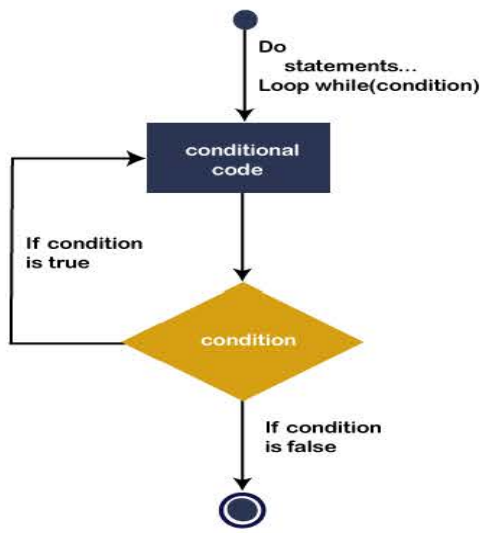


Figure: Diagram Do while loop

```
static void Main(string[] args)
     {
        int i = 0;
         do
         {
              Console.WriteLine("i = {0}", i);
              i++;
         } while (i < 5);
         }
              Console.ReadKey();
      }
```

លទ្ធផល៖

```
i=0
i=1
i=2
i=3
i=4
```

៧.៧.៣ for loop statement

for គឺត្រូវបានប្រើប្រាស់ដើម្បីធ្វើការដំណើរការនូវ Block នៃ Code ដដែលៗ ទៅតាមចំនួនដែលបានកំណត់យ៉ាងត្រឹមត្រូវនៅពេលដែលលក្ខខណ្ឌរបស់ True ។ ចំពោះ while និង do while គឺធ្វើគិតទៅលើ លក្ខខណ្ឌ ដោយមិនគិតពីចំនួនដងឡើយ ប៉ុន្តែ for វិញគឺធ្វើឲ្យគិតទៅលើចំនួនដង។ ជាទូទៅចំពោះការ loop គេនិយមប្រើប្រាស់ for ពីព្រោះយើងអាចដឹងពីចំនួនដែលវាត្រូវធ្វើការនៅក្នុង loop ។

```
Syntax:  ❶                ❷                ❹
for( initialization; Boolean expression; update control variable)
    {          ❸
        statement;
    }
```

1.Initialization: គឺជាតម្លៃចាប់ផ្តើមដំណើរការ loop ហើយវាធ្វើការតែម្តងប៉ុណ្ណោះ។

2.Boolean expression: គឺជាលក្ខណ្ឌដែលត្រូវត្រួតពិនិត្យប្រសិនបើ True loop នឹងបន្តដំណើរការ ប៉ុន្តែប្រសិនបើ False វិញនោះ Loop និងបញ្ឈប់ដំណើរការ។

3.statement: គឺជា Block នៃ code ដែលត្រូវដំណើរការក្នុង braces { } នៅពេលដែល លក្ខខណ្ឌ True

4.update control variable: គឺជាការដំឡើង ឬបន្ថយ value របស់ variable នៅក្នុង ឲ្យកើនឡើងIncrement ឬថយចុះ Decrement ហើយបន្ទាប់មកវានឹងត្រលប់ទៅដំណើរការនៅក្នុងតំបន់ Condition វិញ។

ដ្យាក្រាម៖

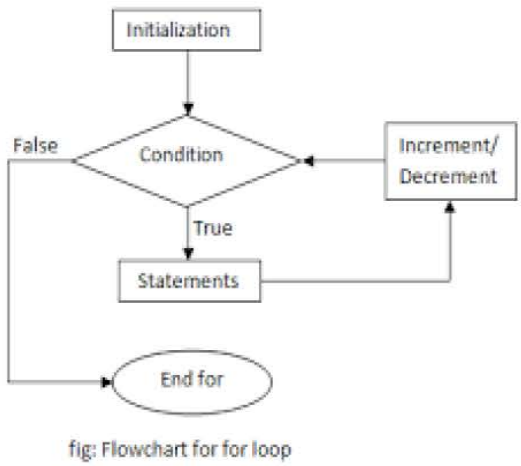
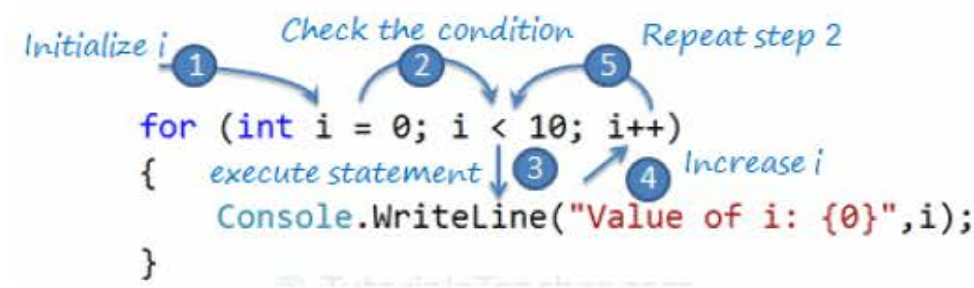


fig: Flowchart for for loop

ឧទាហរណ៍៖

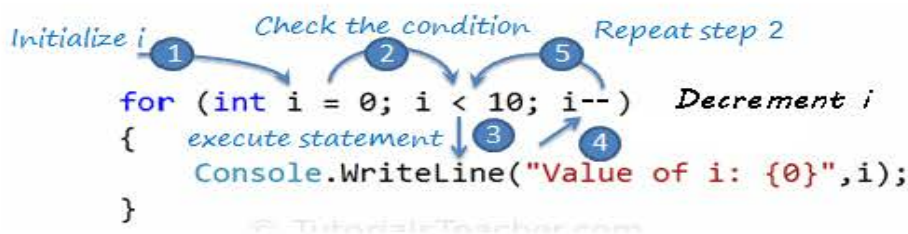



```
static void Main(string[] args)
      {
       for(int i = 0; i < 10; i++)
          {
               Console.WriteLine("Value of i: {0}", i);
          }
               Console.ReadKey();
       }
```

លទ្ធផល៖

```
Value of i: 0
Value of i: 1
Value of i: 2
Value of i: 3
Value of i: 4
Value of i: 5
Value of i: 6
Value of i: 7
Value of i: 8
Value of i: 9
```

ឧទាហរណ៍៖




```
static void Main(string[] args)
      {
       for(int i = 10; i >= 0; i--)
          {
               Console.WriteLine("Value of i: {0}", i);
          }
               Console.ReadKey();
       }
```

លទ្ធផល៖

```
Value of i: 10
```

```
Value of i: 9
Value of i: 8
Value of i: 7
Value of i: 6
Value of i: 5
Value of i: 4
Value of i: 3
Value of i: 2
Value of i: 1
Value of i: 0
```

៧.៧.៤ foreach loop statement

យើងបានសិក្សារួចមកហើយ `foreach loop` ដែលត្រូវបានប្រើទាំងស្រុងដើម្បី `loop` តាមធាតុក្នុង `array` ។

syntax

```
foreach (type variableName in arrayName)
{
  // code block to be executed
}
```

ឧទាហរណ៍ខាងក្រោមបង្ហាញធាតុទាំងអស់នៅក្នុង `cars array` ដោយប្រើ `foreach loop` ៖

Example

```
string[] cars = {"Volvo", "BMW", "Ford", "Mazda"};

foreach (string i in cars)

{
  Console.WriteLine(i);
}
```

៧.៧.៥ Break and continue

អ្នកបានឃើញbreak statement ដែលបានប្រើរួចហើយក្នុងជំពូកមុននៃការបង្រៀននេះ។ វាត្រូវបានប្រើដើម្បី "jump out" នៃ `switch` statement។ `break` statement ក៏អាចត្រូវបានប្រើដើម្បីចាកចេញពី **loop** ផ្ទៃលជុំផងដែរ។

```
for (int i = 0; i < 10; i++)
{
  if (i == 4)
  {
    break;
  }
  Console.WriteLine(i);
}
```

៧.៧.៦ Continue

continue statement breaks ធ្វើម្តងទៀតមួយ (ក្នុង loop រង្វិលជុំ) ប្រសិនបើលក្ខខណ្ឌជាក់លាក់កើតឡើង ហើយបន្តជាមួយការធ្វើម្តងទៀតបន្ទាប់ក្នុង loop រង្វិលជុំ។

```
for (int i = 0; i < 10; i++)
{
  if (i == 4)
  {
    continue;
  }
  Console.WriteLine(i);
}
```

៧.៧.៧ Break and Continue in While Loop

អ្នកប្រើប្រាស់ break និង continue ក្នុង while loops

Break Example

```
int i = 0;
while (i < 10)
{
  Console.WriteLine(i);
  i++;
  if (i == 4)
  {
    break;
  }
}
```

Continue Example

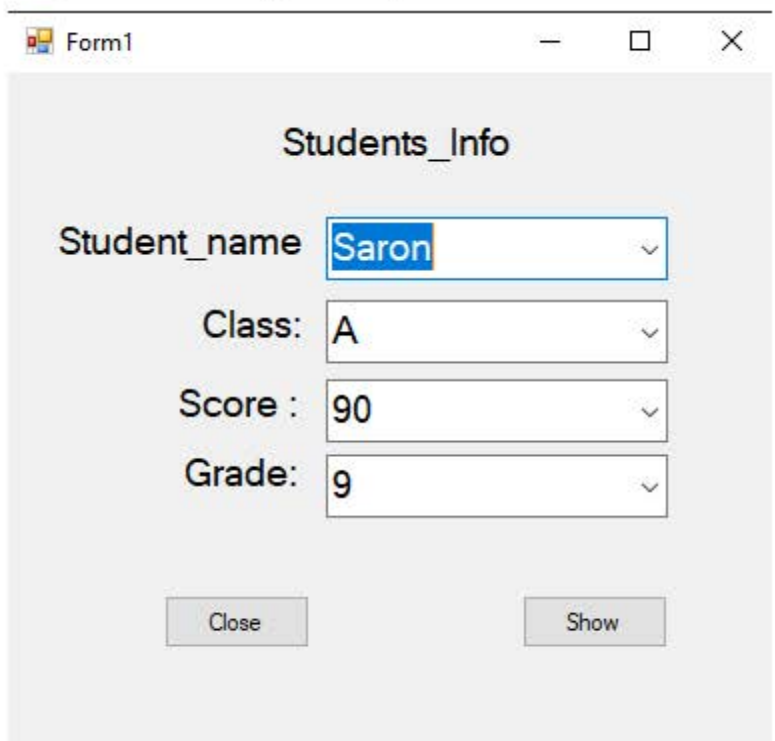
```
int i = 0;
while (i < 10)
{
  if (i == 4)
  {
    i++;
    continue;
```

```
    }
    Console.WriteLine(i);
    i++;
}
```

ជំពូកទី៧

សិក្សាអំពី Array

មេរៀនទាំង៤ដែលលោកអ្នកបានសិក្សាកន្លងមកគឺរាល់ទិន្នន័យត្រូវបានតំណាង ឬរក្សាទុកដោយ variable។ វាជាការពិតណាស់ដែល variables ប្រើសម្រាប់តំណាងឲ្យនូវទិន្នន័យក្នុងប្រភេទណាមួយ ប៉ុន្តែអ្វីដែលអ្នកបានដឹងច្បាស់នោះគឺ variables ទាំងនោះមិនអាចតំណាង ឬរក្សាទិន្នន័យបានលើសពីមួយតម្លៃឡើយក្នុងពេលតែមួយសម្រាប់ variable នីមួយៗ។ សូមមើលឧទាហរណ៍៖



នៅពេលដែលយើងដំណើរការកូដនេះយើងនឹងបានលទ្ធផលដូចរូប ក្រោយពេលដែល ប៊ូតុង Show បានចុច។ យើងឃើញថាគ្រប់ variables ត្រូវបានផ្លាស់ប្តូរតម្លៃ ពីការផ្តល់ឲ្យក្នុងខណៈពេលយើងបានបង្កើតវា ទៅកាន់តម្លៃថ្មីដែលជា តម្លៃដែលបានផ្តល់ឲ្យជាលើកទី២ នេះបញ្ជាក់ថា គ្រប់ variables ដែលយើងបានបង្កើតខាងលើមានលទ្ធភាពអាចផ្ទុក ឬតំណាងបានតែមួយតម្លៃប៉ុណ្ណោះក្នុងពេលតែមួយ។

១. តើអ្វីទៅជា array ?

នៅពេលដែលអ្នកបង្កើត variable ហើយ variable នោះមានលទ្ធភាពក្នុងការតំណាងឲ្យ list នៃទិន្នន័យណាមួយ នោះមានន័យថាអ្នកកំពុងប្រើប្រាស់ array variable។ ជាពិសេសអ្នកត្រូវចងចាំថា បណ្តាទិន្នន័យទាំងឡាយដែលមានប្រភេទទិន្នន័យដូចៗគ្នា ឬអាចនិយាយថាទិន្នន័យនោះមានគោលដៅតំណាង

ឲ្យ list នោះគឺចាំបាច់អ្នកត្រូវប្រើប្រាស់ array។

Array គឺជា variable ពិសេសដែលអនុញ្ញាតឱ្យអ្នក ផ្ទុកនូវតម្លៃបានច្រើននៅក្នុង variable មួយ ។ រាល់តម្លៃនិមួយៗត្រូវបានរក្សាទុកនៅក្នុង index របស់ array ដែលអាច ជាលេខ ឬ ជា តួអក្សរ ។ ជា default ធាតុរបស់ array ដែលជា index គឺចាប់ផ្តើមពីលេខ 0 ។

ប្រសិនបើអ្នកមានតម្លៃចំនួន ៥ ដែលត្រូវរក្សាទុក នោះអ្នកប្រាកដជាត្រូវបង្កើត variable ចំនួនប្រាំផង ដែរ**Array** គឺ flexible ព្រោះវាអាចផ្ទុកតម្លៃបានពីរ ឬ ពីរយ តម្លៃ ដោយពុំ មានការង្កើតនូវ variableថ្មីទៀត ហើយ array ក៏អាចឱ្យអ្នកធ្វើការជាមួយតម្លៃរបស់ វា បានយ៉ាងងាយ ដូចជា ការ loop ធាតុរបស់ array និមួយៗ ឬ តម្រៀបធាតុរបស់វាទៅលំដាប់នៃលេខរៀង ឬ ជាតួអក្សរ ទៅតាមការកំណត់នៅក្នុង system របស់អ្នក ។ ខាងក្រោមគឺជាការបង្ហាញនូវធាតុរបស់ users array ដែលមានធាតុទី៤ ជា index ទី៣ នៃ users ។

ដូច្នេះ **array** គឺជា **list** នៃទិន្នន័យ ដែលតំណាងដោយ **variable** ណាមួយ។

Name_Student					
Data	Daro	Raksa	Bopha	Roza	Tevi
Index	0	1	2	3	4

១. ការបង្កើត Creating Arrays .

អ្នកអាចបង្កើតនូវ array variable ដោយប្រើវិធីពីរយ៉ាងគឺ ការប្រើប្រាស់ array( ) construct ឬ ការប្រើប្រាស់នូវសញ្ញា square brackets ([ ]) ។

ទម្រង់នៃការបង្កើត Array

```
<DataType>[]<Array_Name>=new <DataType>[<Length>];
```

- **<DataType>**គឺតំណាងឲ្យប្រភេទទិន្នន័យនៃទិន្នន័យដែលត្រូវផ្ទុកក្នុង **array** នោះ
- <ArrayName>គឺតំណាងឲ្យឈ្មោះ array ដែលត្រូវបង្កើត
- <Length>គឺតំណាងឲ្យចំនួននៃទិន្នន័យដែលត្រូវផ្ទុកក្នុង array នោះ

សូមមើលឧទាហរណ៍នការបង្កើត array ផ្សេងៗទៀតដូចខាងក្រោម៖

```
Int[]Myint=new int [10];
```

ខាងលើនេះជាការបង្កើត array integer variable ដែលមានឈ្មោះថា MyInt វាអាចផ្ទុកបាន ១០

តម្លៃ integers។

៣. ការផ្តល់តម្លៃទៅកាន់ធាតុនីមួយៗរបស់ array

តាមរយៈការសិក្សាខាងលើអ្នកប្រហែលជាបានដឹងអំពីវិធីនៃការបង្កើត array variable នៅក្នុងកាសាC# របស់អ្នក ហើយបន្តទៀតនេះយើងនឹងនិយាយពីការផ្តល់តម្លៃទៅកាន់ array។

ឧទាហរណ៍លើកមុន បង្ហាញឲ្យឃើញថា Names គឺជា array variable ហើយក្នុងរូបនោះយើងក៏ឃើញថា array ត្រូវបានបែងចែកចេញជាបន្ទប់តូចៗ ដែលយើងហៅថា ធាតុ នៃ array។ ធាតុនីមួយៗរបស់ array គឺមានលេខសម្គាល់របស់វាដែលហៅថា index ដោយចាប់ពីធាតុទី១ត្រូវសម្គាល់ដោយលេខ ០ និង ធាតុទី២ដោយលេខ១ និងបន្តបន្ទាប់តាមលំដាប់។

សូមមើលការផ្តល់តម្លៃទៅកាន់ធាតុនីមួយៗរបស់ Arrays name ដូចខាងក្រោម៖

```
name[0] = "Saron";
name[1] = "Dara";
name[2] = "Rotha";
name[3] = "Makara";
name[4] = "Bundet";
```

៤. ទម្រង់ទូទៅនៃការផ្តល់តម្លៃទៅឲ្យ array

<ArrayName>[<index>] =<Data>;

- <ArrayName>តំណាងឲ្យឈ្មោះ array ដែលត្រូវផ្តល់តម្លៃទៅកាន់ធាតុរបស់វា
- <index>តំណាងឲ្យលេខសម្គាល់ធាតុរបស់ array ដែលត្រូវបញ្ចូលទិន្នន័យ
- <Data> តំណាងឲ្យទិន្នន័យដែលត្រូវបញ្ចូលទៅកាន់ធាតុរបស់ array (ទិន្នន័យ ដែលត្រូវបញ្ចូលត្រូវមានប្រភេទទិន្នន័យដូចគ្នានឹងប្រភេទទិន្នន័យរបស់ array)

ក្រៅពីការផ្តល់តម្លៃទៅកាន់ array តាមវិធីខាងលើ យើងក៏អាចផ្តល់តម្លៃទៅកាន់ array ក្នុងខណៈដែល កំពុងតែធ្វើការបង្កើតផងដែរសូមមើលកូដខាងក្រោម៖

```
string [] name = new string[5] {"Saron","Dara","Rotha","Makara","Bundet"};

string [] name = new string[] {"Saron","Dara","Rotha","Makara","Bundet"};

string [] name =  {"Saron","Dara","Rotha","Makara","Bundet"};
```

៥. ការទាញតម្លៃចេញពី array

យើងពុំពិបាកឡើយក្នុងការទាញតម្លៃចេញពីធាតុនីមួយៗរបស់ array ដោយគ្រាន់តែសរសេរ ឈ្មោះ array និងកំណត់លេខ index នៃធាតុដែលត្រូវទាញតម្លៃនៅពីក្រោយសញ្ញា assign តែប៉ុណ្ណោះ សូមមើល

កូដដូចខាងក្រោម៖

```
string name1=name[];
```

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

namespace ConsoleApplication4
{
 class Program
{
    static void Main(string[] args)
    {
        string[] name = { "Saron", "Dara", "Rotha", "Makara",
                          "Bundet" };

         Console.WriteLine(name[2]);
         Console.ReadKey();
    }
}
}
```

ខាងលើមានន័យថាយើងទាញទិន្នន័យចេញពី name array ក្នុងធាតុទី១ទៅឲ្យ name ។

Console.WriteLine(name[2]);

៦.ការប្រើប្រាស់ parallel arrays

Parallel arrays គឺសំដៅលើរបៀបនៃការចងភ្ជាប់ arrays ជាច្រើនបញ្ចូលគ្នាក្នុងគោលបំណងបង្កើតជាrecords នៃព័ត៌មានអ្វីមួយ សូមពិនិត្យមើលតារាងខាងក្រោម៖

. Table information

ID	Name	Gender
001	Saron	M
002	Bundet	M
003	Tevi	F
004	Makara	F

តារាងខាងលើបង្ហាញថាយើងមានព័ត៌មានមនុស្ស ០៤នាក់ ដែលក្នុងម្នាក់ៗមានព័ត៌មានដូចជា id, name និង Gender។ ដូច្នេះបើយើងធ្វើការបម្លែងតារាងខាងលើទៅជាការប្រើប្រាស់ទម្រង់ parallel arrays

ក្នុងភាសា C# យើងអាចសរសេរដូចនេះខាងក្រោម៖

```
string []id = new string[4]{ "0001", "0002", "0003","004"};

string [] name = new string[4]{ "Saron", "Bundet", "Tevi","Makara"};

string [] sex = new string[4]{ "M", "M", "F", "F"};
```

ឬយើងក៏អាចសរសេរដូចខាងក្រោម៖

```
string [] ID =new string[4];
string[] name = new string[4];
string[] Gender = new string[4];

   ID[0]="001";
   name[0] = "Saron";
   Gender[0] = "M";

   ID[1] = "002";
   name[1] = "Bundet";
   Gender[1] = "M";

   name[2] = "003";
   name[2] = "Tevi";
   name[2] = "Makara";

   ID[3] = "004";
   name[3] = "Makara";
   Gender[3] = "F";
```

```
 1 using System;
 2 using System.Collections.Generic;
 3 using System.Linq;
 4 using System.Text;
 5 using System.Threading.Tasks;
 6
 7 namespace ConsoleApplication4
 8 {
 9     class Program
10     {
11         string [] ID =new string[4];
12              string[] name = new string[4];
13              string[] Gender = new string[4];
14
15                 ID[0]="001";
16                 name[0] = "Saron";
17                 Gender[0] = "M";
18
19                 ID[1] = "002";
20                 name[1] = "Bundet";
21                 Gender[1] = "M";
```

```
22
23
24                  name[2] = "003";
25                  name[2] = "Tevi";
26                  name[2] = "F";
27
28                  ID[3] = "004";
29                  name[3] = "Makara";
30                  Gender[3] = "F";
31
32              Console.WriteLine(ID[0]+name[0] + Gender[0]);
33              Console.ReadKey();
34          }
35      }
36
```

តាមរយៈកូដខាងលើយើងឃើញថា គ្រប់ records នីមួយៗគឺបានកើតឡើងដោយការចងភ្ជាប់ arrays ច្រើនបញ្ចូលគ្នាតាមរយៈការបញ្ជាក់ index ដែលមានតម្លៃដូចៗគ្នា។

៧. ការប្រើប្រាស់ array ច្រើនទំហំ

យើងបានដឹងពីការប្រើប្រាស់ array ហើយអ្វីដែលអ្នកបានដឹងកន្លងមកនេះគឺជាការប្រើប្រាស់ array មួយទំហំ។

Array មួយទំហំវាតំណាងឲ្យទិន្នន័យដែលមានទម្រង់ជា list ឬ column ដូចអ្វីដែលអ្នកបានឃើញកូដខាងលើដូចជា id, name និង sex ដែលជា columns របស់តារាងដែលតំណាងឲ្យជាតុផ្សំនៃព័ត៌មានរបស់មនុស្សម្នាក់ៗ។

ដូចគ្នាផងដែរសម្រាប់ information table ខាងលើយើងក៏អាចបម្លែងទៅជាការប្រើប្រាស់ array ពីរទំហំដូចខាងក្រោម៖

```
string[,] Students = new string[4,4];

            Students[0,0]="001";
            Students[0,1] = "Saron";
            Students[0,2] = "M";

            Students[1,0] = "002";
            Students[1,1] = "Bundet";
            Students[1,2] = "M";

            Students[2,0] = "003";
            Students[2,1] = "Tevi";
            Students[2,2] = "F";

            Students[3,0] = "004";
            Students[3,1] = "Makara";
```

```
            Students[3,2] = "F";
```

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

namespace ConsoleApplication4
{
    class Program
    {
        static void Main(string[] args)
        {
            string[,] Students = new string[4, 4];

            Students[0, 0] = "001";
            Students[0, 1] = "Saron";
            Students[0, 2] = "M";

            Students[1, 0] = "002";
            Students[1, 1] = "Bundet";
            Students[1, 2] = "M";

            Students[2, 0] = "003";
            Students[2, 1] = "Tevi";
            Students[2, 2] = "F";

            Students[3, 0] = "004";
            Students[3, 1] = "Makara";
            Students[3, 2] = "F";

            Console.WriteLine(Students[2,0]);
            Console.WriteLine(Students[2, 1]);
            Console.WriteLine(Students[2, 2]);
            Console.ReadKey();
        }
    }
```

ជាទូទៅការបង្កើត Array variable តែងកំណត់ចំនួនជាតុនៃ Array variable នីមួយៗសម្រាប់ធ្វើការក្បាទុកទិន្នន័យទៅតាមចំនួនកំណត់ មិនថាជា Array មួយទំហំ ឬច្រើនទំហំឡើយ។ នេះជាបញ្ហាចោទមួយបើសិនជាយើងមាន Array variable មួយ ដូចខាងក្រោម៖

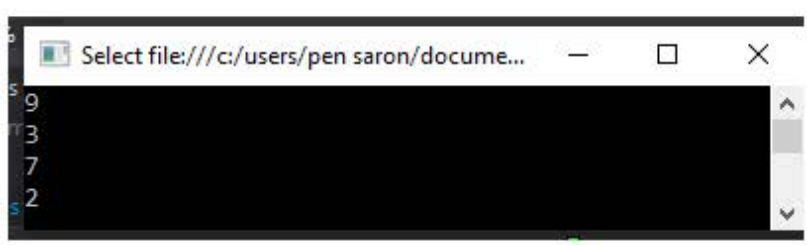
string []names=new string [5];

មានន័យថាខ្ញុំបង្កើត Array variable មួយដែលមាន data type ជា string ហើយមានលទ្ធភាពក្បាទុកទិន្នន័យ, ឈ្មោះជា string បានចំនួន ៥ ឈ្មោះ។

៨.Loop Array

អ្នកអាចធ្វើសារឡើងវិញដោយឆ្លងកាត់បណ្តាជាតុរបស់ array ដែលប្រើ statement ។ code ឧទាហរណ៍ខាងក្រោមនេះសរសេរពីតម្លៃជាតុ array ទៅ console: int[] pins = { 9, 3, 7, 2 };

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

namespace ConsoleApplication4
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] pins = { 9, 3, 7, 2 };
            for (int index = 0; index != pins.Length; index++)
            {
                int pin = pins[index];
                Console.WriteLine(pin);
            }
            Console.ReadKey();
        }
    }

```



៩.Copy **ចំលង** Array

Arrays គឺជាប្រភេទ reference ។ អថេរនៃ array មួយមានផ្ទុក reference មួយដើម្បីបង្កើត array instance ។ នេះមានន័យថាកាលណាអ្នក copy អថេរ array មួយអ្នកបញ្ចប់ជាមួយពីរ references ដែលយោងទៅតាម array instance ដូចគ្នាឧទាហរណ៍:

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

```

```
 7 namespace ConsoleApplication4
 8 {
 9     class Program
10     {
11         static void Main(string[] args)
12         {
13
14             int[] pins = { 9, 3, 7, 2 };
15             foreach (int pin in pins)
16             {
17                 Console.WriteLine(pin);
18             }
19             Console.ReadKey();
20         }
21     }
22 }
```

ប្រសិនបើអ្នកចង់បង្កើត copy នៃ array instance ដែលអថេរ array មួយយោងទៅតាមវា អ្នកធ្វើពីរបញ្ហា ។ ទីមួយ អ្នកត្រូវការបង្កើត array instance ថ្មីនៃប្រភេទដូចគ្នានិងប្រវែង array ដូចគ្នាអ្នកកំពុង copy ដូចក្នុងឧទាហរណ៍នេះ៖

```
int[] pins ={9,3, 7, 2 };
int[] alias = pins; // alias and pins refer to the same array instance
```

ប៉ុន្តែបើអ្នកប្តូរប្រវែងដើមរបស់ array អ្នកត្រូវចាំថាប្តូរទំហំនៃ copy ផង ដែរ វាប្រសើរជាងកំណត់ប្រវែងនៃ array មួយដោយប្រើប្រវែងរបស់វា បង្ហាញក្នុងឧទាហរណ៍ខាងក្រោម៖

```
int[] pins = { 9, 3, 7, 2 };
int[] copy = new int[4];
```

បណ្តាតម្លៃខាងក្នុង copy ឥឡូវនេះបានចាប់ផ្តើមទាំងអស់ទៅក្នុង តម្លៃ default នៃ 0 ។បញ្ហាទីពីរអ្នកត្រូវការធ្វើគឺបង្កើតតម្លៃខាងក្នុង array ថ្មី ទៅតម្លៃដូចគ្នាក្នុង array ដើម ។ អ្នកនឹងអាចធ្វើដូចនេះដោយប្រើ statement, ដូចបង្ហាញក្នុងឧទាហរណ៍ខាងក្រោមនេះ៖

```
int[] pins = { 9, 3, 7, 2 };
int[] copy = new int[pins.Length];
for ( int i = 0; i != copy.Length; i++ )
{
        copy[i] = pins[i];
}
```

ដើម្បីចង្អុលបង្ហាញពីគោលការណ៍នេះ និងពិចារណាពីការប្រតិបត្តិបង្កើត creating និង copying ចំលង array មួយដែលប្រភេទធាតុ គឺជារបស់វាខ្លួនឯង ប្រភេទ reference ។ ឧទាហរណ៍ខាងក្រោមនេះបង្កើត array មួយនៃ 4 triangles: Triangle[] triangles = new Triangle[4];

ប្រសិនបើ Triangle គឺជា class មួយដែលតួនាទី array instance ចាប់ផ្តើមក្នុងធាតុនីមួយៗ គឺត្រូវបានចាប់ផ្តើមទៅ null ។ ម៉្យាងទៀត diagram statement ពីមុនមើលឃើញដូចនេះ ។

១០.ប្រើ System.ArrayClass

ហេតុអ្វីបានជាប្រភេទ int គឺជាប្រភេទតម្លៃគ្រប់ structs ទាំងអស់គឺ ជាប្រភេទតម្លៃ ។

ឧទាហរណ៍: string rep = 42.ToString( );

System.Array class ផ្តត់ផ្តង់ប្រវែងគ្រប់ arrays ទាំងអស់ ។ ម៉្យាងទៀត មិនមានភាពខុសគ្នារវាងពីរ statements:

int[] pips = new int[4]{ 9, 3, 7, 2 };

Array pips = new int[4]{ 9, 3, 7, 2 };

ទោះបីអ្នកបានឃើញ Array class ក្នុង code ។ ក្នុងគោលបំណង នាំប្រភេទ arrayទាំងអស់ទៅក្នុងភាសារួមមួយ common language runtime (CLR) ។ ការចាប់ផ្តើមប្រើប្រាស់ shortcut មិនអាចធ្វើបានកាលណា ចាប់ផ្តើមអថេរ array មួយ : int[] pips = { 9, 3, 7, 2};

១១. អ្វីទៅដែលហៅថា Collection Classes ?

Arrays គឺជាមធ្យោយបាយតែមួយគត់ដើម្បីប្រមូលធាតុនៃប្រភេទដូចគ្នា ក្នុងកម្មវិធី Microsoft .NET Framework មាន namespace ដែលហៅថា System.Collections ដែលផ្តុំកច្រើនរាប់មិនអស់ classes ដែលមានមុខងារ សម្រាប់ប្រមូលធាតុរួមទាំងអស់ ។ បណ្តា class ទាំងនេះផ្តត់ផ្តង់បង្កើតជំណោះស្រាយរួចជាស្រេចគ្រប់ស្ថានភាព កាលណាអ្នកត្រូវការប្រមូលផ្តុំធាតុ ។ គ្រប់ collection classes ទាំងអស់អនុញ្ញាត ជាក់និងត្រឡប់ចូលទៅក្នុងធាតុ របស់គេដូចជា object ។ ប្រភេទធាតុនេះគឺជាការប្រមូលផ្តុំនៃ class ដែល តែងតែមាន object (System.Object) ។

.ArrayList

- BitArray
- Stack
- Queue
- Comparer
- HashTable
- SortedList

១១.១ ការប្រើប្រាស់ ArrayList

ArrayList គឺជាData Structure មួយនៅក្នុង C# Collections។ ArrayList ត្រូវបានរក្សាទុកទិន្នន័យជាលក្ខណៈ list ដែលយើងអាច បញ្ចូល លុប បង្ហាញបានយ៉ាងល្អប្រសើរ និងងាយស្រួលពីព្រោះយើងអាចបង្កើត ដោយមិនចាំបាច់បញ្ជាក់ពីចំនួនធាតុដែលត្រូវរក្សាទុក និង datatype ឡើយ។ គឺវាអាចកែរួមជាលក្ខណៈ Dynamic and Shrink ។ ក្រៅពីនេះ ArrayList អាច តម្រៀប បញ្ចូលទិន្នន័យទៅកាន់ទីតាំងជាក់លាក់ណាមួយ អាចលុបទិន្នន័យពីទីតាំងជាក់លាក់ណាមួយ ដោយ methods សំខាន់ៗ

មួយចំនួន។ ដើម្បីប្រើប្រាស់ ArrayList បានយើងត្រូវប្រើ namespaceSystem.Collections

ទម្រង់ក្នុងការប្រើប្រាស់ Arraylist object

```
ArrayList Array_list_name;

ArrayList list;

ArrayList list = new ArrayList( );
```

Methods		
1	Add	It will add the element as object in the ArrayList
2	AddRange	It will add the collections of elements in the object as individual objects in the ArrayList
3	Clear	It will clear the all objects in the ArrayList
4	BinarySearch	It will return the position of the search object as integer value.
5	Insert	It will insert the element in the specified location of the index in the ArrayList.
6	InsertRange	It will insert the elements in the object as individual objects in the specified location.
7	Remove	It will remove the given object in the first occurrence in the ArrayList.
8	RemoveAt	It will remove the object as specified in the argument.
9	RemoveRange	It will remove the set of objects in the ArrayList from the range specified.
10	Sort	It will do the sorting of the elements in the ascending order.
11	Reverse	It will arrange the elements in the reverse order in the ArrayList.
12	GetEnumerator	It will return the collection of objects in the ArrayList as enumerator.
13	Contains	It checks whether the objects exists or not.

១១.២ របៀបនៃការបញ្ចូល Add និងឆ់យឱ្យ ArrayList Object

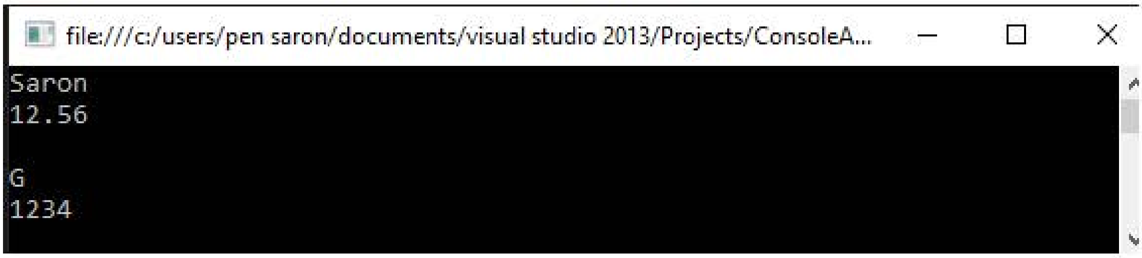
Add Method គឺប្រើប្រាស់ដើម្បីបន្ថែម Object ទៅឱ្យ ArrayList ដែលចង់ បន្ថែមនៃជាតុ ទៅឱ្យ ArrayList បន្ទាប់មកវាធ្វើការពិនិត្យ រួចបញ្ចូលនូវ Object ។

ខាងក្រោមនេះជាវិធីសាស្ត្រក្នុងការបន្ថែមជាតុឱ្យ ArrayList ៖

```
ArrayListObject.Add(element);
```

```
1  using System;
2  using System.Collections;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace ConsoleApplication5
9  {
10     class Program
11     {
12         static void Main(string[] args)
13         {
14             // Creating ArrayList
15             ArrayList My_array = new ArrayList();
16             My_array.Add("Saron");
```

```
            My_array.Add("12.56");
            My_array.Add(null);
            My_array.Add('G');
            My_array.Add(1234);

            foreach (var elements in My_array)
            {
                Console.WriteLine(elements);
            }
            Console.ReadKey();
        }
    }
```



១១.៣ របៀបនៃការបញ្ចូល Insert ទិន្នន័យឱ្យ ArrayList

គេប្រើប្រាស់ Insert ( ) Method Insert ដើម្បីបញ្ចូល ជាតុទៅឱ្យ ArrayList Object List <object> ដែលបានកំណត់ដោយតាមរយៈ Index ។

ខាងក្រោមជាវិធីសាស្ត្រ syntax:

```
ArrayListObject.Insert(index,Object);
```

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

namespace ConsoleApplication5
{
    class Program
    {
        static void Main(string[] args)
        {
            var numbers = new List<int>() { 10, 20, 30, 40 };
            numbers.Insert(1, 11);//inserts 11 at 1st index: after 10.

            foreach (var num in numbers)
            Console.WriteLine(num);
```

```
19              Console.ReadKey();
20          }
21      }
22  }
```

លទ្ធផល៖

១១.៤ របៀបនៃការបញ្ចូល Remove និងនិយមន័យ ArrayList Object

គេប្រើប្រាស់ Remove ( ) Method Remove ដើម្បី ដក ធាតុចេញទៅឱ្យ ArrayList Object List <Object> ដែលបានកំណត់ដោយតាមរយៈ Index ។ ប្រសិនបើយើងប្រើ RemoveAt( ) Method ដើម្បី Remove ធាតុ ចេញតាមរយៈការកំណត់ដោយ Index ។

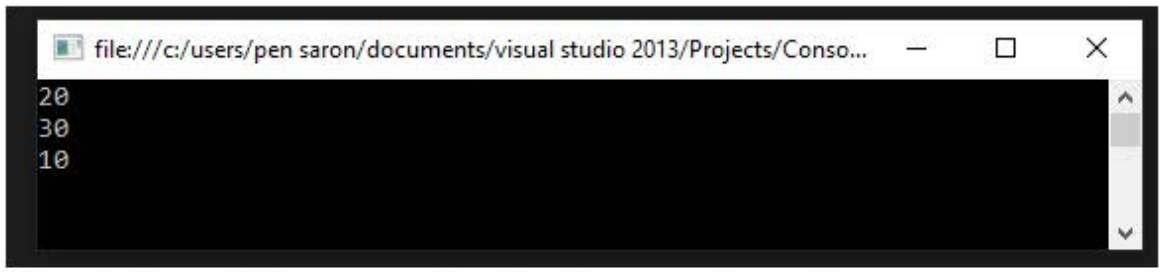
ខាងក្រោមនេះជារបៀបប្រើប្រាស់ remove ArrayList object

Remove() signature: `bool Remove(T item)`

RemoveAt() signature: `void RemoveAt(int index)`

```
1  using System;
2  using System.Collections;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace ConsoleApplication5
9  {
10     class Program
11     {
12         static void Main(string[] args)
13         {
14             var numbers = new List<int>() { 10, 20, 30, 40, 10 };
15
16             numbers.Remove(10); // removes 10 elements from a list
17
18             numbers.RemoveAt(2); //removes the 3rd element (index
19 starts from 0)
20
21             //numbers.RemoveAt(10); //removes the 3rd element (index
22 starts from 0)
23
24             foreach (var num in numbers)
25                 Console.WriteLine(num);
```

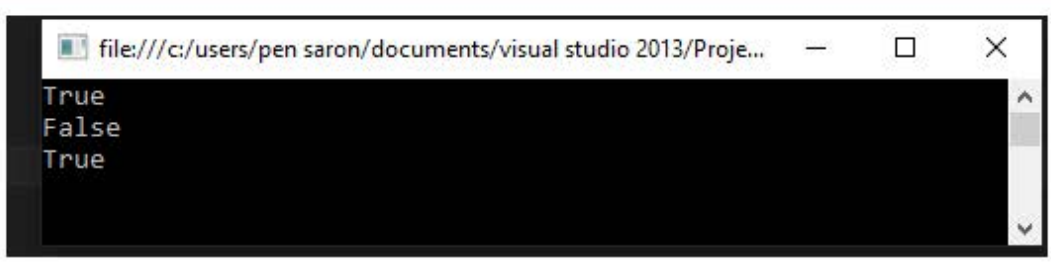
```
26                 Console.ReadKey();
27             }
28         }
29 }
```



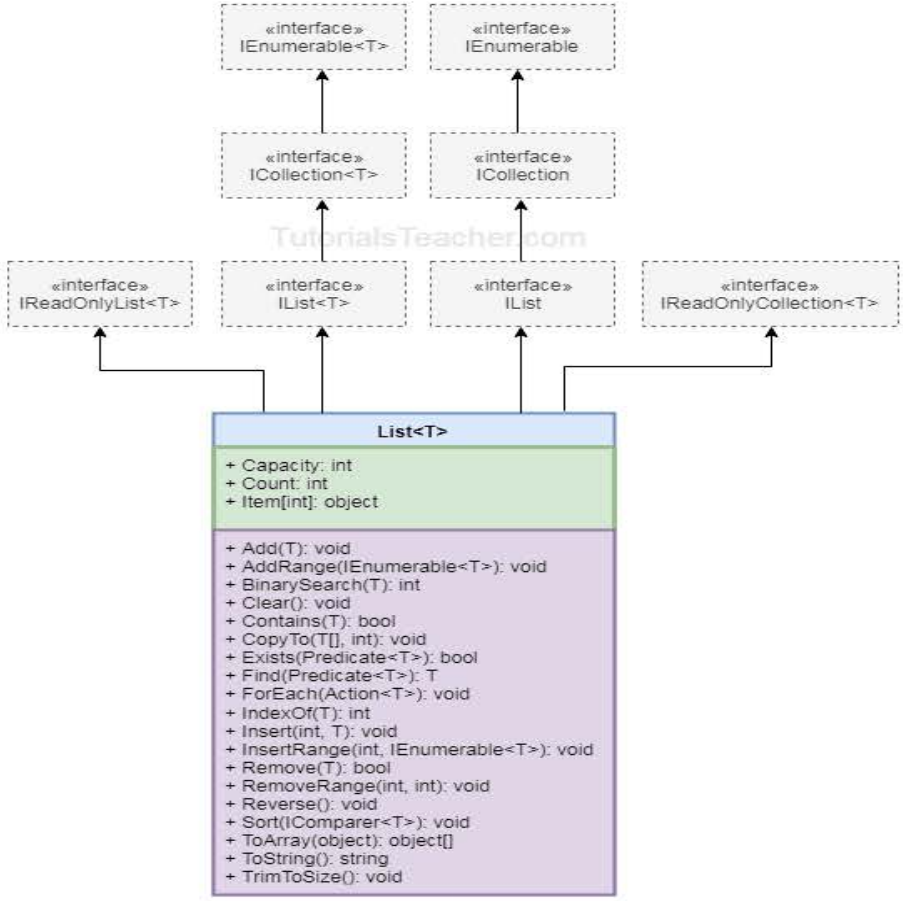
១១.៥ របៀបនៃការពិនិត្យ Check ចំនួនទិន្នន័យឱ្យ ArrayList Object

Check Elements in List ប្រើប្រាស់ Contains( ) method ដើម្បីត្រួពិនិត្យជាតុ របស់ ArrayList វិធីកំណត់ថាតើជាតុ មាននៅក្នុងបញ្ជី List <object> ឬអត់។

```
1  using System;
2  using System.Collections;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace ConsoleApplication5
9  {
10     class Program
11     {
12         static void Main(string[] args)
13         {
14             var numbers = new List<int>() { 10, 20, 30, 40 };
15             Console.WriteLine(numbers.Contains(10));
16             Console.WriteLine(numbers.Contains(11));
17             Console.WriteLine(numbers.Contains(20));
18
19             Console.ReadKey();
20         }
21     }
22 }
```



ជ្បាក្រមនៃ List <T>



១១.៦ រៀប Sort (តម្រៀប) ArrayList Object

បញ្ជីដែលបានតម្រៀប <TKey, TValue> និង SortedList គឺជាថ្នាក់ប្រមូលផ្តុំដែលអាចរក្សាទុក key-value សំខាន់ៗដែលត្រូវបានតម្រៀបដោយ keys ដោយផ្អែកលើការអនុវត្ត IComparer ដែលពាក់ព័ន្ធ។ ឧទាហរណ៍ ប្រសិនបើ keys ជា primitive types ប្រភេទបឋម នោះតម្រៀបតាមលំដាប់ឡើងនៃkeys ។

C# គាំទ្របញ្ជី SortedList ទូទៅ generic និង non-generic SortedList ។

វាត្រូវបានណែនាំឱ្យប្រើ SortedList<TKey, TValue> ព្រោះវាដំណើរការលឿនជាង និង error-prone មិនសូវមានកំហុសជាង non-generic SortedList ។

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

```
using System.Threading.Tasks;

namespace ConsoleApplication5
{
    class Program
    {
        static void Main(string[] args)
        {
            SortedList<int, string> numberNames = new SortedList<int,
             string>()
                                {
                                    {3, "Three"},
                                    {1, "One"},
                                    {2, "Two"}
                                };

            Console.WriteLine(numberNames[1]); //output: One
            Console.WriteLine(numberNames[2]); //output: Two
            Console.WriteLine(numberNames[3]); //output: Three

            numberNames[2] = "TWO"; //updates value
            numberNames[4] = "Four"; //adds a new key-value if a key
                                     // does not exists

      foreach (var kvp in numberNames)
       Console.WriteLine("key: {0}, value: {1}", kvp.Key, kvp.Value);

          Console.ReadKey();
       }
    }
}
```

១២. ArrayList

នៅក្នុង C # បញ្ជីអារេ ArrayList គឺជាការប្រមូល non-generic collection of object ដែលមិនមែនជាទូទៅនៃ object ដែលទំហំដែលទំហំកើនឡើងយ៉ាង ដោយការបញ្ចូល។ វាដូចគ្នានឹងអារេដែរលើកលែងតែទំហំរបស់វាកើនឡើងយ៉ាងស្វាហាប់។ ArrayList មួយអាចត្រូវបាន ប្រើ ដើម្បីបន្ថែមទិន្នន័យដែលមិនស្គាល់ដែលអ្នកមិនដឹងប្រភេទនិងទំហំទិន្នន័យ។

១២.១ ការបង្កើត Create an ArrayList

ArrayList Class ត្រូវបានបញ្ចូលនៅក្នុង using System.Collections namespace ។ ការបង្កើត Object នៃ ArrayList ដោយប្រើប្រាស់នូវ new KeyWord ។

Syntax:

```
sing System.Collections;
ArrayList arlist = new ArrayList();
// or
var arlist = new ArrayList(); // recommended
```

១២.២ ការបន្ថែម Adding Elements in ArrayList

យើងប្រើ Add( ) Method ឬ Object initializer syntax ដើម្បី បន្ថែម ធាតុមួយ ឬច្រើន ទៅក្នុង AraayList ។

```
1  using System;
2  using System.Collections;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace ConsoleApplication5
9  {
10     class Program
11     {
12         static void Main(string[] args)
13         {
14             // adding elements using ArrayList.Add() method
15                 var arlist1 = new ArrayList();
16                 arlist1.Add(1);
17                 arlist1.Add("Bill");
18                 arlist1.Add(" ");
19                 arlist1.Add(true);
20                 arlist1.Add(4.5);
21                 arlist1.Add(null);
22
23                 // adding elements using object initializer syntax
24                 var arlist2 = new ArrayList()
25                     {
26                         2, "Steve", true, 4.5, null
27                     };
28                 Console.WriteLine("ArrayList 1 Elements");
29
30                 for(int i = 0; i< arlist1.Count; i++)
31                         Console.WriteLine(arlist1[i]);
```

```
32
33                  Console.WriteLine("ArrayList 2 Elements");
34
35                  for(int i = 0; i< arlist2.Count; i++)
36                          Console.WriteLine(arlist2[i]);
37          Console.ReadKey();
38          }
39      }
```

គេប្រើ AddRange(ICollection c) method ដើម្បី Add entire Array, HashTable, SortedList, ArrayList, BitArray, Queue និង Stack ក្នុង ArrayList.

```
1  using System;
2  using System.Collections;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace ConsoleApplication5
9  {
10     class Program
11     {
12         static void Main(string[] args)
13         {
14             var arlist1 = new ArrayList();
15
16             var arlist2 = new ArrayList()
17                         {
18                           1, "Bill", " ", true, 4.5, null
19                         };
20
21             int[] arr = { 100, 200, 300, 400 };
22
23             Queue myQ = new Queue();
24             myQ.Enqueue("Hello");
25             myQ.Enqueue("World!");
26
27             arlist1.AddRange(arlist2); //adding arraylist in
28                                        //arraylist
29             arlist1.AddRange(arr); //adding array in arraylist
30             arlist1.AddRange(myQ); //adding Queue in arraylist
31
32             Console.WriteLine("ArrayList Elements");
33
```

```
            for (int i = 0; i < arlist1.Count; i++)
                Console.WriteLine(arlist1[i]);
        Console.ReadKey();
        }
    }
}
```

១២.៣ ដំណើរការ Accessing an ArrayList

The ArrayList class implements the IList interface. So, elements can be accessed using indexer, in the same way as an array. Index starts from zero and increases by one for each subsequent element.

១៣. List

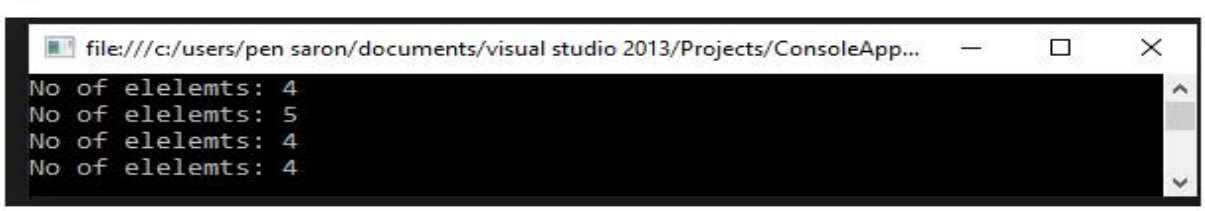
List <Object> គឺជា Collection នៃប្រភេទការ តម្រៀប Object ដែលអាចដំណើរការបានដោយ Index ហើយនិង Method សម្រាប់ធ្វើការ តម្រៀប Sort data ការស្វែងរក ការផ្លាស់ប្តូរ List ។ វាគឺជា Generic version នៃ ArrayList ដែលបានមកពី system System.Collections.Generic namespace ។

១៣. ១ ការបង្កើត List Creating a List

List <Object> គឺជា generic collection ដូចនេះអ្នកត្រូវតែ កំណត់ ប្រភេទប៉ារ៉ាម៉ែត្រ DataType Parameter សម្រាប់ ប្រភេទ នៃ ទិន្នន័យ វាអាច Store ទិន្នន័យ Class Object ។
Soure code:

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

namespace ConsoleApplication5
{
    class Program
```

```
    {
      // Class object() method
        class Student
        {
            public int Id { get; set; }
            public string Name { get; set; }
        }

        static void Main(string[] args)
        {
            // adding elements using add() method
            var primeNumbers = new List<int>();
            primeNumbers.Add(1);
            primeNumbers.Add(3);
            primeNumbers.Add(5);
            primeNumbers.Add(7);

            Console.WriteLine("No of elelemts: " +
                              primeNumbers.Count);

            var cities = new List<string>();
            cities.Add("New York");
            cities.Add("London");
            cities.Add("Mumbai");
            cities.Add("Chicago");
            cities.Add(null); // null is allowed

            Console.WriteLine("No of elelemts: " + cities.Count);

            // adding elements using collection initializer syntax
            var bigCities = new List<string>() { "New York", "London",
                              "Mumbai", "Chicago" };

            Console.WriteLine("No of elelemts: " + bigCities.Count);

            var students = new List<Student>() {
                new Student(){ Id = 1, Name="Bill"},
                new Student(){ Id = 2, Name="Steve"},
                new Student(){ Id = 3, Name="Ram"},
                new Student(){ Id = 4, Name="Abdul"}
            };

            Console.WriteLine("No of elelemts: " + students.Count);
            Console.ReadKey();
        }
    }
}
```

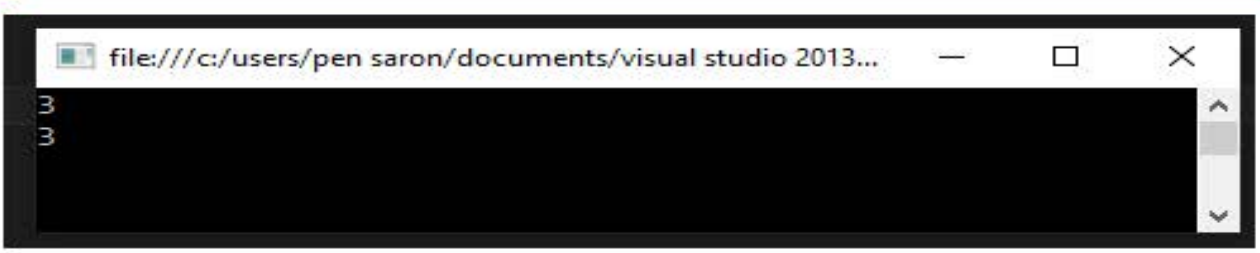


១៣.២ ការបន្ថែម Adding an Array in a List

គេប្រើ AddRange( ) method ដើម្បី Add ទាំងអស់ នៃជាតុ ដែលបានមកពី Array ឬ ក៏ Collection ផ្សេងទៀត សម្រាប់ List ។

AddRange( ) signature: void AddRange(IEnumerable<Objects> collection);

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

namespace ConsoleApplication5
{
    class Program
    {

        static void Main(string[] args)
        {
          string[] cities = new string[3] { "Mumbai", "London", "New York" };

            var popularCities = new List<string>();

            // adding an array in a List
            popularCities.AddRange(cities);

            var favouriteCities = new List<string>();

            // adding a List
            favouriteCities.AddRange(popularCities);

            Console.WriteLine(popularCities.Count);
            Console.WriteLine(favouriteCities.Count);
            Console.ReadKey();
        }
    }
}
```

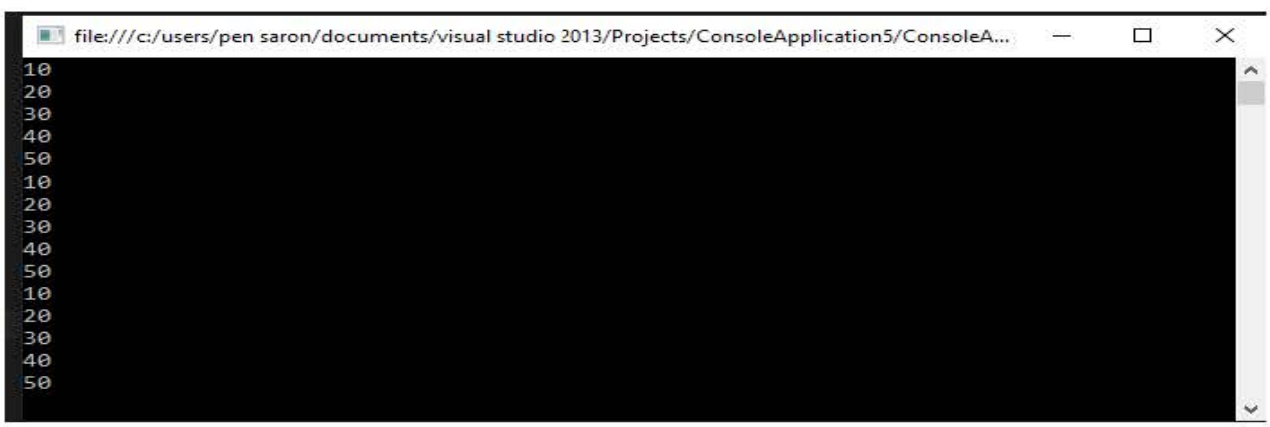


១៣.៣ ដំណើរការ Accessing a List

List ដំណើរការបានដោយ Index ធ្វើការដោយប្រើប្រាស់នូវ for/foreach loop និងប្រើប្រាស់ LINQ queries ។ Indexes នៃ List វាធ្វើការ Loop ដោយចាប់ផ្តើម ពី Index [0] ។ ហើយវាធ្វើការ បម្លែង Index

នៅក្នុង Brackets ដើម្បី Access ដោយផ្តាល់ list items ដូចទៅនឹង Array ដែរ។
ខាងក្រោមនេះយើងប្រើ loop ដើម្បី Access list

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

namespace ConsoleApplication5
{
    class Program
    {

        static void Main(string[] args)
        {
          List<int> intList = new List<int>() { 10, 20, 30, 40, 50 };

            intList.ForEach(el => Console.WriteLine(el));

            foreach (var el in intList)
                Console.WriteLine(el);

            for (int i = 0; i < intList.Count; i++)
                Console.WriteLine(intList[i]);
            Console.ReadKey();
        }

    }
```



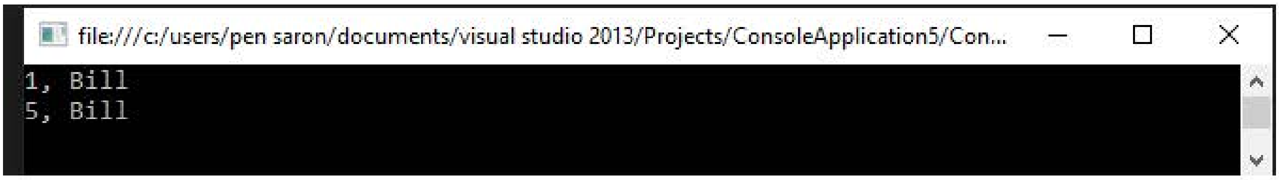
១៣.៤ ដំណើរការ Accessing a List using LINQ

List<Objects> ធ្វើការអនុវត្ត IEnumerable interface ដូចនេះយើងអាច query List ដោយប្រើប្រាស់ LINQ query syntax ឬ Method syntax ដូចខាងក្រោម ៖

sourer code:

```
using System;
using System.Collections;
```

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

namespace ConsoleApplication5
{
    class Program
    {
        public class Student
        {
            public int Id { get; set; }
            public string Name { get; set; }
        }

        static void Main(string[] args)
        {
            var students = new List<Student>() {
                new Student(){ Id = 1, Name="Bill" },
                new Student(){ Id = 2, Name="Steve" },
                new Student(){ Id = 3, Name="Ram" },
                new Student(){ Id = 4, Name="Abdul" },
                                new Student(){ Id = 5, Name="Bill" }
                };

            //get all students whose name is Bill
            var studNames = from s in students
                            where s.Name == "Bill"
                            select s;

            foreach (var student in studNames)

            Console.WriteLine(student.Id + ", " + student.Name);
            Console.ReadKey();
        }

    }
}
```



១៣.៥ Insert Elements in List

ប្រើប្រាស់ Insert( ) Method ដើម្បីបញ្ចូលធាតុ element ទៅឱ្យ List collection ដែលយើងបានកំណត់តាមរយៈ index ។

Insert( ) signature: void Insert(int index, object item);

១៣.៥.១ របៀបនៃការបញ្ចូល Insert និងនិយមន័យ ArrayList

គេប្រើប្រាស់ Insert ( ) Method Insert ដើម្បីបញ្ចូល ជាតុទៅឱ្យ ArrayList Object List <object> ដែលបានកំណត់ដោយតាមរយៈ Index ។

ខាងក្រោមជាវិធីសាស្ត្រ syntax:

```
ArrayListObject.Insert(index,Object);
```

```
1  using System;
2  using System.Collections;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace ConsoleApplication5
9  {
10     class Program
11     {
12         static void Main(string[] args)
13         {
14             var numbers = new List<int>() { 10, 20, 30, 40 };
15             numbers.Insert(1, 11);
16             //inserts 11 at 1st index: after 10.
17             foreach (var num in numbers)
18             Console.WriteLine(num);
19             Console.ReadKey();
20         }
21     }
22 }
```

លទ្ធផល៖

១៣.៥.២ របៀបនៃការបញ្ចូល Remove និងនិយមន័យឱ្យ ArrayList Object

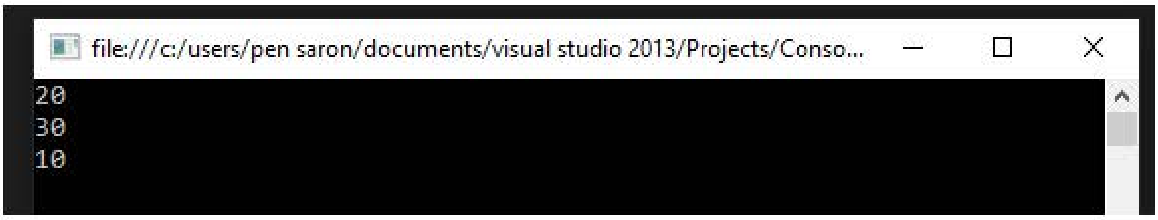
គេប្រើប្រាស់ Remove ( ) Method Remove ដើម្បី ដក ជាតុចេញទៅឱ្យ ArrayList Object List <Object> ដែលបានកំណត់ដោយតាមរយៈ Index ។ ប្រសិនបើយើងប្រើ RemoveAt( ) Method ដើម្បី Remove ជាតុ ចេញតាមរយៈការកំណត់ដោយ Index ។

ខាងក្រោមនេះជារបៀបប្រើប្រាស់ remove ArrayList object

Remove() signature: `bool Remove(T item)`

RemoveAt() signature: `void RemoveAt(int index)`

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

namespace ConsoleApplication5
{
    class Program
    {
        static void Main(string[] args)
        {
          var numbers = new List<int>() { 10, 20, 30, 40, 10 };

            numbers.Remove(10);
             // removes 10 elements from a list

            numbers.RemoveAt(2);
                //removes the 3rd element (index starts from 0)

            //numbers.RemoveAt(10);
             //removes the 3rd element (index starts from 0)

            foreach (var num in numbers)
                Console.WriteLine(num);
                Console.ReadKey();
        }
    }
}
```

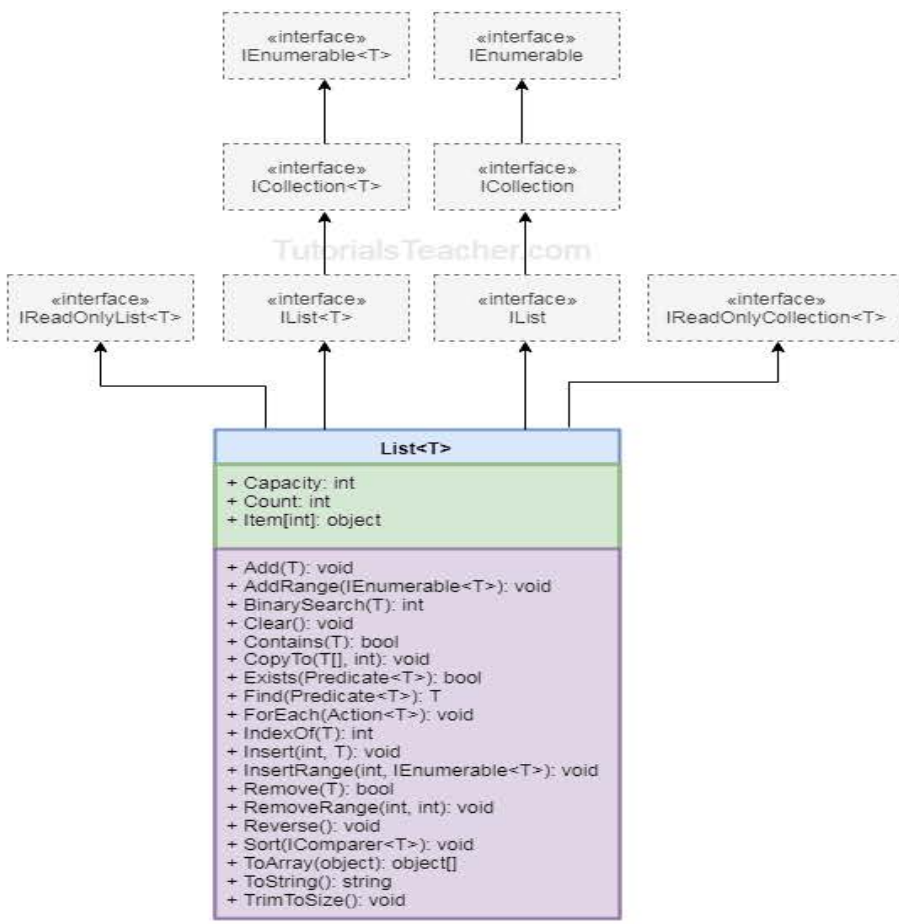


១៣.៥.៣ របៀបនៃការពិនិត្យ Check ទិន្នន័យឱ្យ ArrayList Object

Check Elements in List ប្រើប្រាស់ Contains( ) method ដើម្បីត្រួតពិនិត្យធាតុ របស់ ArrayList ដើម្បីកំណត់ថាតើធាតុ មាននៅក្នុងបញ្ជី List <object> ឬអត់។

```
1  sing System;
2
3  namespace ConsoleApplication5
4  {
5      class Program
6      {
7          static void Main(string[] args)
8          {
9              var numbers = new List<int>() { 10, 20, 30, 40 };
10             Console.WriteLine(numbers.Contains(10));
11             Console.WriteLine(numbers.Contains(11));
12             Console.WriteLine(numbers.Contains(20));
13
14             Console.ReadKey();
15         }
16     }
17 }
```

ដ្យាក្រាមនៃ List <T>



១៤. SortedList <TKey, TValue>

SortList <Key, Value> ហើយនឹង SortList គឺជា collection Class ដែលអាច Store key-value The SortedList<TKey, TValue>, and SortedList are collection classes that can store key-value pairs that are sorted by the keys based on the associated IComparer implementation. For example, if the keys are of primitive types, then sorted in ascending order of keys.

១៤.១ ការបង្កើត Creating a SortedList

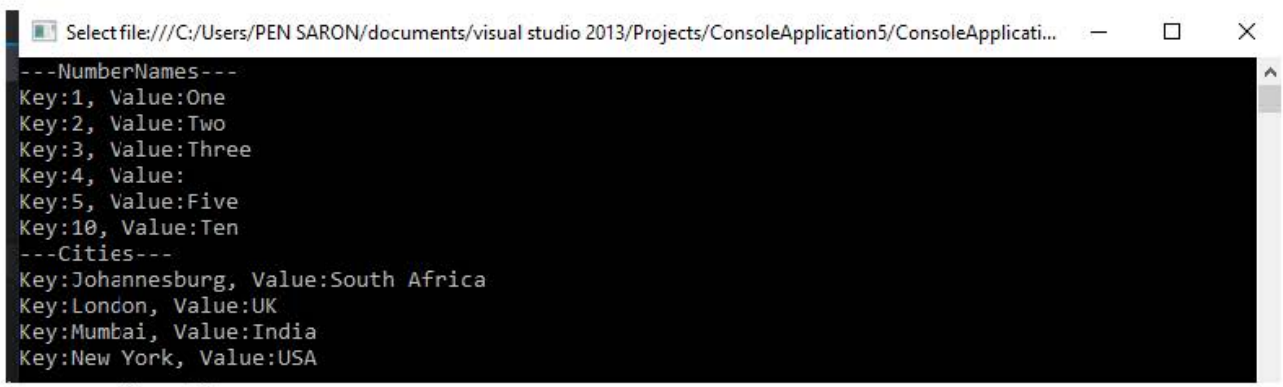
SortedList<TKey, TValue> object គឺជាការបង្កើត ប្រភេទobjects នៃ Keys និង Values ដើម្បីធ្វើការរក្សាទុក Store ។ SortedList<int, string> គឺវា store keys នៃ ប្រភេទចំនួនគត់ int type និង Values ជា ប្រភេទតួអក្ស String type ។

Add( ) method គឺប្រើដើម្បី Add key-value មួយ ដែលមាននៅក្នុង SortedList ។ Keys មិនអាច តម្លៃ ជា Null ឬ តម្លៃស្មុនគ្នា duplicate ទេ ។ ប្រសិនបើ វាស្វែងរកឃើញ វានឹងបោះចោលការលើកលែងពេលវាកំពុងដំណើរការ លើសំណើ ។ Values អាច មានតម្លៃស្មុនគ្នា Duplicate និង Null ប្រសិនបើ តម្លៃប្រភេទ nullable ។

sourer code:

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

namespace ConsoleApplication5
{
  class Program
   {
   public class Student
   {
       public int Id { get; set; }
       public string Name { get; set; }
   }

   static void Main(string[] args)
   {
       //SortedList of int keys, string values
     SortedList<int, string>numberNames=new SortedList<int,string>();
     numberNames.Add(3, "Three");
     numberNames.Add(1, "One");
     numberNames.Add(2, "Two");
     numberNames.Add(4, null);
     numberNames.Add(10, "Ten");
     numberNames.Add(5, "Five");
```

```
28
29
30          Console.WriteLine("---NumberNames---");
31          foreach (var kvp in numberNames)
32          Console.WriteLine("Key:{0}, Value:{1}", kvp.Key, kvp.Value);
33          SortedList<string, string> cities = new SortedList<string,
34                                      string>()
35                      {
36                          {"London", "UK"},
37                          {"New York", "USA"},
38                          { "Mumbai", "India"},
39                          {"Johannesburg", "South Africa"}
40                      };
41          Console.WriteLine("---Cities---");
42          foreach (var kvp in cities)
43          Console.WriteLine("Key:{0}, Value:{1}", kvp.Key, kvp.Value);
44          Console.ReadKey();
45      }
46
47      }
48  }
```

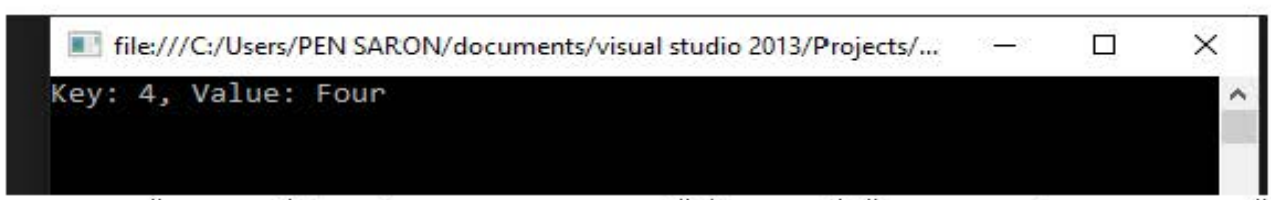


១៤.២ ដំណើរការ Accessing SortedList

បញ្ជាក់ key ធ្វើការ sortedList[key] នៅក្នុង index ដើម្បីចាប់ទទួលតម្លៃ get ឬ set values នៅក្នុង ប្រភេទ SortedList ។ ដើម្បីបង្ហាញលើសំណើនេះ គេប្រើប្រាស់ ContainsKey( ) ឬ TryGetValue( ) methods ដូចក្នុងឧទាហរណ៍ខាងក្រោម៖ Sourer code:

```
1  using System;
2  using System.Collections;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace ConsoleApplication5
9  {
10     class Program
11     {
12    public class Student
13    {
```

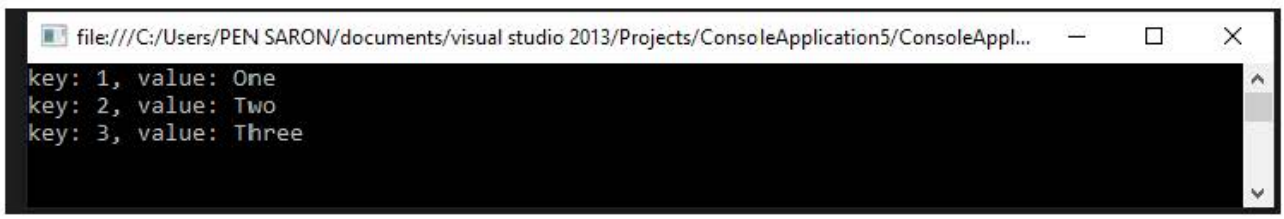
```
14          public int Id { get; set; }
15          public string Name { get; set; }
16      }
17
18      static void Main(string[] args)
19      {
20      SortedList<int, string> numberNames = new SortedList<int, string>()
21                  {
22                      {3, "Three"},
23                      {1, "One"},
24                      {2, "Two"}
25                  };
26
27          if (!numberNames.ContainsKey(4))//check if key exists
28          {
29              numberNames[4] = "Four";
30          }
31
32          string result;
33
34          if (numberNames.TryGetValue(4, out result))
35              //try to get value of 4 key
36          Console.WriteLine("Key: {0}, Value: {1}", 4, result);
37          Console.ReadKey();
38      }
39
40       }
41  }
```



តែប្រើ Keys និង Values properties បើសិន អ្នកចង់ធ្វើការ SortedList អ្នកអាចប្រើប្រាស់ for loop ។

```
1   using System;
2   using System.Collections;
3   using System.Collections.Generic;
4   using System.Linq;
5   using System.Text;
6   using System.Threading.Tasks;
7
8   namespace ConsoleApplication5
9   {
10      class Program
11      {
12
13      static void Main(string[] args)
14      {
15      SortedList<int, string> numberNames = new SortedList<int, string>()
16              {
```

```
17
18              {3, "Three"},
19              {1, "One"},
20              {2, "Two"}
21          };
22
23        for (int i = 0; i < numberNames.Count; i++)
24        {
25          Console.WriteLine("key: {0}, value: {1}",numberNames.Keys[i],
26           numberNames.Values[i]);
27        }
28        Console.ReadKey();
29    }
```



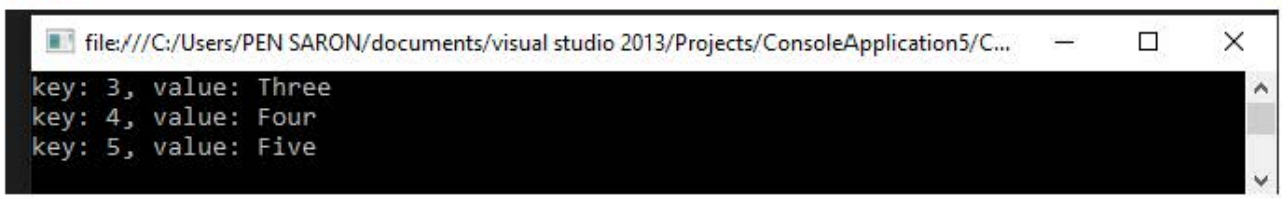
១៤.៣ ការលុប Remove Elements from SortedList

គេប្រើ Remove(key) និង RemoveAt(index) methods ដើម្បី Remove key-value ចេញពីទីតាំង SortedList ។

sourer code:

```
1  using System;
2  using System.Collections;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace ConsoleApplication5
9  {
10  class Program
11    {
12
13    static void Main(string[] args)
14    {
15    SortedList<int, string> numberNames = new SortedList<int, string>()
16                        {
17                            {3, "Three"},
18                            {1, "One"},
19                            {2, "Two"},
20                            {5, "Five"},
21                            {4, "Four"},
22                        };
```

```
23              numberNames.Remove(1);//removes key 1 pair
24              numberNames.Remove(10);//removes key 10.
25
26              numberNames.RemoveAt(0);
27                              //removes key-value pair from index 0
28
29          foreach (var kvp in numberNames)
30
31           Console.WriteLine("key: {0}, value: {1}", kvp.Key, kvp.Value);
32           Console.ReadKey();
33         }
34
35     }
   }
```



១៥. ការប្រើប្រាស់ Dictionary

Dictionary ជា class មួយដែលជាពពួក Collection ដែរដែលវាប្រើ key និង value ដើម្បីអាច ប្រើប្រាស់ Dictionary បានត្រូវបន្ថែម namespaceSystem.Collections.Gemerics ។

Dictionary Characteristics

- Dictionary<TKey, TValue> stores key-value pairs.
- Comes under System.Collections.Generic namespace.
- Implements IDictionary<TKey, TValue> interface.
- Keys must be unique and cannot be null.
- Values can be null or duplicate.
- Values can be accessed by passing associated key in the indexer e.g. myDictionary[key]
- Elements are stored as KeyValuePair<TKey, TValue> objects.

ទម្រង់ទូទៅនៃការបង្កើត Dictionary object

```
Dictionary<keys, Values> dictionary_name;
Dictionary<string, string> dict;
Dictionary<string, string> dict=new Dictionary<string, string>();
```

១៥.១ របៀបនៃការ add ធាតុយៃទៅកាន់ Dictionary object

```
Dict = new Dictionary<string, string>();
Dict.Add("ABC","Beer");
```

១៥.២ ការបង្កើត Creating a Dictionary

អ្នកអាចបង្កើត Dictionary<keys, Values> objects ដោយចម្លងប្រភេទ នៃKeys និង Values ដែលវាអាចរក្សាទុក ។

ឧទាហរណ៍៖

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

namespace ConsoleApplication5
{
 class Program
{

  static void Main(string[] args)
  {
      IDictionary<int, string> numberNames = new Dictionary<int, string>();
      numberNames.Add(1, "One"); //adding key/value using the Add() method
      numberNames.Add(3, "Three");
      numberNames.Add(2, "Two");

      foreach (KeyValuePair<int, string> kvp in numberNames)
          Console.WriteLine("Key: {0}, Value: {1}", kvp.Key, kvp.Value);

      //creating a dictionary using collection-initializer syntax
      var cities = new Dictionary<string, string>(){
                  {"UK","London, Manchester, Birmingham"},
                  {"USA","Chicago, New York, Washington"},
                  {"India","Mumbai, New Delhi, Pune"}
          };

      foreach (var kvp in cities)
          Console.WriteLine("Key: {0}, Value: {1}", kvp.Key, kvp.Value);
          Console.ReadKey();
  }
}
```

```
Select file:///C:/Users/PEN SARON/documents/visual studio 2013/Projects/ConsoleApplic...
Key: 1, Value: One
Key: 3, Value: Three
Key: 2, Value: Two
Key: UK, Value: London, Manchester, Birmingham
Key: USA, Value: Chicago, New York, Washington
Key: India, Value: Mumbai, New Delhi, Pune
```

១៥.៣ ការ Update Dictionary

ធ្វើបច្ចុប្បន្នភាពតម្លៃ Update value នៃ key ដោយបញ្ជាក់ key នៅក្នុង index ។ វានឹងបោះ KeyNotFoundException ប្រសិនបើ key មិនមាននៅក្នុង dictionary ដូច្នេះសូមប្រើ ContainsKey( ) method មុនពេលចូលប្រើ key ដែលមិនស្គាល់។

```
var cities = new Dictionary<string, string>()
{
       {"UK", "London, Manchester, Birmingham"},
       {"USA", "Chicago, New York, Washington"},
       {"India", "Mumbai, New Delhi, Pune"}
};
cities["UK"] = "Liverpool, Bristol"; // update value of UK key
cities["USA"] = "Los Angeles, Boston"; // update value of USA key
//cities["France"]    =    "Paris";    //throws    run-time    exception:
KeyNotFoundException

if(cities.ContainsKey("France")){
   cities["France"] = "Paris";
}
```

១៥.៤ ការ Remove Elements in Dictionary

វិធីសាស្ត្រ method Remove( ) លុបគូតម្លៃ key-value ដែលមានស្រាប់ចេញពី dictionary ចែនានុក្រម។ method Clear( ) លុបធាតុទាំងអស់នៃ dictionary ចែនានុក្រម។

```
var cities = new Dictionary<string, string>(){
       {"UK", "London, Manchester, Birmingham"},
       {"USA", "Chicago, New York, Washington"},
       {"India", "Mumbai, New Delhi, Pune"}
};
cities.Remove("UK"); // removes UK

if(cities.ContainsKey("France")){ // check key before removing it
   cities.Remove("France");
}
cities.Clear(); //removes all elements
```

១៥.៥ ដំណើរការ Access Dictionary Elements

Dictionary អាចដំណើរការទៅបានដោយប្រើប្រាស់តាមរយៈ index ។ បញ្ជាក់គន្លឹះដើម្បីទទួលបានតម្លៃ ។ អ្នកក៏អាចប្រើ ElementAt( ) method ដើម្បីទទួលបាន Key Value ដែលកំណត់តាមរយៈ index ។

```
var cities = new Dictionary<string, string>()
{
       {"UK", "London, Manchester, Birmingham"},
       {"USA", "Chicago, New York, Washington"},
       {"India", "Mumbai, New Delhi, Pune"}
};
Console.WriteLine(cities["UK"]); //prints value of UK key
Console.WriteLine(cities["USA"]);//prints value of USA key
//use ContainsKey() to check for an unknown key
if(cities.ContainsKey("France")){
    Console.WriteLine(cities["France"]);
}
//use TryGetValue() to get a value of unknown key
string result;
if(cities.TryGetValue("France", out result))
{
    Console.WriteLine(result);
}
//use ElementAt() to retrieve key-value pair using index
for (int i = 0; i < cities.Count; i++)
{
    Console.WriteLine("Key: {0}, Value: {1}",
                                        cities.ElementAt(i).Key,
                                        cities.ElementAt(i).Value);
}
```

១៦. Hashtable

Hashtable គឺជា non-generic collection ដែលរក្សាទុកនូវទាំង key-value ជាទូទៅវាស្រដៀងទៅនឹង dictionary<TKey, Tvalue> collection ការប្រមូល។ វាបង្កើនប្រសិទ្ធភាពការរកមើលដោយគណនា hash code នៃ key នីមួយៗ ហើយ stores រក្សាទុកវានៅក្នុង bucket ធុងផ្សេងគ្នានៅខាងក្នុង ហើយបន្ទាប់មកត្រូវគ្នានឹងលេខ hash code នៃ key ដែលបានបញ្ជាក់ time នៅពេល accessing values ចូលប្រើតម្លៃ។

Hashtable គឺជាប្រភេទមួយនៃ collection ដែលធាតុនីមួយៗរបស់វាអាចផ្ទុកទិន្នន័យជា object ដែលអាចឲ្យយើង អាច access វាបានតាមរយៈ Key។ Key គឺជាតំលៃមួយដែលមិនស្ទួនក្នុង Hashtable ហើយវាជាតំណាងឲ្យ value ណាមួយក្នុង Hashtable ។

Property	Description
Count	រាប់ចំនួនធាតុជាក់ស្តែងក្នុង ArrayList
IsFixedSize	ចង់ដឹងថាតើ ArrayList វា fixed size រឺអត់
IsReadOnly	ចង់ដឹងថាតើ ArrayList វា read-only រឺអត់
Item	ផ្តល់តំលៃ រឺ ចាប់ធាតុណាមួយរបស់ ArrayList តាមរយៈ Key

Methods	Description
Add(object key, object value)	បន្ថែម ១ ធាតុចូលទៅក្នុង Hashtable
Clear( );	លុបធាតុទាំងអស់របស់ Hashtable
ContainsKey(object key);	ត្រួតពិនិត្យ check មើលថាតើធាតុនឹងមួយមានក្នុង Hashtable នេះរឺអត់ តាមរយៈ key
ContainsValue (object value)	ត្រួតពិនិត្យ check មើលថាតើធាតុនឹងមួយមានក្នុង Hashtable នេះរឺអត់ តាមរយៈ value

Remove(object key);	លុបធាតុ Hashtable តាមរយៈ key

១៦.១ Hashtable Characteristics

`Hashtable` stores key-value pairs.

- Comes under `System.Collections` namespace.
- Implements IDictionary interface.
- Keys must be unique and cannot be null.
- Values can be null or duplicate.
- Values can be accessed by passing associated key in the indexer e.g. `myHashtable[key]`
- Elements are stored as DictionaryEntry objects.

១៦.២ ការបង្កើត Hashtable

ឧទាហរណ៍ខាងក្រោមបង្ហាញពីការបង្កើត Hashtable និង adding elements បន្ថែមធាតុ។

```
Hashtable numberNames = new Hashtable();
numberNames.Add(1,"One"); //adding a key/value using the Add() method
numberNames.Add(2,"Two");
numberNames.Add(3,"Three");

//The following throws run-time exception: key already added.
//numberNames.Add(3, "Three");

foreach(DictionaryEntry de in numberNames)
    Console.WriteLine("Key: {0}, Value: {1}", de.Key, de.Value);

//creating a Hashtable using collection-initializer syntax
var cities = new Hashtable(){
       {"UK", "London, Manchester, Birmingham"},
       {"USA", "Chicago, New York, Washington"},
       {"India", "Mumbai, New Delhi, Pune"}
};
foreach(DictionaryEntry de in cities)
    Console.WriteLine("Key: {0}, Value: {1}", de.Key, de.Value);
```

ការប្រមូល Hashtable អាចរួមបញ្ចូលធាតុទាំងអស់នៃ dictionary ដូចដែលបានបង្ហាញខាងក្រោម។

```
Dictionary<int, string> dict = new Dictionary<int, string>();
dict.Add(1, "one");
dict.Add(2, "two");
dict.Add(3, "three");

Hashtable ht = new Hashtable(dict);
```

១៦.៣ ការUpdate Hashtable

អ្នកអាចទាញយកតម្លៃនៃសៅដែលមានស្រាប់ពី Hashtable ដោយឆ្លងកាត់ key នៅក្នុង index។ Hashtable គឺជា non-generic collection បណ្តុំដែលមិនមែនជាទូទៅ ដូច្នេះអ្នកត្រូវតែវាយបញ្ចូលប្រភេទតម្លៃ cast ខណៈពេលដែលទាញយកវាមកវិញ។

```
//creating a Hashtable using collection-initializer syntax
var cities = new Hashtable(){
        {"UK", "London, Manchester, Birmingham"},
        {"USA", "Chicago, New York, Washington"},
        {"India", "Mumbai, New Delhi, Pune"}
};

string citiesOfUK = (string) cities["UK"]; //cast to string
string citiesOfUSA = (string) cities["USA"]; //cast to string

Console.WriteLine(citiesOfUK);
Console.WriteLine(citiesOfUSA);

cities["UK"] = "Liverpool, Bristol"; // update value of UK key
cities["USA"] = "Los Angeles, Boston"; // update value of USA key

if(!cities.ContainsKey("France")){
    cities["France"] = "Paris";
}
```

១៦.៤ ការ Remove Elements in Hashtable

Remove( ) វិធីសាស្ត្រលុបនូវ key-value ដែលត្រូវគ្នានឹងអ្វីដែលបានបញ្ជាក់នៅក្នុង Hashtable។ វាបោះ KeyNotfoundException ប្រសិនបើ key ដែលបានបញ្ជាក់មិនត្រូវបានរកឃើញនៅក្នុង Hashtable ដូច្នេះសូមពិនិត្យមើល key ដែលមានស្រាប់ដោយប្រើវិធី method ContainsKey( ) មុននឹងលុបចេញ removing ។ ប្រើ method Clear( ) ដើម្បីលុបធាតុទាំងអស់ចេញក្នុងសំណុំ ។

Example: Remove Elements from Hashtable

```
var cities = new Hashtable(){
        {"UK", "London, Manchester, Birmingham"},
        {"USA", "Chicago, New York, Washington"},
        {"India", "Mumbai, New Delhi, Pune"}
};

cities.Remove("UK"); // removes UK

if(cities.ContainsKey("France")){ // check key before removing it
    cities.Remove("France");
}
cities.Clear(); //removes all elements
```

១៧. Stack

វាត្រូវបានគេហៅថា last-in(ចូលក្រោយ, first out(ចេញមុន) collection of object។ វាត្រូវបានគេប្រើ ជាមួយ item ទាំងឡាយណាដែលជាប្រភេទ last-in, first out ។ ពេលយើង add item គេហៅថា pushing ហើយពេលដែលយើង remove item ចេញវិញគេហៅថា popping ។

Stack គឺជាប្រភេទ collection ពិសេស ដែលរក្សាទុក ធាតុនៅក្នុង LIFO style (Last In First Out) ។ C# រួមបញ្ចូលថ្នាក់ទូទៅ Stack និងnon-generic Stack collection classes ដែលមិនមែនជាទូទៅ។ វាត្រូវបានណែនាំឱ្យប្រើ generic Stack<T> collection បណ្តុំ Stack ទូទៅ។

Stack គឺមានប្រយោជន៍ក្នុងការរក្សាទុកទិន្នន័យបណ្តោះអាសន្នក្នុងរចនាប័ទ្ម LIFO ហើយអ្នកប្រហែលជាចង់លុប delete ធាតុមួយបន្ទាប់ពីទាញយកតម្លៃរបស់វា។

Stack is Last In First Out collection.

•It comes under System.Collection.Generic namespace.

•Stack can contain elements of the specified type. It provides compile-time type checking and doesn't perform boxing-unboxing because it is generic.

•Elements can be added using the Push( ) method. Cannot use collection-initializer syntax.

•Elements can be retrieved using the Pop( ) and the Peek( ) methods. It does not support an indexer.

១៧.១ ការបង្កើត Stack

អ្នកអាចបង្កើត object នៃ Stack ដោយបញ្ជាក់ប៉ារ៉ាម៉ែត្រប្រភេទសម្រាប់ប្រភេទនៃធាតុដែលវាអាចផ្ទុកបាន។ ឧទាហរណ៍ខាងក្រោមបង្កើត និងបន្ថែមធាតុនៅក្នុង Stack ដោយប្រើវិធី Push( ) ។ Stack អនុញ្ញាតឱ្យចាត់ទុកជា null (សម្រាប់ប្រភេទឯកសារយោង reference types) និងតម្លៃស្ទួន។

Example: Create and Add Elements in Stack

```
Stack<int> myStack = new Stack<int>();
myStack.Push(1);
myStack.Push(2);
myStack.Push(3);
myStack.Push(4);

foreach (var item in myStack)
    Console.Write(item + ","); //prints 4,3,2,1,
```

Example: Create and Add Elements in Stack

```
int[] arr = new int[]{ 1, 2, 3, 4};
Stack<int> myStack = new Stack<int>(arr);

foreach (var item in myStack)
    Console.Write(item + ","); //prints 4,3,2,1,
```

Stack Properties and Methods:

Property	Usage
Count	Returns the total count of elements in the Stack.

Method	Usage stack.
Push(T)	Inserts an item at the top of the stack.
Peek()	Returns the top item from the stack.
Pop()	Removes and returns items from the top of the
Contains(T)	Checks whether an item exists in the stack or not.
Clear()	Removes all items from the stack.

១៨. Queue

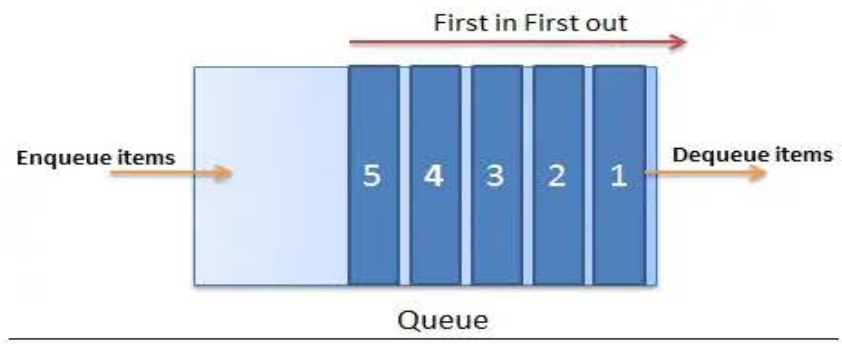
Queue គឺជាប្រភេទពិសេសនៃការប្រមូល collection ដែលរក្សាទុកធាតុនៅក្នុង FIFO style (ដំបូងក្នុងការចេញដំបូង First In First Out) ផ្ទុយស្រឡះពី stack collection ។ វាមាន contains ធាតុនៅក្នុងលំដាប់ដែលពួកគេត្រូវបានបន្ថែម។ C# រួមបញ្ចូល Queue ទូទៅ និង non-generic Queue collection មិនទូទៅ។

១៨.១ Queue Characteristics

Queue គឺជា FIFO (First In First Out) collection

- វាស្ថិតនៅក្រោម System.Collection.Generic namespace
- Queue អាចមានធាតុនៃប្រភេទដែលបានបញ្ជាក់។ វាផ្តល់នូវការត្រួតពិនិត្យ compile-time type និងមិនអនុវត្តការបើកប្រអប់ boxing-unboxing ព្រោះវាមានលក្ខណៈទូទៅ។
- ធាតុអាចត្រូវបានបន្ថែម added ដោយប្រើវិធីសាស្ត្រ method Enqueue( ) ។ មិនអាចប្រើវាក្យសម្ព័ន្ធការប្រមូលផ្តុំដំបូង collection-initializer syntax ។
- ធាតុអាចត្រូវបានទាញយកដោយប្រើវិធី Dequeue( ) និង Peek( ) methods វាមិនគាំទ្រអ្នកបង្កើត index ទេ។

figure illustrates the Queue collection ខាងក្រោម៖



១៨.២ ការបង្កើត Queue

អ្នកអាចបង្កើតវត្ថុនៃ Queue ដោយបញ្ជាក់ ប៉ារ៉ាម៉ែត្រប្រភេទសម្រាប់ប្រភេទនៃធាតុដែលវាអាចផ្ទុកបាន។ ឧទាហរណ៍ខាងក្រោមបង្កើត និងបន្ថែមធាតុនៅក្នុង Queue ដោយប្រើ method Enqueue( ) ។ ការប្រមូលផ្ដុំមួយអនុញ្ញាតឱ្យទុកជា null (សម្រាប់ reference types) និងតម្លៃស្ទួន។

Example: Create and Add Elements in the Queue

```
Queue<int> callerIds = new Queue<int>();
callerIds.Enqueue(1);
callerIds.Enqueue(2);
callerIds.Enqueue(3);
callerIds.Enqueue(4);

foreach(var id in callerIds)
    Console.Write(id); //prints 1234
```

១៨.៤ Properties និង Methods របស់ Queue

Property	Usage
Count	Returns the total count of elements in the Queue.

Method	Usage
Enqueue(T)	Adds an item into the queue.
Dequeue	Returns an item from the beginning of the queue and removes it from the queue.
Peek()	Returns an first item from the queue without removing it.
Contains(T)	Checks whether an item is in the queue or not
Clear()	Removes all the items from the queue.

១៨.៥ Retrieve Elements from a Queue

method Dequeue( ) និង method Peek( ) ត្រូវបានប្រើដើម្បីទាញយកធាតុទីមួយនៅក្នុង queue collection បណ្តុំជួរមួយ។ Dequeue( ) removes ដកចេញ និងបញ្ជូនធាតុទីមួយចេញពី queue ពីព្រោះ queue ផ្ទុកធាតុនៅក្នុងលំដាប់ FIFO ។ ការហៅទៅវិធីសាស្ត្រ method Dequeue( ) នៅលើ queue ទទេ (empty queue) នឹងបោះបង់ ការលើកលែង InvalidOperation ។

ដូច្នេះ សូមពិនិត្យមើលថាចំនួនសរុបនៃ queue គឺធំជាងសូន្យមុនពេលហៅវា។

Example: Reading Queue

```
Queue<string> strQ = new Queue<string>();
strQ.Enqueue("H");
strQ.Enqueue("e");
strQ.Enqueue("l");
strQ.Enqueue("l");
strQ.Enqueue("o");
Console.WriteLine("Total elements: {0}", strQ.Count); //prints 5
while (strQ.Count > 0)
    Console.WriteLine(strQ.Dequeue()); //prints Hello
    Console.WriteLine("Total elements: {0}", strQ.Count); //prints 0
```

វិធីសាស្ត្រ Peek( ) តែងតែត្រឡប់ធាតុដំបូងពី queue collection ដោយមិនដកវាចេញពី queue ។ ការហៅទូរសព្ទទៅវិធី method Peek( ) នៅលើ queue ទទេ នឹងបោះបង់ការលើកលែង run-time ពេលដំណើរការ InvalidOperationException។

Example: Peek()

```
Queue<string> strQ = new Queue<string>();
strQ.Enqueue("H");
strQ.Enqueue("e");
strQ.Enqueue("l");
strQ.Enqueue("l");
strQ.Enqueue("o");

Console.WriteLine("Total elements: {0}", strQ.Count); //prints 5

if(strQ.Count > 0){
    Console.WriteLine(strQ.Peek()); //prints H
    Console.WriteLine(strQ.Peek()); //prints H
}
Console.WriteLine("Total elements: {0}", strQ.Count); //prints 5
```

១៨.៦ ការប្រើ Contains ( )

method Contains( ) method ពិនិត្យថាតើមានធាតុនៅក្នុង queue ឬអត់។ វាត្រឡប់ true ពិត ប្រសិនបើ item ធាតុដែលបានបញ្ជាក់មាន បើមិនដូច្នោះទេ ត្រឡប់មិនពិត false ។

Contains( ) Signature: bool Contains(object obj);

Example: Contains()

```
Queue<int> callerIds = new Queue<int>();
callerIds.Enqueue(1);
callerIds.Enqueue(2);
callerIds.Enqueue(3);
callerIds.Enqueue(4);

callerIds.Contains(2); //true
callerIds.Contains(10); //false
```

ជំពូកទី ៨

Methods

ក្នុងមេរៀននេះយើងស្វែងយល់គោលបំណង និងបញ្ហាដែលកើតមានរាល់កម្មវិធីដែលយើងបម្រុងនិងបង្កើត ។ ការយល់ពីបញ្ហាទាំងនោះនឹងធ្វើឱ្យយើងអាចកំណត់បាននូវអ្វីដែលយើងត្រូវធ្វើវា និងជាចំណុចចាប់ផ្តើមក្នុងការបំបែកបញ្ហាទាំងនោះឱ្យទៅជាផ្នែកតូចៗក្នុងការសរសេរកូដ ដោយការធ្វើបែបនេះវានឹងនាំឱ្យអ្នកងាយស្រួលក្នុងការគ្រប់គ្រងកូដ និងជាពិសេសនៅពេលមានការស្វែងរកកំហុស និងការ Update កូដរបស់អ្នក ។ ការសរសេរដើម្បីដោះស្រាយបញ្ហាតូចៗទាំងមូលនោះវាគឺជាការបង្កើត Method ដែលមានន័យប្រហាក់ប្រហែលគ្នានឹងពាក្យ "អនុគមន៍ (Function)" ។ យើងនឹងពិភាក្សាប្រើប្រាស់ Method ដោយចាប់ផ្តើមពីការស្វែងយល់នៃកម្មវិធីក្នុងរូប ខាងក្រោយ ។

វិធីសាស្ត្រគឺជាប្លុកនៃកូដដែលដំណើរការតែនៅពេលដែលវាត្រូវបានហៅ។អ្នកអាចបញ្ជូនទិន្នន័យដែលគេស្គាល់ថាជាប៉ារ៉ាម៉ែត្រទៅក្នុងវិធីសាស្ត្រមួយ។ វិធីសាស្ត្រត្រូវបានប្រើដើម្បីអនុវត្តសកម្មភាពជាក់លាក់ហើយវាត្រូវបានគេស្គាល់ថាជាមុខងារផងដែរ។

ហេតុអ្វីត្រូវប្រើវិធីសាស្ត្រ? ដើម្បីប្រើកូដឡើងវិញ៖ កំណត់កូដម្តង ហើយប្រើវាច្រើនដង។ Method ត្រូវបានកំណត់ដោយឈ្មោះនៃ method បន្តដោយវង់ក្រចក ()។

C# ផ្តល់នូវវិធីសាស្ត្រដែលបានកំណត់ជាមុនមួយចំនួន ដែលអ្នកធ្លាប់ស្គាល់រួចមកហើយ ដូចជា Main() ប៉ុន្តែអ្នកក៏អាចបង្កើតវិធីសាស្ត្រផ្ទាល់ខ្លួនរបស់អ្នក ដើម្បីអនុវត្តន៍សកម្មភាពមួយចំនួន៖

១. ទម្រង់ទូទៅការបង្កើត Method

យើងអាចបង្កើត Method បានតាមការអនុវត្តទម្រង់ទូទៅដូចខាងក្រោម៖

```
<Date Type><Method_Name> (<Parameters>)
{
        <Block_Code>
}
```

- <Data_Type> បញ្ជាក់ពីប្រភេទនៃការផ្តល់តម្លៃត្រឡប់មកវិញដោយMethod ក្នុងដែលយើងប្រើប្រាស់ Void ជា Data Type របស់ Method ។ Void ប្រើសម្រាប់បញ្ជាក់ថាគ្មានទិន្នន័យដែលត្រូវផ្តល់ឱ្យដោយ Method ។
- <Method Name> ជាការដាក់ឈ្មោះឱ្យ Method ដែលត្រូវបង្កើត ។ ការកំណត់ឈ្មោះ Method ត្រូវអនុវត្តន៍ដូចគ្នានឹងការដាក់ឈ្មោះឱ្យ Variable ។
- <Parameters> ជា Local variable របស់ Method ដែលផ្តល់លទ្ធភាពឱ្យ method អាចចាប់យកតម្លៃពីផ្នែកផ្សេងៗទៅអនុវត្តន៍នៅក្នុងដំណើរធ្វើការរបស់ Method នោះលើសពីនេះ Parameters ក៏អាច

ធ្វើការផ្លាស់ប្តូរតម្លៃរបស់ Variable ដែលបានភ្ជាប់ទៅកាន់ Parameters ទាំងនោះផងដែរ ។ យើងនឹងបានសិក្សាពីការប្រើប្រាស់ Parameters នៅផ្នែកចុងក្រោយនៃមេរៀននេះ ។

- <Block_Code> ជាបណ្តុំនៃកូដដែលត្រូវសរសេរសម្រាប់ការធ្វើការរបស់ Method ហើយវាត្រូវតែសរសេរ ក្នុងចន្លោះសញ្ញា {} ។

២. ប្រកាស Method

ក្នុង method គឺជាការប្រកាស statement ដែលបានដាក់ឈ្មោះ ។ ក្នុង method នីមួយៗ មានឈ្មោះ និងខ្លួនរបស់វា ។ ក្នុង method body មានផ្នែកសេចក្តីប្រកាស ជាច្រើនការប្រកាសពិតប្រាកដ៏ដើម្បីតំណើរការកាលណា method ត្រូវបានហៅមកប្រើ ។ ឈ្មោះ method និងមានន័យពេញលេញ (meaningful identifier) ដែលបង្ហាញគោលបំណងរួមនៃ method (ឧទាហរណ៍ដូចជា CalculateIncomeTax) ។ មាន method ជាច្រើនអាច បញ្ជូនផ្តល់ data មួយចំនួនដើម្បីឱ្យវាដំណើរការ និងព័ត៌មានត្រឡប់ដែលតែងតែផ្តល់លទ្ធផលនៃដំណើរការ ។ Method នេះគឺជាមូលដ្ឋានដែលមាន កម្លាំងខ្លាំង ។ ចំណុចសំខាន់ៗនៃ Method Declaration Syntax Microsoft Visual C# មានដូចខាងក្រោមនេះ

```
returnType methodName ( parameterList )
{
// method body statements go here
}
// បណ្តុ Command line របស់ method
}
```

- < returnType> គឺជាឈ្មោះនៃប្រភេទនិងលក្ខណៈប្រភេទនៃពត៌មានក្នុង method returns ។ ប្រភេទនេះគឺជាឈ្មោះនៃប្រភេទ ដូចជា int ឬ string ។ ប្រសិនបើសរសេរ method មួយដែលមិនត្រលប់ទៅតម្លៃអ្នកត្រូវប្រើ keyword ក្នុងកន្លែងប្រភេទ return type ។
- <methodName> គឺជាឈ្មោះដែលប្រើដើម្បីហៅ method ។ ដែល ឈ្មោះ Method names ត្រូវធ្វើតាមដូចតួនាទីដូចជាឈ្មោះ variable name ឧទាហរណ៍ addValues គឺជាឈ្មោះ method name ត្រឹមត្រូវ ទោះបីជា add$Values មិនបានកំណត់ ត្រឹមត្រូវ ។ ឥឡូវនេះអ្នកនឹងប្រើ camelCase សម្រាប់ឈ្មោះ method names ហើយអ្នកនឹងបង្កើតឈ្មោះ method names ។
- <parameterList> គឺជាប្រភេទពត៌មានដែលអធិប្បាយពីប្រភេទនិង នៃពត៌មានដែល method អនុញ្ញាត ។ អ្នកសរសេរ parameters ចន្លោះពីឆ្វេង ទៅស្តាំដៃដូចជាឆ្លងកាត់ការប្រកាសអថេរ: ឈ្មោះនៃប្រភេទដែលមានឈ្មោះនៃ parameter ។ ប្រសិនបើ method ដែលអ្នកកំពុងសរសេរឬមាន parameters, ជាច្រើនអ្នកត្រូវបែងចែកវាដោយប្រើសញ្ញាcommas ។
- <method body statements> គឺជាបន្ទាត់នៃ code ដែលត្រូវតំណើរការកាលណា method ត្រូវបានហៅចេញ ។ វាប្រើសញ្ញាបើក ({) និង សញ្ញាបើក (}) ។

ខាងក្រោមនេះនិយមន័យនៃ method ត្រូវបានហៅ addValues ដែលតម្លៃត្រឡប់ int និងមាន int parameters ពីរដែលហៅថា lhs និង rhs (short for left-hand side និង right-hand side):

```
int addValues( int leftHandSide, int rightHandSide )
{
        // ...
        // method body statements go here
        // ...
}
         // បណ្តា command ក្នុង method body
}
```

ខាងក្រោមនេះគឺជានិយមន័យនៃ method ដែលហៅថា showResult ដែលតម្លៃមិនត្រឡប់គឺមាន single int គឺជាប្រភេទ parameter ដែលហៅថា answer:

```
void showResult( int answer )
{
        // method body statements go here
}
```

ចំណាំ: អ្នកប្រើ keyword ដើម្បីបង្ហាញ method មិនធ្វើការត្រឡប់។ អ្នកសរសេរ Visual Basic programmers នឹងកំណត់ចំណាំ C# មិនប្រើច្បាស់ keywords រវាង method ដែលតម្លៃត្រឡប់ (ក្នុងអនុគមន៍) ហើយ method ដែលតម្លៃមិនត្រឡប់ (procedure ឬ subroutine) ។ ប្រសិនបើអ្នកចង់បាន method របស់អ្នកដើម្បីត្រឡប់ព័ត៌មានអ្នកត្រូវ សរសេរ return statement នៅក្នុង method ។ អ្នកធ្វើបែបនេះដោយប្រើ keyword return ដែលធ្វើតាមដោយប្រើ expression ដែលគណនាតម្លៃត្រឡប់និង semicolon ។ ប្រភេទ expression ត្រូវដូចនិងលក្ខណៈពិសេស អនុគមន៍ ។ អនុគមន៍ត្រឡប់ function returns int និងការប្រកាសត្រឡប់ return statement ត្រូវត្រឡប់ int ។

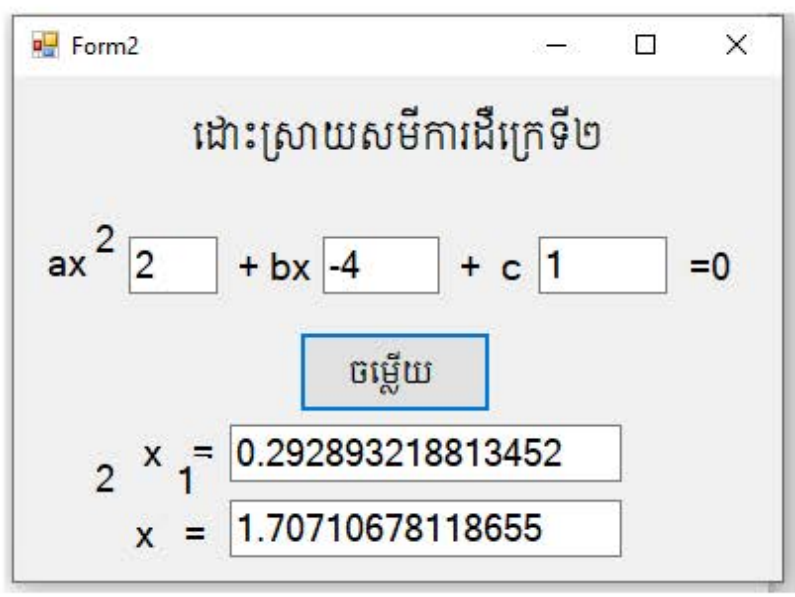
ឧទាហរណ៍

```
int addValues( int leftHandSide, int rightHandSide )
{
// ...
return leftHandSide + rightHandSide;
}
```

ការប្រកាស return statement ត្រូវចុងក្រោយបំផុតរបស់ method របស់អ្នកវាបណ្តាលឱ្យ method បញ្ចប់ finish ។ ក្នុងសេចក្តីប្រកាសបន្ទាប់ពី ប្រកាសត្រឡប់ return statement និងមិនប្រតិបត្តិការ (ឆ្លងកាត់ compiler និង ព្រមានអំពីបញ្ហា problem នេះប្រសិនបើ បញ្ចូល សេចក្តីប្រកាស statements បន្ទាប់ពី return statement) ។ ប្រសិនបើអ្នកមិនចង់បាន method របស់អ្នក ព័ត៌មានត្រឡប់អ្នកអាចប្រើផ្លាស់ប្តូរនៃ return statement ដែលបណ្តាលពីមូលហេតុបញ្ចប់ភ្លាមៗពី method ។ អ្នកសរសេរ keywordreturn?ទាហរណ៍

```
void showResult( int answer )
{
        //statement
}
```

ប្រសិនបើ method របស់អ្នកមិន return អ្វីទាំងអស់អ្នកអាចលុប return statement ពីព្រោះ method នឹងបញ្ចប់ដោយ automatically កាលណាការប្រតិបត្តិចូលមកដល់ក្នុង ពេល បិទ ឬការបញ្ចប់នៃ method ។ ទោះបីជា ការអនុវត្តធម្មតាតែងតែត្រិះរិះពី style ដ៏ល្អៗ ។ Methods ដែលត្រឡប់ return តម្លៃមួយដែលមានផ្តុក return statement ។ មិនមានទំហំ minimum size សម្រាប់ method ។ ប្រសិនបើជំនួយ method ដើម្បីជៀសវាងការច្រឡំផ្នែនៗ និងបង្កើតកម្មវិធីបាន ភ្លាមៗឱ្យល់ ពី method ដែលមានប្រយោជន៍ដែលយើងមិនបានគិតដល់ ។



sourer code:

```
1  using System;
2  using System.Collections;
3  using System.Collections.Generic;
```

```
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
  public partial class Form2 : Form
   {
     public Form2()
      {
        InitializeComponent();
      }
     public  float a, b, c, delta; // parameter

     private void Discriminantt()
      {
          //ការបង្កើត Method Discriminantt( ) សម្រាប់គណនា Dalta
             delta = (b * b) - (4 * a * c);
      }
    private void FindResult()
      {
         //ការបង្កើត Method FindResult( ) សម្រាប់គណនារកឫស Dalta
        if (delta == 0)
          {
            //សមីការមានឫសឌុប x1=x2=(-b/(2*a));

             txt_result_x2.Text = (-b / (2 * a)).ToString();
             txt_resut_x1.Text = (-b / (2 * a)).ToString();
          }
         else if (delta < 0)
         {
           //សមីការគ្មានឫស

           txt_result_x2.Text = "សមីការគ្មានឫស No result";

            txt_resut_x1.Text = "សមីការគ្មានឫស No result";
         }
         else
         {
             txt_resut_x1.Text = ((-b - Math.Sqrt(delta)) / (2 *
                                 a)).ToString();
             txt_result_x2.Text = ((-b + Math.Sqrt(delta)) / (2 *
                                    a)).ToString();
          }
        }

        private void btResult_Click(object sender, EventArgs e)
        {

```

```
59              //ចម្លើយ
60              a = float.Parse(txt_ax.Text);
61              b = float.Parse(txt_bx.Text);
62              c = float.Parse(txt_c.Text);
63              Discriminantt();
64              FindResult();
65          }
66      }
67  }
```

តាមរយៈកូដខាងលើយើងឃើញថា ក្រោយពេលដែល user បញ្ចូលតម្លៃនៃសមីការ និងចុចប៊ូតុងចម្លើយ តម្លៃក្នុង txt_ax, txt_bx, និង txt_c ដែលជា textbox ត្រូវបានផ្តល់តម្លៃទៅ a, b និង c ដែលជា float variable បន្ទាប់មកកូដបានហៅ Discriminant( ) method ដែលជា method បង្កើតឡើងសម្រាប់ធ្វើការរកតម្លៃរបស់ delta គឺ

delta = (b * b) - (4 * a * c);

បន្ទាប់ពីនេះក្រោយពេល Discriminant( ) method អនុវត្តន៍ចប់កូដនឹងធ្វើការហៅ FindResult( ) method ដែល ជា method ប្រើសម្រាប់គណនារកតម្លៃឫសនៃសមីការគឺ X1 និង X2 ទៅតាមលក្ខណ្ឌនៃតម្លៃ delta។

ឥឡូវនេះអ្នកទើបបានឃើញពីការអនុវត្តន៍នៃការប្រើប្រាស់ method ប៉ុន្តែអ្វីដែលជាចម្ងល់គឺ ការបើប្រាស់ method មានប្រយោជន៍អ្វី?

វាជាការត្រឹមត្រូវដែលយើងត្រូវដឹងពីហេតុផលនៃការធ្វើអ្វីមួយ បើមិនដូច្នេះទេវានឹងមិនខុសពីអ្វីដែលអ្នកបើកម៉ូតូដោយមិនដឹងពីគោលដៅថាតើខ្លួននឹងត្រូវទៅណានោះឡើយ។

Method នឹងជួយកាត់បន្ថយនូវភាពសាំញ៉ាំនៃការសរសេរកូដ ដោយសារវាតំណាងឲ្យនូវផ្នែកតូចៗនៃការដោះស្រាយបញ្ហា។ ជាពិសេស method នឹងនាំមកនូវភាពងាយស្រួលក្នុងការស្វែងរកកំហុសនៅពេលដែល បញ្ហាមិនត្រូវបានដោះស្រាយត្រឹមត្រូវនិងងាយស្រួលក្នុងការ updates កូដ។

៣. ការប្រើប្រាស់ Return Keyword

យើងបានឃើញកូដរបស់កម្មវិធីដោះស្រាយសមីការដឺក្រេទី២ខាងលើ Discriminant( ) និង FindResult( ) សុទ្ធតែជា method ដែលគ្មានការផ្តល់តម្លៃ ឬអាចនិយាយថា method ទាំងនោះពុំបានតំណាងឲ្យតម្លៃណាមួយ ឡើយក្រោយពីបញ្ចប់ការងាររបស់វា នេះក៏ដោយសារយើងបានប្រើប្រាស់ void ជា data type នៃ method ទាំងនោះ។ delta គឺជា global variable របស់ Form1 ដែលទទួលតម្លៃបានដោយសារការប្រើប្រាស់វា ក្នុង block របស់ Discriminant( ) method ។

```
private void Discriminant( )
    {
        delta = ( b * b ) - ( 4 * a * c );
    }
```

យើងនឹងបញ្ឈប់ការប្រើប្រាស់ delta variable ដោយការកែប្រែកូដរបស់ Discriminant( ) method ដូចខាងក្រោម៖

```
private float Discriminant( )
        }
                return (b * b) - (4 * a * c);
        }
```

កូដខាងលើនេះយើងបានប្រើប្រាស់ពាក្យ return ដែលបញ្ជាក់ថា Discriminant( ) method នឹងផ្តល់តម្លៃ ឬ នឹងតំណាងឲ្យតម្លៃដែលបានសរសេរពីក្រោយពាក្យ return នោះ ។

return (bb)-(4 ac);

លទ្ធផលនៃប្រមាណវិធីនេះនឹងត្រូវបានផ្តល់ឲ្យដោយ Discriminant}} method ក្រោយពីបញ្ចប់ការធ្វើការរបស់ method នេះ

- នៅពេលដែល method ណាមួយមានការប្រើប្រាស់ពាក្យ return នោះ data type នៃ method មិនអាចជា void បានឡើយ។ តម្លៃដែលសរសេរពីក្រោយពាក្យ return គឺជាចំខាតមាន data type ដូចគ្នានឹង data type របស់ method។

នៅពេលដែល return ត្រូវបានអនុវត្តន៍ ក៏មានន័យថា method ក៏នឹងបញ្ចប់សកម្មភាពរបស់វាទន្ទឹមគ្នា នោះដែរ យើងក៏នឹងទទួលបានតម្លៃដែលបានសរសេរពីក្រោយពាក្យ return នោះ។

ខាងក្រោមនេះគឺជាកូដដែលត្រូវកែប្រែសម្រាប់ click event របស់ ប៊ូតុង "ចម្លើយ" និង FindResult || method ក្រោយពី បានកែប្រែកូដរបស់ Discriminant( ) ដូចខាងលើ។

sourer code:

```
1  using System;
2  using System.Collections;
3  using System.Collections.Generic;
4  using System.ComponentModel;
5  using System.Data;
6  using System.Drawing;
7  using System.Linq;
8  using System.Text;
9  using System.Threading.Tasks;
10 using System.Windows.Forms;
11
12 namespace WindowsFormsApplication1
13 {
14     public partial class Form2 : Form
15     {
16      public Form2()
17      {
18        InitializeComponent();
19      }
20
```

```
21  public  float a, b, c, delta; // parameter
22
23  private float Discriminantt()
24  {
25      //ការបង្កើត Method Discriminantt( ) សម្រាប់គណនា Dalta
26
27      return ((b * b) - (4 * a * c));
28  }
29  private void FindResult()
30  {
31      //ការបង្កើត Method FindResult( ) សម្រាប់គណនារកឫស Dalta
32      if (Discriminantt() == 0)
33      {
34
35          //"គណនារកឫសសមីការ ";
36          txt_resut_x1.Text = (-b / (2 * a)).ToString();
37          txt_result_x2.Text = (-b / (2 * a)).ToString();
38      }
39      else if (Discriminantt() < 0)
40      {
41          txt_resut_x1.Text = "សមីការគ្មានឫស No result";
42
43          txt_result_x2.Text = "សមីការគ្មានឫស No result";
44      }
45      else
46      {
47          txt_resut_x1.Text = ((-b - Math.Sqrt(Discriminantt())) /
48
49                               (2 * a)).ToString();
50
51          txt_result_x2.Text = ((-b + Math.Sqrt(Discriminantt())) /
52
53                              (2 * a)).ToString();
54      }
55  }
56
57  private void btResult_Click(object sender, EventArgs e)
58  {
59      a = float.Parse(txt_ax.Text);
60      b = float.Parse(txt_bx.Text);
61      c = float.Parse(txt_c.Text);
62
63      FindResult();
64  }
65
```

៤. ការប្រើប្រាស់ Parameters

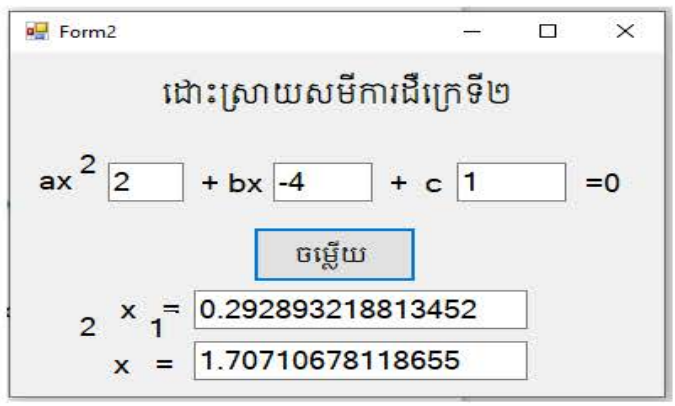
Value Types Parameters

Value type parameters សំដៅលើ Parameters ដែលគ្រាន់តែធ្វើការថតចម្លងតម្លៃពីក្រៅទៅអនុវត្តន៍ក្នុងMethod នោះប៉ុណ្ណោះ Value_parameters() ទៅកាន់ Method_name() ។

Value types parameters សំដៅលើ parameters ដែលគ្រាន់តែធ្វើការថតចម្លងតម្លៃពីក្រៅទៅអនុវត្តន៍ក្នុង method នោះតែប៉ុណ្ណោះ សូមមើលកូដនិង រូបភាពខាងក្រោមបង្ហាញពីការភ្ជាប់តម្លៃពី textboxes ទៅកាន់ FindResult( ) method FindResult( ) Discriminant( ) method"]

សញ្ញាព្រួញគឺនឹងបង្ហាញពីទិសដៅនៃតម្លៃត្រូវបានភ្ជាប់ពីមួយទៅមួយ។ ក្នុងកូដខាងក្រោមក៏បានបង្ហាញ ថាយើងពុំមានការប្រើប្រាស់ a,b និង c ជា gloabal variables ទៀតឡើយគឺយើងបានភ្ជាប់តម្លៃដោយផ្ទាល់ពី textbox ទៅកាន់ parameters នីមួយៗរបស់ FindResult( ) ដែលមាន a, b, c និង dalta ជា arguments ឬអាចនិយាយបានថា a, b និង c ជា local variables របស់ FindResult( ) method។

```
public float a, b, c, delta; // parameter
```




```
using System;
using System.Collections;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
    public partial class Form2 : Form
    {
   public Form2()
   {
       InitializeComponent();
   }

   private void btResult_Click(object sender, EventArgs e)
   {
    // "តម្លៃរបស់ Parameter ";
    FindResult(float.Parse(txt_ax.Text),float.Parse(txt_bx.Text)

                                    , float.Parse(txt_c.Text));
```

```
27
28      }
29
30      private float Discriminant(float a, float b, float c)
31      {
32          //"រកតម្លៃ Dalta ";
33          return((b*b)-(4*a*c));
34      }
35      private void FindResult(float a, float b, float c)
36      {
37
38           //"គណនារកឫសសមីការ ";
39          if (Discriminant(a, b, c) == 0)
40          {
41              txt_resut_x1.Text = (-b / (2 * a)).ToString();
42              txt_result_x2.Text = (-b / (2 * a)).ToString();
43
44          }
45          else if (Discriminant(a, b, c) < 0)
46          {
47             txt_resut_x1.Text = "សមីការគ្មានឫស No result";
48
49             txt_result_x2.Text = "សមីការគ្មានឫស No result";
50          }
51          else
52          {
53            txt_resut_x1.Text= ((-b - Math.Sqrt(Discriminant(a, b, c)))/
54                                (2 * a)).ToString();
55            txt_result_x2.Text= ((-b + Math.Sqrt(Discriminant(a, b, c)))/
56                                 (2 * a)).ToString();
57          }
58      }
59        }
60  }
61
```

៥. Reference parameter

ref ត្រូវបានប្រើសម្រាប់បង្កើត reference parameter ដែលអាចឱ្យប្រភពនៃតម្លៃមានការប្រែប្រួលតម្លៃផងដែរ ។

Reference parameter វាមិនមែនជា local variable របស់ method នោះទេប៉ុន្តែវា គឺជាឈ្មោះតំណាងសម្រាប់ប្រភពតម្លៃពីខាងក្រៅដែលអាចយកទៅប្រើប្រាស់ក្នុង method នោះតែប៉ុណ្ណោះ។

៦. **ស្វែងយល់** Scope

អ្នកបានស្គាល់គឺអាចបង្កើតអថេរដែលអ្នកអាចបង្កើតនៅក្នុង method ។ អថេរ គឺត្រូវបានបង្កើតក្នុង statement វាបានផ្តល់និយមន័យ។ បណ្តា statements ផ្សេងទៀតក្នុង method ដែលស្ថិតនៅអាចប្រើអថេរ។ អថេរអាចប្រើជាកន្លែងផ្ទុកបន្ទាប់ពីវាបានបង្កើត ។ ម្តងទៀត method បាន បញ្ចប់អថេរបាត់ភ្លាម ។ ប្រសិនបើ completely ហើយបើសិនប្រើលក្ខណៈពិសេសទីតាំងក្នុងកម្មវិធី។ Scope អនុវត្តទៅក្នុង

methods ដែលជាអថេរល្អ ប្រសើរ ។ អត្តសញ្ញាណនៃ scope ភ្ជាប់ជាមួយទីតាំងនៃការប្រកាស declaration ដែលណែនាំពីអត្តសញ្ញាណ ទៅក្នុង កម្មវិធីដូចជាអ្នក និងរៀនពេលឥឡូវនេះ។ បង្កើត Local Scope ជាមួយ Method នៅខាងឆ្វេង និងស្តាំដៃដែល form នៃ method បង្កើត scope មួយ ។ អថេរខ្លះអ្នក declare នៅផ្នែកខាងក្នុង body នៃ method មួយ។ អថេរទាំងនេះត្រូវបានហៅថាជា local variable ព្រោះវាតាំងនៅ method ដែលវាត្រូវបានប្រកាស វាមិនមាននៅក្នុង scope ក្នុង method ផ្សេងទៀត ។ ការរៀបចំ ជាមធ្យោបាយដែលអ្នកមិនអាចប្រើ local variable ដើម្បីចែកចាយព័ត៌មានរវាង methods ។

ឧទាហរណ៍:

```
class Example
{
     void firstMethod()
     {
          int myVar;
          …
     }
     void anotherMethod()
     {
          myVar = 42; // error - variable not in scope
          …
     }
}
```

៧. បង្កើត Class Scope ជាមួយ Class

នៅខាងឆ្វេងនិងស្តាំដៃដែលមាន form body នៃ class មួយដែល បង្កើត scope មួយ ។ អថេរ variables" ខ្លះអ្នកប្រកាស declare នៅខាង ក្រៅ body នៃ class (ប៉ុន្តែមិននៅខាងក្រៅ method) ត្រូវបាន scope ទៅក្នុង class ។ ដូចជាយើងបានមើលឃើញក្នុងជំពូកទីពីឈ្មោះ C# name សម្រាប់អថេរទាំងនេះគឺជាបណ្តា fields ។

```
 1 class Example
 2 {
 3      void firstMethod()
 4      {
 5           myField = 42; // ok
 6           …
 7      }
 8      void anotherMethod()
 9      {
10           myField = 42; // ok
11           …
12      }
13      int myField = 0;
14 }
```

៨. Overloading method

Overloading method ជា method ទាំងឡាយណាដែលមានឈ្មោះដូចៗគ្នា ប៉ុន្តែមានលក្ខណៈខុសគ្នាត្រង់

- data type របស់ parameter ឬ method
- ចំនួន ឬ លក្ខណៈរបស់ parameter។

ឧទាហរណ៍ខ្ញុំមាន method ពីរដែលមានឈ្មោះដូចគ្នា calculate( ) តែការប្រើប្រាស់មិនដូចគ្នាទេ ។ ដែល method ទី១ ប្រើសម្រាប់គណនារកផលបូកពីរតម្លៃជាចំនួនគត់ឯ method ទី២ មានតួនាទីសម្រាប់គណនារកផលបូកពីរតម្លៃជាចំនួនទសភាគ។ សូមរកមើលចំណុចខុសគ្នារវាង method ទាំងពីរ

ឧទាហរណ៍ទី១៖

```
int calculate( int  val1, int val2, Char op)
{
     switch (op)
     {
          case '+':
               return val1 + val2;
          case '-':
               return val1 - val2;
          case ***:
          return val1 * val2;
          case '/':
               return val1 / val2;
          case '%':
               return val1 % val2;
          default:
          MessageBox.Show("Invalid operator!");
          return 0;
     }
}
```

ឧទាហរណ៍ទី២៖

```
int calculate( float val1, float val2, char op)
{
     switch (op)
     {
          case '+':
               return val1 + val2;
          case '-':
               return val1 - val2;
          case ***:
          return val1 * val2;
          case '/':
               return val1 / val2;
          case '%':
```

```
14                  return val1 % val2;
15              default:
16              MessageBox.Show("Invalid operator!");
17              return 0;
18          }
19  }
```

ជំពូកទី ៩

សិក្សា Object Oriented Programming

១. សេចក្តីណែនាំ C# OOP

១.១ What is OOP ?

OOP មកពីពាក្យ Object-Oriented Programming.

OOP គឺជាវិធីសាស្ត្រដ៏មានសារសំខាន់ក្នុងការសរសេរកម្មវិធីកុំព្យូទ័រ ដោយការប្រើប្រាស់ Classes or Object ដែលចងភ្ជាប់កូដជាមួយគ្នា ដោយមិនចាំបាច់ក្នុងការសរសេរកូដដដែលៗ និងដំណើរការប្រតិបត្តិការយ៉ាងសាមញ្ញ ។ Objects គឺជា Classes ដ៏មានសរៈសំខាន់ដែលប្រមូលផ្តុំ (Functions) បញ្ចូលគ្នា និងចងពួកវាទុកក្នុងការប្រើប្រាស់លើកក្រោយ ដោយមិនចាំបាច់សរសេរកូដ ច្រើនសារជាដដែលៗនៅគ្រប់ពេលដែលអ្នកត្រូវការធ្វើអ្វីមួយនោះទេ។ Procedural Programming ធ្វើការតាមជំហានពីលើចុះក្រោមនៃទំព័រនីមួយៗ ដូចដែល Server ធ្វើការអាន គ្រប់ទិន្នន័យនៅលើ Server របស់អ្នកដែរ ។ ជាមួយ OOP វាមានមួយឬពីរ Objects ដែលកំពុងត្រូវបានដំណើរការជាថ្មី វាអាចដំណើរការពីរ បី មួយរយ ឬ មួយពាន់ Objects ផ្សេងៗទៀតដែលមាន ដំណើរការជាក់លាក់អាស្រ័យលើ variables ដែលត្រូវបានជាក់បញ្ចូលចូលទៅក្នុង Object នោះ។ OOP មានដំណើរការលឿន សាមញ្ញ និងមានភាពងាយស្រួលក្នុងការដោះស្រាយបញ្ហាប្រើប្រាស់ធនធាន Server តិច និងការសរសេរកូដតិច ដំណើរការលឿន និងត្រឹមត្រូវជាងមុនក្នុងការធ្វើការនៅពេលដែលអ្នកដឹងពីមូលដ្ឋានគ្រឹះរបស់វា។ OOP មានការផ្លាស់ប្តូរនូវស្មាយក្នុងការអភិវឌ្ឍន៍កម្មវិធីរបស់លោកអ្នកជារៀងរហូត។ Object-oriented programming មានសារសំខាន់សម្រាប់ក្នុងការសរសេរ programming ដូចជា៖

- OOP គឺលឿន និងងាយស្រួលក្នុងការប្រតិបត្តិ
- OOP ផ្តល់នូវរចនាសម្ព័ន្ធច្បាស់លាស់សម្រាប់សរសេរកម្មវិធីត្រឹមត្រូវតាមលំដាប់លំដោយ
- OOP ជួយក្នុងការសរសេរកូដដដែលៗ ច្រើនដង និងងាយស្រួលក្នុង កែប្រែ ការរកកំហុស
- OOP ធ្វើឱ្យវាអាចបង្កើតបានដើម្បីបង្កើតកម្មវិធីដែលអាចប្រើឡើងវិញបានពេញលេញជាមួយនឹងលេខកូដតិច និងពេលវេលាអភិវឌ្ឍន៍ខ្លី

លក្ខណៈរបស់ OOP

Encapulation ៖ ការលាក់ភាពសម្ងិតនៃទិន្នន័យនឹងប្រតិបត្តិការខាងក្រោយកម្មវិធី។

Polymophism ៖ ការប្រើឈ្មោះឬ សញ្ញាដូចគ្នាបាន (ក្នុង Class មានឈ្មោះ Variable នោះហើយ ប៉ុន្តែនៅក្នុង Class ផ្សេងក៏អាចប្រើឈ្មោះនោះបានដែរ)។

Inheritance ៖ ការផ្ទេរតម្លៃ (Object អាចទទួលបានតាមការកំណត់របស់ Class) ។

២. ការបង្កើត Class និង Object s

Class គឺជាប្រភេទគម្រូមុខងារមួយដែលកើតឡើងពីបណ្តុំ Variable និង Function មួយឬច្រើន ហើយវាក៏ជាប្រភេទ Data Type ពិសេសមួយផងដែរសម្រាប់បង្កើតជា Object ។ ហើយ វាក៏ជាបន្សំឡើងដោយ Data Member , Property និង Methode ផងដែរ។

Object គឺជាអញ្ញាតជំពិសេសមួយ ដែលមានប្រភេទជា Class ហើយវាមានតួនាទីដូច Class អញ្ចឹង។

រូបមន្តបង្កើត class

```
class className [: BaseClass, Interfaces]
    {
        //attributes;
        //method;
    }
```

៣. Classes & Objects

C# គឺជាភាសារ Object Oriented Programming (OOP) Language ដែលរាល់គ្រប់យ៉ាងមានទៅជោយ classes នឹង objects ជាមួយនឹង attributes និង methods របស់វា។ ជាឧទាហរណ៍នៅក្នុង ជីវិតរស់នៅជាក់ស្តែង ដូចជា ឡាន (car) គឺត្រូវបានចាត់ទុកថាជា Object។ ឡាន (Car) គឺមាន attributes ដូចជា ទម្ងន់ (weight) និង ពណ៌ (color) រួមទាំង methods ដូចជា បើកបរ (drive) និង ចាប់ហ្វ្រាង់ (brake) ជាដើម។ តាមនិយមន័យ Class គឺជា object constructor ឬក៏ ពុម្ពគម្រូ blueprint" សម្រាប់បង្កើត objects។

៣.១. របៀបបង្កើត Class

Class គឺជាប្រភេទគម្រូមុខងារមួយដែលកើតឡើងពីបណ្តុំ Variable និង Function មួយឬច្រើន ហើយវាក៏ជាប្រភេទ Data Type ពិសេសមួយផងដែរសម្រាប់បង្កើតជា Object ។ ហើយ វាក៏ជាបន្សំឡើង ដោយ Data Member , Property និង Methode ផងដែរ។

ដើម្បីបង្កើត class បានយើងត្រូវប្រើប្រាស់នូវពាក្យគន្លឹះមួយដែលមានឈ្មោះថា "class"

បង្កើត class មួយដែលមានឈ្មោះថា "Car" ជាមួយនឹងអថេរ (variable) មួយ ឈ្មោះ ថា "Car"

```
class Car
    {
        string color = "red";
    }
```

នៅពេលដែលអថេរ (variable) ត្រូវបានប្រកាសដោយផ្ទាល់នៅក្នុង class វាជារឿយៗត្រូវបានគេចាត់ទុកថាជា field (ឬ attribute)

៣.២. របៀបបង្កើត Object

Object ត្រូវបានគេបង្កើតចេញពី class។ យើងបានបង្កើតនូវ class មួយដែលមានឈ្មោះថា "Car" រួចរាល់ហើយ ដូច្នេះយើងអាចប្រើប្រាស់ class នេះដើម្បីធ្វើការបង្កើតនូវ objects បាន។ ដើម្បីបង្កើត object របស់ class "Car" ត្រូវបញ្ជាក់ពី class name រួចអនុលោមតាម ឈ្មោះរបស់ object ដោយប្រើប្រាស់ keyword "new" ដើម្បីធ្វើការបង្កើត object នោះ

ឧទាហរណ៍: បង្កើត object មួយឈ្មោះ ថា "myObj" ហើយធ្វើការប្រើប្រាស់ object នោះ ដើម្បីធ្វើការ print តម្លៃរបស់ color

```
class Car
{
        string color = "red";
        static void Main( string[] args )
        {
                Car myObj = new Car( );
                Console.WriteLine( myObj.color );
        }
}
```

Object គឺជាអញ្ញាតដ៏ពិសេសមួយ ដែលមានប្រភេទជា Class ហើយវាមានតួនាទីដូច Class អញ្ចឹង។

ចំណាំ ថាយើងប្រើប្រាស់សញ្ញាចុច ( ) ដើម្បីធ្វើការ access នូវ variables/fields នៅ class (myObj.color)

៣.៣. របៀបប្រើប្រាស់ Multiple Objects

យើងអាចធ្វើការបង្កើត multiple objects របស់ class មួយបាន ដូចឧទាហរណ៍ខាងក្រោម បង្កើត Object ចំនួន ២ ចេញពី class "Car"

```
class Car
{
```

```
        string color="red";
        static void Main( string[] args )
        {
            Car myObj1= new Car( );
            Car myObj2 = new Car( );
            Console.WriteLine( myObj1.color );
            Console.WriteLine( myObj2.color );
        }
    }
```

៣.៤. របៀបប្រើប្រាស់ Multiple Classes

យើងក៏អាចបង្កើត object មួយនៃ class មួយ ហើយក៏អាចដំណើរការវានៅក្នុង class មួយផ្សេងទៀតបានដែរ។ ជារឿយៗវិធីនេះត្រូវបានប្រើប្រាស់ដើម្បីរៀបចំ class ឱ្យកាពប្រសើរ (class មួយមាននូវ fields និង methods ទាំងអស់ ខណៈពេលដែល class ដទៃទៀត មានតែ Main( ) method តែប៉ុណ្ណោះ(ជាកន្លែងដែលត្រូវបាន execute code)។

ឧបមាថា យើងមាន class ចំនួន២

```
class Car
    {
    public string color = "red";
    }
```

```
class Program
    {
        static void Main( string[] args )
        {
            Car myObj = new Car( );
```

```
            Console.WriteLine( myObj.color );
        }
    }
```

សូមកត់សម្គាល់ផងដែរថា Keyword "public" គឺជា access modifier ដែលបញ្ជាក់អំពី variable/field "color" iufu class "car" fun access 19ins classes

ផ្សេងទៀតផងដែរ ជូជា class "Program" ។

៣.៥ Member Class

នៅក្នុងអត្ថបទនេះនឹង និយាយអំពីកាការប្រើប្រាស់ Member of Class ហើយ Member របស់ Class នោះមានដូចជា Constructor, Inheritance, property , method , Field ...ល។ ហើយក្នុងអត្ថបទនោះខ្ញុំនឹងធ្វើការលើកយក Member មួយចំនួនបង្ហាញជូនលោកអ្នកដូចខាងក្រោម។

Fields និង methods ស្ថិតនៅក្នុងថ្នាក់ត្រូវបានសំដៅជាញឹកញាប់ថាជា "Class Members"។

Create a Car class with three class members: **two fields** and **one method**.

```
// The class
class MyClass
{
  // Class members
  string color = "red";         // field
  int maxSpeed = 200;           // field
  public void fullThrottle()    // method
  {
    Console.WriteLine("The car is going as fast as it can!");
  }
}
```

៣.៥.១ Fields

យើងសិក្សាពីរបៀបហៅ Fields Variable ពីក្នុង Class ហើយអ្នកអាចដំណើរការបង្កើត Object នៃ Class ដោយប្រើប្រាស់ នូវសញ្ញា (.) ។

ឧទាហរណ៍ខាងក្រោមនឹងបង្កើត Object នៃ Car class ដោយមានឈ្មោះ myObj។ បន្ទាប់មកយើងបង្ហាញតម្លៃនៃ fields color និង maxSpeed ៖

```
1  class Car
2  {
3    string color = "red";
4    int maxSpeed = 200;
5
6    static void Main(string[] args)
7    {
8      Car myObj = new Car();
9      Console.WriteLine(myObj.color);
10     Console.WriteLine(myObj.maxSpeed);
11   }
12 }
```

៣.៥ .២ Object Methods

Methods ជាធម្មតាជាកម្មសិទ្ធិរបស់ class ហើយពួកវាកំណត់ពីរបៀបដែល object នៃ class behaves ។ ដូចគ្នានឹង fields ដែរ អ្នកអាចចូលប្រើ methods ដោយប្រើសញ្ញា (.) ។ ទោះជាយ៉ាងណាក៏ដោយ ចំណាំថា methods ត្រូវតែប្រកាសជាសាធារណៈ។ ហើយត្រូវចាំថា យើងប្រើឈ្មោះនៃ methods ដែលតាមពីក្រោយដោយ parantheses( ) ពីរ និង semicolon (;) មួយ ដើម្បីហៅ (execute) methods ។

```
1  class Car
2  {
3    string color;                 // field
4    int maxSpeed;                 // field
5    public void fullThrottle()    // method
6    {
7      Console.WriteLine("The car is going as fast as it can!");
8    }
9
10   static void Main(string[] args)
11   {
12     Car myObj = new Car();
13     myObj.fullThrottle();  // Call the method
14   }
15 }
```

៣.៥.៣ ការប្រើប្រាស់ Class ច្រើន Multiple Classes

យើងអាចប្រើ multiple classes ថ្នាក់ច្រើនសម្រាប់ស្តាប័នល្អជាងមុន (មួយសម្រាប់ fields , methods ហើយនឹង class មួយទៀតសម្រាប់ execution) ។

Programming_1.cs

```
1  class Car
2  {
3    public string model;
```

```
  public string color;
  public int year;
  public void fullThrottle()
  {
    Console.WriteLine("The car is going as fast as it can!");
  }
}
```

Programming_2.cs

```
class Program
{
  static void Main(string[] args)
  {
    Car Ford = new Car();
    Ford.model = "Mustang";
    Ford.color = "red";
    Ford.year = 1969;

    Car Opel = new Car();
    Opel.model = "Astra";
    Opel.color = "white";
    Opel.year = 2005;

    Console.WriteLine(Ford.model);
    Console.WriteLine(Opel.model);
  }
}
```

៣.៦. ការប្រើ Member Constructor

Constructor ជា Method មួយដែលមានឈ្មោះដូចទៅនឹង Class ហើយ Method នោះត្រូវបាន execute នៅពេលដែល Object កើតចញ្ចពី Class ។ ។ ដើម្បីបង្កើត Constructor គឺគ្រាន់តែបង្កើត Sub procedure មួយឈ្មោះ New នៅក្នុង Class ជាការស្រេច។

៣.៧. Overloading Method

អនុញ្ញាតឱ្យ Class មួយមាន methods ច្រើនមានឈ្មោះដូចគ្នា ប៉ុន្តែ methods ទាំងនោះខុសគ្នាត្រង់ចំនួន Arguments or Datatype របស់ Argument នោះ។

៤. សិក្សាអំពី Constructors

Constructors គឺជាប្រភេទ method ពិសេសនៅក្នុង class ដែលត្រូវបានហៅដោយស្វ័យប្រវត្តិ ដោយប្រព័ន្ធដើម្បីកំណត់ស្ថានភាពដំបូងនៃ object ឬ class ។ ពួកគេមិនអាចត្រូវបានគេហៅថាដោយប្រើ សេចក្តីថ្លែងការណ៍ CALL METHOD ទេ។ constructors តែងតែមានវត្តមាននៅក្នុង class ប៉ុន្តែដើម្បី អនុវត្ត constructor វាត្រូវតែត្រូវបានប្រកាសយ៉ាងច្បាស់ជាមួយនឹង METHODS ឬ CLASS-METHODS statements ។

Constructor មានឈ្មោះដូច Class ហើយពុំមាន return type នោះទេ។ ផល ប្រយោជន៍នៃការប្រើ Constructor គឺកំណត់តម្លៃចាប់ផ្តើមនៃ Fields និងសម្រាប់បង្កើតតម្លៃទៅឲ្យ Object ។ នៅក្នុង C# constructors អាចត្រូវបានបែងចែកជា 5 ប្រភេទ៖

1.Default Constructor
2.Parameterized Constructor
3.Copy Constructor
4.Static Constructor
5.Private Constructor

៤.១. សិក្សាអំពី Default Constructor

Default Constructor : សំដៅទៅលើ Constructor ដែលមិនមាន return type និងមិនមានប៉ារ៉ាម៉ែត្រ។ constructor ដោយគ្មានប៉ារ៉ាម៉ែត្រណាមួយត្រូវបានគេហៅថា default constructor; នៅក្នុងពាក្យផ្សេងទៀត constructor ប្រភេទនេះមិនយកប៉ារ៉ាម៉ែត្រទេ។ គុណវិបត្តិនៃ default constructor គឺថា រាល់ instance នៃ class នឹងត្រូវបានផ្តួចផ្តើមឱ្យមានតម្លៃដូចគ្នា ហើយវាមិនអាចចាប់ផ្តើម instance នីមួយៗនៃ class ជាមួយនឹងតម្លៃផ្សេងគ្នាបានទេ។

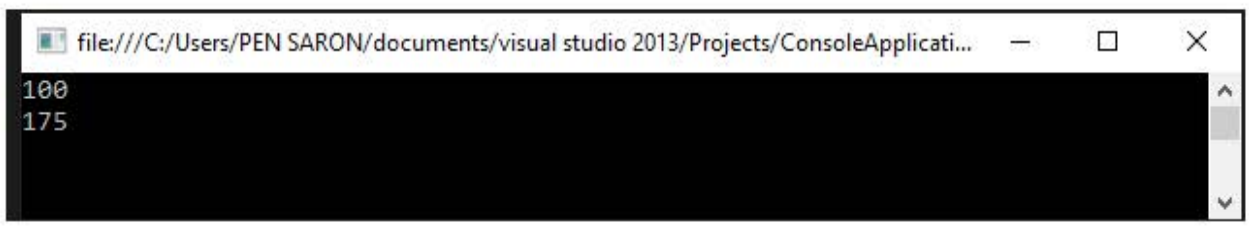
default constructor ចាប់ផ្តើមធ្វើការ:

numeric fields ទាំងអស់ នៅក្នុង class ចាប់ពី zero

string និង object fields ទាំងអស់អាចជា null

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

namespace ConsoleApplication6
{
    class Program
    {
        class addition
        {
            int a, b;
            public addition()    //default contructor
            {
                a = 100;
                b = 175;
            }

            public static void Main()
            {
                addition obj = new addition();
                 //an object is created , constructor is called
                Console.WriteLine(obj.a);
```

```
                Console.WriteLine(obj.b);
                Console.Read();
            }
        }
    }
}
```



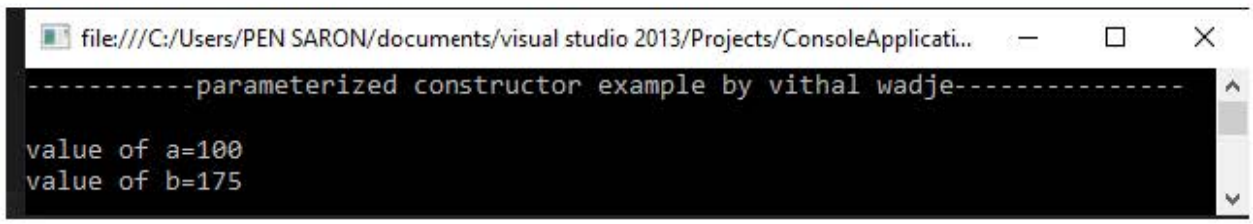
៤.២.សិក្សាអំពី Parameterized Constructor

Parameterized Constructor : សំដៅទៅលើ Constructor ដែលមានប៉ារ៉ាម៉ែត្រ ក្នុងគោលបំណង set(កំណត់) តម្លៃចាប់ផ្តើមឲ្យ instance variables នៅពេលបង្កើត Object ។

Constructor ដែលមានប៉ារ៉ាម៉ែត្រយ៉ាងហោចណាស់មួយត្រូវបានគេហៅថា constructor ប៉ារ៉ាម៉ែត្រ។ អត្ថប្រយោជន៍នៃ parameterized constructor គឺថាអ្នកអាចចាប់ផ្តើម instance នីមួយៗនៃ class ជាមួយនឹងតម្លៃផ្សេងគ្នា។

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

namespace ConsoleApplication6
{
    class paraconstrctor
    {
        public int a, b;
        public paraconstrctor(int x, int y)
        // decalaring Paremetrized Constructor with ing x,y parameter
        {
            a = x;
            b = y;
        }
    }
    class MainClass
    {
        static void Main()
        {
            paraconstrctor v = new paraconstrctor(100, 175);
       // Creating object of Parameterized Constructor and ing values
            Console.WriteLine("---parameterized constructor---");
            Console.WriteLine("\t");
            Console.WriteLine("value of a=" + v.a);
```

```
            Console.WriteLine("value of b=" + v.b);
28          Console.Read();
29      }
30  }
}
```



៤.៣. សិក្ខាអំពី Copy Constructor

Constructor ដែលបង្កើត object ដោយចម្លង variable ពី object ផ្សេងទៀតត្រូវបានគេហៅថា copy constructor ។ គោលបំណងនៃអ្នកបង្កើតច្បាប់ចម្លងគឺដើម្បីចាប់ផ្តើម instance ថ្មីទៅនឹងតម្លៃនៃ instance ដែលមានស្រាប់។ Syntax:

```
1 public employee(employee emp)
2 {
3      name=emp.name;
4      age=emp.age;
5 }
```

Copy Constructor ត្រូវបានហៅដោយ instantiating Object នៃប្រភេទ employee ហើយដោយឆ្លងកាត់វា object ដែលត្រូវចម្លង។ The copy constructor is invoked by instantiating an object of type employee and by passing it the object to be copied

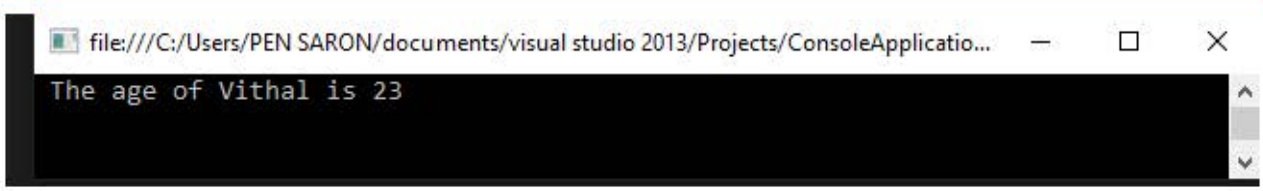
ឧទាហរណ៍៖

```
1  Employee emp1 = new employee (emp2);
```

បន្ទាប់មកយើងធ្វើការ ចម្លង emp1 ទៅ emp2.

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace ConsoleApplication6
8  {
9      class employee
10     {
11         private string name;
12         private int age;
13         public employee(employee emp)
14
15
```

```
16              // declaring Copy constructor.
17          {
18              name = emp.name;
19              age = emp.age;
20          }
21          public employee(string name, int age)
22              // Instance constructor.
23          {
24              this.name = name;
25              this.age = age;
26          }
27          public string Details      // Get deatils of employee
28          {
29              get
30              {
31                  return " The age of " + name + " is " + age.
32                                                      ToString();
33              }
34          }
35      }
36      class empdetail
37      {
38          static void Main()
39          {
40              employee emp1 = new employee("Vithal", 23);
41                // Create a new employee object.
42              employee emp2 = new employee(emp1);
43                // here is emp1 details is copied to emp2.
44              Console.WriteLine(emp2.Details);
45              Console.ReadLine();
46          }
47      }
48  }
```



៤.៤ សិក្ខាអំពី Static Constructor

នៅពេលដែល constructor ត្រូវបានបង្កើតដោយប្រើពាក្យគន្លឹះបិតរួច វានឹងត្រូវបានគេហៅតែម្តងគត់សម្រាប់ instances ទាំងអស់នៃ class ហើយវាត្រូវបានហៅក្នុងអំឡុងពេលបង្កើត instance ដំបូងនៃ class ឬការយោងដំបូងទៅកាន់ static member នៅក្នុង class ។ static constructor ត្រូវបានប្រើដើម្បីចាប់ផ្តើមវាលបិតរួចនៃថ្នាក់ និងដើម្បីសរសេរកូដដែលចាំបាច់ត្រូវប្រតិបត្តិតែម្តងប៉ុណ្ណោះ។

ចំនុចសំខាន់ៗមួយចំនួនរបស់ static constructor គឺ៖

១. static constructor គឺមិនទទួលយកការកែប្រែ ឬ មាន ប៉ារ៉ាម៉ែត្រ

២. static constructor គឺហៅដោយស្វ័យប្រវត្តិ ដើម្បីចាប់ Class ជា មុនពេលដែល instance ទីមួយត្រូវបានបង្កើត ឬសមាជិក static ណាមួយត្រូវ គឺជា referenced។

៣. static constructor មិនអាច Call ដោយផ្ទាល់បានទេ

៤. អ្នកប្រើមិនមានការគ្រប់គ្រងលើពេលដែល static constructor ត្រូវបានប្រតិបត្តិក្នុងកម្មវិធីនោះទេ។

៥. ការប្រើប្រាស់ធម្មតានៃ static constructors គឺនៅពេលដែល class ប្រើប្រាស់ log file ហើយ constructor ត្រូវបានប្រើដើម្បីសរសេរជាតុទៅក្នុង file នេះ។

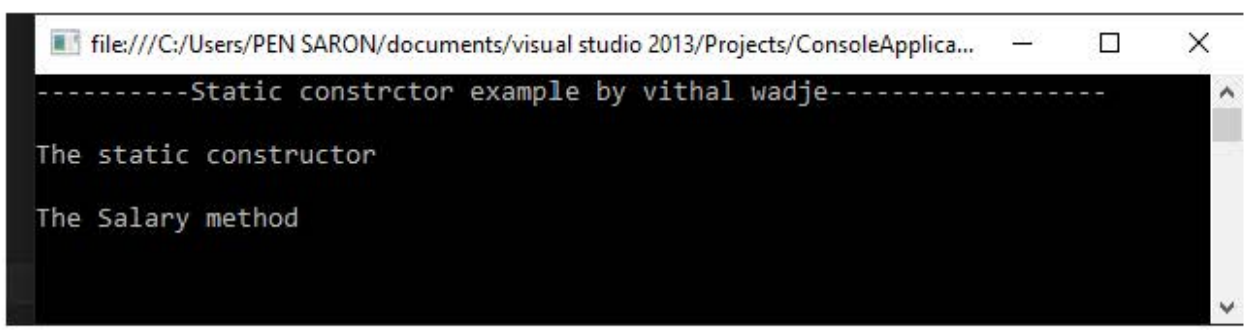
Syntax:

```
Employee employee
{
  //static contructor
  static employee()
  {
   //statement;
  }
}
```

ឧទាហរណ៍ទី១៖

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

namespace ConsoleApplication6
{
    class employee
    {
        private string name;
        private int age;
        public employee(employee emp) // declaring Copy constructor.
        {
            name = emp.name;
            age = emp.age;
        }
        public employee(string name,int age)//Instance constructor.
        {
            this.name = name;
            this.age = age;
        }
        public string Details     // Get deatils of employee
        {
```

```
25              get
26              {
27                return " The age of " + name + " is " + age.ToString();
28              }
29          }
30      }
31      class empdetail
32      {
33          static void Main()
34          {
35              employee emp1 = new employee("Vithal", 23);
36                      // Create a new employee object.
37              employee emp2 = new employee(emp1);
38                      // here is emp1 details is copied to emp2.
39              Console.WriteLine(emp2.Details);
40              Console.ReadLine();
41          }
42      }
43  }
```



៤.៥ សិក្ខាអំពី Private Constructor

នៅពេលដែល constructor ត្រូវបានបង្កើតជាមួយ private specifier វាមិនអាចទៅរួចទេសម្រាប់ class ផ្សេងទៀតដើម្បីទាញយកពី class នេះ ហើយក៏មិនអាចបង្កើត instance នៃ class នេះដែរ។ ជាធម្មតាពួកវាត្រូវបានប្រើនៅក្នុងថ្នាក់ដែលមានតែសមាជិកបិតិរន្តប៉ុណ្ណោះ។
ចំណុចសំខាន់ៗមួយចំនួនរបស់ Private constructor គឺ៖

១. ការប្រើប្រាស់មួយរបស់ constructor ឯកជនគឺនៅពេលដែលយើងមានសមាជិកតែមួយ static members

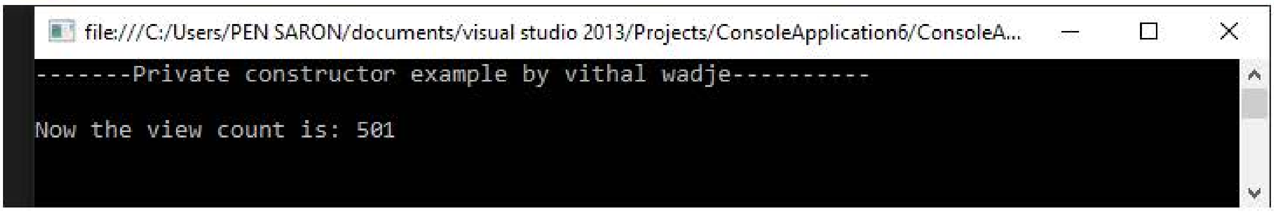
២. វាផ្តល់នូវការអនុវត្តនៃគំរូថ្នាក់ singleton

៣. នៅពេលដែលយើងផ្តល់ constructor ដែលមានលក្ខណៈឯកជន ឬសាធារណៈ ឬណាមួយនោះ compiler នឹងមិនបន្ថែម parameter-less public constructor ទៅក្នុង class ទេ។

ឧទាហរណ៍ទី១៖

```
1 using System;
2 using System.Collections.Generic;
```

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

namespace ConsoleApplication6
{
    public class Counter
    {
        private Counter()    //private constrctor declaration
        {

        }
        public static int currentview;
        public static int visitedCount()
        {
            return ++currentview;
        }
    }
    class viewCountedetails
    {
        static void Main()
        {
       // Counter aCounter = new Counter();    // Error
       Console.WriteLine("--Private constructor example ----");
       Console.WriteLine();
       Counter.currentview = 500;
       Counter.visitedCount();
       Console.WriteLine("Now the view count is: {0}",
                         Counter.currentview);
       Console.ReadLine();
        }
    }
}
```



៥. សិក្សាអំពី Access Modifiers

នៅក្នុង C# អ្នកកែប្រែការចូលប្រើបញ្ជាក់ភាពងាយស្រួលនៃប្រភេទ (ថ្នាក់ ចំណុចប្រទាក់ ។ល។) និងប្រភេទសមាជិក (វាល វិធីសាស្ត្រ ។ល។)។

Access Modifiers នៅក្នុងភាសា C# គឺមានចំនួនបួន៖

- public: អនុញ្ញាតឲ្យ class members អាច access បានគ្រប់ class

- private: អនុញ្ញាតឲ្យ class members អាច access បានតែក្នុង class របស់ខ្លួនឯង ប៉ុណ្ណោះ និងមិនអាច access ទៅ class ផ្សេងៗបានឡើយ
- protected: អនុញ្ញាតឲ្យ class members អាច access ទៅ class ផ្សេងបានលុះត្រាតែប្រើប្រាស់នូវ inheritance
- internal: អនុញ្ញាតឲ្យ class members អាច access បានតែក្នុង assembly របស់ខ្លួនឯងតែប៉ុណ្ណោះ។

យើងសិក្សាអំពី public, private, protected និង internal access modifiers ក្នុង C#

access modifiers specify ភាពងាយស្រួល នៃប្រភេទ Class (classes, interfaces, etc) និង type members (fields, methods, etc) ។

```
class Student {

  public string name;

  private int num;

  }
```

name public field ដែលអាច accessed ពីកន្លែង

num private field ដែលអាចចូលប្រើបានតែនៅក្នុង Class ប៉ុណ្ណោះ។

៥.១. public access modifier

នៅពេលដែលយើងប្រកាសប្រភេទអញ្ញាត ឬ ប្រភេទសមាជិក type member public វាអាច accessed ពីគ្រប់កន្លែង។

ឧទាហរណ៍៖

```
using System;

namespace MyApplication {

  class Student {
    public string name = "Sheeran";

    public void print() {
      Console.WriteLine("Hello from Student class");
     }
```

```
  }

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

      // creating object of Student class
      Student student1 = new Student();

      // accessing name field and printing it
      Console.WriteLine("Name: " + student1.name);

      // accessing print method from Student
      student1.print();
      Console.ReadLine();
    }
  }
}
```

Output

```
Name: Sheeran
Hello from Student class
```

នៅក្នុងឧទាហរណ៍ខាងលើ យើងអាចបង្កើត Class មួយឈ្មោះ Student ជាមួយ field name និង method print( ) ។

```
// accessing name field and printing it
Console.WriteLine("Name: " + student1.name);

// accessing print method from Student
student1.print();
```

ចាប់ផ្តើម field និង method គឺ public យើងអាច accessed ពួកគេមកពី Program class ។

៥.២ private access modifier

យើងប្រកាស type member ជា private access modifier វាអាច accessed បានតែនៅក្នុង class ឬ struct ដូចគ្នា ។

ឧទាហរណ៍ទី ៖

```
using System;

namespace MyApplication {

  class Student {
    private string name = "Sheeran";

    private void print() {
```

```
      Console.WriteLine("Hello from Student class");
    }
  }

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

      // creating object of Student class
      Student student1 = new Student();

      // accessing name field and printing it
      Console.WriteLine("Name: " + student1.name);

      // accessing print method from Student
      student1.print();

      Console.ReadLine();
    }
  }
}
```

ក្នុងឧទាហរណ៍ខាងលើ យើងបានបង្កើត Class ឈ្មោះ Student ជាមួយ field name និង method print( ) ។

```
// accessing name field and printing it
Console.WriteLine("Name: " + student1.name);

// accessing print method from Student
student1.print();
```

field និង method គឺ private យើងមិនអាចចូលប្រើពួកវាពី program class បានទេ ខាងក្រោមនេះជា code generate error ។

```
Error    CS0122    'Student.name' is inaccessible due to its protection level
Error    CS0122    'Student.print()' is inaccessible due to its protection level
```

៥.៣. protected access modifier

នៅពេលយើងប្រកាសសមាជិក member ប្រភេទថាត្រូវបានការពារ type member protected វាអាចចូលប្រើបានតែពីថ្នាក់ដូចគ្នា និងថ្នាក់ដែលបានមកពីរបស់វា type member as protected ។ ឧទាហរណ៍ ៖

```
using System;

namespace MyApplication {

  class Student {
    protected string name = "Sheeran";
```

```
  }

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

      // creating object of student class
      Student student1 = new Student();

      // accessing name field and printing it
      Console.WriteLine("Name: " + student1.name);
      Console.ReadLine();
    }
  }
}
```

ក្នុងឧទាហរណ៍ខាងលើ យើងបានបង្កើត class ឈ្មោះ student មាន field name ។ field គឺ protected យើងមិនអាច access វាពី class program បានទេ។

code នឹង generate បង្ហាញភាព error ដូចខាងក្រោម៖

```
Error    CS0122    'Student.name' is inaccessible due to its protection level
```

Now, let's try to access the `protected` member from a derived class.

```
using System;

namespace MyApplication {

  class Student {
    protected string name = "Sheeran";
  }

  // derived class
  class Program : Student {
    static void Main(string[] args) {

      // creating object of derived class
      Program program = new Program();

      // accessing name field and printing it
      Console.WriteLine("Name: " + program.name);
      Console.ReadLine();
    }
  }
}
```

Output

```
Name: Sheeran
```

នៅក្នុងឧទាហរណ៍ខាងលើយើងបានបង្កើត class student ជាមួយ protected field name ។ យើងមាន inherited program class ពី student class ។

```
// accessing name field and printing it
Console.WriteLine("Name: " + program.name);
```

ចាប់ពី protected member អាច accessed មកពី derived classes យើងអាចចូលប្រើ access name បានពី Program class ។

៥.៤ internal access modifier

នៅពេលដែលប្រកាសប្រភេទ ឬ type member internal វាអាច accessed បានតែមួយជាមួយ assembly ។

assemly គឺជា Collection នៃ type (Class, interfaces,.....) និង resources (Data) ។ ពួកវាត្រូវបានបង្កើតឡើងដើម្បីធ្វើការរួមគ្នា និងបង្កើតជាឯកតាតក្កវិជ្ជានៃមុខងារ។

នៅពេលដែលយើងដំណើរការ assembly class ទាំងអស់ និង interfaces inside ខាងក្រៅ assembly ធ្វើការជាមួយគ្នា ។

Example: internal within the same Assembly

```
using System;

namespace Assembly {

  class Student {
   internal string name = "Sheeran";
  }

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

      // creating object of Student class
      Student theStudent = new Student();

      // accessing name field and printing it
      Console.WriteLine("Name: " + theStudent.name);
      Console.ReadLine();
    }
  }
}
```

Output

```
Name: Sheeran
```

នៅក្នុងឧទាហរណ៍ខាងលើ យើងបានបង្កើត class មានឈ្មោះ Student ជាមួយ Field name ។ Field គឺជា internal យើងអាចចូលដំណើរការប្រើវាពី Program class ដូចដែលពួកវាស្ថិតនៅក្នុង

assembly ដូចគ្នា ។ ប្រសិនបើយើងប្រើប្រាស់ internal ជាមួយ single assembly វាធ្វើការជា public class modifier ។

Example: internal in different Assembly

```
// Code on Assembly1
using System;

namespace Assembly1 {

  public class StudentName {
    internal string name = "Sheeran";
  }

  class Program {
    static void Main(string[] args) {
    }
  }
}
```

នេះជា code នៅក្នុង Assmbly1 យើងបានបង្កើត create internal field name inside class studentname ។ Field នេះអាច accessed ពី assembly **Assembly1 តែប៉ុណ្ណោះ** ។

យើងបង្កើត create another assembly.

```
// Code on Assembly2
using System;

// access Assembly1
using Assembly1;

namespace Assembly2 {
  class Program {
    static void Main(string[] args) {
      StudentName student = new StudentName();

      // accessing name field from Assembly1
      Console.Write(student.name);
      Console.ReadLine();
     }
  }
}
```

នេះជា Code នៅក្នុង Assembly2 យើងបានព្យាយាមដើម្បី access field name នៃ class studentname (Assembly1) ។ ដើម្បី access fields មកពី Assembly1 ដំបូងយើងត្រូវតែដំឡើង set resference នៃ Assemly1 នៅក្នុង Assembly2 ។

code

```
using Assembly1;
```

ការបង្ហាញការប្រើប្រាស់ code ពី Assembly1 ទៅ Assembly2 នៅពេលដែលយើង try to access field name ពី Assembly2 យើងទទួលបានភាព error ។

```
Error    CS0122    'StudentName.name' is inaccessible due to its protection level
```

This is because name is an internal field present in **Assembly1**.

៥.៥. protected internal access modifier

protected internal គឺជាការបញ្ចូលគ្នារវាង protected និង internal access modifiers ។

ពេលដែលយើង declare member protected internal វាអាចចូលទៅកាន់ assembly ណាដូចគ្នា និង Devived class នៃ containing class ពី assembly ដទៃផ្សេងទៀត ។

```
// Code on Assembly1
using System;

namespace Assembly1 {
  public class Greet {
    protected internal string msg="Hello";
  }

  class Program {
    static void Main(string[] args) {
      Greet greet = new Greet();
      Console.WriteLine(greet.msg);
      Console.ReadLine();
     }
  }
}
```

Output

```
Hello
```

ខាងលើជា code ក្នុង Assembly1 ។

នៅក្នុងឧទាហរណ៍ខាងលើយើងបានបង្កើត class ដែលមានឈ្មោះ Greet ដែលមាន field msg ។

field បន្ទាប់គឺជា protected internal ដែលយើងអាច accessed វាបានពី class program ដូចដែលពួកគេស្ថិតនៅក្នុង assembly ដូចគ្នា ។

យើងទទួលបាន class មកពី Greet នៅក្នុង assembly ផ្សេងៗទៀត និង try to acces protected internal field msg ដែលវាទទួលបាន ។

```
// Code on Assembly2
```

```
using System;

// access Assembly1
using Assembly1;

namespace Assembly2 {

  // derived class of Greet
  class Program: Greet {
    static void Main(string[] args) {
      Program greet = new Program();

      // accessing name field from Assembly1
      Console.Write(greet.msg);
      Console.ReadLine();
    }
  }
}
```

Output

```
Hello
```

ខាងលើជា code ឧទាហរណ៍ របស់ Assembly2 ។ នៅក្នុងឧទាហរណ៍ខាងលើយើងទទួលបាន inherited class program ពី class Greet (From Assembly1) ។

```
// accessing name field from Assembly1
Console.Write(greet.msg);
```

យើងអាច access msg ពី class Greet នៃ Assmbly1 មកពី **Assembly2** ពីព្រោះ msg គឺជា protected internal field និងយើងកំពុងព្យាយាមចូលប្រើវាពី child class នៃ Class Greet ។

៥.៦. private protected access modifier

private protected access modifier គឺវាបញ្ចូលគ្នារវាង private និង protected ។ វាអាចប្រើបានពី C# កំណែversion 7.2 និងជំនាន់ក្រោយ ។

យើងប្រកាសសមាជិក member private protected វាអាច accessed បានតែក្នុង class ប៉ុណ្ណោះ និងវាអាចទទួលបានតែ class assembly ។

For example,

```
// Code in Assembly1
using System;

namespace Assembly1 {
  public class StudentName {
    private protected string name = "Sheeran";
  }
```

```
  //derived class of StudentName class
  class Program1 : StudentName {

    static void Main(string[] args) {

      Program1 student = new Program1();

      //  accessing name field from base class
      Console.Write(student.name);
      Console.ReadLine();
    }
  }
}
```

Output

```
Sheeran
```

ខាលើជា code Assembly1 យើងបង្កើត class studentName វាជា Private protected ដែលមាន Field name ។ យើងទទួល inherited Program1 class មកពី Class studentName ។ ដែលមានលំដាប់ពី Private protected member ដែលអាច accessed ពី Derived class assembly តែមួយប៉ុណ្ណោះ យើងអាចចូលប្រើ name ពី Class Program1 ។

Let's derive a class from `StudentName` in another assembly and try to access the private protected field `name` from it. For example,

```
// Code in Assembly2
using System;
//access Assembly1
using Assembly1;

namespace Assembly2 {

  //derived class of StudentName
  class Program : StudentName {
    static void Main(string[] args) {
      Program student = new Program();

      // accessing name field from Assembly1
      Console.Write(student.name);
      Console.ReadLine();
     }
  }
}
```

ខាងលើជា code Assembly2 ពេលដែលយើងព្យាយាម access field name ពី Derived class នៃ class StudentName យើងទទួលបាន Error ។

```
Error    CS0122    'StudentName.name' is inaccessible due to its protection level
```

ពីព្រោះ name field គឺវានៅក្នុង **Assembly1** ហើយ derived class គឺនៅក្នុង **Assembly2**។

៦ .សិក្សាអំពី Properties តួច Attribute

-ប្រកាស Declare ទទួលទៅ control អនុញ្ញតសរសេរក្នុង properties ។

-បង្កើត interface ដែលប្រកាស declare properties ។

-អនុវត្ត properties នៅក្នុង structs និងបណ្ដ class ដែលទទួលពី interface ដែលមានផ្ទុក properties ។

-អនុវត្ត interface មួយដែលប្រើការអនុវត្តន៍ជាក់ស្តែង interface ។

៦.១. ប្រៀបធៀប Field និច Method

ចូរមើល struct ដែលបង្ហាញខាងក្រោមនេះប្រើសម្រាប់បង្ហាញនៅលើទីតាំងកូអរដោនេ (X, Y) ។ បញ្ហាជាមួយ structនេះគឺវាមិនធ្វើទៅតាមលក្ខណៈតួនាទីនៃ encapsulation វាមិនរក្សាទុកទិន្នន័យដោយឡែក data private:

```
1  struct ScreenPosition
2  {
3  public ScreenPosition(int x, int y)
4  {
5  this.X = rangeCheckedX(x);
6  this.Y = rangeCheckedY(y);
7  }
8  public int X;
9  public int Y;
10 private static int rangeCheckedX(int x)
11 {
12 if (x < 0 || x > 1280)
13 {
14 throw new ArgumentOutOfRangeException("X");
15 }
16 return x;
17 }
18 private static int rangeCheckedY(int y)
19 {
20 if (y < 0 || y > 1024)
21 {
22 throw new ArgumentOutOfRangeException("Y");
23 }
24 return y;
25 }
26 }
```

ទិន្នន័យរួម Public data គឺជាគំនិតមិនប្រសើរ ពីព្រោះវាមិនអាចប្រើ ត្រួតពិនិត្យ checked និង controlled ។ ឧទាហរណ៍ ScreenPosition constructor ឆែកមើលជួរលំដាប់ parameters របស់វា ។ ប៉ុន្តែគ្មានជួរលំដាប់ ណាដែលធ្វើនៅលើ "raw" អនុញ្ញាតទៅក្នុង public fields ។

```
1 ScreenPosition origin = new ScreenPosition(0, 0);
2 …
3 int xpos = origin.X;
4 origin.Y = -100; // Oops
```

មធ្យោបាយរួមដើម្បីដោះស្រាយបញ្ហានេះគឺបង្កើត fields private និង បន្ថែមវិធី method មួយរួចហើយផ្លាស់ប្តូរវិធី method ដើម្បីអាន read និងសរសេរតម្លៃនីមួយៗនៃ private field ដែលបានបង្ហាញ ។ ផ្លាស់ប្តូរវិធី methods អាចឆែកមើលជួរលំដាប់ដែលតម្លៃ field ពិតប្រាកដពីព្រោះ constructor ដែលឆែករួចជាស្រេចតម្លៃ initial field ។ ឧទាហរណ៍ (GetX) និងផ្លាស់ប្តូរ (SetX) សម្រាប់ X field ។ កត់ចំណាំថារបៀបឆែក SetX checks តម្លៃ parameter របស់វា:

```
1 struct ScreenPosition
2 {
3 …
4 public int GetX()
5 {
6 return this.x;
7 }
8 public void SetX(int newX)
9 {
10 this.x = rangeCheckedX(newX);
11 }
12 …
13 private static int rangeCheckedX(int x)
14 {
15  //…statement;
16 }
17 private static int rangeCheckedY(int y)
18 {
19  //…statement;
20 }
21 private int x, y;
```

ឧទាហរណ៍ខាងក្រោមនេះបង្កើនតម្លៃនៃ X ដោយ 10 ។ ដើម្បីធ្វើដូច្នេះ វាបានអាចតម្លៃនៃ X ដោយប្រើ GetX method រួចសរសេរតម្លៃនៃ X ដោយ ប្រើវិធីផ្លាស់ប្តូរ SetX modifier method:

```
int xpos = origin.GetX( );

origin.SetX(xpos + 10);
```

ប្រៀបធៀបតម្លៃនេះជាមួយតម្លៃសមមូលនៃ code ប្រសិនបើ X field គឺជា public: origin.X += 10;

៦.២ សិក្សាអំពី Properties

អ្វីដែលហៅថា Properties ?

property គឺជាការឆ្លងកាត់ logical field និងវិធី physical method អ្នកប្រើ property មួយកំណត់ជាក់ច្បាស់មធ្យោបាយដែលអ្នកប្រើ field មួយ។ បន្ទាប់ពី code segment បង្ហាញក្នុង ScreenPosition struct សរសេរឡើងវិញដោយប្រើ properties ។ កាលណាដែលអាននេះ code កំណត់ចំណាំថាដូចខាងក្រោមនេះ:

- x និង y គឺជា fields ដោយឡែក
- X និង Y គឺជា field រួមដូចជា properties ។

X និង Y គឺមិនមែនជា methods ។ វាគ្មានសញ្ញាវង់ក្រចក វាផ្ទុកគ្រាប់តែទទួលនិងបង្កើត ។

```
1 struct ScreenPosition
2 {
3 public ScreenPosition(int X, int Y)
4 {
5 this.x = rangeCheckedX(X);
6 this.y = rangeCheckedY(Y);
7 }
8 public int X
9 {
10 get { return this.x; }
11 set { this.x = rangeCheckedX(value); }
12 }
13 public int Y
14 {
15 get { return this.y; }
16 set { this.y = rangeCheckedY(value); }
17 }
18 private static int rangeCheckedX(int x)
19 {
20  //statement;
21 }
22 private static int rangeCheckedY(int y)
23 {
24  //statement;
25 }
26 private int x, y;
27 }
```

ឧទាហរណ៍ៈ

```
1 class Person
2 {
3   private string name; // field
```

```
    public string Name    // property
    {
      get { return name; }
      set { name = value; }
    }
}

class Program
{
    static void Main(string[] args)
    {
      Person myObj = new Person();
      myObj.Name = "Liam";
      Console.WriteLine(myObj.Name);
      Console.ReadKry();
    }
}
```

```
Liam
```

៦.២.១ ទទួល get Accessor

ប្រសិនបើប្រើ property មួយក្នុងការអាន read context ពេល compiler automatically ទៅក្នុង field របស់អ្នកដូចជា code នៅក្នុងហៅថាទទួល get accessor នៃ property នោះ ។ ឧទាហរណ៍ compiler automatically ដូចនេះ៖

```
1 ScreenPosition origin = new ScreenPosition(0, 0);
2 int xpos = origin.X
3 int ypos = origin.Y;
```

ធ្វើជាក់ស្តែងដូចនេះ: int x = topLeft.x;

៦.២.២ បង្កើត set Accessor

ប្រសិនបើអ្នកប្រើ property មួយនៅក្នុងការសរសេរ context ពេលដែល compiler automatically ក្នុង field code របស់អ្នកទៅក្នុងហៅទៅ set accessor នៃ property នោះ ។

ឧទាហរណ៍ compiler automatically ដូចនេះ៖

```
1   origin.X = 40;
2   origin.Y = 100;
3
```

៦.២.៣ ការអាន Read-Only Properties

អ្នកត្រូវបានអនុញ្ញាតដើម្បីប្រកាស declare property មួយដែលផ្ទុក ទទួល get accessor ។ ក្នុងករណីនេះអ្នកអាចប្រើ property គ្រាន់តែសម្រាប់ អាចក្នុង context ។ ឧទាហរណ៍ ទីនេះមាន X property មួយនៃ ScreenPosition struct ដែលបានប្រកាស read-only property:

```
1 struct ScreenPosition
2 {
3 …
4 public int X
5 {
6 get { return this.x; }
7 }
8 }
```

X property មិនមានផ្ទុក set accessor; ព្យាយាមដើម្បីប្រើ X ក្នុងការសរសេរ context និងបរាជ័យ fail ។

ឧទាហរណ៍ :

```
origin.X = 140; // compile-time error
```

៦.២.៤ សរសេរ Write-Only Properties

អ្នកត្រូវបានអនុញ្ញាតដើម្បីប្រកាស declare property មួយដែលផ្ទុក set accessor តែមួយគត់ ។ ក្នុងករណីនេះអ្នកអាចគ្រាន់តែប្រើ property ក្នុងការសរសេរ context ។ ឧទាហរណ៍ X property នៃ ScreenPosition struct ដែលបានប្រកាសដូចជា write-only property:

```
1 struct ScreenPosition
2 {
3 …
4 public int X
5 {
6 set { this.x = rangeCheckedX(value); }
7 }
8 }
```

X property មិនផ្ទុក get accessor; ព្យាយាមប្រើ X ក្នុងការអាននៅពេលបរាជ័យ fail ។ ឧទាហរណ៍:

```
Console.WriteLine( origin.X ); // compile-time error

origin.X = 200; // compiles ok

origin.X += 10; // compile-time error
```

៦.២.៥ ស្វែងយល់ពី Property Restriction

- អ្នកអាចចាប់ផ្តើម property មួយនៃ struct ឆ្លងកាត់ accessor ឧទាហរណ៍:

```
ScreenPosition location;
location.X = 40; // compile-time error, location not assigned
```

- អ្នកអាចប្រើ property មួយដូចជា ref ឬ out argument ឧទាហរណ៍:

```
MyMethod( ref location.X ); // compile-time error
```

- អ្នកមិនអាចប្រកាស declare ដែលមាន properties ច្រើននៅក្នុង ការប្រកាស statement ទោល ឧទាហរណ៍:

```
const int X { get { ... } set { ... } } // compile-time error
```

- អ្នកមិនអាចប្រកាស declare const ឬ read only properties

```
ឧទាហរណ៍: const int X {get{...} set{...}}//compile-time error
```

Properties គឺមិនមែន true fields ប៉ុន្តែវាមិនមែនជាវិធី true methods ។ អ្នកអាចប្រកាស declare get ឬ set accessor ដែលទាំងពីរនេះ មាននៅក្នុង property ។

```
1 public int X
2 {
3 Console.WriteLine("common code");//compile-time error
4 get { ....}
5 set {.....}
6 }
```

- អ្នកមិនអាចប្រកាស declare ក្នុង property មួយដែលគឺជាប្រភេទ ទំនេរ ។

```
ឧទាហរណ៍:
        public void X{get{...} set {...}}//compile-time error
```

- មិនអាចប្រកាស declare ក្នុង property មួយដោយគ្មាន parameter

ឧទាហរណ៍:

```
public /*nothing*/ X {get{...}set{...}} //compile-time error
```

- អ្នកមិនអាចប្រកាស declare ក្នុង property មួយជាពីរឬច្រើន parameters:

```
public int,string X { get{...}set{...}}//compile-time error
```

៦.៣. ប្រើ Static Properties

ការប្រកាស declare ពីប្រភេទនៃ fields ក្នុង class មួយឬ struct: instance fields (មួយក្នុង object) និង static fields (មានមួយក្នុង ប្រភេទ) ។ អ្នកអាចប្រើ properties ទៅក្នុង mimic instance fields ហើយ អ្នកអាចប្រើ static properties ទៅក្នុង mimic static fields ។ ដែល syntax សម្រាប់ static property គឺច្បាស់លាស់ដូចជាសម្រាប់ instance property លើកលែងតែអ្នកប្រកាស property នៅខាងមុខជាមួយ ពាក្យគន្លឹះ static ។

ឧទាហរណ៍ ទីនេះមាន ScreenPosition struct ដែលពេលនេះ ជាមួយ static read-only property ដែលតំណាងទំហំនៃ screen:

```
1  struct ScreenPosition
 2 {
 3 public static ScreenPosition Limits
 4 {
 5 get { return limits;}
 6 }
 7 private static int rangeCheckedX(int x)
 8 {
 9 if (x < 0 â",â", x > limits.x)
10 {
11 throw new ArgumentOutOfRangeException("X");
12 }
13 return x;
14 }
15 private static int rangeCheckedY(int y)
16 {
17 if (y < 0 â",â", y > limits.y)
18 {
19 throw new ArgumentOutOfRangeException("Y");
20 }
21 return y;
22 }
23 private static ScreenPosition vgaLimits= new ScreenPosition(600, 800);
24 private static ScreenPosition limits = vgaLimits;
25 private int x, y;
26 }
27
```

៦.៤. ការប្រកាស Interface Properties

អ្នកអាចប្រកាស declare properties នៅក្នុង interface ។ ដើម្បីធ្វើ ដូចនេះ អ្នកប្រកាស declare ទទួលពាក្យគន្លឹះ *get* ឬ *set* ដែលមទាំងពីរនេះ ប៉ុន្តែការប្តូររូបរាងនៃ *get* ឬ *set* accessor ជាមួយសញ្ញា semicolon ។

ឧទាហរណ៍៖

```
1 Interface IScreenPosition
2 {
```

```
3 int X { get; set;}
4 int Y { get; set;}
  }
```

class ឬ Struct ដែលអនុវត្ត interface នេះត្រូវការអនុវត្ត accessors ។

ឧទាហរណ៍៖

```
 1 struct ScreenPosition :IScreenPosition
 2 {
 3 public int X
 4 {
 5 get { return x; }
 6 set { x = rangeCheckedX(value);}
 7 }
 8 public int Y
 9 {
10 get {return y;}
11 set {y= rangeCheckedY(value);}
12 }
13 private int x, y;
14 }
```

ប្រសិនបើអនុវត្ត interface properties ក្នុង class មួយអ្នកអាចប្រកាស declare property ដូចជាការអនុវត្តន៍ជាក់ស្តែង ដែលអនុញ្ញតឱ្យ classes មានលើការអនុវត្តន៍

ឧទាហរណ៍៖

```
 1 class ScreenPosition : IScreenPosition
 2 {
 3 public virtual int X
 4 {
 5 get{…}
 6 set {…}
 7 }
 8 public virtual int Y
 9 {
10 get {…}
11 set {…}
12 }
13 private int x, y;
14 }
```

ទំហំ property ត្រឡប់ទៅ instance នៃ struct មួយដែលហៅថា ទំហំដែលស្ថិតក្នុង System.Drawing namespace ។ ទំហំ Size struct ផ្ទុក public read/write int មួយដែលហៅថា Height និងមួយទៀតហៅថា Width៖

```
1 struct Size
2 {
3 public int Height
4 {
```

```
5
6   get { … }
7   set { … }
8   }
9   public int Width
10  {
11  get { … }
12  set { … }
13  }
14  }
```

ប្រធានបទ	ការអនុវត្តន៍

ប្រកាស Declare read/write property	ប្រកាស Declare ប្រភេទនៃ property ឈ្មោះរបស់វា getAccessor និង set accessor ឧទាហរណ៍:
	struct ScreenPosition
	{
	…
	public int X
	{
	get { … }
	set { … }
	}
	…
	}

ប្រកាស read-only property	ប្រកាស Declare property ជាមួយ get accessor ឧទាហរណ៍:
	struct ScreenPosition
	{
	…
	public int X
	{
	get { … }
	}
	…
	}
	}

ប្រកាស Declare write-only property	ប្រកាស Declare property ជាមួយ set accessor ឧទាហរណ៍:
	struct ScreenPosition
	{
	…
	public int X
	{
	set { … }
	}

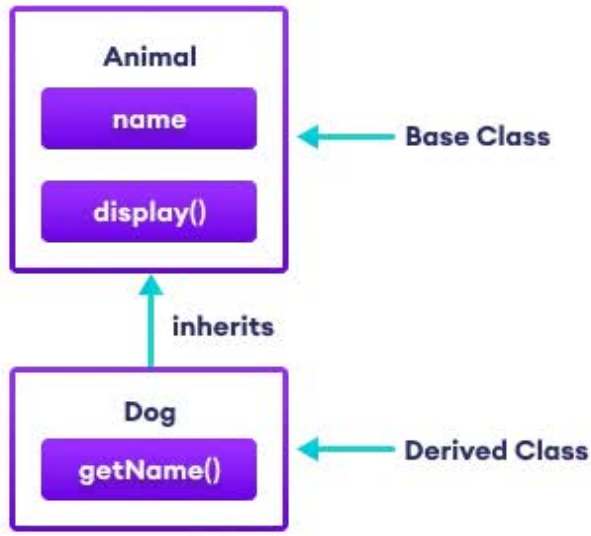
	… }

ប្រកាស Declare property ក្នុង interface	ប្រកាស Declare property ជាមួយ ពាក្យគន្លឹះ get ឬ set
	ឧទាហរណ៍:
	interface IScreenPosition
	{
	int X { get; set; } // no body
	int Y { get; set; } // no body
	}

អនុវត្ត interface property	ក្នុង class ឬ struct ដែលអនុវត្ត interface ប្រកាស declare property និងអនុវត្ត accessors។ឧទាហរណ៍:
	struct ScreenPosition : IscreenPosition
	{
	public int X
	{
	get { … }
	set { … }
	}
	public int Y
	{
	get { … }
	set { … }
	}
	}

៧. សិក្សាអំពី Inheritance

Inheritance ស្ថិតក្នុងផ្នែកមួយនៃ OOP(Object Oriented Programming)ដែលអនុញ្ញាតឲ្យមានការបង្កើត Class ថ្មីចេញមកពី Class ដែលមានស្រាប់។ នៅក្នុងដំណើរការនេះ Class ដែលមានរួចស្រេចគេហៅថា Base Class ឬ Parent Class ឬ Super Class ហើយ Class ដែលបង្កើតថ្មីត្រូវបានគេហៅថា Derived Class ឬ Child Class ឬ Sub Class ។ Inheritance គឺមកពីពាក្យ Inherit ដែលមានន័យថា Class មានស្រាប់ផ្តល់មរតកមក Class ខាងក្រោមឬ Class ថ្មី។




```
class Vehicle  // base class (parent)
{
  public string brand = "Ford";  // Vehicle field
  public void honk()             // Vehicle method
  {
    Console.WriteLine("Tuut, tuut!");
  }
}

class Car : Vehicle  // derived class (child)
{
  public string modelName = "Mustang";  // Car field
}

class Program
{
  static void Main(string[] args)
  {
    // Create a myCar object
    Car myCar = new Car();

    // Call the honk() method (From the Vehicle class) on the
myCar object
    myCar.honk();

    Console.WriteLine(myCar.brand + " " + myCar.modelName);
  }
}
```

៧.១ ការប្រើប្រាស់ sealed Keyword

ប្រសិនបើយើងមិនចង់ឱ្យ Class ផ្សេងទទួល តម្លៃពី inherit Class ទេ យើងអាចប្រើប្រាស់ Keyword sealed ជំនួស

```
sealed class Vehicle
{
   ...
}

class Car : Vehicle
{
   ...
}
```

៧.២ សារប្រយោជន៍នៃការប្រើប្រាស់ Inheritance:

- ប្រើប្រាស់សមត្ថភាពរបស់ Code ឡើងវិញដោយពុំចាំបាច់ត្រូវសរសេរម្តងទៀត
- បង្កើនជំនឿចិត្តនៅលើ Code ដោយមិនចាំបាច់បារម្ភអំពីកំហុស logic នោះទ
- អនុញ្ញាតឲ្យ Class ថ្មីប្រើប្រាស់ fields និង methods ឡើងវិញពី Base Class បើត្រូវការ
- អនុញ្ញាតឲ្យ implement ក្នុង Class ថ្មី បើសិនត្រូវការ(អាចបន្ថែម fields និង methods ថែមបាននៅក្នុង Class ថ្មី)
- ងាយស្រួលគ្រប់គ្រងនិងពង្រីកកម្មវិធី។

៧.៣ ប្រភេទរបស់ Inheritance:

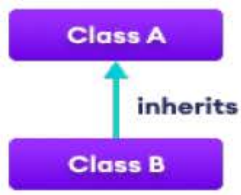
ចំណុចបន្តទៀតនេះគឺយើងនឹងសិក្សាទាក់ទងនឹងប្រភេទរបស់ Inheritance ដែលប្រភេទរបស់ Inheritance មាន ៥ ប្រភេទ

- Single Inheritance
- Multiple Inheritance
- Multilevel Inheritance
- Hierarchical Inheritance
- Hybrid Inheritance

៧.៣.១ Single Inheritance:

Single Inheritance មានន័យថា Derived Class មួយ inherited ចេញពី Base Class មួយ។

The following is an example of Single Inheritance in C#. In the example, the base class is Father and declared like the following code snippet



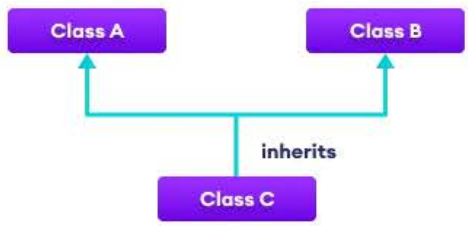

```
1  class Father
2  {
3    public void Display()
4    }
5
6      Console.WriteLine("Display");
7    {
8  }
```

Our derived class is Son and is declared below

```
1  class Son: Father
2  {
3    public void DisplayOne()
4      {
5           Console.WriteLine("DisplayOne");
6      }
7  }
```

៧.៣.២ សិក្ខារំពឹ Multiple Inheritance:

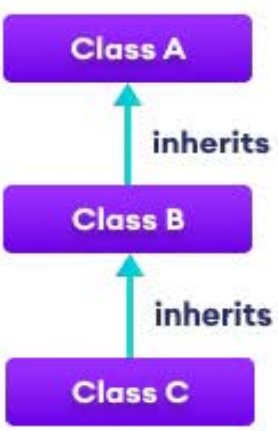
Multiple Inheritance មានន័យថា Derived Class មួយ inherited ចេញពី Base Class ច្រើន។



```
1  Interface IFirst Interface
2  {
3  vold syntethod(); // Interface method
4  }
5  Interface SecondInterface
6  {
7  vold syotherffethod(); // interface method
8  }
9  // Implement multiple interfaces
10 class DonoClass IFirstInterface, IsecondInterface
11 {
12       public void syftethod()
13       {
14          Console.WriteLine("Some text..");
15       }
16       public vold myothereal()
17       {
18          Console.WriteLine("Some other text..."
19       }
20 }
```

៧.៣.៣ Multilevel Inheritance:

Multilevel Inheritance មានន័យថា Derived Class មួយ inherited ចេញពី Derived Class មួយទៀត ហើយវាមាន Base Class តែមួយគត់ ។

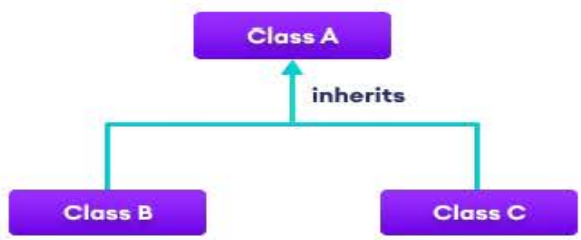



```
1 public class Animal
2 {
3   public void eat()
4   {
5     Console.WriteLine("Eating...");
6   }
```

```
}
public class Dog: Animal
{
    public void bark
    {
        Console.WriteLine("Barking...");
    }
}
public class BabyDog: Dog
{
   public void weep()
    {
      Console.WriteLine("Weeping...");
    }
 }
```

៧.៣.៤ Hierarchical Inheritance:

Hierarchical Inheritance មាននិយមថា Derived Class ច្រើន inherited ចេញពី Base Class តែ
មួយ ។

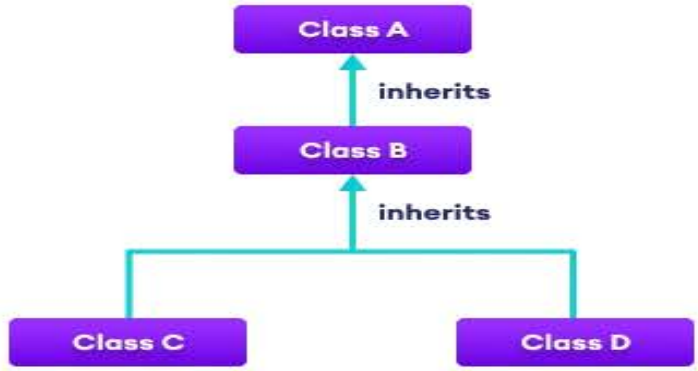



```
class Father
{
   public void display()
   {
     Console.WriteLine("Display...");
   }
}
class Son Father
{
   public void displayOne()
  {
     Console.WriteLine("Display One");
  }
}
class Daughter Father
{
```

```
17      public void displayTwo()
18      {
19          Console.WriteLine("Display Two");
20      }
21  }
```

៧.៣.៤ Hybrid Inheritance:

Hybrid Inheritance : មានន័យថាគឺជាសំណុំឬបណ្តុំ Inheritance ដែលរួមបញ្ចូលនូវប្រភេទ Inheritance sa (Single Inheritance, Multiple Inheritance, Multilevel Inheritance, Hierarchical Inheritance) ។




```
1  public class Moveable
2  {
3      public void Move(int x, int y)
4      {
5          // Move the object to the specified x,y coordinates
6      }
7  }
8  public class Attackable
9  {
10     public void Attack(Gameobject target)
11     {
12         // Attack the specified target object
13     }
14 }
15 public class Interactable
16 {
17     public void Interact(Gameobject target)
18     {
19         // Interact with the specified target object
20     }
```

```
}
public class PlayerCharacter : Moveable, Attackable,
Interactable
{
    // Additional player-specific properties and methods
}
```

ឧទាហរណ៍ទី១៖

```
public interface IButton
{
    void Click();
}
public interface ICheckbox
{
    void Check();
    void Uncheck();
}
public interface ITextbox
{
    void SetText(string text);
    string GetText();
}
public class ButtonCheckboxTextBox : IButton, ICheckbox,
ITextbox
{
    private bool _checked;
    private string _text;
    public void Click()
    {
        Console.WriteLine("Button Clicked");
    }
    public void Check()
    {
        _checked = true;
        Console.WriteLine("Checkbox Checked");
    }
    public void Uncheck()
    {
        _checked = false;
        Console.WriteLine("Checkbox Unchecked");
    }
    public void SetText(string text)
    {
        _text = text;
    }
    public string GetText()
    {
        return _text;
    }
```

```
}
```

ឧទាហរណ៍ទី២៖

```
using System;
namespace consoleapp1{
    class GrandFather {

        public void land() {
                Console.WriteLine("GrandFather's land");
        }

}

class Father : GrandFather {

        public void home() {
                Console.WriteLine("Father's home");
        }

        public void Car() {
                Console.WriteLine("Father's Car");
        }
}

// Inherit /derived / extends
class Son : Father {

        // son constructor
        public Son() {
                Console.WriteLine("Son...");
        }

        public void mobile() {
                Console.WriteLine("Son's mobile");
        }
}

class Daughter : Father {

        // Daughter constructor
        public Daughter() {
                Console.WriteLine("Daughter...");
        }

        public void purse() {
                Console.WriteLine("Daughter's purse");
        }
}

/*
```

```
48  * Test hybrid inheritance
49  */
50 public class TestHybridInheritance {
51
52         public static void Main(String[] args) {
53
54                 // Son object
55                 Son s = new Son();
56                 s.land();// Grand father method
57                 s.Car(); // Father method
58                 s.home();// Father method
59                 s.mobile();// son method
60
61                 // Daughter object
62                 Daughter d = new Daughter();
63                 d.land();// Grand father method
64                 d.Car(); // Father method
65                 d.home();// Father method
66                 d.purse();// son method
67
68         }
69 }
70 }
```

Output

```
Son...
GrandFather's land
Father's Car
Father's home
Son's mobile
Daughter...
GrandFather's land
Father's Car
Father's home
Daughter's purse

[Execution complete with exit code 0]
```

៨.សិក្ខាអំពី Encapsulation

Encapsulation is defined 'as the process of enclosing one or more items within a physical or logical package'. Encapsulation, in object oriented programming methodology, prevents access to implementation details.

Abstraction and encapsulation are related features in object oriented programming. Abstraction allows making relevant information visible and encapsulation enables a programmer to *implement the desired level of abstraction*.

Encapsulation is implemented by using **access specifiers**. An **access specifier** defines the scope and visibility of a class member.

C# supports the following access specifiers −

- Public
- Private
- Protected
- Internal
- Protected internal

៨.១ Public Access Specifier

Public access specifier allows a class to expose its member variables and member functions to other functions and objects. Any public member can be accessed from outside the class.

The following example illustrates this −

```
using System;

namespace RectangleApplication {
   class Rectangle {
      //member variables
      public double length;
      public double width;

      public double GetArea() {
         return length * width;
      }
      public void Display() {
         Console.WriteLine("Length: {0}", length);
         Console.WriteLine("Width: {0}", width);
         Console.WriteLine("Area: {0}", GetArea());
      }
   }//end class Rectangle

   class ExecuteRectangle {
      static void Main(string[] args) {
```

```
         Rectangle r = new Rectangle();
         r.length = 4.5;
         r.width = 3.5;
         r.Display();
         Console.ReadLine();
      }
   }
}
```

When the above code is compiled and executed, it produces the following result −

```
Length: 4.5
Width: 3.5
Area: 15.75
```

In the preceding example, the member variables length and width are declared **public**, so they can be accessed from the function Main() using an instance of the Rectangle class, named **r**.

The member function *Display()* and *GetArea()* can also access these variables directly without using any instance of the class.

The member functions *Display()* is also declared **public**, so it can also be accessed from *Main()* using an instance of the Rectangle class, named **r**.

៨.២ Private Access Specifier

Private access specifier allows a class to hide its member variables and member functions from other functions and objects. Only functions of the same class can access its private members. Even an instance of a class cannot access its private members.

The following example illustrates this −

```
using System;

namespace RectangleApplication {
   class Rectangle {
      //member variables
      private double length;
      private double width;
```

```
      public void Acceptdetails() {
         Console.WriteLine("Enter Length: ");
         length = Convert.ToDouble(Console.ReadLine());
         Console.WriteLine("Enter Width: ");
         width = Convert.ToDouble(Console.ReadLine());
      }
      public double GetArea() {
         return length * width;
      }
      public void Display() {
         Console.WriteLine("Length: {0}", length);
         Console.WriteLine("Width: {0}", width);
         Console.WriteLine("Area: {0}", GetArea());
      }
   }//end class Rectangle

   class ExecuteRectangle {
      static void Main(string[] args) {
         Rectangle r = new Rectangle();
         r.Acceptdetails();
         r.Display();
         Console.ReadLine();
      }
   }
}
```

When the above code is compiled and executed, it produces the following result −

```
Enter Length:
4.4
Enter Width:
3.3
Length: 4.4
Width: 3.3
Area: 14.52
```

In the preceding example, the member variables length and width are declared **private**, so they cannot be accessed from the function Main(). The member functions *AcceptDetails()* and *Display()* can access these variables. Since the member functions *AcceptDetails()* and *Display()* are declared **public**, they can be accessed from *Main()* using an instance of the Rectangle class, named **r**.

៨.៣ Protected Access Specifier

Protected access specifier allows a child class to access the member variables and member functions of its base class. This way it helps in implementing inheritance. We will discuss this in more details in the inheritance chapter.

៨.៤ Internal Access Specifier

Internal access specifier allows a class to expose its member variables and member functions to other functions and objects in the current assembly. In other words, any member with internal access specifier can be accessed from any class or method defined within the application in which the member is defined.

The following program illustrates this −

```
using System;
namespace RectangleApplication
{
   class Rectangle {
      //member variables
      internal double length;
      internal double width;

      double GetArea() {
         return length * width;
      }
      public void Display() {
         Console.WriteLine("Length: {0}", length);
         Console.WriteLine("Width: {0}", width);
         Console.WriteLine("Area: {0}", GetArea());
      }
```

```
   }//end class Rectangle

   class ExecuteRectangle {
      static void Main(string[] args) {
         Rectangle r = new Rectangle();
         r.length = 4.5;
         r.width = 3.5;
         r.Display();
         Console.ReadLine();
      }
   }
}
```

When the above code is compiled and executed, it produces the following result −

```
Length: 4.5
Width: 3.5
Area: 15.75
```

In the preceding example, notice that the member function *GetArea()* is not declared with any access specifier. Then what would be the default access specifier of a class member if we don't mention any? It is **private**.

៨.៥ Protected Internal Access Specifier

The protected internal access specifier allows a class to hide its member variables and member functions from other class objects and functions, except a child class within the same application. This is also used while implementing inheritance.

៩. សិក្សាអំពី Polymorphism

Polymorphism កើតឡើងពីពាក្យក្រិច poly ដែលមានន័យថាច្រើន ផ្សំជាមួយនឹងពាក្យក្រិច morphism ដែលមានន័យថាទម្រង់។ ហេតុនេះ polymorphism មានន័យថាច្រើនទម្រង់។ នៅក្នុង OOP, polymorphism គេសំដៅទៅរក methods ទាំងឡាយមានឈ្មោះដូចគ្នានៅក្នុង Class Hierarchy, ដែលមានការប្រព្រឹត្តផ្សេងៗគ្នាអាស្រ័យទៅតាម Object ដែលវានីមួយៗសំដៅទៅរក។

```
1 class Animal  // Base class (parent)
2 {
3   public void animalSound()
4   {
5     Console.WriteLine("The animal makes a sound");
```

```
  }
}

class Pig : Animal  // Derived class (child)
{
  public void animalSound()
  {
    Console.WriteLine("The pig says: wee wee");
  }
}

class Dog : Animal  // Derived class (child)
{
  public void animalSound()
  {
    Console.WriteLine("The dog says: bow wow");
  }
}
```

១០. សិក្សាអំពី Abstraction

Abstraction គឺជាដំណើរការមួយដែលលាក់ព័ត៌មានលម្អិតជាក់លាក់ និងបង្ហាញតែព័ត៌មានចាំបាច់ដល់អ្នកប្រើប្រាស់។ នៅក្នុង Abstraction មាន class មួយនិង method មួយដែលយើងត្រូវស្គាល់គឺ:

-Abstract Class: គឺជា class មួយដែលមិនអាចបង្កើត object បាន ចឹងបើសិនជាចង់បើ abstract class យើងត្រូវ inherited វាទៅប្រើក្នុង class ផ្សេង។

-Abstract Method: គឺគេប្រើបានតែក្នុង abstract class តែប៉ុណ្ណោះ ហើយវាជា function ដែលគ្មាន body ហើយ body គឺត្រូវបានសរសេរបន្ថែមនៅក្នុង class ដែលបាន inherited ទៅ។

```
abstract class Animal
{
  public abstract void animalSound();
  public void sleep()
  {
    Console.WriteLine("Dog");
  }
}
```

១១. សិក្សាអំពី Interface:

Interface គឺប្រើដើម្បីជំនួស Abstraction ។ ដែលនៅក្នុង Interface គឺអាចផ្ទុកបានតែ abstract methods និង properties តែប៉ុណ្ណោះ ។ Members នៅក្នុង Interface នឹងក្លាយជា abstract និង public ជាស្រេច បើសិនជាយើងមិនដាក់អ្វី នៅពីមុខ members ។

ដើម្បីចូលដំណើរការ interface methods ត្រូវតែត្រូវបាន "implemented" (inherited) ដោយ Class ផ្សេងទៀត ។ ដើម្បី implement Interface មួយ ត្រូវប្រើនិមិត្តសញ្ញា : (ដូចគ្នានឹង inheritance ដែរ)។ តួនៃ interface method (body of the interface method) គឺត្រូវបានផ្តល់ដោយ "implement" Class ។

ចំណាំថាអ្នកមិនចាំបាច់ប្រើពាក្យគន្លឹះ override keyword នៅពេល implementing interface នោះទេ ។

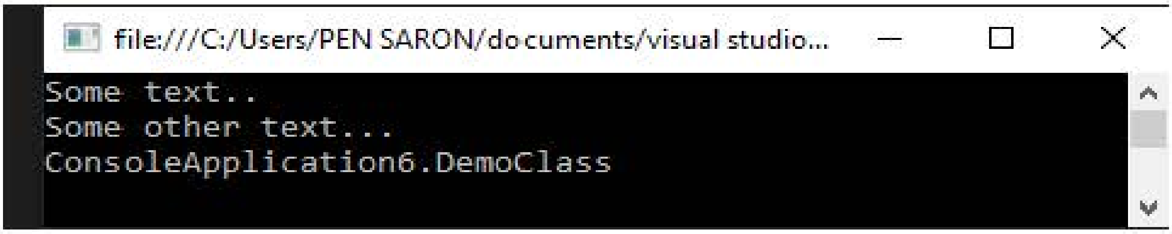
```
// interface
interface Animal
{
  void animalSound(); // interface method (does not have a
body)
  void run(); // interface method (does not have a body)
}
```

១១.១ Multiple Interfaces

ដើម្បីimplement multiple interfaces ច្រើន សូមបំបែកពួកវាដោយសញ្ញាក្បៀស (,) ។

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

namespace ConsoleApplication6
{
    interface IFirstInterface
    {
        void myMethod(); // interface method
    }

    interface ISecondInterface
    {
        void myOtherMethod(); // interface method
    }

    // Implement multiple interfaces
    class DemoClass : IFirstInterface, ISecondInterface
    {
        public void myMethod()
        {
            Console.WriteLine("Some text..");
        }
        public void myOtherMethod()
        {
```

```
                Console.WriteLine("Some other text...");
            }
    }

    class Program
    {
        static void Main(string[] args)
        {
            DemoClass myObj = new DemoClass();
            myObj.myMethod();
            myObj.myOtherMethod();

            Console.WriteLine(myObj + "");
            Console.ReadKey();
        }
    }
}
```



ជំពូកទី ១០

Introduction to ADO.NET

ដើម្បីចូលប្រើទិន្នន័យសហគ្រាស enterprise ពីកម្មវិធី .NET application ត្រូវបានត្រូវការ interface ។ interface នេះដើរតួជាស្ពានរវាងប្រព័ន្ធ RDBMS និងកម្មវិធី .Net ។ ADO.NET គឺជាចំណុចប្រទាក់ interface មួយដែលត្រូវបានបង្កើតឡើងដើម្បីភ្ជាប់កម្មវិធី connect .NET ទៅប្រព័ន្ធ RDBMS systems។

កម្មវិធីសហគ្រាស Enterprise applicationsគ្រប់គ្រងបរិមាណទិន្នន័យច្រើនទិន្នន័យនេះត្រូវបានរក្សាទុកជាចម្បងនៅក្នុងមូលដ្ឋានទិន្នន័យដែលទាក់ទងដូចជា Oracle, SQL Server, Access និងដូច្នេះនៅលើ។ databases មូលដ្ឋានទិន្នន័យទាំងនេះប្រើ Structured Query Language (SQL) សម្រាប់ការទាញយកទិន្នន័យ។ នៅក្នុង .NET framework Microsoft បានចាប់ផ្តើមកំណែទម្រង់ជំនាន់ថ្មីនៃ Active X Data Objects (ADO) ដែលហៅថា ADO.NET ។ កម្មវិធី .NET ណាមួយទាំង Windows-based ឬ web-based អាចធ្វើអន្តរកម្មជាមួយ database ដោយប្រើ classes សម្បូរបែបនៃ ADO.NET library ។ អាចចូលប្រើដំណើរការមូលដ្ឋានទិន្នន័យណាមួយបានដោយប្រើ connected ឬ disconnected architecture ស្ថាបត្យកម្មដែលបានតភ្ជាប់ ឬផ្តាច់។

មានបច្ចេកវិទ្យាចូលដំណើរការប្រើទិន្នន័យជាច្រើនដែលអាចប្រើបានមុន ADO.NET ជាចម្បងដូចខាងក្រោម៖

Open Database Connectivity (ODBC)

Data Access Objects (DAO)

Remote Data Objects (RDO)

Active X Data Objects (ADO)

១. និយមន័យ

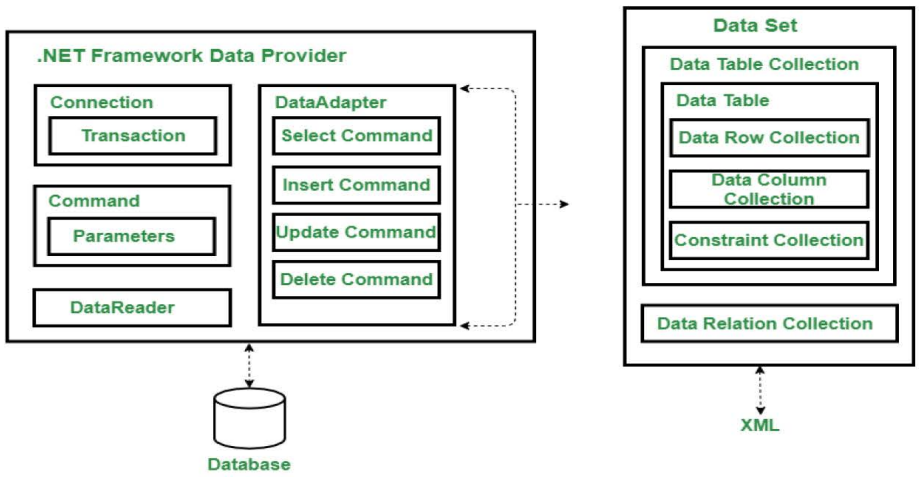
និយមន័យ៖ ADO គឺជាសំណុំសម្បូរបែបនៃ Class, interfaces, structures និង enumerated types ដែលបានរាប់បញ្ចូលការគ្រប់គ្រងដំណើរការចូលប្រើប្រាស់ទិន្នន័យពីប្រភេទផ្សេងៗគ្នានៃកន្លែងផ្ទុកទិន្នន័យ ។

ADO គឺជា interface component-based object-oriented ដែលផ្អែកលើសមាសធាតុសាមញ្ញដើម្បីចូលប្រើទិន្នន័យមិនថាមូលដ្ឋានទិន្នន័យដែលទាក់ទង ឬមិនទាក់ទង relational or non-relational databases។ វា គឺជា successor អ្នកស្នងតំណែងរបស់ DAO និង RDO ។

ADO កាត់បន្ថយចំនួន នៃ objects។ វាមានដូចជា properties, methods និង events។ ADO គឺជា built ត្រូវបានបង្កើតឡើងនៅលើ COM; ជាពិសេស Activex។

ADO គាំទ្រការចូលប្រើទិន្នន័យជាសកលដោយប្រើ Object Linking និង Embedding សម្រាប់ Databases (OLEDB) ។នេះមានន័យថាមិនមានការតឹងលើប្រភេទនៃទិន្នន័យដែលអាចចូលប្រើបានទេ។

២. ប្រភេទ ADO.NET Architecture



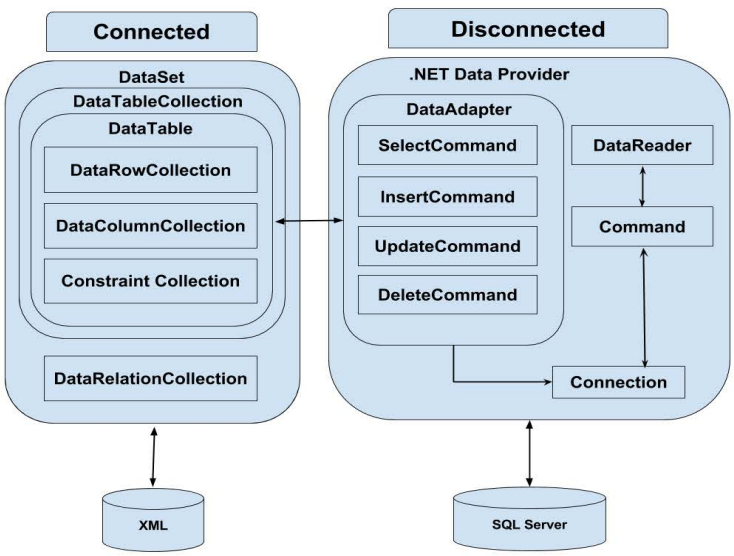
ADO.NET ផ្តល់ជាចម្បងនូវប្រភេទស្ថាបត្យកម្មពីរ two types of architectures ដូចខាងក្រោម៖

១. Connected Architecture (ស្ថាបត្យកម្មភ្ជាប់)

២. Disconnected Architecture (ស្ថាបត្យកម្មផ្តាច់)

៣. Connected and Disconnected Architectures

Figure ខាងក្រោមបង្ហាញពីរបៀបធ្វើការរបស់ connected and disconnected architectures



៣.១ Connected Architecture

connected architecture, ការភ្ជាប់ជាមួយប្រភពទិន្នន័យ data source ត្រូវបានរក្សាទុក Open ជានិច្ចសម្រាប់ការចូលប្រើទាញទិន្នន័យ ក៏ដូចជាប្រតិបត្តិការរៀបចំទិន្នន័យ data manipulation ។

ADO តំណាងឱ្យ active data object ដែលដើរតួជាអ្នកសម្របសម្រួលរវាងផ្នែកអតិថិជន client side និងផ្នែកខាងម៉ាស៊ីនមេ server side ។ ភាគីអតិថិជនClient side មិនអាចធ្វើអន្តរកម្មដោយផ្ទាល់ជាមួយផ្នែកម៉ាស៊ីនមេ server side បានទេ ដូច្នេះមាន ADO.NET ដែលដើរតួជាអ្នកសម្រុះសម្រួលរវាងផ្នែក front end and back end ខាងមុខនិងផ្នែកខាងក្រោយដូចក្នុងរូបខាងក្រោម៖

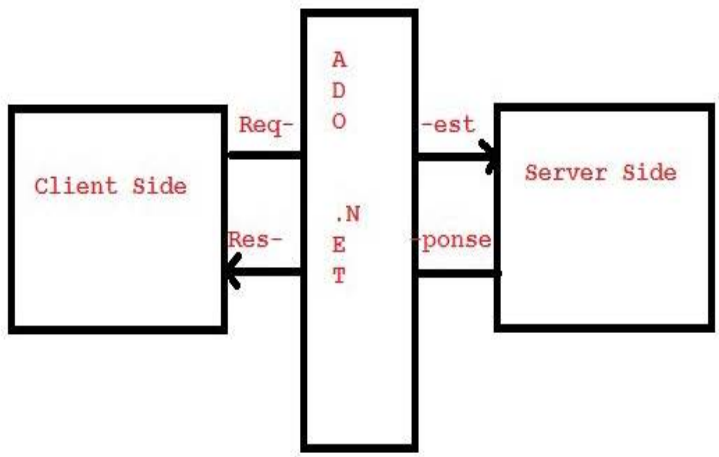


Figure: ADO.NET

Figure: បង្ហាញពីclient-side application interacts ជាមួយ server-side application through ADO.NET ដែលមានទំនាក់ទំនងគ្នា និងដើរតួជាអ្នកសម្របសម្រួលគ្នារវាងfront end និង back end ។

៣.១.១ ប្រភេទ ADO.NET Connected architecture

ADO.NET Connected architecture ប្រើសម្រាប់connection objects មាន៣ ប្រភេទ៖

- SqlConnection
- SqlCommand
- SqlDataReader
- Dataset

៣.១.១.១ Connection

នៅក្នុង connection-oriented architecture ការតភ្ជាប់ មូលដ្ឋានទិន្នន័យ Database ត្រូវបានភ្ជាប់ទៅផ្នែកខាងមុខ front end បន្ទាប់មកពាក្យបញ្ជាបញ្ជូន command ឆ្លងកាត់ query ទៅម៉ាស៊ីនមេពីផ្នែកខាងក្រោយ back end និងនៅលើម៉ាស៊ីនមេ លទ្ធផលដែលត្រូវបានបង្កើត។ លទ្ធផលដែលត្រូវបានបង្កើតនឹងត្រូវបានអានដោយ DataReader ។ នេះជារូបខាងក្រោម៖

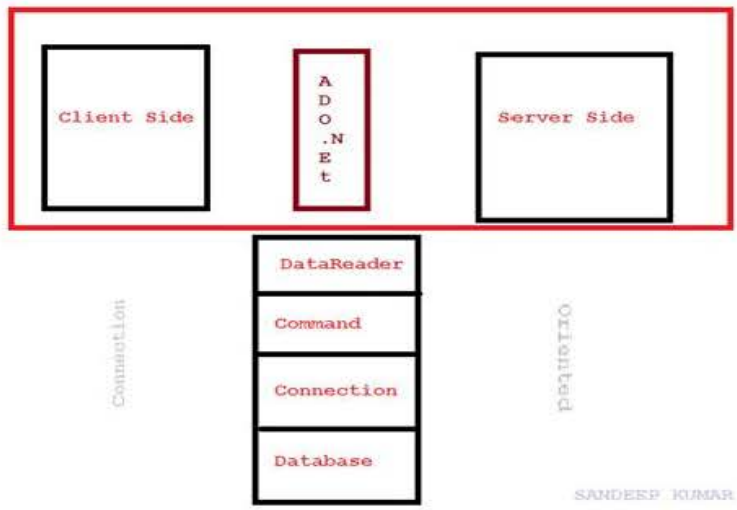


figure បានបង្ហាញ connection-oriented architecture នៃ ADO.NET មានដូចជា Connection, Command និង DataReader ប្រើប្រាស់ដើម្បីបង្កើតតភ្ជាប់ ឬទំនាក់ទំនងគ្នារវាង front end និង back end ហើយឆ្លងកាត់ប្រោះ passing query ពី front end to the back end ។

Connection class ប្រើប្រាស់ដើម្បីបង្កើត connection ភ្ជាប់គ្នារវាង front end និង back end ។

១. ADO.NET objects ដំបូងមួយគឺជា connection object ដែលអនុញ្ញាតឱ្យអ្នកបង្កើតការ connection តភ្ជាប់ទៅប្រភពទិន្នន័យ data source ។

២. connection objects ត្រូវមាន methods សម្រាប់ បើក opening connections និងបិទ closing connections សម្រាប់ធ្វើការបញ្ជូនទិន្នន័យ transaction of data ។

៣ .Net Framework បានផ្តល់នូវ class connection មាន ២ ប្រភេទ ៖ sqlconnection object គឺបង្កើតឡើងដើម្បីធ្វើការ connect ជាមួយ Microsoft SQL Server និង OleDbConnection object បានទូលំទូលាយ ជាមួយ Databases ផ្សេងៗ ដូចជា Microsoft Access និង Oracle ។

៤.connection តម្រូវឱ្យធ្វើអន្តរកម្មជាមួយមូលដ្ឋានទិន្នន័យ ។ Connection object ជួយកំណត់អត្តសញ្ញាណ database server name, user name និង password ដើម្បី connect ជាមួយ database ។ Connection object គឺប្រើ commands ដើម្បី Connection ជាមួយ database

៥. Connection object មានទំនួលខុសត្រូវ responsibility ក្នុង establishing ការបង្កើតការតភ្ជាប់ជាមួយ data store ។

៦. របៀបប្រើប្រាស់ Sqlconnection object ៖

- Instantiate the SqlConnection class.
- Open connection.
- Pass the connection to ADO.NET objects.
- Perform the database operations with ADO.NET object
- Close the connection.

៣.១.១.២ Connection String

No.	Connection String Parameter Name	Description
1	Data Source	Identify the server. Could be a local machine, machine domain name, or IP Address.
2	Initial Catalog	Database name.
3	Integrated Security	Set to SSIP to make a connection with the user's window log in.
4	User ID	Name of user configured in SQL Server.
5	Password	Password matching SQL Server User ID

The connection string is different for each of the various data providers available in .NET 2.0. There are different connection strings for the various types of data sources. You can find a list of all the available providers for creating a connection in a table:

No	Provider	Description
1	System.Data.SqlClient	Provides data for Microsoft SQL Server
2	System.Data.OleDb	Provides data access for data sources exposed using OLE DB
3	System.Data.Odbc	Provides data access for data source exposed using ODBC.
4	System.Data.OracleClient	Provides data access for Oracle.

```
SqlConnection con;
con = new SqlConnection ("Server=Krushna;Database=Anagha;Uid=sa;Pwd=sa");
```

OR

```
SqlConnection con=new SqlConnection("Integrated security=true;initial
catalog=Student;Data source=.");
```

OR

```
SqlConnection Con=new SqlConnection("User id=sa;Password=sa123;Databa
se=Student;Server=.");
```

៣.១.១.៣. Command Object

Command Object គឺជា Object មួយដែលវាមានតួនាទីរក្សា Sql Statement ដើម្បីយកទៅ Execute ។ វាមានសារៈសំខាន់ចំពោះ Connected និង Disconnected Environtment ពីព្រោះវាជាអ្នកបញ្ជាការរាល់ការទាញទិន្នន័យពី Table ដូចនេះវាមានតួនាទីសំខាន់ ក្នុងការបញ្ជ ទិន្នន័យ ដើម្បីយកទៅ Execute ។ ដូចជានៅក្នុង Connected Environtment វាជាអ្នកផ្ញើក Sql Statement ហើយ Execute ដើម្បី បញ្ចូលទិន្នន័យ កែប្រែទិន្នន័យ និងលប់ទិន្នន័យពីក្នុង Table តាមរយៈ Connection ណាមួយ ឧទាហរណ៍ ៖ OledbCommand InsertCmd = new OledbCommand(Sql Statement, Connection String);

Data Command ប្រើប្រាស់សម្រាប់អនុវត្តន៍ដូចជា ការបន្ថែម (Insert), លុប (Delete), កែ (Update) និងស្វែងរក (Select) ជាមួយ Data Source ដោយផ្ទាល់ ដែលមិនចាំបាច់ ពឹងផ្អែក DataAdapter។ យើងអាចប្រើប្រាស់វា សម្រាប់ការងារ ច្រើនដូចជា:

- ដំណើរការជាមួយ DDL (Data Definition Language) commands ដូចជាការ បង្កើត ឬលុប Table ក៏បាន។
- ដំណើរការជាមួយ Store Procedure, Function Procedure លើ Database Server ដូចជាការ Access, លុប (អាស្រ័យទៅតាមការផ្តល់សិទ្ធិ "Permission" ទៅឲ្យវា)។
- ដំណើរការជាមួយ Statement SQL ដូចជា Select, Insert, Update និង Delete ជាមួយ Data Source ដោយផ្ទាល់បាន។ ជាងនេះទៅទៀតយើងក៏អាចទាញទិន្នន័យ (Read Data) ដោយផ្ទាល់ពី Data Source បានដោយប្រើប្រាស់ DataReader class ហើយមិនចាំបាច់ប្រើប្រាស់ DataSet នោះឡើយ។ វាមានទំហំតូចជាង DataSet ហើយយើងអាចប្រើប្រាស់វាបានក្នុងទម្រង់ជា Read-Only និង Forward-Only។

Data Command មានពីរប្រភេទ គឺ SQL Command និង OleDbCommand ។

1.ការប្រើប្រាស់ SQL Command វាត្រូវបានប្រើប្រាស់សម្រាប់បង្កើត Data Command ដែលអាច Access ទៅកាន់ Data Source បានដូចជា SQL Statement, Store Procedure...។

2. Command object executes SQL statements ក្នុង Databases ។ SQL statements អាចមានដូចជា SELECT, INSERT, UPDATE, DELETE ។ វាប្រើ connection object ដើម្បីអនុវត្តសកម្មភាពទាំងនេះនៅលើមូលដ្ឋានទិន្នន័យ database។

Command object executes SQL statements ក្នុង Databases ។ SQL statements អាចមានដូចជា SELECT, INSERT, UPDATE, DELETE ។ វាប្រើ connection object ដើម្បីអនុវត្តសកម្មភាពទាំងនេះនៅលើមូលដ្ឋានទិន្នន័យ database។

ប្រភេទ Connection object ដែលត្រូវអនុវត្តភ្ជាប់ជាមួយមូលដ្ឋានទិន្នន័យ database ដូចជា SELECT, INSERT, UPDATE ឬ DELETE ។

Command object គឺប្រើប្រាស់ដើម្បីអនុវត្តប្រភេទ operations លក្ខខ័ណ្ឌផ្សេងៗនៃប្រតិបត្តិការដូចជា SELECT, INSERT, UPDATE ឬ DELETE នៅលើមូលដ្ឋានទិន្នន័យ database ។

Command class គេប្រើវាដើម្បីតំណាភ្ជាប់រវាង front end និង back end ។ វាមាន query ដែលត្រូវ អនុវត្តពីផ្នែករវាង front end និង back end វាក៏មាន object នៃ Connection Class ផងដែរ ។
ឧទាហរណ៍:

```
1  SqlCommand cmd=new SqlCommand("Query which has to perform",
2
3  Connection Object);
```

- SELECT

```
cmd =new SqlCommand("select * from Employee", con);
```

- INSERT

```
cmd = new SqlCommand("INSERT INTO Employee(Emp_ID,Emp_Name)

        VALUES ('" + aa + "','" + bb + "')", con);
```

- UPDATE

```
SqlCommand cmd =new SqlCommand("UPDATE Employee SET
        Emp_ID ='" + aa + "', Emp_Name ='" + bb + "'
  WHERE Emp_ID = '" + aa + "'", con);
```

- DELETE

```
cmd =new SqlCommand("DELETE FROM Employee

WHERE Emp_ID='" + aa + "'", con);
```

៣.១.១.៤ Execute Command

Execute Command ជា Object មួយដែលមានតួនាទី Execute ពាល់ Command Object រួមមានដូចជា ExcuteReader , ExecuteScalar និង ExecuteNonQuery ។

Command object បង្ហាញ execute methods ជាច្រើនដូចជា៖

- ExecuteScaler()៖ Executes query និង returns ជួរឈរ ទីមួយដំបូង នៃ ជួរដេកទីមួយដំបូង ក្នុងសំណុំ សំណុំលទ្ធផល Result set return ដោយ query ។ Extra columns or rows are ignored ។
- ExecuteReader() ៖បង្ហាញ ជួរឈរទាំងអស់ និងជួរដេកទាំងអស់ ក្នុងបរិស្ថាន client-side environment ។ ពាក្យដែលយើងអាចនិយាយបានថា គឺការបង្ហាញ Data Table Client-side ។
- ExecuteNonQuery()៖ អ្វីមួយត្រូវបានធ្វើឡើងដោយមូលដ្ឋានទិន្នន័យ database ប៉ុន្តែគ្មានអ្វីត្រូវបាន returned ត្រឡប់ដោយមូលដ្ឋានទិន្នន័យ database ទេ។

ភាពខុសគ្នារវាង ExecuteScalar និង ExcuteReader

- ExecuteScalar គឺវា Execute តែ Command Object ណាដែល Return Single Value
  - ExcuteReader គឺវា Execute ពាល់ Command Object ណាដែល Return ច្រើន Column

ឈ្មោះ Name	បរិយាយ
Command_Name	គឺជាឈ្មោះរបស់អញ្ញត (អថេរ) សម្រាប់តំណាង ឲ្យ sqlCommand ។
CommandType	សម្រាប់កំណត់ប្រភេទ Command ដែលត្រូវអនុវត្តន៍
CommandType.Text	សម្រាប់បញ្ជាក់ពីការកំណត់ Command Type ។
CommandText	សម្រាប់កំណត់ពី SQL Statement ។
SQL Statement	សម្រាប់បញ្ជាក់ការកំណត់ SQL Statements ។
Connection	សម្រាប់កំណត់ការភ្ជាប់ទៅកាន់ Data Source ។
DB_Connection	សម្រាប់បញ្ជាក់ពីការកំណត់ Connection ។
ExecuteNonQuery	សម្រាប់បញ្ជាក់ពីដំណើរការ និងទទួលលទ្ធផលពី SQL Server ។

៣.១.១.៥. Data Reader Object

DataReader គឺជា Object មួយប្រភេទដែលត្រូវបានគេប្រើប្រាស់វាដើម្បីអានទិន្នន័យចេញពី Command Object បន្ទាប់ពីបាន Execute SQL Statement ដែលនៅក្នុង Command Object ។ វាទាញទិន្នន័យពី DataSource តាមរយៈការ Execute Command Object ដែល Command Object បានផ្ទុកនូវ SQL Statement ដើម្បីទាញទិន្នន័យចេញពី DataSource ។

DataReader object ប្រើប្រាស់ដើម្បីទទួល results លទ្ធផល ពី SELECT statement ដែលបានមកពី command object ។ សម្រាប់ហេតុផលដំណើរការ ទិន្នន័យដែលបានត្រឡប់ពី data reader អានទិន្នន័យ គឺជាការបញ្ជូនបន្តនៃទិន្នន័យតែប៉ុណ្ណោះ ។ នេះមានន័យថាទិន្នន័យអាចចូលប្រើបានពី stream តាមលំដាប់លំដោយ ។ នេះគឺល្អសម្រាប់ល្បឿន ប៉ុន្តែប្រសិនបើទិន្នន័យចាំបាច់ត្រូវរៀបចំ នោះសំណុំទិន្នន័យ គឺជា object ដែលល្អជាងសម្រាប់ធ្វើការជាមួយ។ DataReader: DataReader គេប្រើប្រាស់វាដើម្បី read data មកពីប្រភព source ឬ Database ។

Stream គឺជា abstraction មួយនៃ sequence នៃ bytes ។ Dot Net បានបែងចែក classes ខុសៗគ្នាដើម្បីបម្រើឲ្យខុសគ្នារបស់ប្រភេទនៃ stream រួមមាន BufferedStream FileStream MemoryStream NetworkStream CryptoStream ហើយ Stream ទាំងនេះមានតួនាទី Reading Writing និង Seeking ។

ឧទាហរណ៍៖

```
1  dr = cmd.ExecuteReader();
2  DataTable dt = new DataTable();
3  dt.Load(dr);
```

របៀបធ្វើការរបស់ DataReader object ៖

•DataReader Object មានសិទ្ធិចូល access សម្រាប់ Read-only .

•វាត្រូវបានប្រើ Connected architecture

•ផ្តល់នូវការអនុវត្តកាន់តែប្រសើរ ។

•DataReader Object Supports តែមួយ Table នៅក្នុង SQL query នៃ មួយ Database ។

•ខណៈដែល DataReader Object ត្រូវបានចងភ្ជាប់ទៅនឹងការគ្រប់គ្រងតែមួយ ។

•DataReader Object មានសិទ្ធិចូលប្រើទិន្នន័យលឿនជាងមុន

•DataReader Object ត្រូវតែសរសេរកូដដោយដៃ

•យើងមិនអាចបង្កើតទំនាក់ទំនងនៅក្នុង datareader បានទេ ។

•whereas Data reader doesn't support ។

•data reader មានទំនាក់ទំនងគ្នាជាមួយ command object ។

•DataReader មិនអាច modify data កែប្រែទិន្នន័យបានទេ ។

៣.១.១.៦. Data Adapter Object

DataAdapter ដើរតួជាអ្នកសម្របសម្រួលរវាង back end និង front end ។ ប៉ុន្តែវាមិនមានលក្ខណៈពិសេសក្នុងការផ្ទុក data ទេ ។ ដូច្នេះមាន dataset សំណុំទិន្នន័យដែលមាន data ទិន្នន័យនៃសំណុំលទ្ធផល result set។

DataAdapter គឺជា Object មួយដែលកើតចេញពី Class មួយវាមានតួនាទី Binding រវាង DataSource និង DataSet ដូចជាការទាញទិន្នន័យពី Table នៅក្នុង Database ត្រូវការ DataAdapter មួយ។ វាមានសារៈសំខាន់ចំពោះ Disconnected Invirontment ពីព្រោះវាជាអ្នក Binding ដែលមានតួនាទីត្រឹមតែ Binding ប៉ុណ្ណោះ នៅពេលដែលការទាញទិន្នន័យត្រូវបានបញ្ចប់វានឹងផ្តាច់ Connection ដោយខ្លួនឯងរហូតទាល់តែមានការបញ្ជាឲ្យភ្ជាប់ម្តងទៀតវានឹងចាប់ផ្តើមភ្ជាប់ជូនម្តងទៀត ដូចនេះយើងពុំចាំបាច់បញ្ជាឲ្យវាឈប់នោះទេវានឹងឈប់ដោយខ្លួនឯង។

• A Data Adapter represents a set of data commands and a database connection to fill the dataset and update a SQL Server database.

• Data Adapter តំណាងសំណុំ នៃ Data commands ទិន្នន័យ និងការភ្ជាប់ database connection ដើម្បី Fill dataset និងការធ្វើបច្ចុប្បន្នភាព SQL Server database ។

• A Data Adapter contains a set of data commands and a database connection to fill the dataset and update a SQL Server database. Data Adapters form the bridge between a data source and a dataset.

• Data Adapters are designed depending on the specific data source. The following table shows the Data Adapter classes with their data source.

Provider-Specific Data Adapter classes	Data Source
SqlDataAdapter	SQL Server
OledbDataAdapter	OLE DB provider
OdbcDataAdapter	ODBC driver
OracleDataAdapter	Oracle

៣.១.១.៦.១ SqlDataAdapter Update Method

1. SqlDataAdapter គឺជាផ្នែកមួយនៃក្រុមហ៊ុនផ្តល់ទិន្នន័យ ADO.NET
2. វាប្រើវិធី Fill Method ដើម្បីទាញយកទិន្នន័យពីប្រភពទិន្នន័យ data sources ហើយ fill វាក្នុង DataSet ។
3. UpdateCommand នៃ SqlDataAdapter Object ធ្វើ updates បច្ចុប្បន្នភាពមូលដ្ឋានទិន្នន័យ database ជាមួយនឹងការកែប្រែទិន្នន័យ data modifications ដែលបានធ្វើឡើងនៅលើ DataSet object ។

Diagram

Steps

1.Create an update query string.
2.Create a connection object.
3.Create a SqlDataAdapter object accompanying the query string and connection object.
4.Use the Update command of the SqlDataAdapter object to execute the update query.

ឧទាហរណ៍ទី១៖

```
1 private void button1_Click(object sender, EventArgs e)
2 {
3     SqlConnection con = new SqlConnection("connetionString");
4     string qry = "SELECT * FROM SOMETABLE";
5     SqlDataAdapter da = new SqlDataAdapter(qry,con);
```

```
6       //Fill the DataSet
7       DataSet ds = new DataSet();
8       da.Fill(ds, "SomeTable");
9       //Update a row in DataSet Table
10      DataTable dt = ds.Tables["SomeTable"];
11      dt.Rows[0]["SomeColumn"] = "xyz";
12
13      string sql = "update sometable set somecolumn = 10 where ...";
14      SqlDataAdapter adapter = new SqlDataAdapter();
15      try
16          {
17           connection.Open();
18           SqlCommand cmd = new SqlCommand(sql, con);
19           //select the update command
20           adapter.UpdateCommand=cmd;
21           //update the data source
22           adapter.Update(ds,"SomeTable");
23           MessageBox.Show ("DataBase updated !! ");
24      }
25      catch (Exception ex)
26      {
27          connection.Close();
28      }
29  }
```

៣.១.១.៧ Data Source

បញ្ជីលំដាប់នៃការ ទាញទិន្នន័យពី DataSource មកក្នុង DataSet

- String Connection
- String Sql Statement
- DataAdapter Object
- DataSet Object
- Fill DataAdapter ទីក្នុង DataSet ដោយបញ្ជាក់ឈ្មោះ Table

contains ស្វែងរកទិន្នន័យក្នុងដែលស្ថិតក្នុងអ្វីមួយ ។

ដំណើរការ Data Source មានដូចជា៖

- Data Adapter object ចូលប្រើប្រាស់ដើម្បីទិន្នន័យ នៅក្នុង disconnected mode ។ វាជា object contains reference មួយដើម្បី connection object ។
- វាត្រូវបានបង្កើតឡើងនៅក្នុងវិធីមួយដោយប្រយោល សម្រាប់ opens និង closes the connection នៅពេលណាដែលត្រូវការ ។
- វាត្រូវបានរក្សាទិន្នន័យនៅក្នុង Dataset Object ។ អ្នកប្រើប្រាស់អាចអានទិន្នន័យប្រសិនបើចាំបាច់ពី dataset read data និង write back សរសេរការផ្លាស់ប្តូរក្នុងក្រុមតែមួយត្រឡប់ទៅមូលដ្ឋានទិន្នន័យ។ នៅពេលដែលធ្វើការបន្ថែម data dataAdapter contains command

object referenced សម្រាប់ SELECT, INSERT, UPDATE, and DELETE ដែលប្រតិបត្តិការណ៍ Data object និង data source ។

- Data Adapter supports គាំទ្រជាចម្បងនូវវិធីសាស្ត្រពីរ two methods ខាងក្រោម៖

• Fill ()

. Fill method បំពេញបំពេញសំណុំទិន្នន័យ dataset ឬ data table object វត្ថុតារាងទិន្នន័យដែលមានទិន្នន័យពីមូលដ្ឋានទិន្នន័យ database ។ វាទាញយកជួរដេក row ពី data source ប្រភពទិន្នន័យដោយប្រើ statement SELECT ដែលបានកំណត់ដោយពាក្យបញ្ជ select command property ។ Fill method ទុកការតភ្ជាប់ក្នុងស្ថានភាពដូចដែលវាជួបនឹងវាមុនពេលបញ្ចូលទិន្នន័យ។ ប្រសិនបើការ Call Method ជាបន្តបន្ទាប់ទិន្នន័យឡើងវិញ គឺត្រូវបានទាមទារឱ្យមានព័ត៌មានPrimary key ជា ចម្បង ។

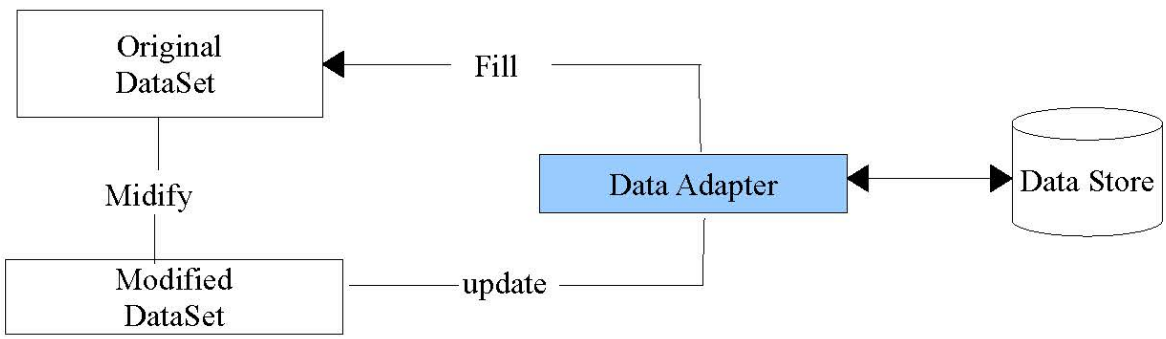
• Update ()

- Update method ផ្ទេរការផ្លាស់ប្តូរទៅមូលដ្ឋានទិន្នន័យ ។វាត្រូវបានវិភាគ Row State នៃ record នៅក្នុង DataSet និង Call the appropriate INSERT, UPDATE, and DELETE statements ។ Data Adapter object គឺត្រូវបានបង្កើតឡើងរវាង disconnected ADO.NET object និង data source ។

ឧទាហរណ៍ទី១៖

```
1  SqlDataAdapter da=new SqlDataAdapter("Select * from Employee",
2  con);
3  da.Fill(ds,"Emp");
4  bldr =new SqlCommandBuilder(da);
5  dataGridView1.DataSource = ds.Tables["Emp"];
```

ដ្យាក្រាម៖



Retrieval & update data using dataAdaper

៣.១.១.៨ DataSet Object

DataSet គឺជាបណ្តុំនៃទិន្នន័យ ហើយវាក៏ជា Virtual Database ផងដែរ សម្រាប់ផ្ទុកទិន្នន័យពីការទាញទិន្នន័យរបស់ DataAdapter ជាច្រើនមករក្សាទុកក្នុងវា ដូចនេះក្នុង DataSet មួយមានច្រើន Table ដែល Table នីមួយៗបានទាញយកមកតាមរយៈ DataAdapter ។ ហើយវាក៏អាច Update Data ទៅកាន់ Database វិញផងដែរតាមរយៈ DataAdapter ទាំងនោះហើយដែលមួយ DataAdapter បាន Binding ជាមួយ Table មួយនៅក្នុង Database Server ។

នៅក្នុង Disconnected ទិន្នន័យដែលបានទាញយកពីមូលដ្ឋានទិន្នន័យត្រូវបានរក្សាទុកក្នុង local បណ្តោះអាសន្នមូលដ្ឋានដែលហៅថា DataSet។ វាត្រូវបានបង្កើតឡើងដើម្បីដំណើរចូលប្រើទិន្នន័យ access data ពី Data source ណាមួយ។ Class ត្រូវបានកំណត់នៅក្នុង System.Data namespace

- Data Set object គឺជា Data Set object ទិន្នន័យគឺជាតំណាងនៅក្នុងអង្គចងចាំ in-memory នៃទិន្នន័យ។ វាត្រូវបានរចនាឡើងជាពិសេសដើម្បីគ្រប់គ្រងទិន្នន័យនៅក្នុងអង្គចងចាំ និងដើម្បីគាំទ្រដល់ប្រតិបត្តិការដែល disconnected ផ្តាច់លើទិន្នន័យ data។
- DataSet គឺជាCollection នៃ DataTable និង DataRelations ។ DataTable គឺជា Collection នៃ DataColumn DataRows និង Constraints ។
- DataTable DataColumn និង DataRows អាចបង្កើតបានដូចខាងក្រោម៖
  - ផ្តល់នូវប្រសិទ្ធភាពទាប។
  - វាត្រូវបានប្រើនៅក្នុង disconnected architecture
  - DataSet Object អាចដំណើរការចូលប្រើ read/write access
  - ataSet Object Support Table ច្រើន ពី Database ផ្សេង
  - DataSet object គឺភ្ជាប់ទៅនឹងការគ្រប់គ្រងច្រើន bound to multiple controls
  - DataSet object ធ្វើយឺតក្នុងចូលប្រើ ទិន្នន័យ
  - DataSet Object គឺត្រូវ Support ដោយ Visual Studio tools
  - យើងអាចបង្កើត Relation នៅក្នុង DataSet
  - DataSet Support បញ្ចូលគ្នាជាមួយ XML
  - DataSet មានទំនាក់ទំនងគ្នាជាមួយ Data Adapter
  - DataSet អាច Modify data កែប្រែទិន្នន័យបាន ។

ឧទាហរណ៍ទី១៖

```
1   DataTable dt = new DataTable();
2   DataColumn col =new DataColumn();
3   Dt.columns.Add(col2);
4   DataRow row = dt.newRow();
```

ដ្យាក្រាម៖

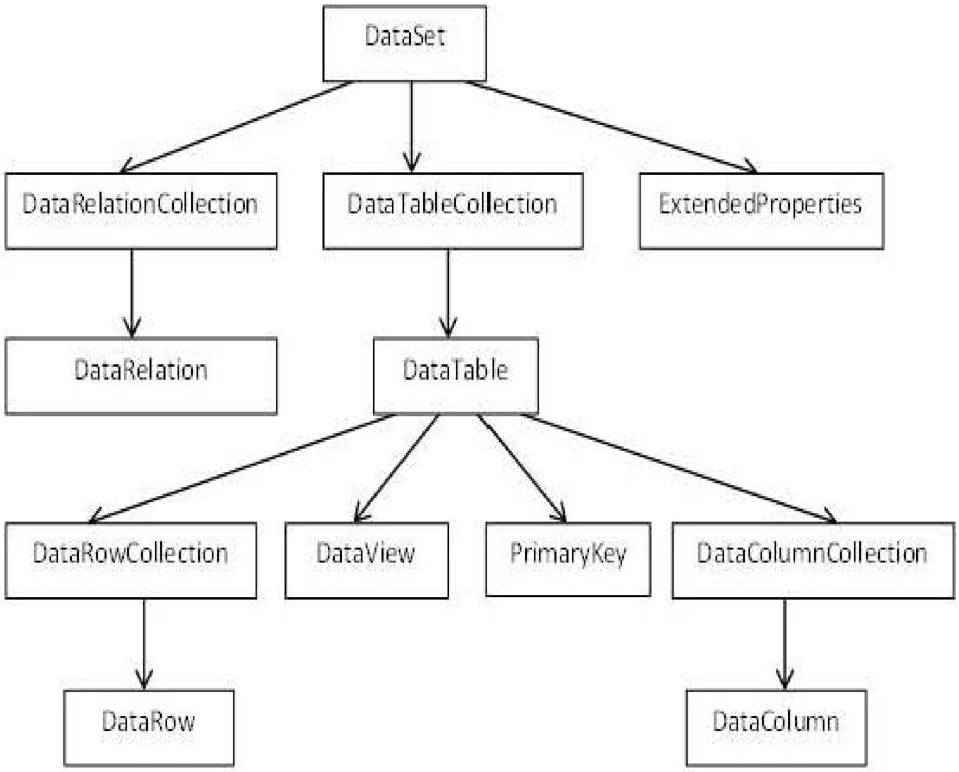


Figure: *Process of DataSet*

៣.១.១.៩ សិក្សាអំពី Command Builder Object

•បង្កើតការបញ្ចូល ធ្វើបច្ចុប្បន្នភាព លុបសំណួរដោយស្វ័យប្រវត្តិ ដោយប្រើលក្ខណៈសម្បត្តិ SelectCommand នៃ DataAdapter ។

•ការបង្កើតស្វ័យប្រវត្តិ សម្រាប់កែកែទិន្នន័យ ដូចជា insert, update, delete queries ប្រើប្រាស់ SelectCommand properties នៃ DataAdapter ។

•Command Builder Object គឺប្រើប្រាស់ដើម្បី buil command សម្រាប់ Modification ពី Object ដែលផ្អែកលើ query table ណាមួយ ។ CommandBuilders គឺបង្កើតឡើងអាស្រ័យដោយការកំណត់ data source ជាក់លាក់។ខាងក្រោមនេះបង្ហាញពី តារាង CommandBuilder class ជាមួយ Data Source ។

Provider-Specific Data Adapter classes	Data Source
SqlDataAdapter	**SQL Server**
OledbDataAdapter	**OLE DB provider**
OdbcDataAdapter	**ODBC driver**
OracleDataAdapter	**Oracle**

ឧទាហរណ៍៖

```
da = new SqlDataAdapter("Select * from Employee", con);
ds = new DataSet();
```

```
3  da.MissingSchemaAction = MissingSchemaAction.AddWithKey;
4  da.Fill(ds, "Emp");
5  bldr = new SqlCommandBuilder(da);
6  dataGridView1.DataSource = ds.Tables["Emp"];
```

៣.១.១.១០ តារាងខុសគ្នារវាង DataReader and DataSet

No	Data Reader	DataSet
1	Used in a connected architecture	used in a disconnected architecture
2	Provides better performance	Provides lower performance
3	DataReader object has read-only access	A DataSet object has read/write access
4	DataReader object supports a single table based on a single SQL query of one database	A DataSet object supports multiple tables from various databases
5	A DataReader object is bound to a single control	A DataSet object is bound to multiple controls
6	A DataReader object has faster access to data	A DataSet object has slower access to data
7	A DataReader object must be manually coded	A DataSet object is supported by Visual Studio tools
8	We can't create a relation in a data reader	We can create relations in a dataset
9	Whereas a DataReader doesn't support data reader communicates with the command object.	A Dataset supports integration with XML Dataset communicates with the Data Adapter only
10	DataReader cannot modify data	A DataSet can modify data

៣.១.១.១១ DataView Object

- DataView គឺដូចគ្នាទៅនឹងសំណុំទិន្នន័យខ្នាតតូចដែលបានតែអាន read-only mini-dataset
- ជាធម្មតាអ្នកផ្ទុកតែសំណុំរងទៅក្នុង DataView ប៉ុណ្ណោះ។
- DataViews ផ្តល់ដោយ Danamic view នៃ data ។ វាត្រូវបានផ្តល់ dataRows ដោយប្រើប្រាស់ DataView ។

៣.២ Disconnected Architecture

Disconnected គឺជាលក្ខណៈសំខាន់ពិសេសចម្បងនៃ .NET framework។ ADO.NET មាន classes ផ្សេងៗដែលគាំទ្រ support architecture នេះ។ .NET application វាមិនធ្វើការភ្ជាប់ជាមួយ Database រហូតទេ ។ Class ត្រូវបានបង្កើតឡើងសម្រាប់ជាមធ្យោបាយភ្ជាប់ Open និង Close connection ដោយស្វ័យប្រវត្តិ ។ data ទិន្នន័យត្រូវបានរក្សាទុកនៅ Client-side ហើយវាត្រូវ បានធ្វើបច្ចុប្បន្នភាព updated នៅក្នុង Databases មូលដ្ឋានទិន្នន័យនៅពេលណាដែលត្រូវការ required ។

ADO.NET Disconnected architecture considers មានប្រភេទ Objects មួយចំនួនដូចជា៖

- DataSet ds;
- SqlDataAdapter da;
- SqlConnection con;
- SqlCommandBuilder bldr;

៣.២.១ ការប្រើប្រាស់ SqlConnection

Connection object គឺប្រើប្រាស់ដើម្បីបង្កើត connection Database ហើយនឹង connection វាដោយខ្លួនឯងនឹងមិនផ្ទេរទិន្នន័យណាមួយឡើយ ។

៣.២.២ ការប្រើប្រាស់ SqlDataAdapter

DataAdapter គឺប្រើប្រាស់បញ្ជូនទិន្នន័យ រវាង database និង Dataset ។ វាមាន commands like select , insert, update និង delete ។

៣.២.៣ ការប្រើប្រាស់ SqlCommand

Select command គឺប្រើដើម្បីទាញយកទិន្នន័យពី Database និង insert, update, delete command គឺប្រើដើម្បីបញ្ជូនផ្លាស់ប្តូរទិន្នន័យនៅក្នុង Dataset ទៅ Database (to send changes to the data in dataset to database)។ វាត្រូវការ Connection ដើម្បីបញ្ជូនទិន្នន័យ ។

៣.២.៤ ការប្រើប្រាស់ SqlCommandBuilder

CommandBuilder : CommandBuilder គឺជាលំនាំដើម default dataAdapter contains មានតែពាក្យបញ្ជ select command ហើយវាមិនមាន contain insert, update and delete commands ទេ។ ដើម្បីបង្កើត insert, update and delete commands ពាក្យបញ្ជសម្រាប់ dataadapter, commandbuilder ដែលត្រូវប្រើប្រាស់ command ។ វាត្រូវបានប្រើសម្រាប់តែបង្កើតពាក្យបញ្ជទាំងនេះសម្រាប់ dataadapter ហើយគ្មានគោលបំណង ឬសំណើរផ្សេងទៀតទេ។

៣.២.៥ ការប្រើប្រាស់ SqlDataSet

DataSet: DataSet គឺប្រើប្រាស់សម្រាប់ Store ទិន្នន័យពី Database ដោយ dataAdapter និងបង្កើតវាតែសម្រាប់ .NET Application ។

Dataset: DataSet contains មានដូចជា table និង relation ។

DataSet គឺជាបណ្តុំនៃទិន្នន័យ ហើយវាក៏ជា Virtual Database ផងដែរ សម្រាប់ផ្ទុកទិន្នន័យពីការទាញទិន្នន័យរបស់ DataAdapter ជាច្រើនមករក្សាទុកក្នុងវា ដូចនេះក្នុង DataSet មួយមានច្រើន Table ដែល Table នីមួយៗបានទាញយកមកតាមរយៈ DataAdapter ។

ហើយវាក៏អាច Update Data ទៅកាន់ Database វិញផងដែរតាមរយៈ DataAdapter ទាំងនោះហើយដែលមួយ DataAdapter បាន Binding ជាមួយ Table មួយនៅក្នុង Database Server ។

Fill data នៅក្នុង dataset fill() method គឺប្រើប្រាស់សម្រាប់ DataAdapter ហើយមានរូបមន្តដូចខាងក្រោម៖

```
1 Da.Fill(Ds,"TableName");
```

ឧទាហរណ៍ ៖

```
1 private void BtSave_Categ_Click(object sender, EventArgs e)
2 {
```

```
    db_Connection cnn = new db_Connection();
    cnn.Conntion_db();
    SqlCommand comm = new SqlCommand();
    comm.Connection = db_Connection.conn;
    comm = new SqlCommand(@"SELECT COUNT(*) FROM Categories
      WHERE Category_Name=@Category_Name",db_Connection.conn);

    comm.Parameters.AddWithValue("@Category_Name",
                                 TxtCategory_Name.Text);

        comm.ExecuteNonQuery();
        SqlDataAdapter DATA = new SqlDataAdapter(comm);
        DataTable dts = new DataTable();
        DATA.Fill(dts);
    }

```

៤. ការប្រើប្រាស់ dataAdapter.Fill Method

និយមន័យ៖ Adds ឬ refreshes rows នៅក្នុង DataSet ដើម្បីផ្គូផ្គងវាត្រូវគ្នានៅក្នុង data source។

```
namespace: System.Data.Common
assembly: System.Data.Common.dll
```

Overloads

Fill(DataSet)	Adds or refreshes rows in the DataSet to match those in the data source.
Fill(DataTable, IDataReader)	Adds or refreshes rows in the DataTable to match those in the data source using the DataTable name and the specified IDataReader.
Fill(DataTable[], IDataReader, Int32, Int32)	Adds or refreshes rows in a specified range in the collection of DataTable objects to match those in the data source.
Fill(DataSet, String, IDataReader, Int32, Int32)	Adds or refreshes rows in a specified range in the DataSet to match those in the data source using the DataSet and DataTable names.

៤.១ Fill(DataSet)

Fill(DataSet) គឺជាការ Adds or refreshes rows នៅក្នុង DataSet ដើម្បីផ្គូផ្គងវាត្រូវគ្នានៅក្នុង data source.

```
public virtual int Fill (System.Data.DataSet dataSet);
```

៤.១.១ Parameters dataSet

Dataset ដើម្បី fill ជាមួយ records ហើយប្រសិនបើចាំបាច់ schema

Returns

int 32

ចំនួនជួរដេកត្រូវបានបន្ថែមដោយជោគជ័យទៅក្នុងសំណុំទិន្នន័យ Dataset ។ នេះមិនរាប់បញ្ចូលជួរដេកដែលប៉ះពាល់ដោយ statements ដែលមិនត្រឡប់ជួរដេក return rows ទេ ។

Implements

Fill(DataSet)

៤.២ Fill(DataTable, IDataReader)

Adds ឬ refreshes row នៅក្នុង DataTable ដើម្បីធ្វើការជ្រើសរើសទិន្នន័យដូចគ្នា នៅក្នុង DataSource ដោយប្រើប្រាស់ឈ្មោះ DataTable ហើយនិងកំណត់ដោយ IdataReader ។

```
protected virtual int Fill (System.Data.DataTable dataTable,
System.Data.IDataReader dataReader);
```

Parameters

dataTable : DataTable

DataTable ដើម្បី fill ជាមួយ records.

dataReader

dataReader: IDataReader

instance of IDataReader.

Returns Int32

ចំនួនជួរដេកដែលបានបន្ថែមដោយជោគជ័យទៅក្នុងតារាងទិន្នន័យ DataTable ។ នេះមិនរាប់បញ្ចូលជួរដេកដែលប៉ះពាល់ដោយ statement ដែលមិនត្រឡប់ជួរដេកទេ។

សម្គាល់ ៖សម្រាប់ DataAdapter.Fill(DataSet) សម្រាប់ additional information ។

៤.៣ Fill(DataTable[], IDataReader, Int32, Int32)

Adds or refreshes rows in a specified range in the collection of DataTable objects to match those in the data source.

```
protected virtual int Fill (System.Data.DataTable[] dataTables,
System.Data.IDataReader dataReader, int startRecord, int
maxRecords);
```

Parameters

dataTables DataTable[]

A collection of DataTable objects to fill with records ។

dataReader IDataReader

An instance of IDataReader

startRecord Int32

The zero-based index of the starting record

maxRecords Int32

An integer indicating the maximum number of records.

Returns Int32

The number of rows successfully added to or refreshed in the DataTable. This does not include rows affected by statements that do not return rows.

Remarks

See the remarks for System.Data.Common.DataAdapter.Fill(System.Data.DataSet) for additional information.

៤.៤ Fill (DataSet, String, IDataReader, Int32, Int32)

Adds or refreshes rows in a specified range in the DataSet to match those in the data source using the DataSet and DataTable names.

```
protected virtual int Fill (System.Data.DataSet dataSet,
string srcTable, System.Data.IDataReader dataReader, int
startRecord, int maxRecords)
```

Parameters

dataSet DataSet

A DataSet to fill with records.

srcTable String

A string indicating the name of the source table.

dataReader IDataReader

An instance of IDataReader.

startRecord Int32

The zero-based index of the starting record.

maxRecords Int32

An integer indicating the maximum number of records.

Returns Int32

The number of rows successfully added to or refreshed in the DataSet. This does not include rows affected by statements that do not return rows.

Remarks

System.Data.Common.DataAdapter.Fill(System.Data.DataSet) for additional information ។

៥. គុណសម្បត្តិរបស់ Connected និង Disconnected Invirontment

+គុណសម្បត្តិរបស់ Connected Invirontment ៖

- ងាយស្រួលប្រើប្រាស់និងសុវត្ថិភាពខ្ពស់
- ងាយស្រួលក្នុងការគ្រប់គ្រងទិន្នន័យ
- ទិន្នន័យបញ្ជូនមកថ្មីៗជានិច្ច

+គុណវិប្បត្តិ

- ត្រូវភ្ជាប់ទៅកាន់ Database ជានិច្ច
- មិនអាចធ្វើ Navigative ដោយផ្ទាល់បាន ដូចជា Next Records, Last Record , First Record និង Previous Records

+ គុណសម្បត្តិរបស់ Disconnected Invirontment

-អាចធ្វើការគ្រប់ពេលនៅលើ Local អាចភ្ជាប់ទៅកាន់ Database Server ខណៈ ពេលដែលត្រូវការ

- មិនធ្វើឲ្យ Database Server មានការវល់ច្រើន
- អាចធ្វើ Navigative ដោយផ្ទាល់បាន ដូចជា Next Records, Last Record , First Record និង Previous Records
- ទាញទិន្នន័យយកមកប្រើតែម្តង ទោះ Database Server ត្រូវបានកាត់ផ្តាច់ក៏មិនមានបញ្ហា ហើយអាច Synchronize បានគ្រប់ពេលដែលមានតំណភ្ជាប់ឡើងវិញ

+គុណវិប្បត្តិ

- ទិន្នន័យមិនអាចថ្មីជានិច្ចបានឡើយ
- មិនមានសុវត្ថិភាពទិន្នន័យ

ជំពូកទី១១

ការបង្កើត Projects

ក្នុងជំពូកនេះយើងនឹងសិក្សាអំពីរបៀបការបង្កើត Projects សម្រាប់ប្រព័ន្ធគ្រប់គ្រងមួយចំនួនដូចជា គ្រប់គ្រង ហាងទំនិញ បុគ្គលិក សិស្ស និស្សិត ក្រុមហ៊ុន ស្តុបន អង្គការ និងការ៉ាស ជាដើម ។ យើងនឹងសិក្សាពី ការបង្កើតកម្មវិធីតូចៗមួយចំនួនសម្រាប់អនុវត្តន៍ និងជាឧទាហរណ៍ ជាគន្លឹះសម្រាប់ប្អូន ក្នុងការសិក្សាសម្រាប់ ភាសា C# ។

១. របៀបបង្កើត Class Object Connection Database system

- db_Connection.cs

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

namespace POS_Systems
{
    class db_Connection
    {
        public static SqlConnection conn = null;
        public static string ServerName = Class1.ServerName;
        //public static string ServerName = "192.168.1.116";
        public static string UserName = Class1.UserName;
        public static string Password = Class1.Password;
        public static string DBMS = Database.DataDMS;
        public void Conntion_db()
        {
            conn = new SqlConnection(@"Data Source=" +
                  ServerName + ";Initial Catalog=" + DBMS + ";
               User ID=" + UserName +";Password=" + Password);
             conn.Open();
        }


    }
}
```

២. ការបង្កើត Form Login

២.១ ការបង្កើត Form Login

Tool Control	Name
Texbox	txtUserName
	txtPassword
Button	btLogin
PictureBox	Photo_Item
	Pic_Closes

២.២ ការបង្កើត Table: tbusers

DESKTOP-QUCR3V0....POS - dbo.tbusers

Column Name	Data Type	Allow Nulls
userId	int	☐
Username	nvarchar(100)	☑
Password	nvarchar(10)	☑
Usertype	nvarchar(100)	☑
Phone	nvarchar(50)	☑
Email	nvarchar(100)	☑
PlaceOfbirth	nvarchar(250)	☑
PlaceOfPressent	nvarchar(250)	☑
Photo	image	☑
Active_user	int	☑

២.៣ ការសរសេរកូដ ក្នុង Form Login

- ប៊ូតុង Login: **LOGIN**

```
1  private void btLogin_Click(object sender, EventArgs e)
2  {
3
4      db_Connection cnn = new db_Connection();
5      cnn.Conntion_db();
6
7      SqlCommand cmd = new SqlCommand();
8      cmd.Connection = db_Connection.conn;
9      cmd = new SqlCommand(@"SELECT * FROM tbusers WHERE Username=
10           N'" + txtUserName.Text + "' AND Password=N'" +
11           txtPassword.Text + "'", db_Connection.conn);
12
13     //cmd.Parameters.AddWithValue("@Active", Active);
14     cmd.ExecuteNonQuery();
15
16     SqlDataReader dr1;
17     dr1 = cmd.ExecuteReader();
18     if (dr1.Read())
19     {
20        userId = (String)dr1["userId"].ToString();
21        Username = (String)dr1["Username"].ToString();
22        USE_mail = (String)dr1["Email"].ToString();
23        UserPermission = (String)dr1["Usertype"].ToString();
24        Phone_User = (String)dr1["Phone"].ToString();
25        PlaceOfbirth = (String)dr1["PlaceOfbirth"].ToString();
26        PlaceOfPressent = (String)dr1["PlaceOfPressent"].ToString();
27
28        userId = userId != "" ? userId : " ";
29        Username = Username != "" ? Username : " ";
30        USE_mail = USE_mail != "" ? USE_mail : " ";
31        UserPermission = UserPermission != "" ? UserPermission : " ";
32        Phone_User = Phone_User != "" ? Phone_User : " ";
33        PlaceOfbirth = PlaceOfbirth != "" ? PlaceOfbirth : " ";
34     PlaceOfPressent = PlaceOfPressent != "" ? PlaceOfPressent : " ";
35        Password = Password != "" ? Password : " ";
36
37     }
38     db_Connection.conn.Close();
39
40     // --end code get user userID, username and Email----;
41
42     db_Connection.conn.Open();
43     SqlDataAdapter da = new SqlDataAdapter(@"SELECT COUNT(*) FROM
44                  tbusers WHERE Active_user=1 AND Username=N'" +
45                        txtUserName.Text + "' AND Password=N'" +
46                        txtPassword.Text + "' ", db_Connection.conn);
47
48     // comm.CommandType = CommandType.Text;
49     DataTable dt = new DataTable();
50     da.Fill(dt);
```

```
51
52      db_Connection.conn.Close();
53
54
55      if (txtUserName.Text == "")
56      {
57          MessageBox.Show("Please input Username", "Information",
58                MessageBoxButtons.OK, MessageBoxIcon.Information);
59          txtUserName.Focus();
60          return;
61      }
62      else if (txtPassword.Text == "")
63      {
64          MessageBox.Show("Please input Password", "Information",
65                 MessageBoxButtons.OK, MessageBoxIcon.Information);
66          txtPassword.Focus();
67          return;
68      }
69      else if (dt.Rows[0][0].ToString() == "1")
70      {
71          //-------get Id to pass to other form
72          //  USE_mail = txtuser_Email_lableuesr.Text;
73          Username = txtUserName.Text;
74          lbtxtuser_id.Text= userId;
75
76          // Pic_user_Log.Image =byte[] Photos.ToString();
77
78          //---------//---------------end code ------------------
79          this.Hide();
80          var form2 = new F_Homes();
81          form2.Closed += (s, args) => this.Close();
82          form2.Show();
83
84      }
85      else
86      {
87          MessageBox.Show("Username Or Password Incorrect !",
88                          "Information", MessageBoxButtons.OK,
89                           MessageBoxIcon.Information);
90          txtPassword.Clear();
91          txtUserName.Clear();
92
93      }
94  }
```

- Button: Close ☒

```
1  private void Pic_Closes_Click(object sender, EventArgs e)
2      {
3          Close();
4      }
```

- កូដទាំងអស់ Form_Login

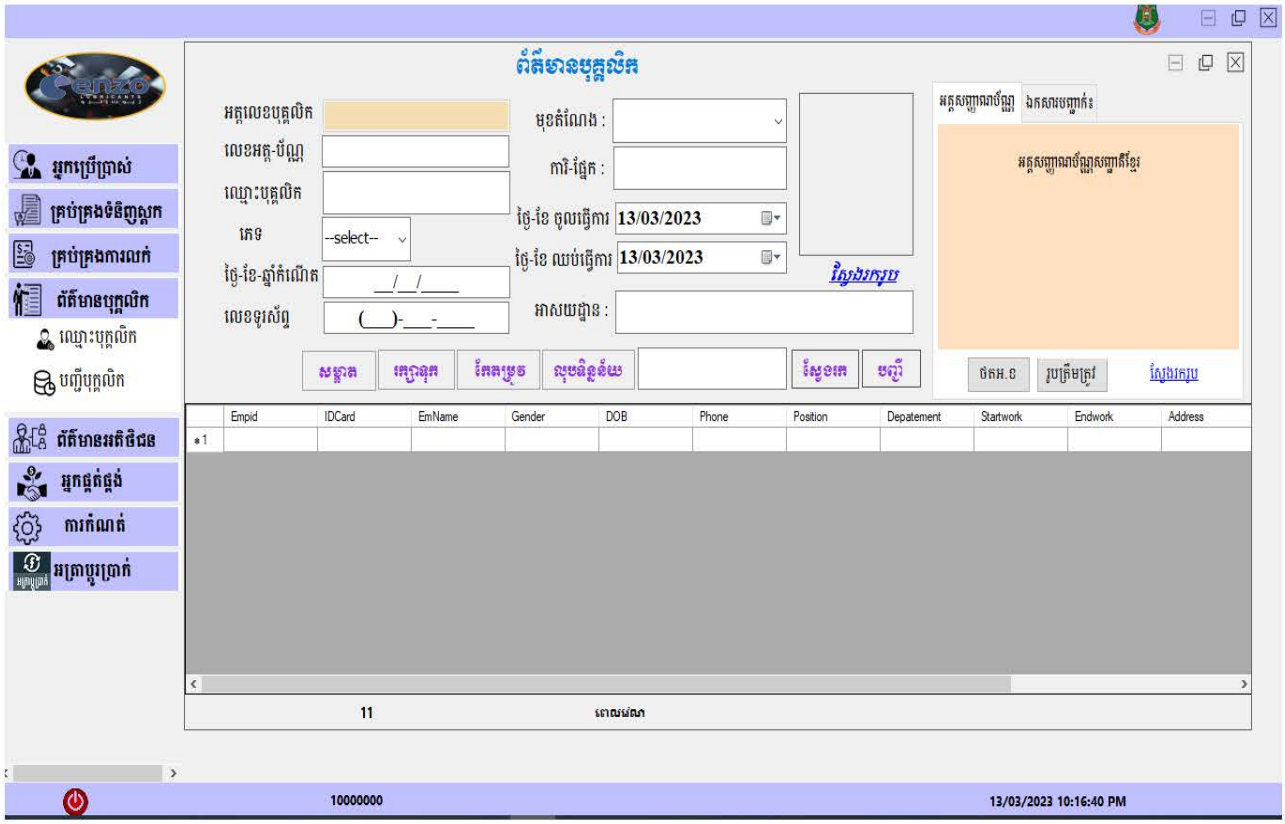
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;
using System.IO;
using System.Drawing.Imaging;
using System.Runtime.Serialization.Formatters.Binary;
using Microsoft.Win32;
using POS_con;

namespace POS_Systems
{
public partial class F_LogIn : Form
{
    public F_LogIn()
    {
        InitializeComponent();
    }

    public static string userId;
    public static string Username;
    public static string USE_mail;
    public static string UserPermission;
    public static string Password;

    public static string Phone_User;
    public static string PlaceOfbirth;
    public static byte [] Photo;
    public static string PlaceOfPressent;

    ////
private void btLogin_Click(object sender, EventArgs e)
{
    db_Connection cnn = new db_Connection();
    cnn.Conntion_db();
    SqlCommand cmd = new SqlCommand();
    cmd.Connection = db_Connection.conn;
    cmd = new SqlCommand(@"SELECT * FROM tbusers WHERE Username=N'" +
          txtUserName.Text + "' AND Password=N'" + txtPassword.Text + "'",
          db_Connection.conn);

    cmd.ExecuteNonQuery();
    SqlDataReader dr1;
    dr1 = cmd.ExecuteReader();
    if (dr1.Read())
    {
        userId = (String)dr1["userId"].ToString();
        Username = (String)dr1["Username"].ToString();
        USE_mail = (String)dr1["Email"].ToString();
```

```
54          UserPermission = (String)dr1["Usertype"].ToString();
55          Phone_User = (String)dr1["Phone"].ToString();
56          PlaceOfbirth = (String)dr1["PlaceOfbirth"].ToString();
57          PlaceOfPressent = (String)dr1["PlaceOfPressent"].ToString();
58
59          userId = userId != "" ? userId : " ";
60          Username = Username != "" ? Username : " ";
61          USE_mail = USE_mail != "" ? USE_mail : " ";
62          UserPermission = UserPermission != "" ? UserPermission : " ";
63          Phone_User = Phone_User != "" ? Phone_User : " ";
64          PlaceOfbirth = PlaceOfbirth != "" ? PlaceOfbirth : " ";
65          PlaceOfPressent = PlaceOfPressent != "" ? PlaceOfPressent : " ";
66          Password = Password != "" ? Password : " ";
67      }
68      db_Connection.conn.Close();
69
70      // -------end code get user userID, username and Email-----------;
71
72      db_Connection.conn.Open();
73      SqlDataAdapter da = new SqlDataAdapter(@"SELECT COUNT(*)
74                  FROM tbusers WHERE Active_user=1 AND Username=N'" +
75                  txtUserName.Text + "' AND Password=N'" + txtPassword.Text
76                  + "' ", db_Connection.conn);
77
78      // comm.CommandType = CommandType.Text;
79      DataTable dt = new DataTable();
80      da.Fill(dt);
81
82      db_Connection.conn.Close();
83
84      if (txtUserName.Text == "")
85      {
86          MessageBox.Show("Please input Username", "Information", Message
87                           BoxButtons.OK, MessageBoxIcon.Information);
88          txtUserName.Focus();
89          return;
90      }
91      else if (txtPassword.Text == "")
92      {
93          MessageBox.Show("Please input Password", "Information", MessageBox
94                          Buttons.OK, MessageBoxIcon.Information);
95          txtPassword.Focus();
96          return;
97      }
98      else if (dt.Rows[0][0].ToString() == "1")
99      {
100         //-------get Id to pass to other form
101         Username = txtUserName.Text;
102         lbtxtuser_id.Text= userId;
103         //---------//---------------end code ------------------
104         this.Hide();
105         var form2 = new F_Homes();
106         form2.Closed += (s, args) => this.Close();
107         form2.Show();
108     }
109     else
110     {
```

```
109
110
111         MessageBox.Show("Username Or Password Incorrect !", "Information",
112                         MessageBoxButtons.OK, MessageBoxIcon.Information);
113         txtPassword.Clear();
114         txtUserName.Clear();
116     }
117 }
118
119  //-------code Form_load-------------
120     private void F_LogIn_Load(object sender, EventArgs e)
121     {
122         lbtxtuser_id.Hide();
123         this.StartPosition = FormStartPosition.Manual;
124         this.Location = new Point(500, 130);
125         this.KeyPreview = true;
126     }
127 //-------code Pic_Closes-------------
128     private void Pic_Closes_Click(object sender, EventArgs e)
129     {
130         Close();
131     }
132
133  }
134 }
13
```

៣. Form Dashboard

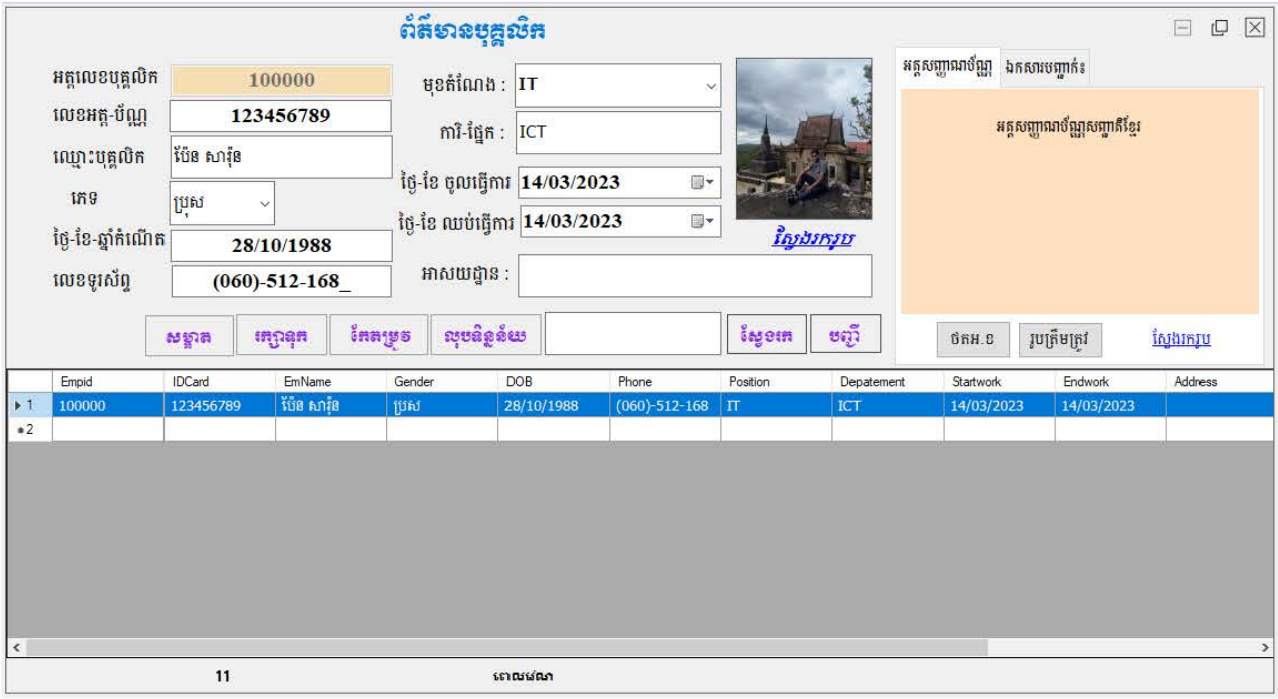
៤. អនុវត្តន៍ ៖



៤.១ Desing Table: tbEmployees

Column Name	Data Type	Allow Nulls
Empid	int	☐
IDCard	nvarchar(50)	☑
EmName	nvarchar(50)	☑
Gender	nvarchar(50)	☑
DOB	date	☑
Phone	nvarchar(50)	☑
Position	nvarchar(50)	☑
Depatement	nvarchar(50)	☑
Startwork	date	☑
Endwork	date	☑
Address	nvarchar(MAX)	☑
Photo	image	☑
ImagIDCard	image	☑

៤.២ Desing Form Employee in C#



៤.៣ Tool Controlled

Tool Control	Name
Textbox	txtId_Emp txtIDCard_Emp txtEmptNames txtIDCard_Emp txtId_Emp txtIDCard_Emp txtAdress
Button	btAdd_Emp btupdate_Emp btdelete_Emp btseach_pro tbList
Combobox	txtGender
DateTimePiker	txtdatejoin_Emp txtdateEnd_Emp
MaskedTextBox	textDob txtPhone
Picturebox	Txtpic_photo_Emp
LinkLabel	Txtseachpic
Label	Titile All Tool box
Tabcontrol	tabcontral_cust

Datalistview	datadataGridView1

៤.៤ Codding សម្រាប់ Form Employee

Forn_load( )

```
1  private void F_Employes_Load(object sender, EventArgs e)
2  {
3     classEmployeeBindingSource.DataSource= new List<Class_Employee>();
4      Despay_Data();
5  }
```

ប៊ូតុង៖ btAdd_Emp_Click

```
1  private void btAdd_Emp_Click(object sender, EventArgs e)
2
3 {
4  if (txtIDCard_Emp.Text == "")
5  {
6
7     MessageBox.Show("Pls input IDcard", "ព័ត៌មាន",
8                     MessageBoxButtons.OKCancel);
9      txtIDCard_Emp.Focus();
10    return;
11
12   }
13   else
14   {
15       //Class_con_data.conn.Open();
16
17       db_Connection connn = new db_Connection();
18       connn.Conntion_db();
19       SqlCommand cmd = new SqlCommand();
20       cmd.Connection = db_Connection.conn;
21       cmd = new SqlCommand(@"INSERT INTO tbEmployees
22             (IDCard,EmName,Gender,DOB,Phone,Position,
23              Depatement,Startwork,Endwork, Address,Photo)
24          Values(@IDCard,@EmpName,@Gender,@Dob,@phone,
25             @Position,@depatement,@startwork,@Endwork,
26             @Address,@Photo)", db_Connection.conn);
27
28      cmd.Parameters.AddWithValue("@IDCard",txtIDCard_Emp.Text);
29      cmd.Parameters.AddWithValue("@EmpName",txtEmptNames.Text);
30      cmd.Parameters.AddWithValue("@Gender", txtCusGender.Text);
31      cmd.Parameters.AddWithValue("@Dob",DateTime.Parse((textDob
32                                  .Text).ToString()));
33      cmd.Parameters.AddWithValue("@phone", txtPhone.Text);
34     cmd.Parameters.AddWithValue("@Position", txtPosition.Text);
35      cmd.Parameters.AddWithValue("@depatement",txtdepatement.
36                                  Text);
37      cmd.Parameters.AddWithValue("@startwork",DateTime.Parse(
```

```
                                (txtdatejoin_Emp.Text).ToString()));
      cmd.Parameters.AddWithValue("@Endwork", DateTime.Parse
                              ((txtdateEnd_Emp.Text).ToString()));
        cmd.Parameters.AddWithValue("@Address", txtAdress.Text);
                  //////// add photo
        MemoryStream Adphoto = new MemoryStream();
    this.Txtpic_photo_Emp.Image.Save(Adphoto, ImageFormat.Png);
         byte[] getphoto = new byte[Adphoto.Length];
         Adphoto.Position = 0;
         Adphoto.Read(getphoto, 0, getphoto.Length);

         cmd.Parameters.AddWithValue("@Photo", getphoto);

         cmd.ExecuteNonQuery();

}
```

ប៊ូតុង៖ btupdate_Emp_Click

```
private void btupdate_Emp_Click(object sender, EventArgs e)
{
    db_Connection con = new db_Connection();
    con.Conntion_db();
    SqlCommand cmd = new SqlCommand();
    cmd.Connection = db_Connection.conn;
    cmd = new SqlCommand(@"UPDATE tbEmployees SET IDCard=@IDCard,
            EmName=@EmName,Gender=@Gender,DOB=@dob, Phone=@Phone,
            Position=@Position,Depatement=@Depatement,Address=
            @Address
      WHERE Empid=@Empid", db_Connection.conn);

    cmd.Parameters.AddWithValue("@Empid",
                        int.Parse((txtId_Emp.Text).ToString()));
    cmd.Parameters.AddWithValue("@IDCard",
                      int.Parse((txtIDCard_Emp.Text).ToString()));
    cmd.Parameters.AddWithValue("@EmName",
                       (txtEmptNames.Text).ToString());
    cmd.Parameters.AddWithValue("@Gender",
                        ((txtCusGender.Text).ToString()));
    cmd.Parameters.AddWithValue("@dob",
                       DateTime.Parse((textDob.Text).ToString()));
    cmd.Parameters.AddWithValue("@Phone",
                        ((txtIDCard_Emp.Text).ToString()));
    cmd.Parameters.AddWithValue("@Position",
                        ((txtId_Emp.Text).ToString()));
    cmd.Parameters.AddWithValue("@Depatement",
                        ((txtIDCard_Emp.Text).ToString()));
    cmd.Parameters.AddWithValue("@Address", txtAdress.Text);

    cmd.ExecuteNonQuery();
   // MessageBox.Show("Hello");
```

```
33      classEmployeeBindingSource.DataSource = null;
34      dataGridView1.Rows.Clear();
35
36      Update_Deslpay();
}
```

ប៊ូតុង៖ tbList_Click()

```
1  private void tbList_Click(object sender, EventArgs e)
2  {
3      if (dataGridView1.Rows.Count - 1 == 0)
4      {
5        this.Enabled = false;
6        var F_SMS = new F_Messages();
7        F_SMS.txtgetMessage.Text = "មិនមានទិន្នន័យ ទេ!
8  សូមធ្វើការទោញទិន្នន័យជាមុន !";
9          F_SMS.Closed += (s, args) => this.Enabled = true;
10         F_SMS.Show();
11         return;
12     }
13   else
14   {
15     Class_Employee obj;
16     classEmployeeBindingSource.Clear();
17     int i = dataGridView1.CurrentRow.Index;
18
19  for (i = 0; i < dataGridView1.Rows.Count - 1; i++)
20   {
21    obj = new Class_Employee();
22    obj.ល.រ = (i + 1).ToString();
23    obj.Empid=dataGridView1.Rows[i].Cells[0].Value.ToString();
24    obj.EmName=dataGridView1.Rows[i].Cells[2].Value.ToString();
25    obj.Phone = dataGridView1.Rows[i].Cells[6].Value.ToString();
26    obj.Position=dataGridView1.Rows[i].Cells[5].Value.ToString();
27    obj.Startwork=dataGridView1.Rows[i].Cells[7].Value.ToString();
28    obj.Endwork = dataGridView1.Rows[i].Cells[8].Value.ToString();
29    obj.Address=dataGridView1.Rows[i].Cells[10].Value.ToString();
30
31     classEmployeeBindingSource.Add(obj);
32    }
33        //classEmployeeBindingSource.MoveLast();
34
35    using (FH_Report_Employees fm = new FH_Report_Employees
36             (classEmployeeBindingSource.DataSource as List
37              <Class_Employee>))
38    {
39        fm.ShowDialog();
40        fm.Show();
41
42
```

```
43      }
    }
```

ប៊ូតុង៖ Txtseachpic_LinkClicked()

```
 1 private void Txtseachpic_LinkClicked(object sender,
 2                 LinkLabelLinkClickedEventArgs e)
 3 {
 4     Image image;
 5
 6     // open file dialog
 7     OpenFileDialog open = new OpenFileDialog();
 8     // image filters
 9     open.Filter = "Image Files(*.jpg;*.*;*.Png;*.png; *.jpeg;
10        *.gif; *.bmp)|*.Png;*.png;*.jpg; *.jpeg; *.gif; *.bmp";
11     if (open.ShowDialog() == DialogResult.OK)
12     {
13         image = Image.FromFile(open.FileName);
14
15         //-------------pic box---------
16         this.Txtpic_photo_Emp.Image = image;
17     }
18 }
```

Method៖ Despay_Data()

```
 1 private void Despay_Data()
 2 {
 3     db_Connection connn = new db_Connection();
 4     connn.Conntion_db();
 5     SqlCommand cmd = new SqlCommand();
 6     cmd.Connection = db_Connection.conn;
 7
 8     cmd = new SqlCommand(@"SELECT * FROM tbEmployees ORDER BY
 9           Empid DESC", db_Connection.conn);
10
11     cmd.ExecuteNonQuery();
12
13     SqlDataAdapter data = new SqlDataAdapter(cmd);
14     DataTable tb = new DataTable();
15     data.Fill(tb);
16     dataGridView1.DataSource = tb;
18 }
```

Method៖ Update_Deslpay()

```
 1 private void Update_Deslpay()
 2 {
 3     db_Connection connn = new db_Connection();
 4     connn.Conntion_db();
```

```
    SqlCommand cmd = new SqlCommand();
    cmd.Connection = db_Connection.conn;

   cmd = new SqlCommand(@"SELECT * FROM tbEmployees
             WHERE Empid=@Empid  ORDER BY Empid DESC",
             db_Connection.conn);
  cmd.Parameters.AddWithValue("@Empid",int.Parse(txtId_Emp.Text));
   cmd.ExecuteNonQuery();

   SqlDataAdapter data = new SqlDataAdapter(cmd);
   DataTable tb = new DataTable();
   data.Fill(tb);
   classEmployeeBindingSource.DataSource = tb;
}
```

ប៊ូតុង៖ btdelete_Emp_Click

```
private void btdelete_Emp_Click(object sender, EventArgs e)
{
    var widd = new F_Company();
    db_Connection con = new db_Connection();
    con.Conntion_db();
    SqlCommand cmd = new SqlCommand();
    cmd.Connection = db_Connection.conn;

    cmd = new SqlCommand(@"DELETE  FROM tbEmployees
            WHERE Empid=@Empid", db_Connection.conn);

    cmd.Parameters.AddWithValue("@Empid", txtId_Emp.Text);
    cmd.ExecuteNonQuery();

    SqlDataAdapter da = new SqlDataAdapter(cmd);
    DataTable dt = new DataTable();
    da.Fill(dt);
    dataGridView1.Refresh();
    dataGridView1.DataSource = dt;
}
```

Datalistview ៖ dataGridView1_SelectionChanged()

```
private void dataGridView1_SelectionChanged(object sender,
        EventArgs e)
{
  if (dataGridView1.Rows[0].Cells[0].Value == null)
  {
    return;
  }
 else
 {
```

```
10      int rowindex = dataGridView1.CurrentRow.Index;
11      if (dataGridView1.SelectedRows.Count == 0)
12     {
13          return;
14     }
15     else
16      {
17         txtId_Emp.Text = (dataGridView1.Rows[rowindex].Cells[0].
18                          Value).ToString();
19          txtIDCard_Emp.Text=(dataGridView1.Rows[rowindex]
20                          .Cells[1].Value).ToString();
21          txtEmptNames.Text = (dataGridView1.Rows[rowindex]
22                              .Cells[2].Value).ToString();
23          txtCusGender.Text = (dataGridView1.Rows[rowindex]
24                              .Cells[3].Value).ToString();
25          textDob.Text = (dataGridView1.Rows[rowindex]
26                          .Cells[4].Value).ToString();
27          txtPhone.Text = (dataGridView1.Rows[rowindex]
28                           .Cells[5].Value).ToString();
29          txtPosition.Text = (dataGridView1.Rows[rowindex]
30                            .Cells[6].Value).ToString();
31          txtdepatement.Text = (dataGridView1.Rows[rowindex]
32                            .Cells[7].Value).ToString();
33          txtAdress.Text = (dataGridView1.Rows[rowindex]
34                             .Cells[10].Value).ToString();
35          txtdatejoin_Emp.Text = (dataGridView1.Rows[rowindex]
36                              .Cells[8].Value).ToString();
37          txtdateEnd_Emp.Text = (dataGridView1.Rows[rowindex]
38                              .Cells[9].Value).ToString();
39
40 if(dataGridView1.Rows[rowindex].Cells[11].Value.ToString()== "")
41       {
42              this.Txtpic_photo_Emp.Image = null;
43              // MessageBox.Show("hello");
44               return;
45          }
46          else
47          {
48              byte[] Photobox1;
49              Photobox1 =(byte[])this.dataGridView1.Rows[rowindex]
50                          .Cells[11].Value;
51              var ms = new MemoryStream(Photobox1);
52              this.Txtpic_photo_Emp.Image = Image.FromStream(ms);
53          }
54       if(dataGridView1.Rows[rowindex].Cells[12].Value.ToString()== "")
55          {
56              this.tpidcard_Emp.Image = null;
57              // MessageBox.Show("hello");
58          return;
59       }
```

```
60      else
61        {
62            byte[] Photobox1;
63            Photobox1 = (byte[])this.dataGridView1.Rows[rowindex]
64                        .Cells[12].Value;
65                var ms = new MemoryStream(Photobox1);
66                this.tpidcard_Emp.Image = Image.FromStream(ms);
67            }
68
69      }
80
81      }
82
83  }
84
```

៤. Form Getegory

៤.១ ការបង្កើត Form

Tool Control	Name	Name_khmer
Textbox	txtcategory_id	
lablelink	btBrowser_Pic	
picturebox	pic_Category	
Button	BtSave_Categ BtEdit_Categ btDelet_Categ btListItem_Categ	រក្សាទុក កែតម្រូវ លុប បញ្ជីឈ្មោះ
Combobox	TxtCategory_Name	
DatadataGridView	datagridView1	

៤.២ ការបង្កើត Table ៖ Categories

DESKTOP-QUCR3V0.d...- dbo.Categories

Column Name	Data Type	Allow Nulls
Cate_id	int	☐
Category_Name	nvarchar(200)	☑
Discription	nvarchar(MAX)	☑
Image	image	☑

៤.៣ ដំណើរការសរសេរ កូដ

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.SqlClient;
using System.Windows.Forms;
using System.IO;
using System.Drawing.Imaging;

namespace POS_Systems
{
 public partial class F_Categories : Form
  {
    public F_Categories()
     {
       InitializeComponent();
       this.StartPosition = FormStartPosition.Manual;
       this.Location = new Point(400, 130);
      }
// --------បង្កើត Method Categories( )----------;
  private void Categories()
   {

    db_Connection con = new db_Connection();
    con.Conntion_db();
    SqlCommand cmd = new SqlCommand();
    cmd = new SqlCommand(@"SELECT Cate_id,Category_Name
          FROM Categories ORDER BY Category_Name ASC",
          db_Connection.conn);
   cmd.Parameters.AddWithValue("@Category_Name",
                               TxtCategory_Name.Text);
   cmd.ExecuteNonQuery();

   SqlDataReader row;
   row = cmd.ExecuteReader();
```

```
40    TxtCategory_Name.Items.Clear();
41    while (row.Read())
42    {
43      string ProductName = (String)row["Category_Name"].ToString();
44      int Cate_id = int.Parse((row["Cate_id"].ToString()).ToString());
45      TxtCategory_Name.Items.Add(ProductName.ToString());
46    }
47   }
48
49 //---------បង្កើត Method Categories_show( )----------បង្ហាញទិន្នន័យ ;
50  private void Categories_Show()
51   {
52     db_Connection con = new db_Connection();
53     con.Conntion_db();
54     SqlCommand cmd = new SqlCommand();
55     cmd = new SqlCommand(@"SELECT Cate_id,Category_Name
56          FROM Categories ORDER BY Cate_id DESC", db_Connection.conn);
57
58     cmd.ExecuteNonQuery();
59     SqlDataAdapter DATA = new SqlDataAdapter(cmd);
60     DataTable dts = new DataTable();
61     DATA.Fill(dts);
62     dataGridView1.DataSource = dts;
63   }
64
65
66
67 //---------ប៊ូតុង រក្សាទុក ----------------;
68  private void BtSave_Categ_Click(object sender, EventArgs e)
69 {
70     db_Connection cnn = new db_Connection();
71     cnn.Conntion_db();
72     SqlCommand comm = new SqlCommand();
73     comm.Connection = db_Connection.conn;
74     comm = new SqlCommand(@"SELECT COUNT(*) FROM Categories
75           WHERE Category_Name=@Category_Name", db_Connection.conn);
76     comm.Parameters.AddWithValue("@Category_Name", TxtCategory_
77                                  Name.Text);
78     comm.ExecuteNonQuery();
79     SqlDataAdapter DATA = new SqlDataAdapter(comm);
80     DataTable dts = new DataTable();
81     DATA.Fill(dts);
82
83     if (TxtCategory_Name.Text == "")
84     {
85       TxtCategory_Name.Focus();
86       this.Enabled = false;
87       var F_SMS = new F_Messages();
88
89       F_SMS.txtgetMessage.Text ="សូមបញ្ចូលឈ្មោះក្រុមប្រភេទទំនិញ ..!";
90       F_SMS.Closed += (s, args) => this.Enabled = true;
91       F_SMS.Show();
92
93       TxtCategory_Name.Focus();
94       return;
```

```
95      }
96
97      else if (dts.Rows[0][0].ToString() == "1")
98      {
99
100         TxtCategory_Name.Focus();
101         this.Enabled = false;
102         var F_SMS = new F_Messages();
103        F_SMS.txtgetMessage.Text = "ប្រភេទក្រុមទំនិញ មានរួចហើយ សូមពិនិត្យម្តងទៀង..!";
104         F_SMS.Closed += (s, args) => this.Enabled = true;
105         F_SMS.Show();
106
107
108         TxtCategory_Name.Focus();
109         return;
110
111      }
112     else
113      {
114
115       if (TxtCategory_Name.Text == "")
116        {
117          TxtCategory_Name.Focus();
118          this.Enabled = false;
119          var F_SMS = new F_Messages();
120          F_SMS.txtgetMessage.Text = "សូមបញ្ចូលឈ្មោះក្រុមប្រភេទទំនិញ ..!";
121          F_SMS.Closed += (s, args) => this.Enabled = true;
122          F_SMS.Show();
123         TxtCategory_Name.Focus();
124         return;
125       }
126      else if (pic_Category.Image == null)
127      {
128        pic_Category.Focus();
129        this.Enabled = false;
130        var F_SMS = new F_Messages();
131        F_SMS.txtgetMessage.Text = "សូមបញ្ចូលរូបភាពប្រភេទក្រុមទំនិញ ..!";
132        F_SMS.Closed += (s, args) => this.Enabled = true;
133        F_SMS.Show();
134        pic_Category.Focus();
135        return;
136      }
137      else
138       {
139        db_Connection con = new db_Connection();
140        con.Conntion_db();
141        SqlCommand cmd = new SqlCommand();
142        cmd.Connection = db_Connection.conn;
143        cmd = new SqlCommand(@"INSERT INTO Categories (Category_Name,
144             Image)VALUES(@CateName,@Images)", db_Connection.conn);
145        cmd.Parameters.AddWithValue("CateName", TxtCategory_Name.Text);
146        //  cmd.Parameters.AddWithValue;
147        //==============photo
148
149
```

```
150         MemoryStream ms = new MemoryStream();
151         pic_Category.Image.Save(ms, ImageFormat.Png);
152         byte[] photo = new byte[ms.Length];
153         ms.Position = 0;
154         ms.Read(photo, 0, photo.Length);
155         cmd.Parameters.AddWithValue("@Images", photo);
156
157         cmd.ExecuteNonQuery();
158
159         this.Enabled = false;
160         var F_SMS = new F_Messages();
161         F_SMS.txtgetMessage.Text = "លោកអ្នកបានបញ្ចូលប្រភេទក្រុមទំនិញថ្មី .ជោគជ័យ.!";
162         F_SMS.Closed += (s, args) => this.Enabled = true;
163         F_SMS.Show();
164         Categories_Data();
165       }
166
167     }
168   }
169
170 //--------Methode Update_Category( ) ------------;
171   private void Update_Category()
172    {
173     if (TxtCategory_Name.Text == "")
174      {
175         TxtCategory_Name.Focus();
176         this.Enabled = false;
177         var F_SMS = new F_Messages();
178         F_SMS.txtgetMessage.Text = "សូមបញ្ចូលឈ្មោះក្រុមប្រភេទទំនិញ ..!";
179         F_SMS.Closed += (s, args) => this.Enabled = true;
180         F_SMS.Show();
181         TxtCategory_Name.Focus();
182         return;
183     }
184     else
185       if (pic_Category.Image == null)
186        {
187          this.Enabled = false;
188          var F_SMS = new F_Messages();
189          F_SMS.txtgetMessage.Text = "សូមបញ្ចូល រូបភាពប្រភេទក្រុមទំនិញ ..!";
190          F_SMS.Closed += (s, args) => this.Enabled = true;
191          F_SMS.Show();
192
193          return;
194        }
195        else
196        {
197
198          db_Connection con = new db_Connection();
199          con.Conntion_db();
200          SqlCommand cmd = new SqlCommand();
201          cmd.Connection = db_Connection.conn;
202          cmd = new SqlCommand(@"UPDATE Categories SET Category_Name
203              =@CateName,Image=@Images
204
```

```
            WHERE Cate_id=@Cate_id", db_Connection.conn);

        cmd.Parameters.AddWithValue("@Cate_id", txtcategory_id.Text);
        cmd.Parameters.AddWithValue("@CateName", TxtCategory_Name.Text);

         //===============photo
         MemoryStream ms = new MemoryStream();
         pic_Category.Image.Save(ms, ImageFormat.Png);
         byte[] photo = new byte[ms.Length];
         ms.Position = 0;
         ms.Read(photo, 0, photo.Length);
         cmd.Parameters.AddWithValue("@Images", photo);

         cmd.ExecuteNonQuery();

         this.Enabled = false;
         var F_SMS = new F_Messages();
         F_SMS.txtgetMessage.Text = "លោកអ្នកបានកែប្រែ ឈ្មោះក្រុមប្រភេទទំនិញ .ជោគជ័យ.!";
         F_SMS.Closed += (s, args) => this.Enabled = true;
         F_SMS.Show();
     }
     Categories();
    }
// ---------បង្កើត កំណត់ ទំហំ រូបភាព ------------;
  public static Image resizeImage(Image imgToResize, Size size)
   {
     return (Image)(new Bitmap(imgToResize, size));
   }

// ---------បង្កើត Method Categories_id ( )---------;
  private void Categories_id()
   {
     db_Connection con = new db_Connection();
     con.Conntion_db();
     SqlCommand cmd = new SqlCommand();
     cmd = new SqlCommand(@"SELECT Cate_id,Category_Name
             FROM Categories
             WHERE Category_Name=@Category_Name ORDER BY Category_Name
                   ASC", db_Connection.conn);
    cmd.Parameters.AddWithValue("@Category_Name", TxtCategory_Name.Text);
        cmd.ExecuteNonQuery();

        SqlDataReader row;
        row = cmd.ExecuteReader();

      while (row.Read())
        {
         int Cate_id = int.Parse((row["Cate_id"].ToString()).ToString());
          txtcategory_id.Text = Cate_id.ToString();
        }
    }

// ---------បង្កើត Method Categories_Data( );
```

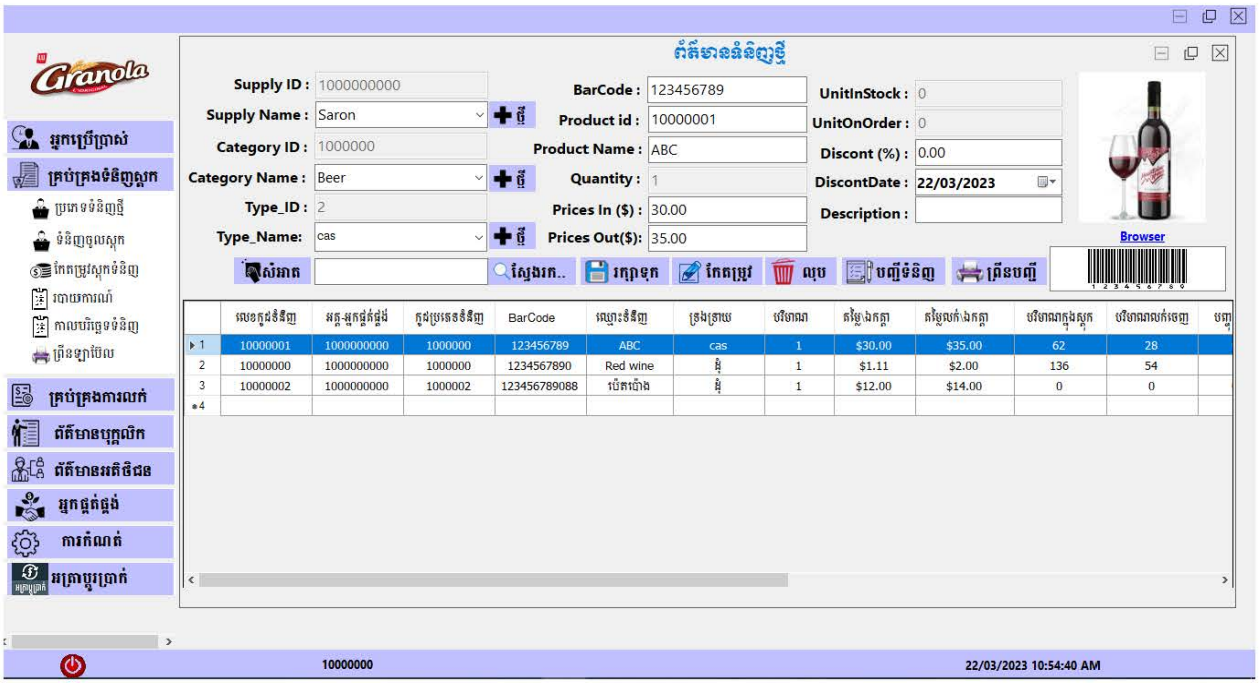
```
260  private void Categories_Data()
261      {
262        db_Connection con = new db_Connection();
263        con.Conntion_db();
264        SqlCommand cmd = new SqlCommand();
265        cmd = new SqlCommand(@"SELECT Cate_id,Category_Name,Image
266                  FROM Categories
267                  ORDER BY Cate_id DESC", db_Connection.conn);
268
269          cmd.ExecuteNonQuery();
270          SqlDataAdapter DATA = new SqlDataAdapter(cmd);
271          DataTable dts = new DataTable();
272          DATA.Fill(dts);
273          dataGridView1.DataSource = dts;
274
275          SqlDataReader row;
276          row = cmd.ExecuteReader();
277
278      }
279
280 //--- --- -------- Method : Play_DataGrid( ) -------- ----
281    public void Play_DataGrid()
282     {
283       if (dataGridView1.SelectedRows.Count == 0)
284         {
285             return;
286         }
287
288          byte[] Photobox1;
289         //  byte[] Photobox4;
290          Photobox1 = (byte[])this.dataGridView1[17,
291                  dataGridView1.SelectedRows[0].Index].Value;
292          var ms = new MemoryStream(Photobox1);
293          this.pic_Category.Image = Image.FromStream(ms);
294      }
295
296
297 //--- --- -------- ប៊ូតុង Browser រូបភាព-------- ----
298    private void btBrowser_Pic_LinkClicked(object sender, LinkLabelLink
299 ClickedEventArgs e)
300      {
301        OpenFileDialog open = new OpenFileDialog();
302        open.Filter = "Image Files(*.jpg;*.*;*.Png;*.png; *.jpeg;
303             *.gif;  *.bmp)|*.Png;*.png;*.jpg; *.jpeg; *.gif; *.bmp";
304         if (open.ShowDialog() == DialogResult.OK)
305          {
306            pic_Category.Text = open.FileName;
307            pic_Category.Image = Image.FromFile(open.FileName);
308            pic_Category.Image = resizeImage(pic_Category.Image,
309                                 new Size(100, 120));
310          }
311      }
312
313
314
```

```
315 //--- --- -------- ប៊ូតុង កែតម្រូវ--------- ----
316     private void BtEdit_Categ_Click(object sender, EventArgs e)
317     {
318         Update_Category();
319     }
320
321 //--- ---  បង្ហាញទិន្នន័យ ពេល Selection idex ក្នុង Commbox Category  ----
322     private void TxtCategory_Name_Click(object sender, EventArgs e)
323     {
324         Categories();
325
326
327     }
328
329 //--- ---  បង្ហាញទិន្នន័យ ពេល Start Form_load -----
330     private void F_Categories_Load(object sender, EventArgs e)
331     {
332         Categories();
333         Categories_Data();
334
335     }
336
337 //---   seletion Row លើ datagrivew-----
338   private void TxtCategory_Name_SelectedIndexChanged(object sender,
339 EventArgs e)
340     {
341         Categories_id();
342     }
343
344 //----ប៊ូតុង x បិទ-----
345     private void Pic_Closes_Click(object sender, EventArgs e)
346     {
347         Close();
348     }
349
350  //----------Even selectionchage datagridView1 ----------;
351  private void dataGridView1_SelectionChanged(object sender,EventArgs e)
352 {
353  try
354   {
355    int rowindex = dataGridView1.CurrentRow.Index;
356
357    if (dataGridView1.Rows[rowindex].Cells[0].Value == null)
358      {
359          return;
360       }
361     else
362      {
363
364       if (dataGridView1.SelectedRows.Count == 0)
365        {
366           return;
```

```
370         }
371     else
372       {
373         txtcategory_id.Text = dataGridView1.Rows[rowindex].Cells[0].
374 Value.ToString();
375          TxtCategory_Name.Text = dataGridView1.Rows[rowindex]
376                                      .Cells[1].Value.ToString();
377
378
379
380       if(dataGridView1.Rows[rowindex].Cells[2].Value.ToString() == "")
381          {
382             this.pic_Category.Image = null;
383             MessageBox.Show("hello");
384               return;
385           }
386        else
387           {
388             byte[] Photobox1;
389             Photobox1 = (byte[])this.dataGridView1.Rows[rowindex].
390 Cells[2].Value;
391             var ms = new MemoryStream(Photobox1);
392             this.pic_Category.Image = Image.FromStream(ms);
393            }
394
395          }
396
397        }
398
399
400      }
401        catch (Exception)
402      {
403
404      }
405    }
406
407 //---------សេចក្តីរៀង DatagridView1 ----------;
408    private void dataGridView1_RowPostPaint(object sender, DataGridView
409 RowPostPaintEventArgs e)
410     {
411         var grid = sender as DataGridView;
412         var rowIdx = (e.RowIndex + 1).ToString();
413         var centerformat = new StringFormat()
414         {
415             Alignment = StringAlignment.Center,
416             LineAlignment = StringAlignment.Center
417
418         };
419         var headerBounds = new Rectangle(e.RowBounds.Left,
420 e.RowBounds.Top, grid.RowHeadersWidth, e.RowBounds.Height);
421         e.Graphics.DrawString(rowIdx, this.Font,
422 SystemBrushes.ControlText, headerBounds, centerformat);
423
424
```

```
425
426     }
427 //----------ប៊ូតុង លុប ----------;
428   private void btDelet_Categ_Click(object sender, EventArgs e)
429    {
430     if (TxtCategory_Name.Text == "")
431      {
432        TxtCategory_Name.Focus();
433        this.Enabled = false;
434        var F_SMS = new F_Messages();
435        F_SMS.txtgetMessage.Text = "សូមបញ្ចូលឈ្មោះក្រុមប្រភេទទំនិញ ..!";
436        F_SMS.Closed += (s, args) => this.Enabled = true;
437        F_SMS.Show();
438        return;
439       }
450      else
451       {
452        db_Connection con = new db_Connection();
453        con.Conntion_db();
454        SqlCommand cmd = new SqlCommand();
455        cmd.Connection = db_Connection.conn;
456        cmd = new SqlCommand(@"DELETE FROM Categories
457                  WHERE Cate_id=@Cate_id", db_Connection.conn);
458        cmd.Parameters.AddWithValue("@Cate_id", txtcategory_id.Text);
459        cmd.ExecuteNonQuery();
460
461        this.Enabled = false;
462        var F_SMS = new F_Messages();
463
464        F_SMS.txtgetMessage.Text = "លោកអ្នកបានលុបប្រភេទភេទទំនិញ .ជោគជ័យ.!";
465        F_SMS.Closed += (s, args) => this.Enabled = true;
466        F_SMS.Show();
467
468        Categories_Data();
469      }
470
471    }
472 }
473
474
```

៥. ការបង្កើត Form ព័ត៌មានអំពីទំនិញ



៥.១ ការបង្កើត Form

Tool Control	Name	Name_khmer
Textbox	txtSupply_Id txtCategry_id txtTypeGoods_Id txtBarcode_pro TxtBarcode txtProduct_id TxtProduct_Name TxtQuantity txtPrice_in TxtPrices_out txtUnitIn_Stock TxtUnit_Order TxtDiscont_sel txtDiscription_pro	
lablelink	btBrowser_Pic	
picturebox	pic_Product	
Button	btClear_Items btsearch_prod btSave_Pro btEdite	សម្អាត ស្វែងរក រក្សាទុក កែតម្រូវ

	btdelete_pro btList_Product btPrint_List_Product btSupplyees btAddnew_Categ btAddNew_Brand	លុប បញ្ជីទំនិញ ព្រីនបញ្ជី
Combobox	TxtCategory_Name txtSupply_Name txtTypegoods_Name	
DatadataGridView	datagridView1	
dateTimePicker1	TxtDiscont_date	

៥.២ ការបង្កើត Table ៖ Products

DESKTOP-QUCR3V0....OS - dbo.Products

Column Name	Data Type	Allow Nulls
Product_id	int	☐
Supplyeer_id	int	☑
Categorys_id	int	☑
user_id	int	☑
BarCode	nvarchar(50)	☑
ProductName	nvarchar(MAX)	☑
ProductTypes_id	int	☑
QuantityPerUnit	int	☑
UnitPrice_in	float	☑
UnitPrice_Out	float	☑
UnitInStock	float	☑
UnitOnOrder	float	☑
Discont	float	☑
DiscontDate	date	☑
Images_item	image	☑
Date_at	date	☑
Date_up	date	☑
Action	int	☑

៦. កូដ

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Drawing.Imaging;
```

```
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.SqlClient;
using System.Windows.Forms;

namespace POS_Systems
{

public partial class F_AddNew_Product : UserControl
{
    public F_AddNew_Product()
    {
        InitializeComponent();

    }
    private InputLanguages InputLang = new InputLanguages();

//----------ប៊ូតុង រក្សាទុក----------;
private void Save_Productditail()
{
    db_Connection cnn = new db_Connection();
    cnn.Conntion_db();
    SqlCommand comm = new SqlCommand();
    comm.Connection = db_Connection.conn;
    comm = new SqlCommand(@"SELECT COUNT(*) FROM tbProdType_Detail
              WHERE Supplyeer_id=@Supplyeer_id
              AND Category_id =@Category_id
               AND ProType_id=@ProType_id ", db_Connection.conn);

    comm.Parameters.AddWithValue("@Supplyeer_id", txtSupply_Id.Text);
    comm.Parameters.AddWithValue("@Category_id", txtCategry_id.Text);
    comm.Parameters.AddWithValue("@ProType_id", txtTypeGoods_Id.Text);
    comm.ExecuteNonQuery();

    SqlDataAdapter da = new SqlDataAdapter(comm);
    DataTable dt = new DataTable();
    da.Fill(dt);
    if (txtSupply_Id.Text == "")
    {
        txtSupply_Name.Focus();
        this.Enabled = false;
        var F_SMS = new F_Messages();
        F_SMS.txtgetMessage.Text = "សូមជ្រើសរើស (Supplyeer_Name) ឈ្មោះអ្នកផ្គត់ផ្គង់ .!";
        F_SMS.Closed += (s, args) => this.Enabled = true;
        F_SMS.Show();
        txtSupply_Name.Focus();
        return;
    }
    else if (TxtCategory_Name.Text == "")
     {
      TxtCategory_Name.Focus();
      this.Enabled = false;
      var F_SMS = new F_Messages();
     F_SMS.txtgetMessage.Text ="សូមជ្រើសរើស (Category_Name) ឈ្មោះប្រភេទទំនិញ .!";
```

```
64          F_SMS.Closed += (s, args) => this.Enabled = true;
65          F_SMS.Show();
66          TxtCategory_Name.Focus();
67          return;
68       }
69      else if (txtTypegoods_Name.Text == "")
70       {
71          txtTypegoods_Name.Focus();
72          this.Enabled = false;
73          var F_SMS = new F_Messages();
74          F_SMS.txtgetMessage.Text = "សូមជ្រើសរើស (Type_Name) ឈ្មោះទ្រង់ទ្រាយទំនិញ .!";
75           F_SMS.Closed += (s, args) => this.Enabled = true;
76           F_SMS.Show();
77           return;
78
79
80         }
81        else if (dt.Rows[0][0].ToString() == "1")
82         {
83
84            this.Enabled = false;
85             var F_SMS = new F_Messages();
86        F_SMS.txtgetMessage.Text = "សូមពិនិត្យម្តងទៀត ប្រភេទទំនិញរបស់អ្នកផ្គត់ផ្គង់មានរួចហើយ !";
87           F_SMS.Closed += (s, args) => this.Enabled = true;
88           F_SMS.Show();
89           return;
90
91         }
92       else
93         {
94
95          db_Connection con = new db_Connection();
96          con.Conntion_db();
97          SqlCommand cmd = new SqlCommand();
98          cmd.Connection = db_Connection.conn;
99          cmd = new SqlCommand(@"INSERT INTO tbProdType_Detail
100                      (Supplyeer_id ,Category_id,ProType_id)
101               Values(@Supplyeer_id,@Category_id,@ProType_id)",
102                      db_Connection.conn);
103
104         cmd.Parameters.AddWithValue("@Supplyeer_id", txtSupply_Id.Text);
105         cmd.Parameters.AddWithValue("@Category_id", txtCategry_id.Text);
106         cmd.Parameters.AddWithValue("@ProType_id", txtTypeGoods_Id.Text);
107         cmd.ExecuteNonQuery();
108         MessageBox.Show("Successfully !");
109
110         this.Enabled = false;
111         var F_SMS = new F_Messages();
112         F_SMS.txtgetMessage.Text = "លោកអ្នកបានបញ្ចូលប្រភេទទំនិញរបស់អ្នកផ្គត់ផ្គង់.ជោគជ័យ.!";
113         F_SMS.Closed += (s, args) => this.Enabled = true;
114         F_SMS.Show();
115
116         Show_data();
117      }
118   }
119
120 //=======ប៊ីតុង រក្សាទុក ----------ទំនិញ==========;
```

```
121 private void btSave_Pro_Click(object sender, EventArgs e)
122  {
123   if (txtSupply_Name.Text == "")
124     {
125         txtSupply_Name.Focus();
126         this.Enabled = false;
127         var F_SMS = new F_Messages();
128         F_SMS.txtgetMessage.Text = "សូមបញ្ចូលក្រុមហ៊ុន អ្នកផ្គត់ផ្គង់..!";
129         F_SMS.Closed += (s, args) => this.Enabled = true;
130         F_SMS.Show();
131         txtSupply_Name.Focus();
132         return;
133     }
134     else if (txtCategry_id.Text == "" && TxtCategory_Name.Text=="")
135     {
136         txtCategry_id.Focus();
137         TxtCategory_Name.Focus();
138         this.Enabled = false;
139         var F_SMS = new F_Messages();
140         F_SMS.txtgetMessage.Text = "សូមបញ្ចូល Categories ប្រភេទមុខទំនិញ..!";
141         F_SMS.Closed += (s, args) => this.Enabled = true;
142         F_SMS.Show();
143         return;
144     }
145     else if (txtTypegoods_Name.Text == "")
146     {
147         txtTypegoods_Name.Focus();
148         this.Enabled = false;
149         var F_SMS = new F_Messages();
150         F_SMS.txtgetMessage.Text = "សូមបញ្ចូល ទ្រង់ទ្រាយ Type_Name ប្រភេទទំនិញ..!";
151         F_SMS.Closed += (s, args) => this.Enabled = true;
152         F_SMS.Show();
153         return;
154     }
155     else if (txtBarcode_pro.Text == "")
156     {
157         txtBarcode_pro.Focus();
158         this.Enabled = false;
159         var F_SMS = new F_Messages();
160         F_SMS.txtgetMessage.Text = "សូមបញ្ចូល លេខ៖ Barcodes ទំនិញ..!";
161         F_SMS.Closed += (s, args) => this.Enabled = true;
162         F_SMS.Show();
163         return;
164     }
165     else if (TxtProduct_Name.Text == "")
166     {
167         TxtProduct_Name.Focus();
168         this.Enabled = false;
169         var F_SMS = new F_Messages();
170         F_SMS.txtgetMessage.Text = "សូមបញ្ចូល ប្រភេទទំនិញ..!";
171         F_SMS.Closed += (s, args) => this.Enabled = true;
172         F_SMS.Show();
173         return;
174     }
175     else if (TxtQuantity.Text == "")
176
177
```

```
178     {
179         TxtQuantity.Focus();
180         this.Enabled = false;
181         var F_SMS = new F_Messages();
182         F_SMS.txtgetMessage.Text = "សូមបញ្ចូល ប្រភេទទំនិញ..!";
183         F_SMS.Closed += (s, args) => this.Enabled = true;
184         F_SMS.Show();
185         return;
186     }
187     else if (txtPrice_in.Text == "" && txtPrice_in.Text == "0.00")
188     {
189         txtPrice_in.Focus();
190         this.Enabled = false;
191         var F_SMS = new F_Messages();
192         F_SMS.txtgetMessage.Text = "សូមបញ្ចូល តម្លៃចូល ទំនិញ..!";
193         F_SMS.Closed += (s, args) => this.Enabled = true;
194         F_SMS.Show();
195         return;
196     }
197     else if (TxtPrices_out.Text == "0.00" && TxtPrices_out.Text == "")
198     {
199         TxtPrices_out.Focus();
200         this.Enabled = false;
201         var F_SMS = new F_Messages();
202         F_SMS.txtgetMessage.Text = "សូមបញ្ចូល តម្លៃលក់ចេញ ទំនិញ..!";
203         F_SMS.Closed += (s, args) => this.Enabled = true;
204         F_SMS.Show();
205         return;
206     }
207     else if (txtUnitIn_Stock.Text == "")
208     {
209         txtUnitIn_Stock.Focus();
210         this.Enabled = false;
211         var F_SMS = new F_Messages();
212         F_SMS.txtgetMessage.Text = "សូមបញ្ចូល □□□□□ □□□□□..!";
213         F_SMS.Closed += (s, args) => this.Enabled = true;
214         F_SMS.Show();
215         return;
216     }
217     else if (TxtUnit_Order.Text == "")
218     {
219         TxtUnit_Order.Focus();
220         this.Enabled = false;
221         var F_SMS = new F_Messages();
222         F_SMS.txtgetMessage.Text = "សូមបញ្ចូលលក់ចេញ ទំនិញ..!";
223         F_SMS.Closed += (s, args) => this.Enabled = true;
224         F_SMS.Show();
225         return;
226     }
227     else if (TxtDiscont_sel.Text == "")
228     {
229         TxtDiscont_sel.Focus();
230         this.Enabled = false;
231         var F_SMS = new F_Messages();
232
233
234
```

```
235         F_SMS.txtgetMessage.Text = "សូមបញ្ចូល បញ្ចៃតម្លៃ ទំនិញ..!";
236         F_SMS.Closed += (s, args) => this.Enabled = true;
237         F_SMS.Show();
238         return;
239     }
240     else if (txtdate_time.Text == "")
241     {
242         txtdate_time.Focus();
243         return;
244     }
245     else if (pic_Product.Image == null)
246     {
247         var F_SMS = new F_Messages();
248         F_SMS.txtgetMessage.Text = "សូមបញ្ចូល រូបភាព សម្គាល់របស់ទំនិញ..!";
249         F_SMS.Closed += (s, args) => this.Enabled = true;
250         F_SMS.Show();
251         return;
252     }
253     else
254     {
255         AddNew_Product_Admin();
256         Save_Productditail();
257         ShowData_Products();
258     }
259
260  }
261 //=======ប៊ូតុង បិទ X ----------ទំនិញ==========;
262 private void Pic_Closes_Click(object sender, EventArgs e)
263  {
264    Dispose();
265  }
266
267 private DateTime today_work = DateTime.Today;
268 //=======Method: AddNew_Product_Admin for user Admin----------ទំនិញ==========;
269 private void AddNew_Product_Admin()
270 {
271   if (pic_Product.Image == null)
272   {
273     MessageBox.Show("Pls insert Photo product !");
274     return;
275   }
276  else
277   {
278     db_Connection con = new db_Connection();
279     con.Conntion_db();
280     SqlCommand cmd = new SqlCommand();
281     cmd.Connection = db_Connection.conn;
282     int mark = 1;
283     cmd = new SqlCommand(@"INSERT INTO Products (user_id,Supplyeer_id,
284               Categorys_id,Barcode,ProductName,ProductTypes_id,
285               QuantityPerUnit,UnitPrice_in,UnitPrice_Out,UnitInStock,
286               UnitOnOrder,Discont,DiscontDate, Images_item, Date_at,
287               Action)
288         VALUES(@userid,@Suppleer_id, @Categorys_id,@Barcode,
289               @ProductName,@Product_Types,@QTY, @unitPrice_in,
290
291
```

```
292                 @UnitPrice_Out,@UnitInStock,@UnitOnOrder,@Discont,
293                 @DiscontDate, @Images, @Date_at, @Action)",
294             db_Connection.conn);
295
296         cmd.Parameters.AddWithValue("@userid", int.Parse(txtUser_Id.Text));
297         cmd.Parameters.AddWithValue("@Suppleer_id", txtSupply_Id.Text);
298         cmd.Parameters.AddWithValue("@ProductName", TxtProduct_Name.Text);
299         cmd.Parameters.AddWithValue("@Product_Types",txtTypeGoods_Id.Text);
300         cmd.Parameters.AddWithValue("@Categorys_id",int.Parse(txtCategry_id.Text));
301         cmd.Parameters.AddWithValue("@Barcode", txtBarcode_pro.Text);
302         cmd.Parameters.AddWithValue("@QTY", TxtQuantity.Text);
303         cmd.Parameters.AddWithValue("@unitPrice_in", txtPrice_in.Text);
304         cmd.Parameters.AddWithValue("@unitPrice_out", TxtPrices_out.Text);
305         cmd.Parameters.AddWithValue("@UnitInStock", txtUnitIn_Stock.Text);
306         cmd.Parameters.AddWithValue("@UnitOnOrder", TxtUnit_Order.Text);
307         cmd.Parameters.AddWithValue("@Discont", TxtDiscont_sel.Text);
308         cmd.Parameters.AddWithValue("@Date_at", DateTime.Parse(txtdate_time.Text));
309        cmd.Parameters.AddWithValue("@DiscontDate",DateTime.Parse(txtdate_time.Text));
310         cmd.Parameters.AddWithValue("@Action", mark);
311             //===============photo
312          MemoryStream ms = new MemoryStream();
313          pic_Product.Image.Save(ms, ImageFormat.Png);
314          byte[] photo = new byte[ms.Length];
315          ms.Position = 0;
316          ms.Read(photo, 0, photo.Length);
317          cmd.Parameters.AddWithValue("@Images", photo);
318          cmd.ExecuteNonQuery();
319
320           this.Enabled = false;
321           var F_SMS = new F_Messages();
322           F_SMS.txtgetMessage.Text ="លោកអ្នក បានធ្វើការបញ្ចូលទំនិញថ្មី.ជោគជ័យ.!";
323           F_SMS.Closed += (s, args) => this.Enabled = true;
324           F_SMS.Show();
325
326
327       }
328
329   }
330 //=======Method: AddNew_Product_Users for user  users----------ទំនិញ==========;
331   private void AddNew_Product_Users()
332   {
333     if (pic_Product.Image == null)
334     {
335        MessageBox.Show("Pls insert Photo product !");
336        return;
337     }
338    else
339    {
340    db_Connection con = new db_Connection();
341    con.Conntion_db();
342    SqlCommand cmd = new SqlCommand();
343    cmd.Connection = db_Connection.conn;
344    int mark = 1;
345    cmd = new SqlCommand(@"INSERT INTO Products (user_id, Categorys_id,
346           Barcode, ProductName, QuantityPerUnit, UnitPrice_in,
347           Images_item, Date_at, Action)
348      values(@userid, @Categorys_id,@Barcode, @ProductName,@QTY,
```

```
349         @unitPrice_in, @Images, @Date_at, @Action)",db_Connection.conn);
350
351   cmd.Parameters.AddWithValue("@userid", int.Parse(txtUser_Id.Text));
352   cmd.Parameters.AddWithValue("@ProductName", TxtProduct_Name.Text);
353   cmd.Parameters.AddWithValue("@Categorys_id",int.Parse(txtCategry_id.Text));
354   cmd.Parameters.AddWithValue("@Barcode", TxtBarcode.Text);
355   cmd.Parameters.AddWithValue("@QTY", TxtQuantity.Text);
356 cmd.Parameters.AddWithValue("@unitPrice_in",float.Parse(txtPrice_in.Text));
357   cmd.Parameters.AddWithValue("@Date_at",DateTime.Parse(txtdate_time.Text));
358   cmd.Parameters.AddWithValue("@Action", mark);
359   //==============photo
360   MemoryStream ms = new MemoryStream();
361   pic_Product.Image.Save(ms, ImageFormat.Png);
362   byte[] photo = new byte[ms.Length];
363   ms.Position = 0;
364   ms.Read(photo, 0, photo.Length);
365   cmd.Parameters.AddWithValue("@Images", photo);
366   cmd.ExecuteNonQuery();
367
368         MessageBox.Show("Successfully!");
369     }
370 }
371 //=======Method: Update_Product_Admin for user Admin ======គ្រប់គ្រងទំនិញ==========;
372
373 private void Updated_Product_Admin()
374 {
375  try
376   {
377     db_Connection con = new db_Connection();
378    con.Conntion_db();
379    SqlCommand cmd = new SqlCommand();
380    cmd.Connection = db_Connection.conn;
381   //  int mark = 1;
382    cmd = new SqlCommand(@"UPDATE Products SET Categorys_id=@Categorys,
383          BarCode=@Barcode,Brand_id=@Brand_id,ProductName= @ProductName,
384          UnitPrice_Out=@UnitPrice_Out,Discont=@Discont,
385          QuantityPerUnit= @QTY,UnitPrice_in=@unitPrice_in,
386          Images_item=@Images, Date_up=@Date_up
387    WHERE Product_id=@Product_id AND user_id= @userid",db_Connection.conn);
388
389   cmd.Parameters.AddWithValue("@Product_id",int.Parse(txtProduct_id.Text));
390   cmd.Parameters.AddWithValue("@userid", int.Parse(txtUser_Id.Text));
391   cmd.Parameters.AddWithValue("@ProductName", TxtProduct_Name.Text);
392   cmd.Parameters.AddWithValue("@Categorys", int.Parse(txtCategry_id.Text));
393   cmd.Parameters.AddWithValue("@Barcode", txtBarcode_pro.Text);
394   cmd.Parameters.AddWithValue("@Brand_id", txtTypeGoods_Id.Text);
395   cmd.Parameters.AddWithValue("@QTY", TxtQuantity.Text);
396   cmd.Parameters.AddWithValue("@unitPrice_in", txtPrice_in.Text);
397   cmd.Parameters.AddWithValue("@UnitPrice_Out", TxtPrices_out.Text);
398   cmd.Parameters.AddWithValue("@Discont", float.Parse(TxtDiscont_sel.Text));
399   cmd.Parameters.AddWithValue("@Date_up",DateTime.Parse(txtdate_time.Text));
400
401   //==============photo
402   MemoryStream ms = new MemoryStream();
403   pic_Product.Image.Save(ms, ImageFormat.Png);
404   byte[] photo = new byte[ms.Length];
405   ms.Position = 0;
   ms.Read(photo, 0, photo.Length);
```

```
406   cmd.Parameters.AddWithValue("@Images", photo);
407   cmd.ExecuteNonQuery();
408
409   this.Enabled = false;
410   var F_SMS = new F_Messages();
411   F_SMS.txtgetMessage.Text = "លោកអ្នក កែប្រែទិន្នន័យ ទំនិញ បានជោគជ័យ..!";
412
413   F_SMS.Closed += (s, args) => this.Enabled = true;
414   F_SMS.Show();
415
416     }catch (Exception)
417     {
418
419     }
420
421   }
422 //=======Method: Update_Product_Users for user difaul Users --------ទំនិញ==========;
423 private void Updated_Product_Users()
424 {
425   if (pic_Product.Image == null)
426   {
427       MessageBox.Show("Pls insert Photo product !");
428       return;
429   }
430   else
431   {
432      db_Connection con = new db_Connection();
433      con.Conntion_db();
434      SqlCommand cmd = new SqlCommand();
435      cmd.Connection = db_Connection.conn;
436       // int mark = 1;
437      cmd = new SqlCommand(@"INSERT Products SET(Categorys_id=
438            @Categorys_id, Barcode=@Barcode, ProductName=@ProductName,
439              QuantityPerUnit= @QTY, UnitPrice_in=@unitPrice_in,
440              Images_item= @Images, Date_up=@Date_up, Action)
441           WHERE(user_id=@userid AND Product_id=@Product_id)",
442                db_Connection.conn);
443
444    cmd.Parameters.AddWithValue("@Product_id",int.Parse(txtProduct_id.Text));
445     cmd.Parameters.AddWithValue("@userid", int.Parse(txtUser_Id.Text));
446     cmd.Parameters.AddWithValue("@ProductName", TxtProduct_Name.Text);
447     cmd.Parameters.AddWithValue("@Categorys_id",int.Parse(txtCategry_id.Text));
448     cmd.Parameters.AddWithValue("@Barcode", TxtBarcode.Text);
449     cmd.Parameters.AddWithValue("@QTY", TxtQuantity.Text);
450     cmd.Parameters.AddWithValue("@unitPrice_in",float.Parse(txtPrice_in.Text));
451     cmd.Parameters.AddWithValue("@Date_up", DateTime.Parse(txtdate_time.Text));
452       //==============photo
453     MemoryStream ms = new MemoryStream();
454     pic_Product.Image.Save(ms, ImageFormat.Png);
455     byte[] photo = new byte[ms.Length];
456      ms.Position = 0;
457      ms.Read(photo, 0, photo.Length);
458      cmd.Parameters.AddWithValue("@Images", photo);
459      cmd.ExecuteNonQuery();
460
461    }
462
```

```
463     }
464
465
466 //==========Method: Select_Product_Users for user Admin =========ទំនិញ==========;
467
468 private void Select_Product_Users()
469 {
470     db_Connection con = new db_Connection();
471     con.Conntion_db();
472     SqlCommand cmd = new SqlCommand();
473     cmd.Connection = db_Connection.conn;
474     // int mark = 1;
475     cmd = new SqlCommand(@"SELECT *FROM  Products_Select
476                     WHERE(user_id=@userid AND Date_at=@Date_at)",
477                         db_Connection.conn);
478
479     cmd.Parameters.AddWithValue("@Date_at", today_work);
480     cmd.Parameters.AddWithValue("@userid", int.Parse(txtUser_Id.Text));
481     cmd.ExecuteNonQuery();
482
483     SqlDataAdapter da = new SqlDataAdapter(cmd);
484     DataTable dt = new DataTable();
485     da.Fill(dt);
486     dataGridView1.DataSource = dt;
487 }
488
489 //=======Method: Delete_Product ----------ទំនិញ==========;
490 private void Delete_Product()
491 {
492     db_Connection con = new db_Connection();
493     con.Conntion_db();
494     SqlCommand cmd = new SqlCommand();
495     cmd.Connection = db_Connection.conn;
496     // int mark = 1;
497     cmd = new SqlCommand(@"DELETE FROM  Products
498               WHERE(Product_id=@Product_id)", db_Connection.conn);
499
500     cmd.Parameters.AddWithValue("@Product_id", txtProduct_id.Text);
501
502     //===============photo
503     MemoryStream ms = new MemoryStream();
504     pic_Product.Image.Save(ms, ImageFormat.Png);
505     byte[] photo = new byte[ms.Length];
506     ms.Position = 0;
507     ms.Read(photo, 0, photo.Length);
508     cmd.ExecuteNonQuery();
509
510     SqlDataAdapter da = new SqlDataAdapter(cmd);
511     DataTable dt = new DataTable();
512     da.Fill(dt);
513     dataGridView1.DataSource = dt;
514
515     this.Enabled = false;
516     var F_SMS = new F_Messages();
517     F_SMS.txtgetMessage.Text = "លោកអ្នកបានលុបទំនិញ ចេញពីប្រព័ន្ធគ្រប់គ្រង ជោគជ័យ..!";
518     F_SMS.Closed += (s, args) => this.Enabled = true;
519
```

```
520     F_SMS.Show();
521 }
522
523 //=======Method: Search_Proname----------ទំនិញ==========;
524
525 private void Search_Proname()
526 {
527     if (txtSearch_proName.Text == "")
528     {
529         txtSearch_proName.Focus();
530         this.Enabled = false;
531         var F_SMS = new F_Messages();
532         F_SMS.txtgetMessage.Text = "សូមបញ្ចូល Barcodes ដើម្បីស្វែងរក..!";
533         F_SMS.Closed += (s, args) => this.Enabled = true;
534         F_SMS.Show();
535         txtSearch_proName.Focus();
536     }
537     else
538     {
539
540       db_Connection con = new db_Connection();
541       con.Conntion_db();
542       SqlCommand cmd = new SqlCommand();
543       cmd.Connection = db_Connection.conn;
544
545       cmd = new SqlCommand(@"SELECT * FROM  tbAddNew_Products
546                           WHERE BarCode=@BarCode", db_Connection.conn);
547
548       cmd.Parameters.AddWithValue("@BarCode", txtSearch_proName.Text);
549       cmd.ExecuteNonQuery();
550
551       SqlDataAdapter da = new SqlDataAdapter(cmd);
552       DataTable dt = new DataTable();
553       da.Fill(dt);
554       dataGridView1.DataSource = dt;
555
556    }
557  }
558
559 //=======Method: ShowData_Product s ----------ទំនិញ==========;
560
561 private void ShowData_Products()
562 {
563     db_Connection con = new db_Connection();
564     con.Conntion_db();
565     SqlCommand cmd = new SqlCommand();
566     cmd.Connection = db_Connection.conn;
567
568     cmd = new SqlCommand(@"SELECT * FROM  tbAddNew_Products
569                         WHERE BarCode=@BarCode", db_Connection.conn);
570
571     cmd.Parameters.AddWithValue("@BarCode", txtBarcode_pro.Text);
572     cmd.ExecuteNonQuery();
573     SqlDataAdapter da = new SqlDataAdapter(cmd);
574     DataTable dt = new DataTable();
575     da.Fill(dt);
576     classProductBindingSource.DataSource = dt;
```

```
577
578  }
579 //=======Method: ShowAll_Products ----------ទំនិញ==========;
580 private void ShowAll_Products()
581  {
582     db_Connection con = new db_Connection();
583     con.Conntion_db();
584     SqlCommand cmd = new SqlCommand();
585     cmd.Connection = db_Connection.conn;
586
587     cmd = new SqlCommand(@"SELECT * FROM  Products_Select
588                         ORDER BY ProductName ASC", db_Connection.conn);
589     cmd.ExecuteNonQuery();
590
591     SqlDataAdapter da = new SqlDataAdapter(cmd);
592     DataTable dt = new DataTable();
593     da.Fill(dt);
594     dataGridView1.DataSource = dt;
595
596 }
597
598 //=======Method: Call_Product_id ----------ទំនិញ==========;
599 private void Call_Product_id()
600 {
601     db_Connection con = new db_Connection();
602     con.Conntion_db();
603     SqlCommand cmd = new SqlCommand();
604     cmd.Connection = db_Connection.conn;
605     cmd = new SqlCommand(@"SELECT Categorys_id,Product_id, BarCode,
606                          ProductName,Images_item FROM Products
607                   WHERE ProductName=@ProductName", db_Connection.conn);
608
609     cmd.Parameters.AddWithValue("@ProductName", TxtProduct_Name.Text);
610     cmd.ExecuteNonQuery();
611
612     SqlDataReader row;
613     row = cmd.ExecuteReader();
614     if (row.Read())
615     {
616         string ProductName = (String)row["ProductName"].ToString();
617         string Categorys_id = (String)row["Categorys_id"].ToString();
618         string BarCode = (String)row["BarCode"].ToString();
619         string Product_id = (String)row["Product_id"].ToString();
620
621         TxtBarcode.Text = BarCode.ToString();
622         txtProduct_id.Text = Product_id.ToString();
623         txtCategry_id.Text = Categorys_id.ToString();
624         TxtProduct_Name.Text = ProductName.ToString();
625         txtBarcode_pro.Text = BarCode.ToString();
626         //     get photo to picbox
627         Byte[] Photo;
628         Photo = (Byte[])row["Images_item"];
629         Photo = Photo != null ? Photo : null;
630         var ms = new MemoryStream(Photo);
631         this.pic_Product.Image = Image.FromStream(ms);
632
633
```

```
634     }
635
636     db_Connection.conn.Close();
637 }
638
639 public static Image resizeImage(Image imgToResize, Size size)
640  {
641     return (Image)(new Bitmap(imgToResize, size));
642  }
643
644 //=======Method: Browser រូបភាព======ទំនិញ==========;
645
646
647 private void btBrowser_Pic_LinkClicked(object sender, LinkLabelLinkCli
648 ckedEventArgs e)
649 {
650     OpenFileDialog open = new OpenFileDialog();
651     open.Filter = "Image Files(*.jpg;*.*;*.Png;*.png; *.jpeg;
652                    *.gif;*.jfif, *.bmp)|*.Png;*.jfif,*.png;*.jpg;
653                    *.jpeg; *.gif; *.bmp";
654
655   if (open.ShowDialog() == DialogResult.OK)
656   {
657     pic_Product.Text = open.FileName;
658     pic_Product.Image = Image.FromFile(open.FileName);
659     pic_Product.Image = resizeImage(pic_Product.Image, new Size(200, 220));
660     }
661 }
662
663 //===========ពេលវេលា ==========ទំនិញ==========;
664 private void timer1_Tick(object sender, EventArgs e)
665 {
666     DateTime RunTime = DateTime.Now;
667     this.txtdate_time.Text = RunTime.ToString();
668 }
669
670
671 //=======Method: Company ( ) ----------ទំនិញ==========;
672 private void Company()
673 {
674     db_Connection con = new db_Connection();
675     con.Conntion_db();
676     SqlCommand cmd = new SqlCommand();
677     cmd = new SqlCommand(@"SELECT Supplyeer_id,Supplyeer_Name
678                           FROM tbSupplyeer ORDER BY Supplyeer_Name ASC",
679                            db_Connection.conn);
680     cmd.ExecuteNonQuery();
681
682     SqlDataReader row;
683     row = cmd.ExecuteReader();
684
685     txtSupply_Name.Items.Clear();
686     while (row.Read())
687     {
688       string Company_Name = (String)row["Supplyeer_Name"].ToString();
689       int ComBrand_id =int.Parse((row["Supplyeer_id"].ToString()).ToString());
690       txtSupply_Name.Items.Add(Company_Name.ToString());
```

```
691         }
692     }
693
694
695     //==========Method: Categories ( ) ==========ទំនិញ==========;
696
697     private void Categories()
698     {
699         db_Connection con = new db_Connection();
700         con.Conntion_db();
701         SqlCommand cmd = new SqlCommand();
702         cmd = new SqlCommand(@"SELECT Cate_id,Category_Name FROM Categories
703                                ORDER BY Category_Name ASC", db_Connection.conn);
704
705         cmd.Parameters.AddWithValue("@Category_Name", TxtCategory_Name.Text);
706         cmd.ExecuteNonQuery();
707
708         SqlDataReader row;
709         row = cmd.ExecuteReader();
710
711         TxtCategory_Name.Items.Clear();
712         while (row.Read())
713         {
714             string ProductName = (String)row["Category_Name"].ToString();
715             int Cate_id = int.Parse((row["Cate_id"].ToString()).ToString());
716             TxtCategory_Name.Items.Add(ProductName.ToString());
717         }
718       }
719
720     //=======Method: Call_ProductName ( ) ----------ទំនិញ==========;
721     private void Call_ProductName()
722       {
723
724         db_Connection con = new db_Connection();
725         con.Conntion_db();
726         SqlCommand cmd = new SqlCommand();
727         cmd.Connection = db_Connection.conn;
728         cmd = new SqlCommand(@"SELECT ProductName,UnitInStock,UnitOnOrder
729                                FROM Products
730                                WHERE Categorys_id=@Categorys_id",
731                                      db_Connection.conn);
732
733         cmd.Parameters.AddWithValue("@Categorys_id", txtCategry_id.Text);
734         cmd.ExecuteNonQuery();
735         TxtProduct_Name.Text = "";
736         SqlDataReader row;
737         row = cmd.ExecuteReader();
738         while (row.Read())
739         {
740             string ProductName = (String)row["ProductName"].ToString();
741         }
742     }
743     //=======Method: Categories_id( ) ----------ទំនិញ==========;
744     private void Categories_id()
745     {
746         db_Connection con = new db_Connection();
747
```

```
748     con.Conntion_db();
749     SqlCommand cmd = new SqlCommand();
750     cmd = new SqlCommand(@"SELECT Cate_id,Category_Name FROM Categories
751                         WHERE Category_Name=@Category_Name ORDER BY
752                               Category_Name ASC", db_Connection.conn);
753
754     cmd.Parameters.AddWithValue("@Category_Name", TxtCategory_Name.Text);
755     cmd.ExecuteNonQuery();
756
757     SqlDataReader row;
758     row = cmd.ExecuteReader();
759     while (row.Read())
760     {
761         int Cate_id = int.Parse((row["Cate_id"].ToString()).ToString());
762         txtCategry_id.Text = Cate_id.ToString();
763     }
764 }
765
766 //==========Form Load -==========ទំនិញ==========;
767
768 private void F_AddNewProduct_Load(object sender, EventArgs e)
769  {
770     txtUser_Id.Text = F_LogIn.userId;
771     txtUser_Id.Hide();
772     txtdate_time.Hide();
773
774     timer1.Start();
775     Categories();
776     Company();
777     classProductBindingSource.DataSource = new List<Class_Product>();
778  }
779
780 //========== ប៊ូតុង ពង្រីក Pic_Max-==========ទំនិញ==========;
781 private void Pic_Max_Click(object sender, EventArgs e)
782 {
783     //  base.WindowState = FormWindowState.Maximized;
784     this.Pic_Max.Visible = false;
785     this.Pic_maxi.Visible = true;
786 }
787
788
789 //========== ប៊ូតុង ពង្រួម Pic_Min-==========ទំនិញ==========;
790 private void Pic_maxi_Click(object sender, EventArgs e)
791  {
792     // base.WindowState = FormWindowState.Normal;
793     this.Pic_Max.Visible = true;
794     this.Pic_maxi.Visible = false;
795  }
796
797 //========== ជ្រើសប្រភេទ Categories ==========ទំនិញ==========;
798 private void TxtCategory_Name_Click(object sender, EventArgs e)
799   {
800     Categories();
801   }
802
803
804 //========== ប៊ូតុង លុប ==========ទំនិញ==========;
```

```
805 private void btdelete_pro_Click(object sender, EventArgs e)
806 {
807     Delete_Product();
808     Select_Product_Users();
809 }
810
811 //========== ជ្រើសរើស ជួរដេក datagridView1 =========ទំនិញ==========;
812
813 public void Play_DataGrid()
814 {
815 try
816 {
817
818 int rowindex = dataGridView1.CurrentRow.Index;
819
820 if (dataGridView1.Rows[rowindex].Cells[0].Value == null)
821 {
822     return;
823 }
824 else
825 {
826
827  if (dataGridView1.SelectedRows.Count == 0)
828   {
829      return;
830   }
831  else
832  {
833
834   txtProduct_id.Text=
835 dataGridView1.Rows[rowindex].Cells[0].Value.ToString();
836   txtBarcode_pro.Text=dataGridView1.Rows[rowindex].Cells[3].Value.ToString();
837   txtCategry_id.Text=dataGridView1.Rows[rowindex].Cells[2].Value.ToString();
838   TxtProduct_Name.Text=dataGridView1.Rows[rowindex].Cells[4].Value.ToString();
839   TxtQuantity.Text=dataGridView1.Rows[rowindex].Cells[6].Value.ToString();
840   txtPrice_in.Text=double.Parse(dataGridView1.Rows[rowindex]
841                   .Cells[7].Value.ToString()).ToString("N");
842    TxtPrices_out.Text=double.Parse(dataGridView1.Rows[rowindex]
843                         .Cells[8].Value.ToString()).ToString("N");
844    TxtBarcode.Text=dataGridView1.Rows[rowindex].Cells[3].Value.ToString()
845 ;
846
847  //display images from dataGriview to picturebox
848   if (dataGridView1.Rows[rowindex].Cells[16].Value.ToString() == "")
849      {
850             this.pic_Product.Image = null;
851             MessageBox.Show("hello");
852             return;
853      }
854      else
855      {
856         byte[] Photobox1;
857         Photobox1=(byte[])this.dataGridView1.Rows[rowindex].Cells[16].Value;
858          var ms = new MemoryStream(Photobox1);
859          this.pic_Product.Image = Image.FromStream(ms);
860       }
861
      }
```

```
862
863         }
864       }
865      catch(Exception)
866       {
867       }
868     }
869 //========== ប៊ូតុង ព្រីនបញ្ជី =========ទំនិញ==========;
870
871 private void Print_Report_Product()
872 {
873  if (dataGridView1.Rows[0].Cells[0].Value == null)
874   {
875     return;
876   }
877  else
878   if(dataGridView1.Rows.Count-1==0)
879   {
880     return;
881   }
882  else
883   {
884     Class_Product obj;
885     classProductBindingSource.Clear();
886     int i = dataGridView1.CurrentRow.Index;
887
888     for (i = 0; i < dataGridView1.Rows.Count - 1; i++)
889     {
890       obj = new Class_Product();
891
892       obj.No = (i + 1).ToString();
893       obj.Product_id = dataGridView1.Rows[i].Cells[0].Value.ToString();
894       obj.ProductName = dataGridView1.Rows[i].Cells[2].Value.ToString();
895       classProductBindingSource.Add(obj.Product_id);
896     }
897     //  classProductBindingSource.MoveLast();
898     using (rfmReport_AddProdeuct Frm = new rfmReport_AddProdeuct
899           (classProductBindingSource.DataSource as List<Class_Product>))
900       {
901         Frm.ShowDialog();
902        }
903      }
904   }
905
906 //========== ចុចលើ datagridView1 =========ទំនិញ==========;
907 private void dataGridView1_Click(object sender, EventArgs e)
908 {
909     Play_DataGrid();
910  }
911
912 //========== ជ្រើសរើស ជួរដេក datagridView1 =========ទំនិញ==========;
913 public void Search_Products()
914 {
915       db_Connection con = new db_Connection();
916       con.Conntion_db();
917       SqlCommand cmd = new SqlCommand();
918
```

```
919         cmd.Connection = db_Connection.conn;
920         cmd = new SqlCommand(@"SELECT COUNT(*)
921                                FROM Products
922                                WHERE Product_id=@Product_id  AND
923         BarCode=@BarCode", db_Connection.conn);
924         cmd.Parameters.AddWithValue("@Product_id", txtProduct_id.Text);
925          AND BarCode=@BarCode//
926         cmd.Parameters.AddWithValue("@BarCode", TxtBarcode.Text);
927         cmd.ExecuteNonQuery();
928
929         SqlDataAdapter da = new SqlDataAdapter(cmd);
930         DataTable dt = new DataTable();
931         da.Fill(dt);
932
933      if (dt.Rows[0][0].ToString() == "1")
934       {
935        var F_SMS = new F_Mesager_Admin();
936        F_SMS.txtgetMessage.Text = "តើអ្នកចង់ លុប ទំនិញមែនទេ□   ?";
937        F_SMS.Closed += (s, args) => this.Enabled = true;
938        F_SMS.Delete_pro = "F_AddNew_Product";
939        F_SMS.Delete_Product_id = txtProduct_id.Text;
940        F_SMS.Show();
941       }
942      else
943       {
944        this.Enabled = false;
945        var F_SMS = new F_Messages();
946        F_SMS.txtgetMessage.Text ="សូមពិនិត្យលេខ Barcode និងលេខសម្គាល់ ទំនិញមិនត្រឹមត្រូវទេ !";
947
948        F_SMS.Closed += (s, args) => this.Enabled = true;
949        F_SMS.Show();
950       }
951      }
952
953     //========== ប៊ូតុង លុប =========ទំនិញ==========;
954    private void btdelete_pro_Click_1(object sender, EventArgs e)
955    {
956        if (txtBarcode_pro.Text == "")
957        {
958            txtBarcode_pro.Focus();
959            this.Enabled = false;
960            var F_SMS = new F_Messages();
961
962           F_SMS.txtgetMessage.Text = "សូមបញ្ចូល លេខ៖ Barcodes ទំនិញ.ដើម្បីលុប.!";
963            F_SMS.Closed += (s, args) => this.Enabled = true;
964            F_SMS.Show();
965            return;
966        }
967        else if (txtProduct_id.Text == "")
968        {
969            txtProduct_id.Focus();
970            this.Enabled = false;
971            var F_SMS = new F_Messages();
972           F_SMS.txtgetMessage.Text = "សូមបញ្ចូល អត្តលេខ Products Code ទំនិញ ដើម្បីលុប..!";
973
974            F_SMS.Closed += (s, args) => this.Enabled = true;
975            F_SMS.Show();
            return;
```

```
976         }
977         else
978         {
979             Search_Products();
980         }
981 
982     }
983     //========== ប៊ូតុង ស្វែងរក =========ទំនិញ==========;
984     private void btsearch_prod_Click(object sender, EventArgs e)
985     {
986         Search_Proname();
987     }
988     //========== ជ្រើសរើស ស្វែងរក =========ទំនិញ==========;
989     private void txtSearch_proName_TextChanged(object sender, EventArgs e)
990     {
991         Search_Proname();
992     }
993     //========== ប៊ូតុង កែតម្រូវ =========ទំនិញ==========;
994     private void btEdite_Click(object sender, EventArgs e)
995     {
996         if (txtSupply_Name.Text == "")
997         {
998             txtSupply_Name.Focus();
999             this.Enabled = false;
1000            var F_SMS = new F_Messages();
1001            F_SMS.txtgetMessage.Text = "សូមបញ្ចូលក្រុមហ៊ុន អ្នកផ្គត់ផ្គង់..!";
1002            F_SMS.Closed += (s, args) => this.Enabled = true;
1003            F_SMS.Show();
1004            txtSupply_Name.Focus();
1005            return;
1006        }
1007        else if (txtCategry_id.Text == "" && TxtCategory_Name.Text == "")
1008        {
1009            txtCategry_id.Focus();
1010            TxtCategory_Name.Focus();
1011            this.Enabled = false;
1012            var F_SMS = new F_Messages();
1013            F_SMS.txtgetMessage.Text = "សូមបញ្ចូល Categories ប្រភេទមុខទំនិញ..!";
1014            F_SMS.Closed += (s, args) => this.Enabled = true;
1015            F_SMS.Show();
1016            return;
1017        }
1018        else if (txtTypegoods_Name.Text == "")
1019        {
1020            txtTypegoods_Name.Focus();
1021            this.Enabled = false;
1022            var F_SMS = new F_Messages();
1023            F_SMS.txtgetMessage.Text = "សូមបញ្ចូល ទ្រង់ទ្រាយ Type_Name ប្រភេទទំនិញ..!";
1024            F_SMS.Closed += (s, args) => this.Enabled = true;
1025            F_SMS.Show();
1026            return;
1027        }
1028        else if (txtBarcode_pro.Text == "")
1029        {
```

```
1033            txtBarcode_pro.Focus();
1034            this.Enabled = false;
1035            var F_SMS = new F_Messages();
1036            F_SMS.txtgetMessage.Text = "សូមបញ្ចូល លេខ៖ Barcodes ទំនិញ..!";
1037            F_SMS.Closed += (s, args) => this.Enabled = true;
1038            F_SMS.Show();
1039            return;
1040        }
1041        else if (TxtProduct_Name.Text == "")
1042        {
1043            TxtProduct_Name.Focus();
1044            this.Enabled = false;
1045            var F_SMS = new F_Messages();
1046            F_SMS.txtgetMessage.Text = "សូមបញ្ចូល ប្រភេទទំនិញ..!";
1047            F_SMS.Closed += (s, args) => this.Enabled = true;
1048            F_SMS.Show();
1049            return;
1050        }
1051        else if (TxtQuantity.Text == "")
1052        {
1053            TxtQuantity.Focus();
1054            this.Enabled = false;
1055            var F_SMS = new F_Messages();
1056            F_SMS.txtgetMessage.Text = "សូមបញ្ចូល បរិមាណទំនិញ..!";
1057            F_SMS.Closed += (s, args) => this.Enabled = true;
1058            F_SMS.Show();
1059            return;
1060        }
1061        else if (txtPrice_in.Text == "")
1062        {
1063            txtPrice_in.Focus();
1064            this.Enabled = false;
1065            var F_SMS = new F_Messages();
1066            F_SMS.txtgetMessage.Text = "សូមបញ្ចូល តម្លៃទំនិញ..!";
1067            F_SMS.Closed += (s, args) => this.Enabled = true;
1068            F_SMS.Show();
1069            return;
1070
1071        }
1072        else if (TxtPrices_out.Text == "")
1073        {
1074            TxtPrices_out.Focus();
1075            var F_SMS = new F_Messages();
1076            F_SMS.txtgetMessage.Text = "សូមបញ្ចូល តម្លៃទំនិញលក់ចេញ..!";
1077            F_SMS.Closed += (s, args) => this.Enabled = true;
1078            F_SMS.Show();
1079            return;
1080        }
1081        else if (txtUnitIn_Stock.Text == "")
1082        {
1083            txtUnitIn_Stock.Focus();
1084            var F_SMS = new F_Messages();
1085            F_SMS.txtgetMessage.Text = "សូមបញ្ចូល បរិមាណទំនិញស្តុក..!";
1086            F_SMS.Closed += (s, args) => this.Enabled = true;
1087
1088
1089
```

```
1090            F_SMS.Show();
1091            return;
1092        }
1093        else if (TxtUnit_Order.Text == "")
1094        {
1095            TxtUnit_Order.Focus();
1096        }
1097        else if (TxtDiscont_sel.Text == "")
1098        {
1099            TxtDiscont_sel.Focus();
1100            var F_SMS = new F_Messages();
1101            F_SMS.txtgetMessage.Text = "សូមបញ្ចូល បញ្ចុះតម្លៃទៅមិញ..!";
1102            F_SMS.Closed += (s, args) => this.Enabled = true;
1103            F_SMS.Show();
1104            return;
1105        }
1106        else if (txtdate_time.Text == "")
1107        {
1108            txtdate_time.Focus();
1109            return;
1110        }
1111        else if (pic_Product.Image == null)
1112        {
1113            var F_SMS = new F_Messages();
1114            F_SMS.txtgetMessage.Text = "សូមបញ្ចូល រូបភាព សម្គាល់របស់ទំនិញ..!";
1115            F_SMS.Closed += (s, args) => this.Enabled = true;
1116            F_SMS.Show();
1117            return;
1118        }
1119        else
1120        {
1121            Updated_Product_Admin();
1122            Select_Product_Users();
1123        }
1124
1125    }
1126
1127    //========== ព្រីនបញ្ជីទំនិញ =========ទំនិញ==========;
1128
1129    private void btPrint_List_Product_Click(object sender, EventArgs e)
1130     {
1131    // Print_Report_Product();
1132     if (dataGridView1.Rows.Count - 1 == 0)
1133      {
1134        this.Enabled = false;
1135        var F_SMS = new F_Messages();
1136        F_SMS.txtgetMessage.Text = "មិនមានទិន្នន័យ ទេ! សូមធ្វើការទាញទិន្នន័យជាមុន !";
1137        F_SMS.Closed += (s, args) => this.Enabled = true;
1138        F_SMS.Show();
1139        txtProduct_id.Focus();
1140        return;
1141      }
1142     else
1143      {
1144        Class_Product obj;
1145
1146
```

```
1147      classProductBindingSource.Clear();
1148      int i = dataGridView1.CurrentRow.Index;
1149
1150      for (i = 0; i < dataGridView1.Rows.Count - 1; i++)
1151      {
1152        obj = new Class_Product();
1153        obj.No = (i + 1).ToString();
1154        obj.Product_id = dataGridView1.Rows[i].Cells[0].Value.ToString();
1155        obj.BarCode = (dataGridView1.Rows[i].Cells[3].Value.ToString());
1156        obj.Categorys_id = dataGridView1.Rows[i].Cells[2].Value.ToString();
1157        obj.Images_item = (byte[])dataGridView1.Rows[i].Cells[16].Value;
1158        obj.Product_Types=(dataGridView1.Rows[i].Cells[5].Value.ToString())
1159 ;
1160        obj.UnitPrice_in =dataGridView1.Rows[i].Cells[7].Value.ToString();
1161        obj.UnitPrice_Out=dataGridView1.Rows[i].Cells[8].Value.ToString();
1162        obj.UnitInStock = dataGridView1.Rows[i].Cells[9].Value.ToString();
1163        obj.UnitOnOrder = dataGridView1.Rows[i].Cells[10].Value.ToString();
1164        obj.Discont = dataGridView1.Rows[i].Cells[11].Value.ToString();
1165        obj.DiscontDate=DateTime.Parse(dataGridView1.Rows[i].Cells[12].
1166                                       Value.ToString()).ToShortDateString();
1167          classProductBindingSource.Add(obj);
1168      }
1169
1170    using (rfmReport_AddProdeuct fm = new rfmReport_AddProdeuct
1171           (classProductBindingSource.DataSource as List<Class_Product>))
1172      {
1173        fm.ShowDialog();
1174        fm.Show();
1175      }
1176    }
1177  }
1178
1179 private void dataGridView1_SelectionChanged(object sender, EventArgs e)
1180 {
1181     Play_DataGrid();
1182 }
1183
1184 private void Clear_text_Box()
1185 {
1186     txtCategry_id.Clear();
1187     txtProduct_id.Clear();
1188     TxtProduct_Name.Text = "";
1189     txtCategry_id.Text = "";
1190     pic_Product.Image = null;
1191
1192     txtCategry_id.Clear();
1193     TxtProduct_Name.Text = "";
1194     txtProduct_id.Clear();
1195     TxtCategory_Name.Text = "";
1196     txtSupply_Id.Clear();
1197     txtTypeGoods_Id.Clear();
1198     txtSupply_Name.Text = "";
1199     txtTypegoods_Name.Text = "";
1200     txtProduct_id.Clear();
1201
1202 }
1203 //==========ស្តែនបាកូដ Scan_Barcode =========ទំនិញ==========;
```

```
1204 private void Scan_Barcode()
1205 {
1206     if (txtBarcode_pro.Text == "")
1207     {
1208         Clear_text_Box();
1209         return;
1210     }
1211     else
1212     {
1213       db_Connection con = new db_Connection();
1214       con.Conntion_db();
1215       SqlCommand cmd = new SqlCommand();
1216       cmd.Connection = db_Connection.conn;
1217       cmd = new SqlCommand(@"SELECT * FROM Products_Select
1218                       WHERE BarCode=@BarCode", db_Connection.conn);
1219       cmd.Parameters.AddWithValue("@BarCode", txtBarcode_pro.Text);
1220       cmd.ExecuteNonQuery();
1221       SqlDataReader row;
1222       row = cmd.ExecuteReader();
1223     while (row.Read())
1224       {
1225         string ProductName = (String)row["ProductName"].ToString();
1226         string Categorys_id = (String)row["Categorys_id"].ToString();
1227         string Product_id = (String)row["Product_id"].ToString();
1228         string Product_Types = (String)row["Product_Types"].ToString();
1229         string Category_Name = (String)row["Category_Name"].ToString();
1230         string Supplyeer_id = (String)row["Supplyeer_id"].ToString();
1231         string ProductTypes_id=(String)row["ProductTypes_id"].ToString();
1232         string Supplyeer_Name = (String)row["Supplyeer_Name"].ToString();
1233
1234         TxtBarcode.Text = txtBarcode_pro.Text;
1235         txtCategry_id.Text = Categorys_id.ToString();
1236         TxtProduct_Name.Text = ProductName.ToString();
1237         txtProduct_id.Text = Product_id.ToString();
1238         TxtCategory_Name.Text = Category_Name.ToString();
1239         txtSupply_Id.Text = Supplyeer_id.ToString();
1240         txtTypeGoods_Id.Text = ProductTypes_id.ToString();
1241         txtSupply_Name.Text = Supplyeer_Name.ToString();
1242         txtTypegoods_Name.Text = Product_Types.ToString();
1243         txtProduct_id.Text = Product_id.ToString();
1244             //     get photo to picbox
1245             Byte[] Photo;
1246             Photo = (Byte[])row["Images_item"];
1247             Photo = Photo != null ? Photo : null;
1248             var ms = new MemoryStream(Photo);
1249             this.pic_Product.Image = Image.FromStream(ms);
1250         }
1251
1252         db_Connection.conn.Close();
1253     }
1254  }
1255
1256 //========== Method: Goods_Types =========ទំនិញ==========;
1257
1258 private void Goods_Types_All()
1259 {
1260     db_Connection con = new db_Connection();
     con.Conntion_db();
```

```
    SqlCommand cmd = new SqlCommand();
    cmd = new SqlCommand(@"SELECT * FROM Products_Types
             WHERE Category_id=@Category_id
             ORDER BY Product_Types ASC", db_Connection.conn);

    cmd.Parameters.AddWithValue("@Category_id", txtCategry_id.Text);
    cmd.ExecuteNonQuery();

    SqlDataReader row;
    row = cmd.ExecuteReader();
    txtTypegoods_Name.Items.Clear();

    while (row.Read())
    {
        string Good_TypeName = (String)row["Product_Types"].ToString();
        txtTypegoods_Name.Items.Add(Good_TypeName.ToString());
    }
}
//========== Method: Goods_Types_id ( )=========ទំនិញ==========;
private void Goods_Types_id()
{
    db_Connection con = new db_Connection();
    con.Conntion_db();
    SqlCommand cmd = new SqlCommand();
    cmd = new SqlCommand(@"SELECT * FROM tbType
       WHERE Product_Types=@Product_Type  ORDER BY Product_Types ASC",
            db_Connection.conn);

  cmd.Parameters.AddWithValue("@Product_Type", txtTypegoods_Name.Text);
  cmd.ExecuteNonQuery();

    SqlDataReader row;
    row = cmd.ExecuteReader();

    while (row.Read())
    {
      int ProType_id = int.Parse((row["ProType_id"].ToString()).ToString());
      txtTypeGoods_Id.Text = ProType_id.ToString();

    }
}
//========== ជ្រើសរើស Combobox: Gategories_Name=========ទំនិញ==========;
private void TxtCategory_Name_SelectedIndexChanged(object sender, Even
tArgs e)
{
    Categories_id();
    Goods_Types();
}
//========== ជ្រើសរើស Combobox:  txtBarcode_Name =========ទំនិញ==========;
private void txtBarcode_pro_TextChanged(object sender, EventArgs e)
{
    Scan_Barcode();
    TxtBarcode.Text = txtBarcode_pro.Text;
}
//========== ប៊ូតុង សម្អាត =========ទំនិញ==========;
```

```
1318 private void btClear_Items_Click(object sender, EventArgs e)
1319 {
1320     Textclear();
1321 }
1322
1323 //========== Method: Company_id( ) =========ទំនិញ==========;
1324
1325 private void Company_id()
1326 {
1327     db_Connection con = new db_Connection();
1328     con.Conntion_db();
1329     SqlCommand cmd = new SqlCommand();
1330     cmd = new SqlCommand(@"SELECT * FROM tbSupplyeer
1331           WHERE Supplyeer_Name=@Company_Name", db_Connection.conn);
1332
1333     cmd.Parameters.AddWithValue("@Company_Name", txtSupply_Name.Text);
1334     cmd.ExecuteNonQuery();
1335
1336     SqlDataReader row;
1337     row = cmd.ExecuteReader();
1338
1339     while (row.Read())
1340     {
1341         string ComBrand_id = (string)(row["Supplyeer_id"].ToString());
1342         txtSupply_Id.Text = ComBrand_id.ToString();
1343     }
1344  }
1345
1346 //========== Method: Brand( ) =========ទំនិញ==========;
1347 private void Brand()
1348 {
1349     db_Connection con = new db_Connection();
1350     con.Conntion_db();
1351     SqlCommand cmd = new SqlCommand();
1352     cmd = new SqlCommand(@"SELECT Brand_id FROM tbBrand
1353                   WHERE Brand_Name=@Brand_Name", db_Connection.conn);
1354
1355     cmd.Parameters.AddWithValue("@Brand_Name", txtTypegoods_Name.Text);
1356     cmd.ExecuteNonQuery();
1357
1358     SqlDataReader row;
1359     row = cmd.ExecuteReader();
1360
1361     while (row.Read())
1362     {
1363         string Brand_id = (string)(row["Brand_id"].ToString());
1364         txtTypeGoods_Id.Text = Brand_id.ToString();
1365     }
1366 }
1367 //========== Method: Brand_id( ) =========ទំនិញ==========;
1368
1369 private void Brand_id()
1370 {
1371     db_Connection con = new db_Connection();
1372     con.Conntion_db();
1373     SqlCommand cmd = new SqlCommand();
1374     cmd = new SqlCommand(@"SELECT Company_id,Brand_Name FROM tbBrand
```

```
                    WHERE Company_id=@Company_id", db_Connection.conn);

    cmd.Parameters.AddWithValue("@Company_id", txtSupply_Id.Text);
    cmd.ExecuteNonQuery();

    SqlDataReader row;
    row = cmd.ExecuteReader();

    txtTypegoods_Name.Items.Clear();
    while (row.Read())
    {
        string Brand_Name = (string)(row["Brand_Name"].ToString());
        txtTypegoods_Name.Items.Add(Brand_Name.ToString());
    }
 }
//========== ជ្រើសរើស Combobox: txtSupply_Name =========ទំនិញ==========;
private void txtSupply_Name_SelectedIndexChanged(object sender, EventArgs
e)
  {
    Company_id();
  }

//========== ជ្រើសរើស Combobox: txtBrand_Name =========ទំនិញ==========;
private void txtBrand_Name_SelectedIndexChanged(object sender, EventArgs
e)
 {
    Goods_Types_id();
 }
//========== បង្ហាញ Form Company ពេលចុច =========ទំនិញ==========;
private void btSupplyees_Click(object sender, EventArgs e)
{
    this.Enabled = false;
    var f_s = new F_Company();
    f_s.Closed += (s, args) => this.Enabled = true;
    f_s.Show();
}

//========== បិតុង ក្បាទុក btAddNew_Brand =========ទំនិញ==========;
private void btAddNew_Brand_Click(object sender, EventArgs e)
{
    this.Enabled = false;

    var f_s = new F_TypeProducts();
    f_s.Closed += (s, args) => this.Enabled = true;
    f_s.Show();
}

//========== បិតុង ក្បាទុក btAddNew_Gateg =========ទំនិញ==========;
private void btAddnew_Categ_Click(object sender, EventArgs e)
{
    this.Enabled = false;
    var f_s = new F_Categories();
    f_s.Closed += (s, args) => this.Enabled = true;
    f_s.Show();
```

```
1432 }
1433
1434 //========== ប៊ូតុង បញ្ចីទំនិញ =========ទំនិញ==========;
1435 private void btList_Product_Click(object sender, EventArgs e)
1436 {
1437     ShowAll_Products();
1438 }
1439
1440
1441 //========== Method: Goods_Types =========ទំនិញ==========;
1442 private void Goods_Types()
1443 {
1444
1445     db_Connection con = new db_Connection();
1446     con.Conntion_db();
1447     SqlCommand cmd = new SqlCommand();
1448     cmd = new SqlCommand(@"SELECT ProType_id,Product_Types FROM tbType
1449                         ORDER BY Product_Types ASC", db_Connection.conn);
1450
1451     cmd.ExecuteNonQuery();
1452
1453     SqlDataReader row;
1454     row = cmd.ExecuteReader();
1455
1456     txtTypegoods_Name.Items.Clear();
1457     while (row.Read())
1458     {
1459      string Good_TypeName = (String)row["Product_Types"].ToString();
1460      int ProType_id = int.Parse((row["ProType_id"].ToString()).ToString());
1461      txtTypegoods_Name.Items.Add(Good_TypeName.ToString());
1462
1463     }
1464  }
1465
1466 //========== ជ្រើសរើស ទ្រង់ទ្រាយទំនិញ =========ទំនិញ==========;
1467  private void txtTypegoods_Name_Click(object sender, EventArgs e)
1468  {
1469     Goods_Types();
1470  }
1471 //========== ជ្រើសរើស អ្នកផ្គត់ផ្គង់ =========ទំនិញ==========;
1472  private void txtSupply_Name_Click(object sender, EventArgs e)
1473  {
1474     Company();
1475   }
1476 /========== ជ្រើសរើស ប្រភេទទំនិញ =========ទំនិញ==========;
1477  private void TxtCategory_Name_Click_1(object sender, EventArgs e)
1478   {
1479     Categories();
1480   }
1481 }
1482
1483
1484
```

ឯកសារយោង

[1]. Paul, Harvey Deitel, 2012, Visual C# 2012, Deitel & Associates,. Inc, Microsoft® and Windows® , Microsoft Corporation in the U.S.A

[2]. Men Saravuth Msc, 2006, Introduction to Advanced Programming with Visual Basic.NET, Lecture Notes .

[3]. Herbert Schildt , 2009 , Beginners Guide C# 3.0, McGraw-Hill Companies , United States of America

[4]. Jesse Liberty and Donald Xie, 2008, Programming C# 3.0 , O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472 , United States of America

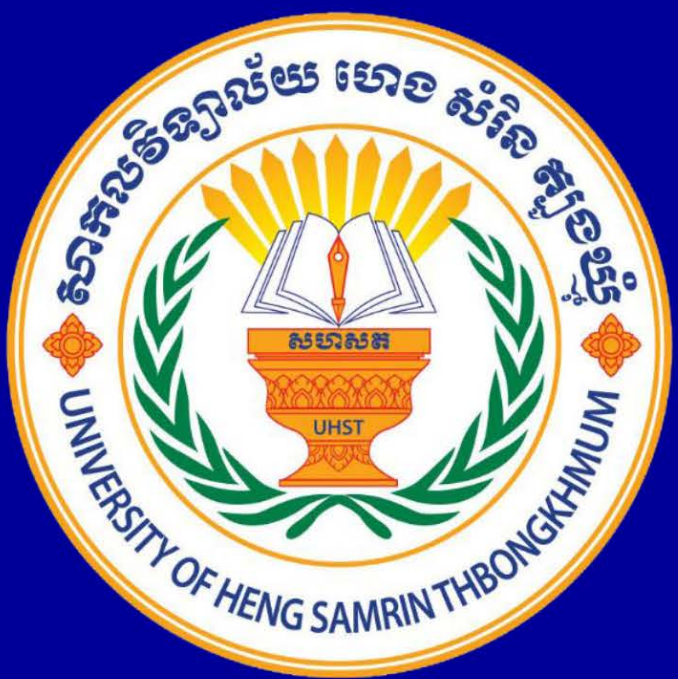
[5]. Jack Purdum,2012, Beginning Object-Oriented Programming with C sharp

[6]. Sam A. Abolrous, 2008, Learn C# Includes the C# 3.0 Features ,Wordware Publishing, Inc. Plano, Texas.

Website

1. https://www.geeksforgeeks.org
2. https://www.tutorialsteacher.com
3. https://www.w3schools.com
4. https://www.programiz.com
5. https://www.decodejava.com

មូលដ្ឋានគ្រឹះភាសា C# នៃសាកលវិទ្យាល័យ ហេង សំរិន ត្បូងឃ្មុំ


សាកលវិទ្យាល័យ ហេង សំរិន ត្បូងឃ្មុំ
UHST
UNIVERSITY OF HENG SAMRIN THBONGKHMUM