

C# Programming Fundamentals Course

PART – I

(မြန်မာ ဘာသာဖြင့် ရေးသားထားသည်)

Author :

Pyae Phyo Maung Maung

B.C (Tech) (Hons :)

MCTS , MCPD .

Web site : <http://myanmaraspnet.multiply.com>

Unconditional ဆိုသည်မှာ

Method နှင့် Function အခေါ်အဝေါ်

Method Call သည် ဆိုသည်မှာ

Method Call ပုံ

Method Call ပုံ နှင့် return Value အကြောင်း

Conditional ဆိုသည်မှာ

If အကြောင်း

Nested If အကြောင်း

Switch အကြောင်း

Looping

goto အကြောင်း

while, do while , for အကြောင်း

foreach အကြောင်း

Unconditional ဆိုသည်မှာ

Unconditional ဆိုသည်မှာ ရိုးရှင်းသော Method Call ပုံ မျိုးကို ဆိုလိုသည် ။ တနည်းအားဖြင့် ထို Method ထဲတွင် မည်သည့် Conditional Statement မှ မပါဝင်သော Method မျိုး ကို ဆိုလိုသည် ။

Method နှင့် Function အခေါ်အဝေါ်

တကယ်တော့ Method နှင့် Function ဆိုတာ အခေါ်အဝေါ် ကွာပေမယ့် သူတို့အလုပ်လုပ်ပုံ ဟာ အတူတူပဲ ဖြစ်ပါတယ် ။ C Programmers များက Function လို့ခေါ်ဝေါ် သုံးစွဲခဲ့တာဖြစ်ပြီး C++ / Java / C# Programmers များကတော့ Method လို့ ခေါ်ဝေါ် အသုံးပြုခဲ့တာ ဖြစ်ပါတယ် ။

Method Call

Method Call သည် ဆိုသည်မှာ Program ၏ တစ်နေရာတွင် ခွဲရေးထားသော Method တစ်ခုကို မိမိခေါ်လိုသော Event , Method စသည်တို့မှ ခေါ်ယူ အသုံးပြုခြင်း ကို ဆိုလိုသည် ။

ဘာကြောင့် Method ခွဲရေးသင့်ပါသလဲ ?

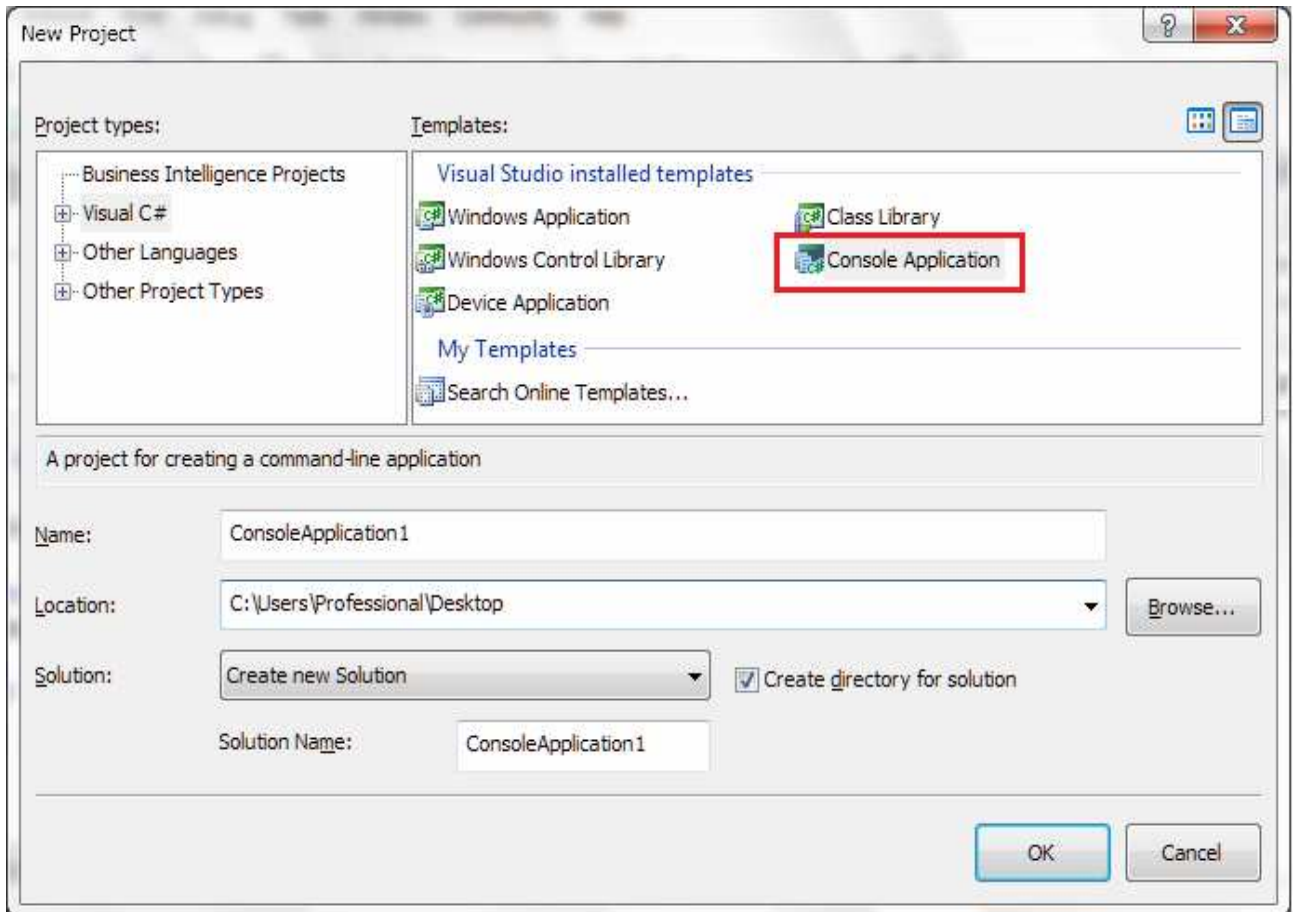
Method တစ်ခုထဲတွင် Coding များ စုပြုံရေးထားခြင်းသည် ရှင်းလင်းမှု မရှိခြင်း ၊ သဘောတရား တူညီသော Coding များကို ထပ်ခါထပ်ခါ ရေးရခြင်း စသော အခက်အခဲများကို ရှောင်ရှားရန် အတွက် Method ကို ခွဲရေး ကြခြင်း ဖြစ်သည် ။

Method Call သည် ဆိုရာဝယ် Method နှစ်မျိုး သာ ရှိသည် ။ Return ပြန်သော Method နှင့် Return မပြန်သော Method တို့ဖြစ်ကြသည် ။

အောက်တွင် Return မပြန်သော Method နှင့် Return ပြန်သော Method တို့ကို ဆက်လက် ဖော်ပြပါမည် ။

Method Call ဝံ

စတင်အသုံးပြုမယ့် သူအားလုံးအတွက် ရည်ရွယ်ပြီး Console Application ကို A သုံးပြုပါမယ် ။



၁ . Visual Studio .Net မှ File \ New Project ကို ရွေးပါ ။

၂ . ထို့နောက် Templates ထဲမှ Console Application ကို ရွေးပါ ။

၃ . Main ထဲတွင် ဒီနှစ်ကြောင်းကိုရေးပါ ။

```
Console.WriteLine("I'm in Main!");  
FunctionOne();
```

၄ . Main ဖော့ဘက်တွင် ဒီလိုရေးပါ ။

```
static void FunctionOne()  
{
```

```

        Console.WriteLine("I'm in functionOne!");
        FunctionTwo();
    }

    static void FunctionTwo()
    {
        Console.WriteLine("I'm in functionTwo!");
    }

```

၅ . အဆင့် ၄ အထိ Programm ရေးပြီးရင် ဒီလို မြင်ရမှာပါ ။

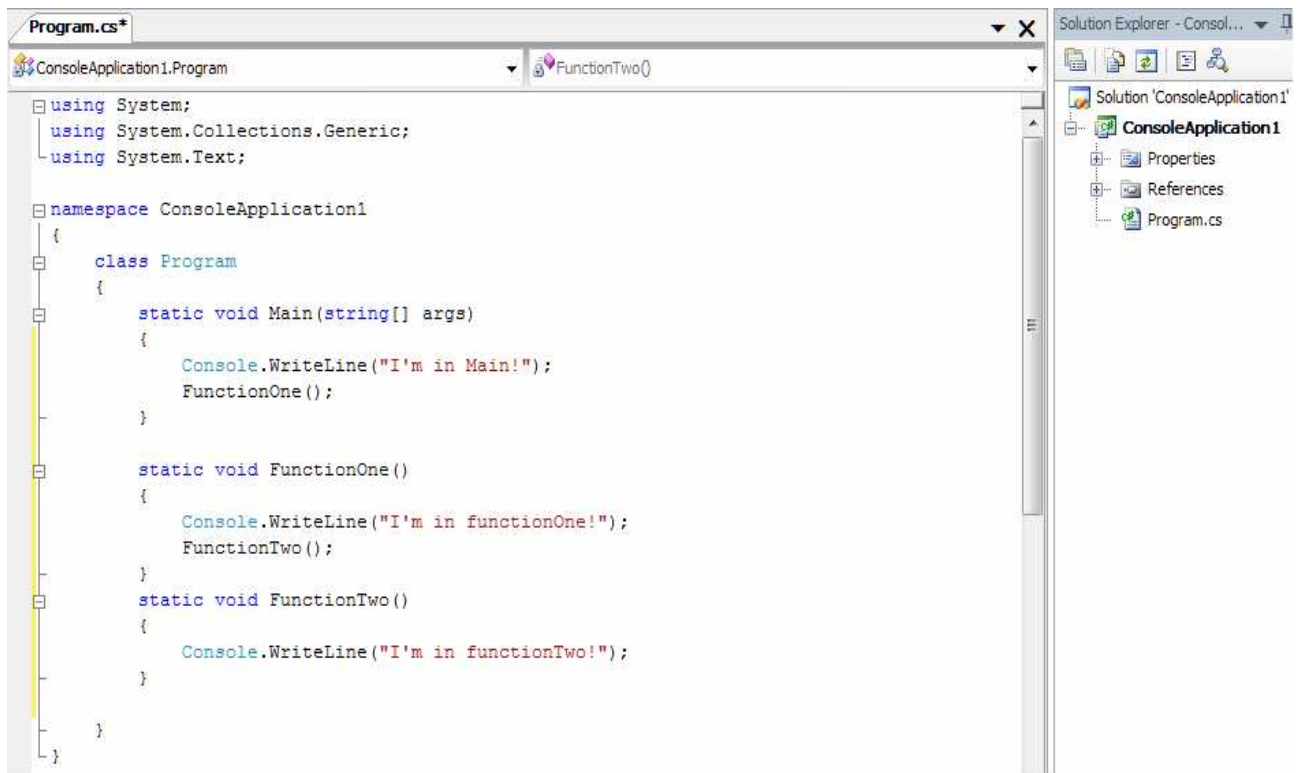


Fig : Program - 1

၆ . ရေးထားတဲ့ Program ကို **SHIFT + CTRL + B** နဲ့ **Compile** လုပ်ပါမယ် ။

၇ . Build succeeded ဆိုတဲ့ စာသားလေးကို Window အောက်ခြေနားမှာ မြင်ရပြီဆိုရင် **Compile** လုပ်တာ အောင်မြင်ပါပြီ ။

၈ . Program Run ဖို့အတွက် CTRL + F5 ကို နှိပ်ပါ ။

၉ . Result က ဒီလို မြင်ရမှာပါ ။

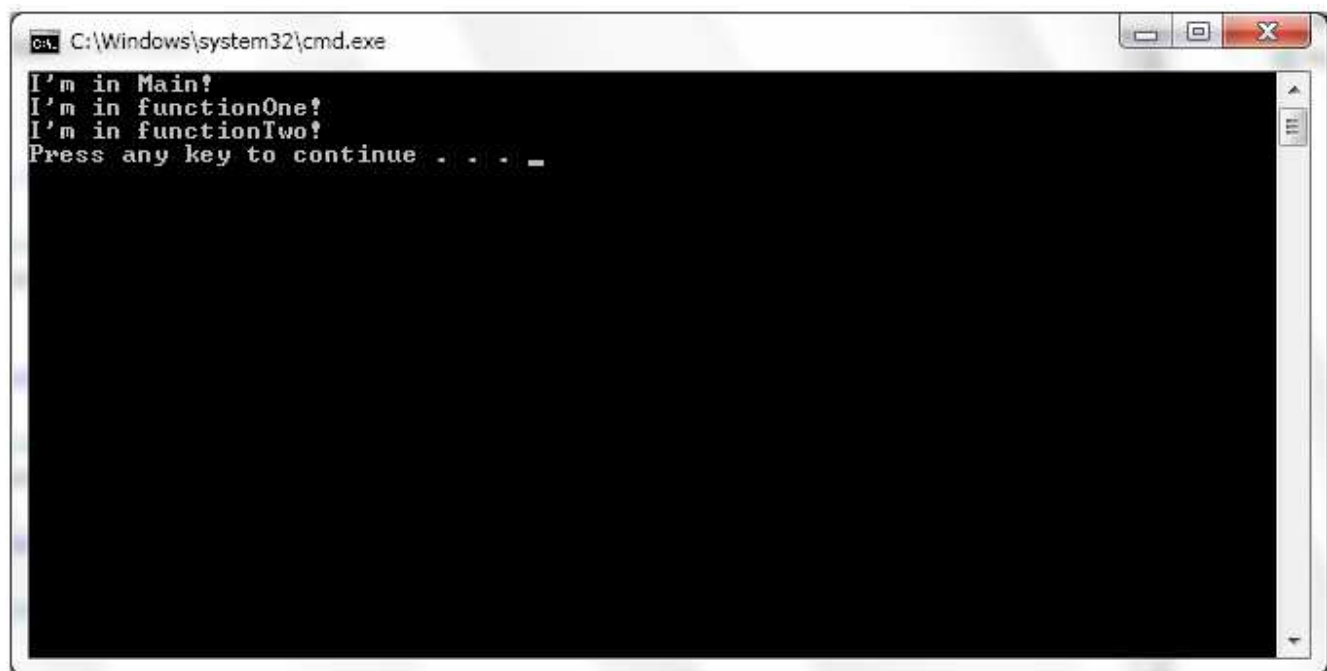


Fig : Program – 1 (Result)

၁၀ . Result ထွက်လာရင် Programmer တစ်ယောက် မဖြစ်မနေ လုပ်ရမယ့် အလုပ်တစ်ခု ကို စတင်လုပ်ရပါမယ် ။ အဲဒါကတော့ **Trace** လိုက်ရမှာပါ ။

၁၁ . Main ထဲမှာ ရှိတဲ့ `Console.WriteLine ("I'm in Main ! ");` ဆိုတဲ့ နေရာကို **F9** နှိပ်ပါ ။
(**Break Point** ထောက်တယ်လို့ ခေါ်ပါတယ် ။)

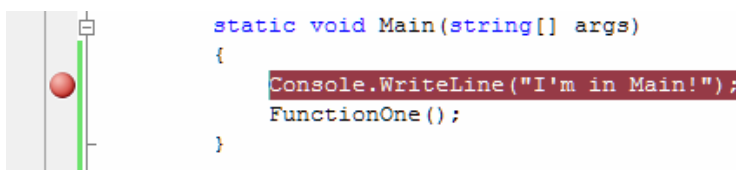


Fig : Program_Trace

၁၂ . Programm ကို Run ဖို့အတွက် **F5** ကို ပြန်နှိပ်ပါ ။
(**CTRL + F5** ကို နှိပ်ရင် **Trace** လိုက်ပေးမှာ မဟုတ်ပဲ Program ဟာ တန်းပြီး အလုပ်လုပ်သွားမှာပါ)

၁၃ . အခုဆိုရင် Program ကို **Trace** လိုက်ရန် အသင့်ဖြစ်ပါပြီ ။

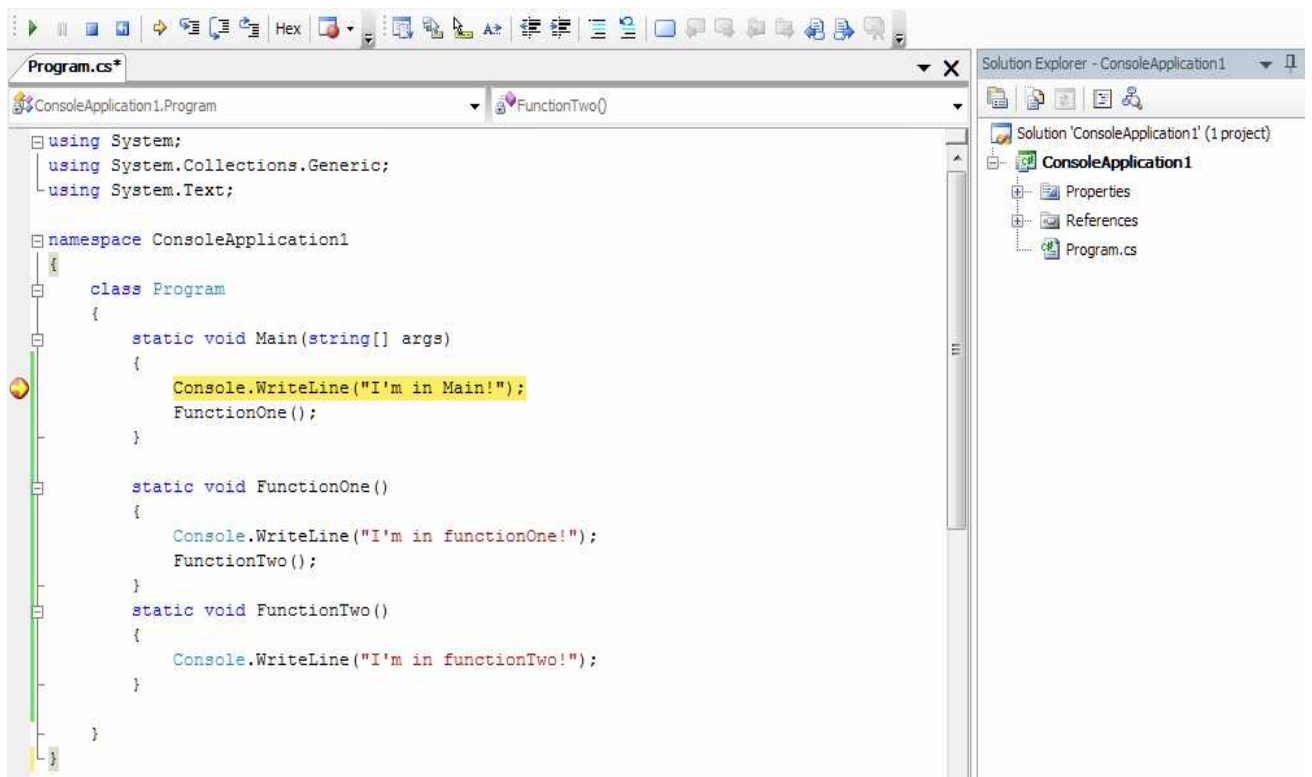


Fig : Program_1_Trace_With_F11

၁၄ . F11 ကို အသုံးပြုပြီး Trace တစ်ကြောင်းခြင်း လိုက်လို့ရပါပြီ ။

Method Call ပုံ နှင့် return Value အကြောင်း

ဒီ တစ်ခါ Method Call ပုံ နှင့် Return Value အကြောင်းကို ပြောပါမယ် ။

အပေါ်မှာ ရေးခဲ့တဲ့ Coding ကို ပဲ ပြင်ရေးမှာပါ ။

၁ . FunctionOne() ရဲ့ Coding ကို အနည်းငယ်ပြောင်းလဲထားပြီး FunctionTwo() နေရာမှာ `int` ကို `return` ပြန်ပေးမယ့် `Doubler(int originalValue)` ကို ရေးပါမယ် ။

```
using System;
using System.Collections.Generic;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("I'm in Main!");
            FunctionOne();
        }

        static void FunctionOne()
        {
            Console.WriteLine("I'm in functionOne!");
            int x = 5;
            int y;

            y = Doubler(x);

            Console.WriteLine("x was {0} and y is {1}", x, y);
        }

        static int Doubler(int originalValue)
        {
            int doublevalue = originalValue * 2;
            return doublevalue;
        }
    }
}
```

ရှင်းလင်းချက် ။ ။

`y = Doubler(x);` ဆိုတဲ့ အခြေအနေမှာ x ရဲ့ တန်ဖိုးဟာ အပေါ်မှာ ထည့်ထားတဲ့ 5 ပါ ။

အဲဒီနောက်

```
static int Doubler(int originalValue)
```



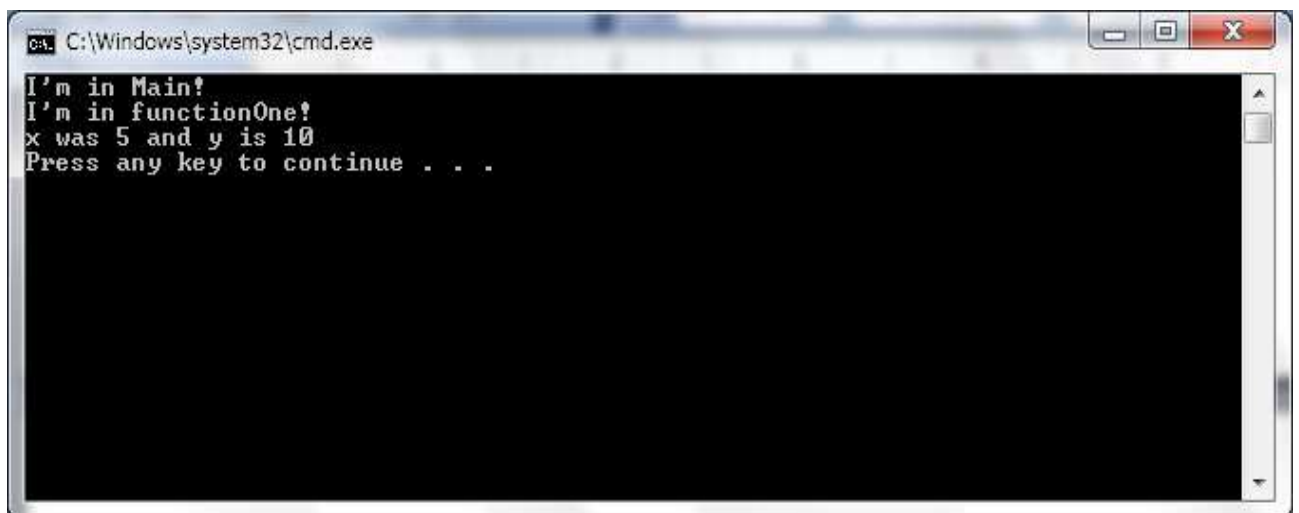
```
{
    int doublevalue = originalValue * 2;
    return doublevalue;
}
```

ကို သွားခေါ်လိုက်တဲ့ အခါ `originalValue` ထဲကို `x` ရဲ့ တန်ဖိုး 5 ရောက်သွားပါတယ် ။

Doubler Function ဟာ ဝင်လာတဲ့ တန်ဖိုး (`originalValue`) ကို ၂ ဆ မြှောက်လိုက်ပြီး `doublevalue` ထဲသို့ ထည့်လိုက်ပါတယ် ။

ထို့နောက် `return` ကို အသုံးပြုပြီး ရလာတဲ့ `int` တန်ဖိုး 10 ကို `y` ထဲသို့ ရောက်စေပါတယ် ။

Result Screen ကတော့ ဒီလိုပါ ။



A screenshot of a Windows command prompt window. The title bar shows the path 'C:\Windows\system32\cmd.exe'. The command prompt displays the following text: 'I'm in Main!', 'I'm in functionOne!', 'x was 5 and y is 10', and 'Press any key to continue . . .'. The window has standard Windows window controls (minimize, maximize, close) in the top right corner.

Conditional ဆိုသည်မှာ

Conditional ဆိုသည်မှာ လက်တွေ့တွင် ဖြစ်ပျက်နေသော အကြောင်းအရာများကို အနီးစပ်ဆုံး တူညီအောင် ဖော်ပြရန်အတွက် အသုံးပြုခြင်း ဖြစ်သည်။

Conditional မပါသော Program ဆိုသည် မှာ မရှိလောက်အောင် ရှားပါသည်။

IF အကြောင်း

ဥပမာ ။ ။

မိုးရွာလျှင် ထီးဆောင်းမည် ။ Condition တစ်ခု ကိုသာ စစ်လိုသော အခါ If ကို အသုံးပြုသည်။

```
if (isRaining)
    OpenUmbrella();
```

နှစ်မျိုးသာ ဖြစ်နိုင်သော အကြောင်းအရာများကို စစ်လိုသော အခါ If else ကို သုံးသည်။

```
if ( myAge > 40 )
    Console.WriteLine("You are over 40");
else
    Console.WriteLine("Youth abounds!");
```

တစ်ခုထက်ပို ဖြစ်ပျက်သော အကြောင်းအရာများကို စစ်လိုသော အခါ If ,else if ,else ကို သုံးနိုင်သလို Nested If ကို လည်း အသုံးပြုနိုင်သည်။

If else if else,

```
if(Money == "Dollar")
{
    Console.WriteLine("Dollar");
}

else if(Money == "Yum" )
{
    Console.WriteLine("Yum");
}

else
{
    Console.WriteLine("Kyat");
}
```

Nested If

```
if( temperature > freezingPoint )
{
    if ( temperature < boilingPoint )
```

```

        Console.WriteLine("Water is liquid");
    else
        Console.WriteLine("Water is gaseous");
}

else
    Console.WriteLine("Water is solid");

```

Switch အကြောင်း

```

using System;
using System.Collections.Generic;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main()
        {
            Console.Write("(1) Walk (2) Run (3) Crawl (4) Falling
[1,2,3,4]: ");

            string choice = Console.ReadLine();
            int menuChoice = Convert.ToInt32(choice);

            switch (menuChoice)
            {
                case 1:
                    Console.WriteLine("Walking...");
                    break;

                case 2: Console.WriteLine("Running...");
                    goto case 4;

                case 3:
                    Console.WriteLine("Crawling...");
                    break;

                case 4:
                    Console.WriteLine("Falling...");
                    break;
            }
        }
    }
}

```

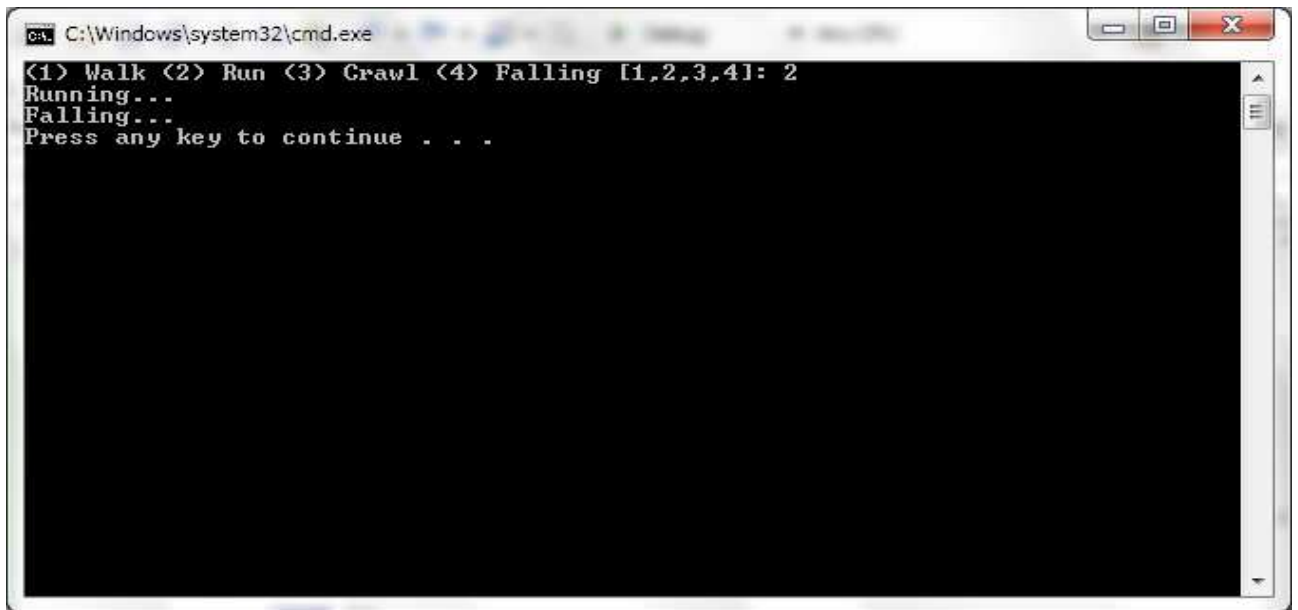
မှတ်ချက် ။ ။ Program ကို Run တဲ့ အခါ CTRL+F5 ကို နှိပ်မှ Result ကို မြင်ရမှာပါ ။

goto case 4; ဆိုတဲ့ နေရာကို သတိပြုမိမှာပါ ။ Run Box မှာ 2 လို့ ရိုက်ထည့်လိုက်ပါ ။ အဲဒါဆိုရင် goto အလုပ်လုပ်သွားပုံ ကို အလွယ်တကူ နားလည်နိုင်မှာပါ ။

goto အသုံးပြုခြင်း ရည်ရွယ်ချက်ကတော့ သူနဲ့ သက်ဆိုင်တဲ့ အကြောင်းအရာ တစ်ခုကို ပူးတွဲ ပြချင်တဲ့ အခါမျိုးမှာ သုံးလေ့ ရှိပါတယ် ။

အောက်က Result Screen မှာ 2 လို့ ရိုက်ထည့်လိုက်တဲ့ အခါ Running ... Falling.... ဆိုပြီး ပါလာပါတယ် ။

Result Screen



```
cmd C:\Windows\system32\cmd.exe
<1> Walk <2> Run <3> Crawl <4> Falling [1,2,3,4]: 2
Running...
Falling...
Press any key to continue . . .
```

Fig : Switch

LOOPING အကြောင်း

C# မှာ Supports လုပ်တဲ့ Looping Instructions များကတော့

- goto
- while
- do...while
- for
- foreach

goto အကြောင်း

goto ဟာ if နဲ့ လည်း တွဲဖက် အသုံးပြုလို့ ရပါတယ် ။

```

using System;
using System.Collections.Generic;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main()
        {
            // goto
            Console.WriteLine("goto...");
            int i = 0;
            repeat:
            Console.Write("{0} ", i);
            i++;
            if (i < 10)
                goto repeat;
        }
    }
}

```

Result Screen

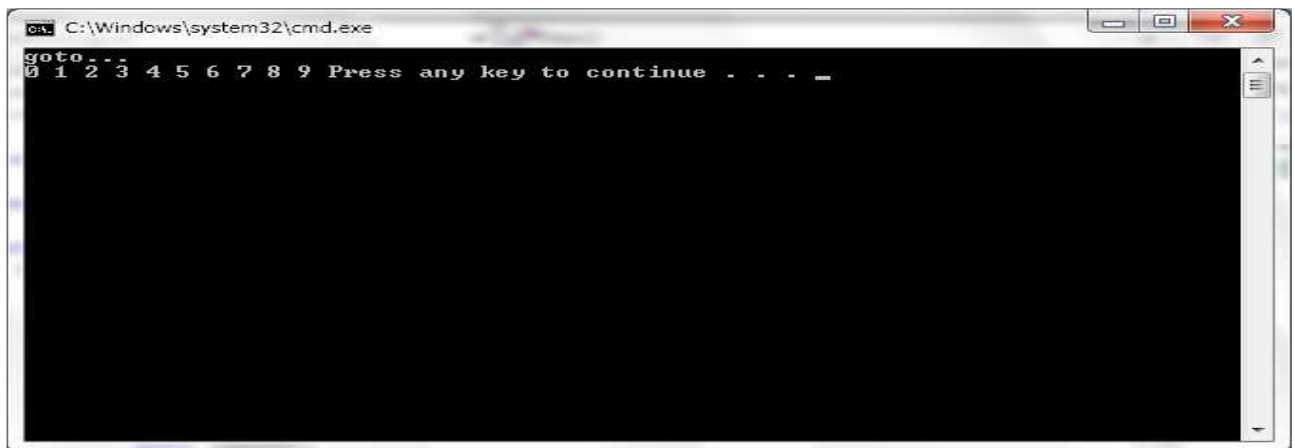


Fig : goto

while , do.. while , for and do:

```

using System;

```

```

using System.Collections.Generic;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main()
        {
            // while
            Console.WriteLine("While...");
            int i = 0;
            while (i < 10)
            {
                Console.Write("{0} ", i);
                i++;
            }

            // do...while
            Console.WriteLine("\n\nDo While...");
            i = 0;
            do
            {
                Console.Write("{0} ", i);
                i++;
            } while (i < 10);

            // for
            Console.WriteLine("\n\nFor...");
            for (i = 0; i < 10; i++)
            {
                Console.Write("{0} ", i);
            }
        }
    }
}

```

Result Screen

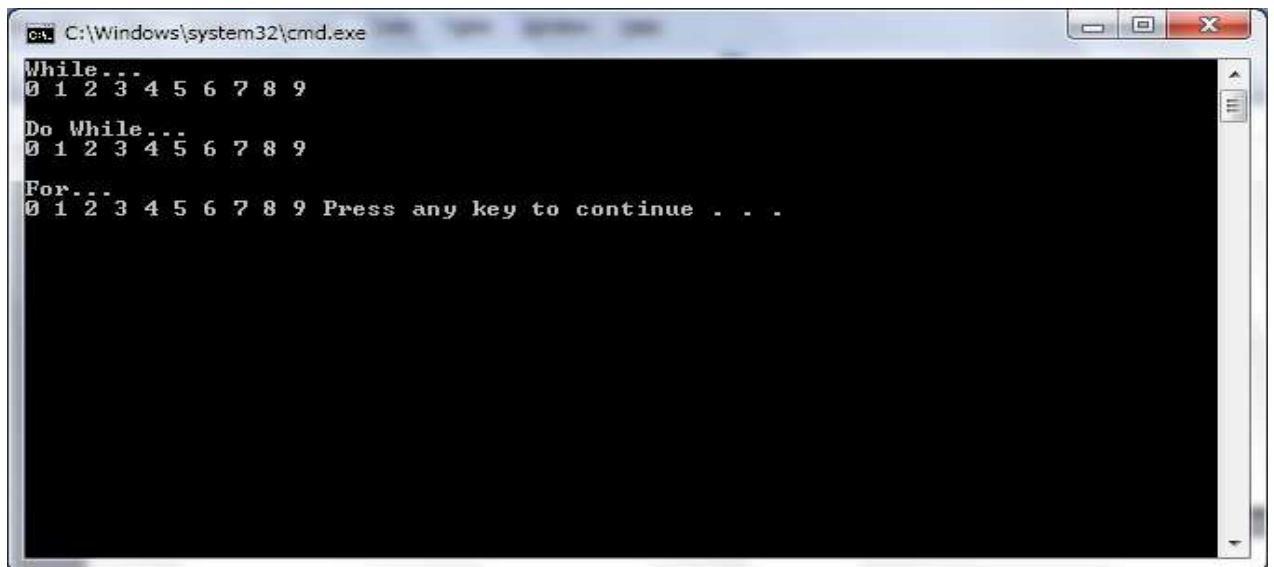


Fig : While_DoWhile_For

foreach အကြောင်း

Integer Array တစ်ခုနှင့် တွဲ အသုံးပြုသော Foreach

<code>int[] intArray = new int[5] ;</code>	အခန်း 5 ခန်းပါ Integer Array တစ်ခု ဆောက်လုပ်
<code>int sum = 0 ;</code>	sum ဆိုသော Integer ထဲသို့ 0 ကို ထည့်ပုံ (= ကို Assign လုပ်သည်ဟုခေါ်သည်)
<code>foreach (int i in intArray)</code>	<code>foreach (Containerထဲရှိ အမျိုးအစား variableName in ကိုယ်အသုံးပြုလိုသည့် Container)</code>
<code>{</code>	1
<code> sum += i ;</code>	2 3 4
<code>}</code>	ဒီ လေးမျိုး မဖြစ်မနေပါရပါမယ်

END