동국대학교
dongguk university

C# 기본 문법

HyoJoon Han
동국대학교
han6343@dongguk.edu

목차

변수와 데이터형

- 변수의 개념
  - 변수
    - 프로그램이 실행되는 동안 사용자가 입력한 값이나 계산한 출력 값 등을 저장하는 기억 공간
  - 변수 선언
    - 프로그래머가 프로그램을 작성하다가 기억 공간이 필요하면 변수의 이름과 크기를 컴퓨터에 알려주는 것
    - 컴퓨터는 요청 받은 크기만큼 기억 공간을 확보하고 시작 주소를 프로그램에 알려준다.
    - 프로그램은 할당된 시작 주소와 선언된 변수명을 매칭하여 사용한다.

변수와 데이터형

- 변수의 선언
  - 변수의 크기를 선언할 때는 직접 몇 바이트인지 입력하지 않고 해당 데이터형을 입력한다.
  - 변수 선언 형식

형식 | 데이터형 변수명; or 데이터형 변수명 = 값;

예 | int num;
String str = "Hello";

  - 변수명 작성 규칙
    - 영문자, 한글, 숫자, 밑줄(_) 을 조합하여 사용할 수 있으며 다른 특수 기호는 사용할 수 없다.
    - 첫 글자는 반드시 영문자나 한글이어야 하고 기호나 숫자는 사용할 수 없다.
    - 키워드(while, for, int ,void …)는 사용할 수 없다.
    - 255자를 넘지 않아야 한다.

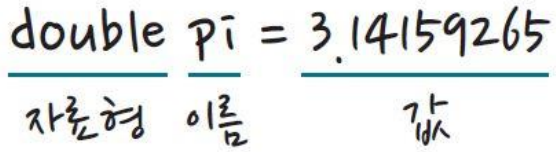


그림 2-12 변수 선언 예

변수와 데이터형

- 데이터형

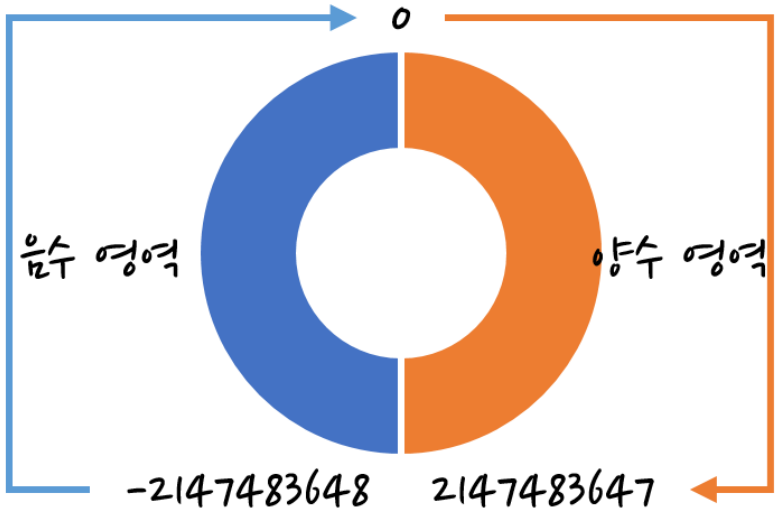
데이터 형식	설명	크기(바이트)	담을 수 있는 값의 범위
byte	부호 없는 정수	1(8비트)	0~255
sbyte	signed byte 정수	1(8비트)	–128~127
short	정수	2(16비트)	–32,768~32,767
ushort	unsigned short 부호 없는 정수	2(16비트)	0~65,535
int	정수	4(32비트)	–2,147,483,648~2,147,483,647
uint	unsigned int 부호 없는 정수	4(32비트)	0~4,294,967,295
long	정수	8(64비트)	–922,337,203,685,477,508~922,337,203,685,477,507
ulong	unsigned long 부호 없는 정수	8(64비트)	0~18,446,744,073,709,551,615
char	유니코드 문자	2(16비트)	

변수와 데이터형

- 정수 자료형

표 2-12 정수 자료형

키워드	설명
int	4바이트의 정수
long	8바이트의 정수



- int 자료형
  - 일반적으로 정수를 만들 때 사용
  - 오버플로우 : int 자료형의 범위를 넘는 현상
    - Ex) int a = 2147483640; int b = 59233;
      int c = a+b;
- long 자료형
  - int보다 더 큰 정수를 표현할 때 사용
  - int형과 구분하기 위해 숫자 뒤에 'L' 을 붙임

변수와 데이터형

- 오버플로우 문제 해결 방법 : 자료형 변환

- Ex) int a = 2000000000;
  int b = 1000000000;

  long c = (long)a + (long)b; or long c = a + b;

변수와 데이터형

- unsigned 자료형 (부호가 없는 자료형)
  - 정수의 음수 범위를 양수로 포함하여 더 큰 범위의 숫자를 표현
  - uint와 ulong 데이터형 사용

  - Ex) uint unsignedInt = 4147483647;
    ulong unsignedLong = 112233720368547758O8L;

변수와 데이터형

- 실수 자료형

키워드	설명
float	4바이트의 실수
double	8바이트의 실수

- double 자료형
  - 일반적으로 실수 데이터를 표현할 때 사용

- float 자료형
  - double보다 작은 데이터를 처리하거나 빠른 데이터 처리가 필요할 때 사용
  - 값 뒤에 f 를 붙여서 float 자료형을 표시
  - ex) float pi = 3.1415f;

변수와 데이터형

- 문자열 자료형
  - 문자열을 표현할 수 있는 데이터형태
  - string 자료형만 struct가 아닌 class로 시작

표 2-15 문자열 자료형

키워드	설명
string	문자열 자료형

표 2-16 기본 자료형의 원형

기본 자료형	원형	기본 자료형	원형
int	struct System.Int32	float	struct System.Single
long	struct System.Int64	double	struct System.Double
char	struct System.Char	bool	struct System.Boolean
string	class System.String		

변수와 데이터형

- 불 자료형
  - True 또는 False 로 표현되는 자료형
  - $0 = \text{False}$, $0 \neq \text{True}$ 로 사용됨

표 2-17 불 자료형

키워드	설명
bool	불 자료형

변수와 데이터형

- var 키워드

표 2-22 var 키워드

var 키워드	설명
var	자료형을 자동으로 지정합니다.

  - 데이터형이 지정되지 않은 변수를 선언할 때 사용
  - 한 번 지정된 자료형은 계속 유지
  - int 자료형으로 선언된 변수를 string 자료형으로 바꾸는 것은 불가능

```
var numberA = 100L;    // long 자료형
var numberB = 100.0;   // double 자료형
var numberC = 100.0F;  // float 자료형
```

변수와 데이터 형

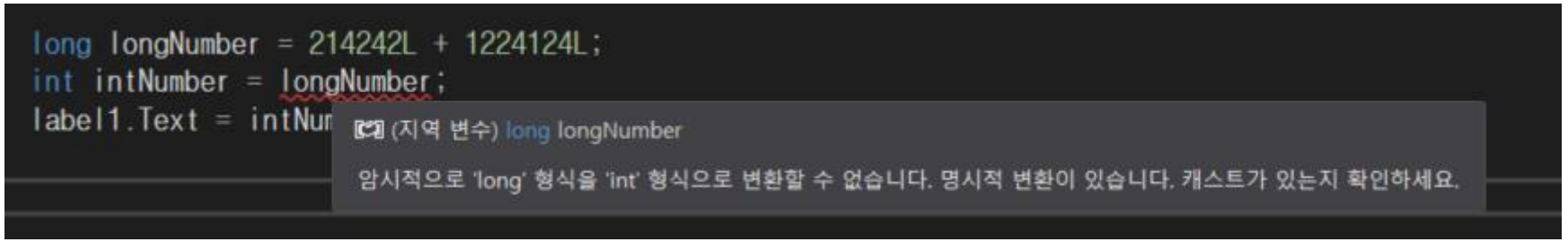
- 자료형 변환
  - 한 자료형을 다른 자료형으로 바꾸는 것
  - 묵시적 형 변환 or 자동 형 변환

  - 예

```
int intNumber = 214242 + 1224124;
long longNumber = intNumber;
label1.Text = intNumber.ToString();
```

변수와 데이터형

- 강제 자료형 변환



  - 자동 형 변환이 불가능한 경우도 있음
  - 강제로 데이터 형을 지정하여 변형해줌
  - 강제 형 변환 or 명시적 형 변환

코드 2-57 **강제 자료형 변환** /2장/Casts

```
var a = (int)10.0;
var b = (float)10;
var c = (double)10;
```

변수와 데이터형

- 강제 자료형 변환

```
long longNumber = 214242L + 1224124L;
int intNumber = (int)longNumber;
label1.Text = intNumber.ToString();
```

  - 강제 형 변환 시 데이터 손실이 발생 할 수 있음
  - 위와 같은 예 에서는 longNumber가 int 로 변환되어도 int가 표현할 수 있는 범위이므로 데이터 손실 발생하지 않음

변수와 데이터형

- 자동 자료형 변환
  - 변환하고자 하는 자료형을 지정해주지 않아도 자동으로 형을 변환할 수 있는 형태

표 2-24 자동 자료형 변환

기존 자료형	자동 변환되는 자료형
int	long, float, double
long	float, double
char	int, long, float, double
float	double

변수와 데이터형

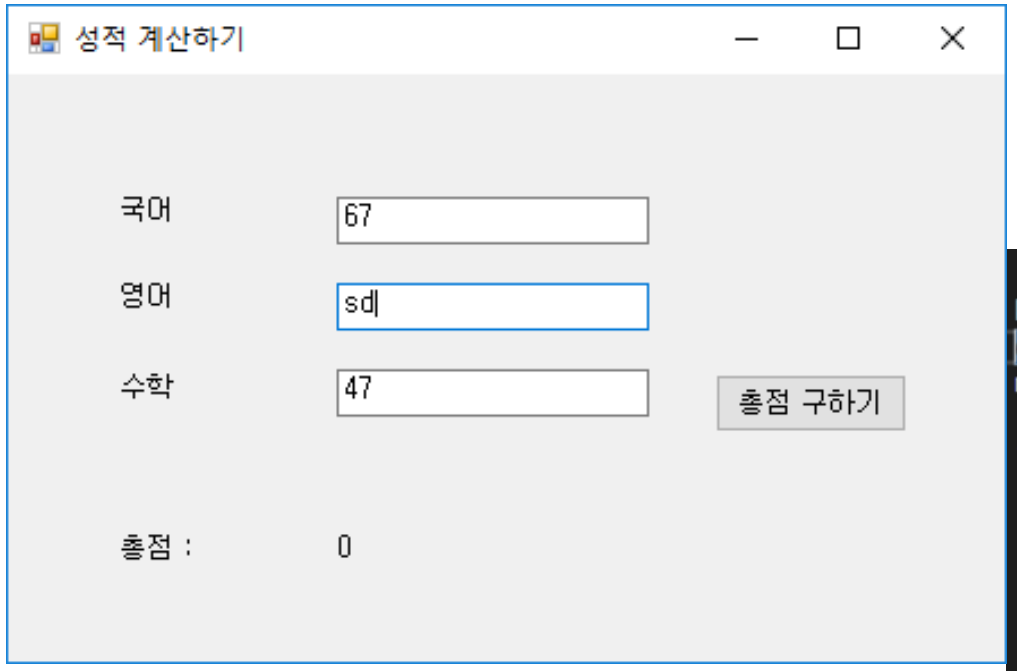
- 다른 자료형을 숫자로 변환
  - 주로 String 자료형을 숫자로 변환하고자 할 때 사용

표 2-25 문자열을 숫자로 변환하는 메서드

메서드 이름	설명
int.Parse()	다른 자료형을 int 자료형으로 변경합니다.
long.Parse()	다른 자료형을 long 자료형으로 변경합니다.
float.Parse()	다른 자료형을 float 자료형으로 변경합니다.
double.Parse()	다른 자료형을 double 자료형으로 변경합니다.

변수와 데이터형

- FormatException 예외
  - 다음과 같이 문자를 숫자로 변경하고자 하는 경우 등에 발생하는 오류




```
int kor, eng, mat, total;
kor = int.Parse(textBox1.Text);
eng = int.Parse(textBox2.Text);
mat = int.Parse(textBox3.Text);

total = kor + eng + mat;
label5.Text = total.ToString();
```

예외가 처리되지 않음

System.FormatException: '입력 문자열의 형식이 잘못되었습니다.'

자세히 보기 | 세부 정보 복사

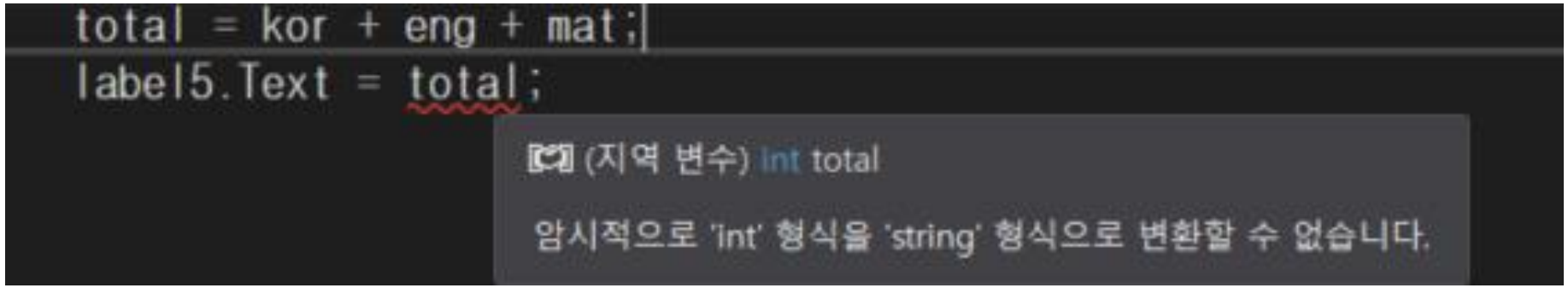
▷ 예외 설정

변수와 데이터형

- 다른 자료형을 문자열로 변환
  - C#의 모든 자료형은 ToString() 메소드를 가지고 있음
  - 문자열로 표현하고 싶은 변수 뒤에 .ToString() 메소드를 사용하여 문자열로 변경 가능

표 2-26 문자열로 변환하는 메서드

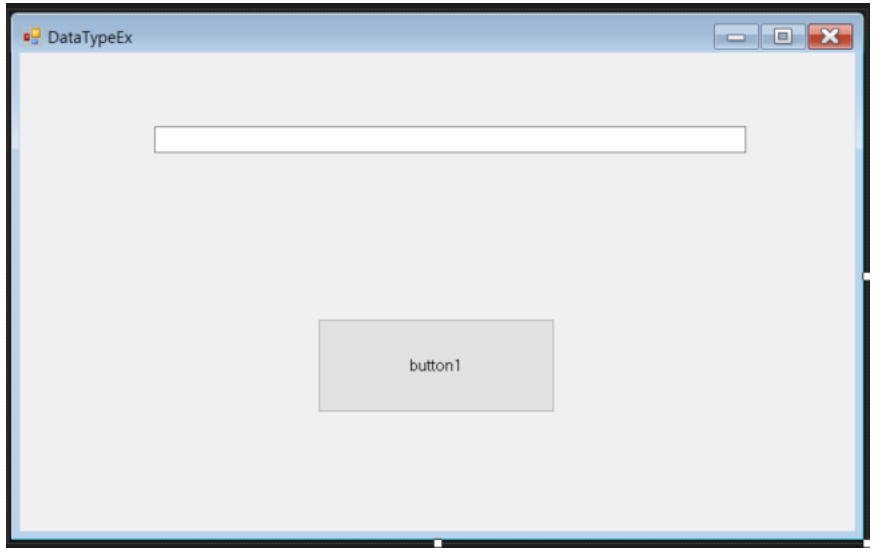
메서드	설명
ToString()	문자열로 변환합니다.



변수와 데이터형

- 실습 – 변수 선언하고 사용하기
  - Score라는 변수에 점수를 저장 후 <출력>을 클릭하면 텍스트박스에 출력하는 프로그램을 작성

1) 윈도우 폼 디자인 , 속성 설정



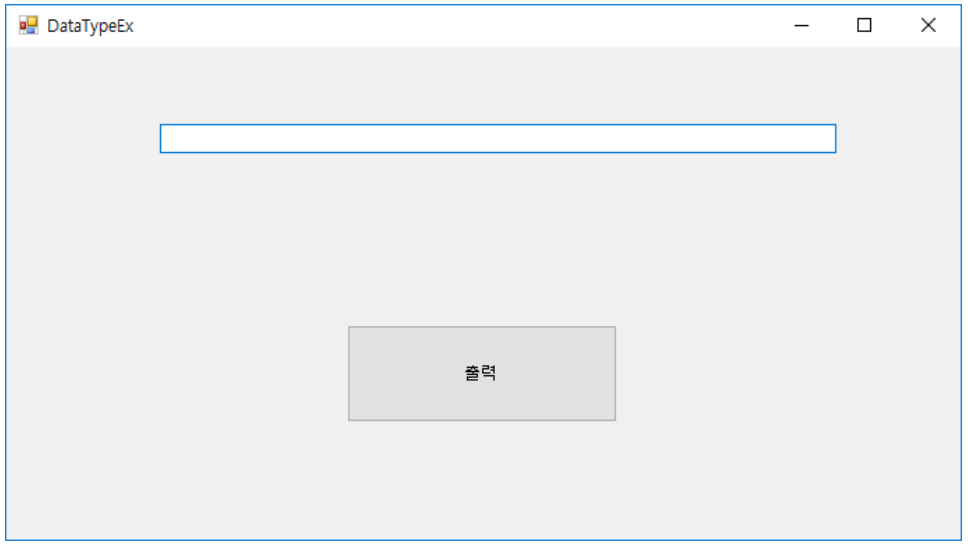
컨트롤	속성	속성값
Form1	Text	변수 사용
TextBox1	Text	(빈칸)
Button1	Text	출력

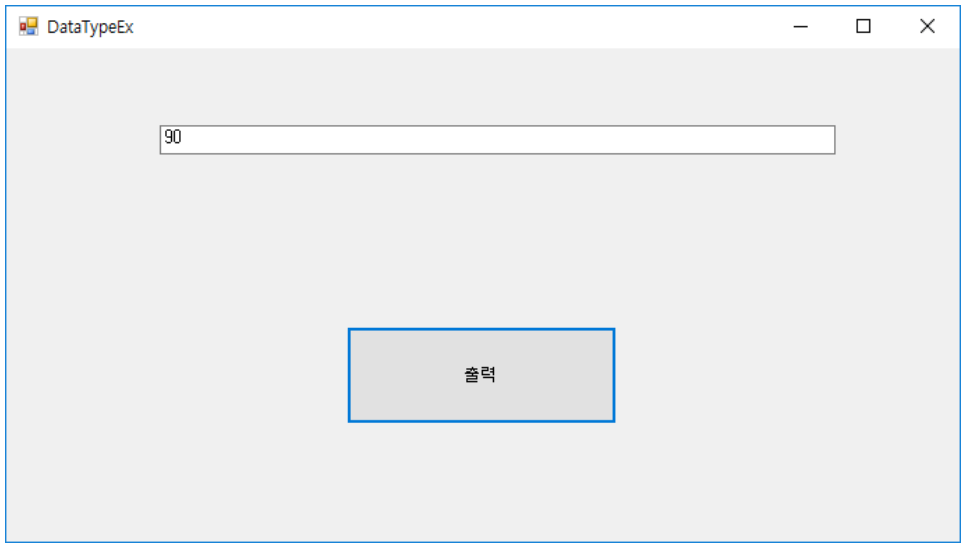
변수와 데이터형

2) 코드 작성

```
private void button1_Click(object sender, EventArgs e)
{
    int Score = 90;
    textBox1.Text = Score.ToString();
}
```

3) 실행 결과

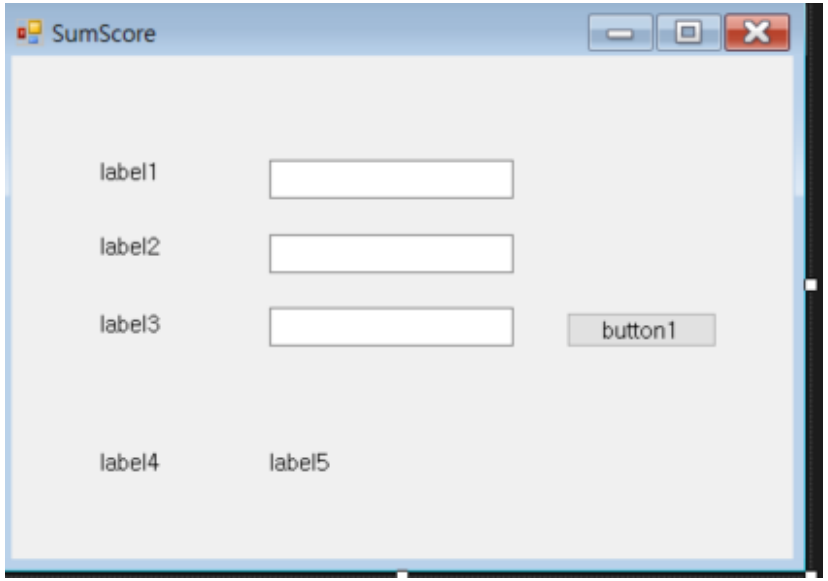




변수와 데이터형

- 실습 – 성적 합계 구하기
  - 국어, 영어, 수학 과목의 점수를 입력 받아 총점을 구하고 출력하는 프로그램 작성

1) 윈도우 폼 디자인



2) 속성 설정

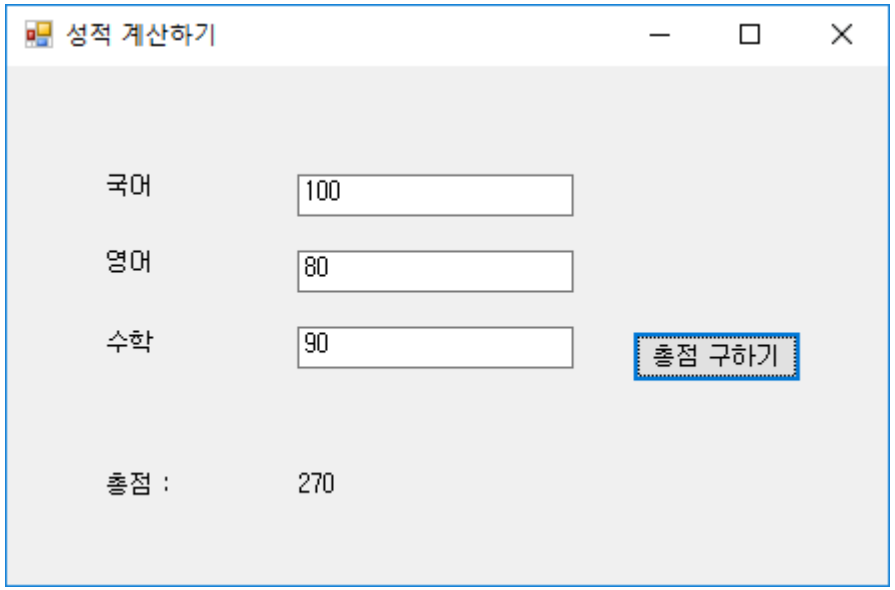
컨트롤	속성	속성값
Form1	Text	성적 계산하기
	StartUpPosition	CenterScreen
Label1	Text	국어
Label2	Text	영어
Label3	Text	수학
TextBox1	Text	(빈칸)
TextBox2	Text	(빈칸)
TextBox3	Text	(빈칸)
Label4	Text	총점 :
Label5	Text	0
Button1	Text	총점 구하기

변수와 데이터형

3) 코드 작성

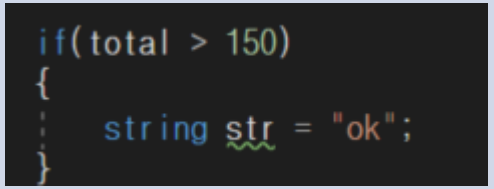
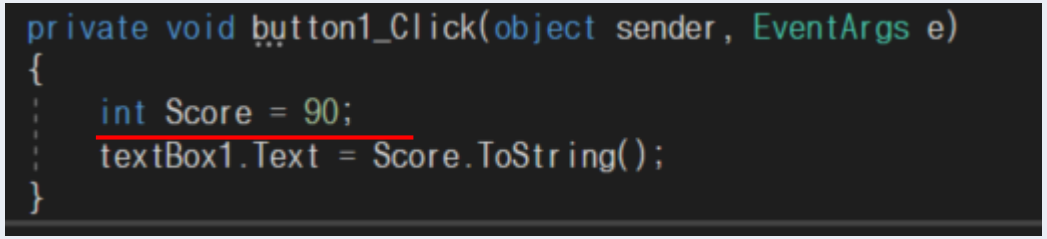
```
private void button1_Click(object sender, EventArgs e)
{
    int kor, eng, mat, total;
    kor = int.Parse(textBox1.Text);
    eng = int.Parse(textBox2.Text);
    mat = int.Parse(textBox3.Text);

    total = kor + eng + mat;
    label5.Text = total.ToString();
}
```

4) 실행 결과



변수와 데이터형

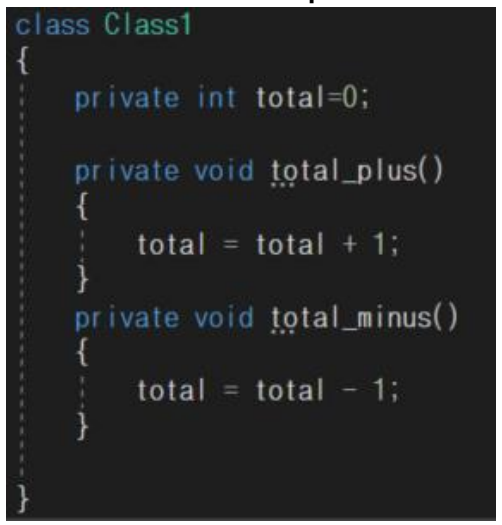
- 변수의 수명과 유효 범위
  - 변수는 자신이 선언되는 코드 범위 안에서만 값을 가질 수 있다.
  - 유효 범위
    - 변수가 메모리를 차지하고 값을 유지하는 구간

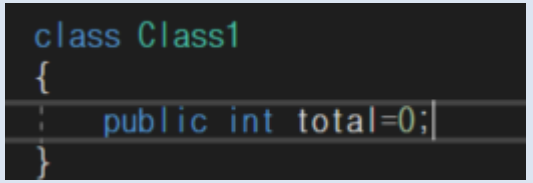
유효 범위		설명
지역변수	블록	• 선언 시작과 종료 문 사이를 말함 : for { ... } if(...) { ... } • 블록 내에 선언된 변수는 해당 블록 내에서만 사용 가능 • 예 
	메소드	• 메소드 내에서 선언된 변수는 해당 메소드 내에서만 사용 가능 • 이 수준의 변수를 '지역변수 ' 라 함 • 예 

```
if(total > 150)
{
    string str = "ok";
}
```

```
private void button1_Click(object sender, EventArgs e)
{
    int Score = 90;
    textBox1.Text = Score.ToString();
}
```

변수와 데이터형

<table>
<tr>
<td>클래스변수</td>
<td>클래스</td>
<td>
• 클래스 수준에서 private으로 선언된 변수로 클래스 내의 메소드에서만 사용 가능
• 예


</td>
</tr>
<tr>
<td>전역변수</td>
<td>모든 폼 및 클래스</td>
<td>
• 클래스 수준에서 public 으로 선언된 변수로 모든 폼과 클래스에서 사용 가능
• 여러 개의 폼으로 구성된 경우에 모든 폼에서 사용 가능
• 예


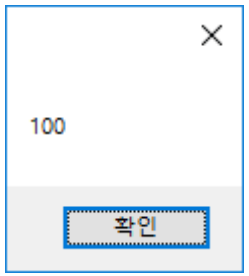
</td>
</tr>
</table>

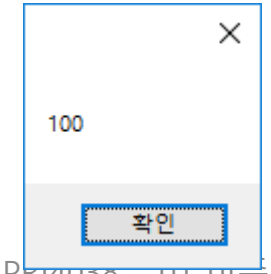
변수와 데이터형

- 지역 변수의 사용
  - 지역 변수는 메모리 반납 여부에 따라 var과 static 변수로 나뉜다.
  - var 변수 – 메소드 실행이 종료되면 메모리를 반납하고 값이 없어진다.
  - static 변수 – 메모리를 반납하지 않고 저장된 값을 그대로 유지하므로, 메소드가 종료되더라도 그 값을 유지해야 할 경우에 사용한다.

```
private void button1_Click(object sender, EventArgs e)
{
    int intTest = 0; ①
    intTest = intTest + 100; ②
    MessageBox.Show(intTest.ToString()); ③
} ④
```

① 정수형 변수 intTest가 생성되고 초기 값으로 0이 저장됨 ② intTest에 정수 100을 더함
③ intTest 값을 출력함 ④ intTest 변수가 소멸됨



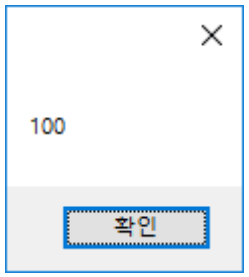


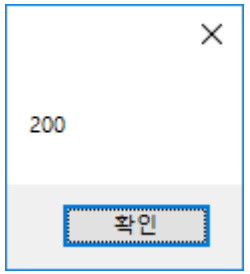
변수와 데이터형

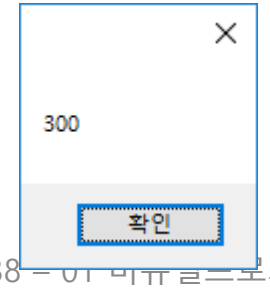
- static 지역 변수의 사용

```
public partial class localVar : Form
{
    int staticInt = 0; ①
    public localVar()
    {
        InitializeComponent();
    }

    private void button1_Click(object sender, EventArgs e)
    {
        staticInt = staticInt + 100; ②
        MessageBox.Show(staticInt.ToString()); ③
    }④
}
```

① 정수형 변수 staticInt가 생성되고 초기 값으로 0이 저장됨 ② staticInt에 정수 100을 더함
③ staticInt 값을 출력함 ④ staticInt 변수가 소멸되지 않고 값이 유지됨







변수와 데이터형

- 클래스 변수의 선언
  - 메소드 내부가 아닌 코드 편집 창 상단 부분의 클래스 내부에 private로 선언
  - 선언된 클래스변수는 클래스 내에 기술된 모든 메소드에서 이용할 수 있다.

- 전역 변수의 선언
  - 프로젝트 내의 모든 메소드에서 변수를 공유하고자 할 때는 public 전역변수 선언자를 사용
  - 한 번 선언하면 프로젝트가 종료될 때까지 메모리를 유지
  - 한 개 이상의 폼에서 공유할 수 있다.

변수와 데이터형

여기서 잠깐

지역변수의 장점

일반적으로 변수는 유효 범위를 최대한 좁게 하여 필요한 부분에서만 지역변수로 사용하길 권한다. 그 이유는 다음과 같다.

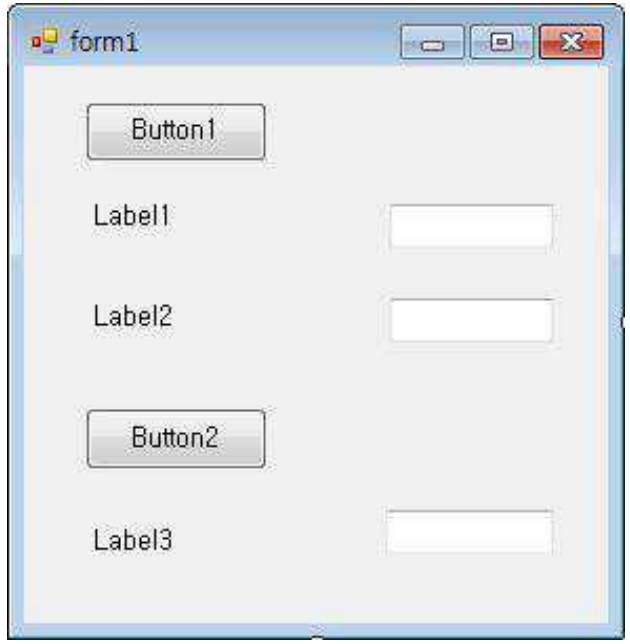
① 혼자가 아닌 여러 사람이 협업하여 프로그램을 작성하는 경우가 많다.
여러 사람이 각각 담당할 부분을 나누어 작업하다 보면 비슷한 변수명을 사용하는 경우가 많이 발생한다. 만약 변수의 유효 범위를 넓게 설정하여 사용한다면 변수명이 서로 중복되어 혼란을 야기할 수 있다. 따라서 유효 범위를 최대한 좁게 설정하는 것이 좋다. 각 프로시저에서 선언한 변수는 그곳에서만 사용 및 소멸되도록 프로그래밍하면, 설령 프로시저별로 중복된 변수명이 있더라도 오류가 발생할 위험이 줄어든다.

② 메모리를 효율적으로 사용할 수 있다.
지역변수는 해당 프로시저가 실행되는 동안에만 메모리를 할당받아 사용하고 종료되면 바로 메모리를 반납한다. 필요한 시점에만 메모리를 사용하므로 효율적이다.

변수와 데이터형

- 실습 – 지역변수와 클래스변수 사용하기

1) 윈도우 폼 디자인



2) 속성 설정

컨트롤	속성	속성값
Button1	Text	지역변수
Button2	Text	모듈변수
Label1	Text	지역변수 var_pro 값
Label2	Text	모듈변수 var_mod 값
Label3	Text	모듈변수 var_mod 값
TextBox1, 2, 3	Text	(빈칸)

변수와 데이터형

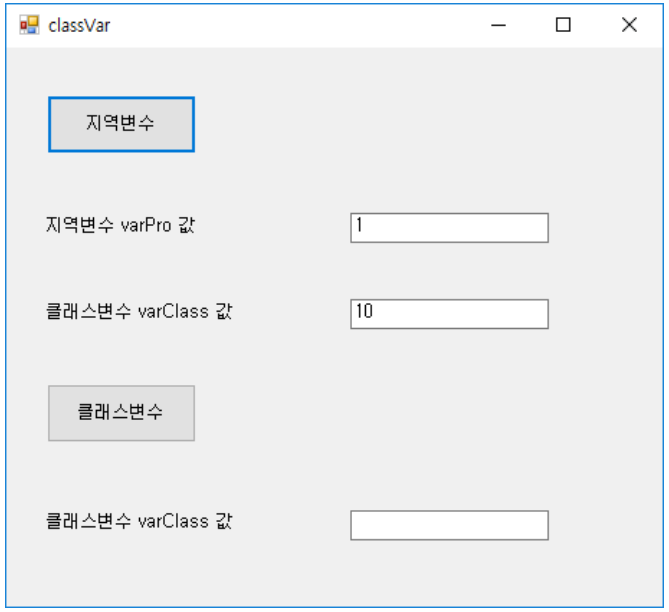
3) 코드 작성

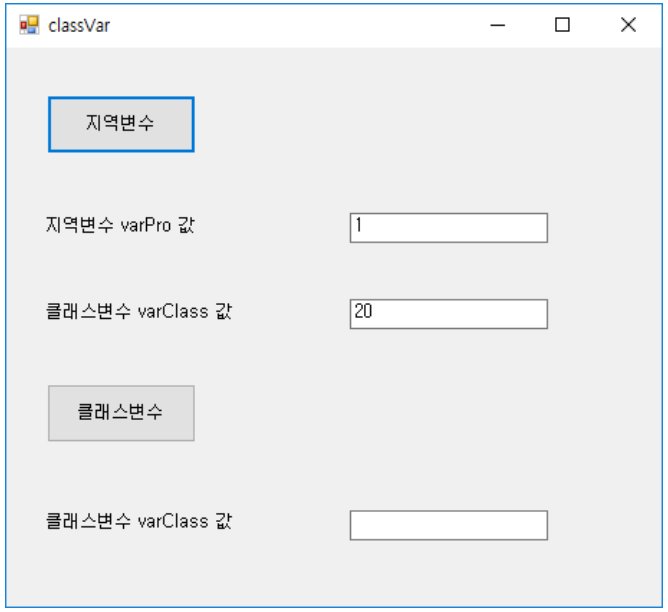
```
namespace Chapter04
{
    public partial class classVar : Form
    {
        private int varClass = 0;

        public classVar()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            int varPro = 0;

            varPro = varPro + 1;
            varClass = varClass + 10;

            textBox1.Text = varPro.ToString();
            textBox2.Text = varClass.ToString();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            varClass = varClass + 100;
            textBox3.Text = varClass.ToString();
        }
    }
}
```

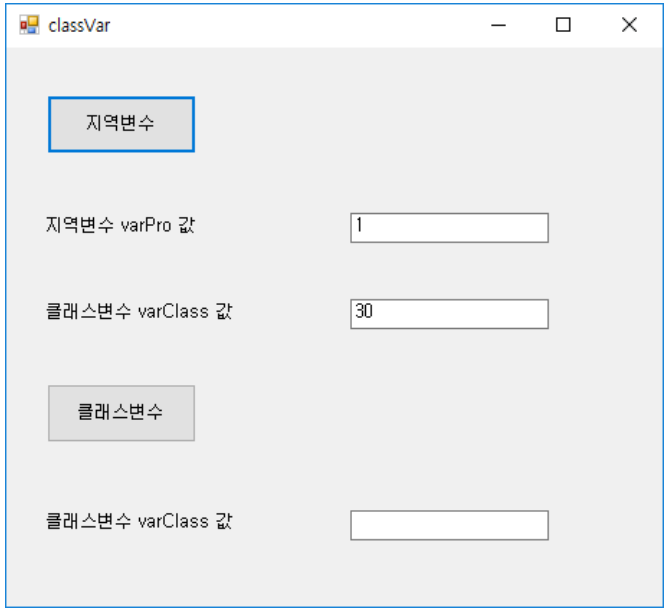
변수와 데이터형

4) 프로그램 실행. 버튼을 클릭하여 지역 변수와 클래스 변수의 사용 결과를 비교해본다.

5) <지역변수>를 세 번 클릭하면 지역변수 varPro값은 동일하지만 클래스 변수 varClass값은 증가한다.

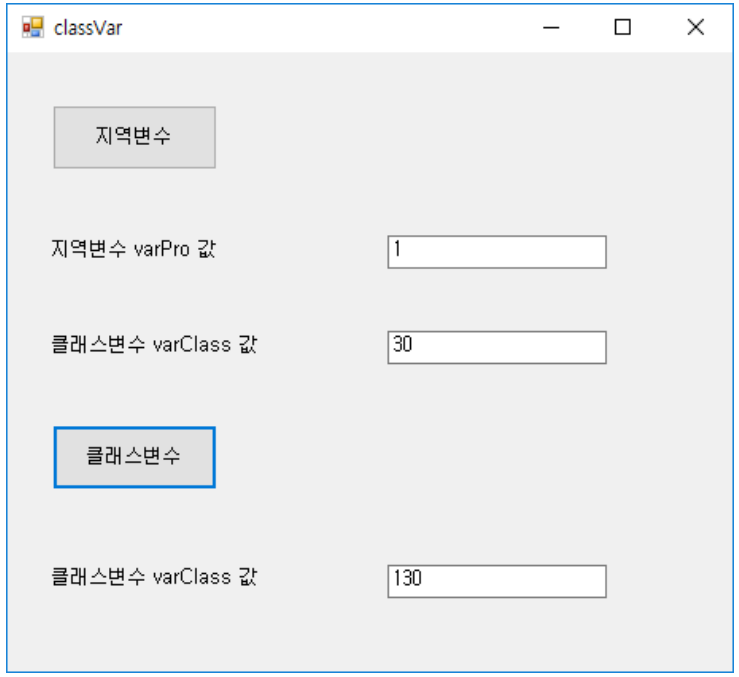


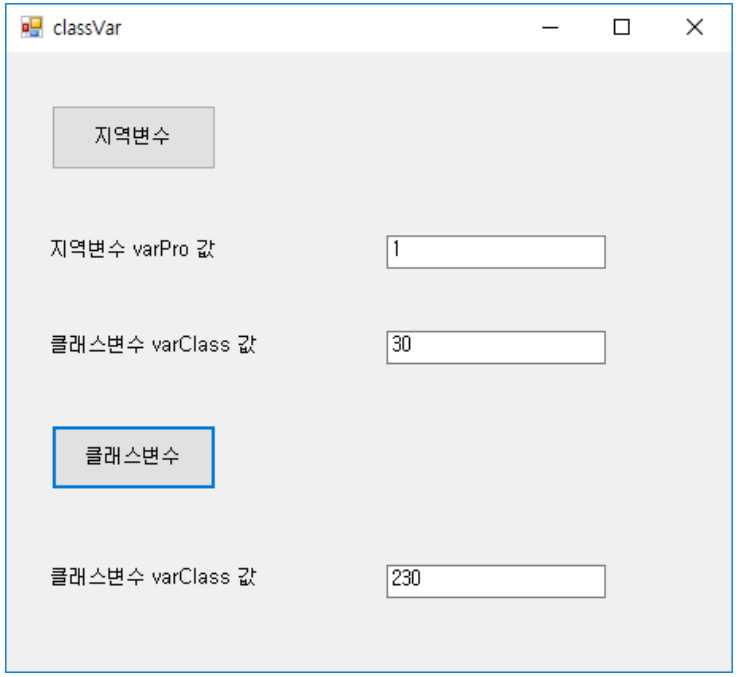


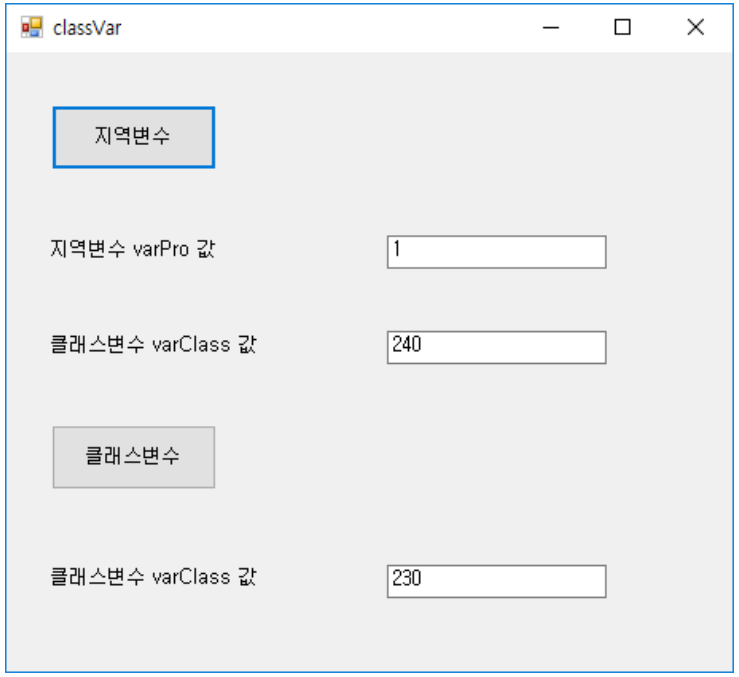


변수와 데이터형

6) <클래스 변수를 두 번 클릭하면 varClass 값이 '130', '230' 으로 증가한다. 다시 <지역변수> 를 한 번 클릭하면 지역변수 varPro는 역시 '1'이지만 varClass는 '240'으로 증가한다.



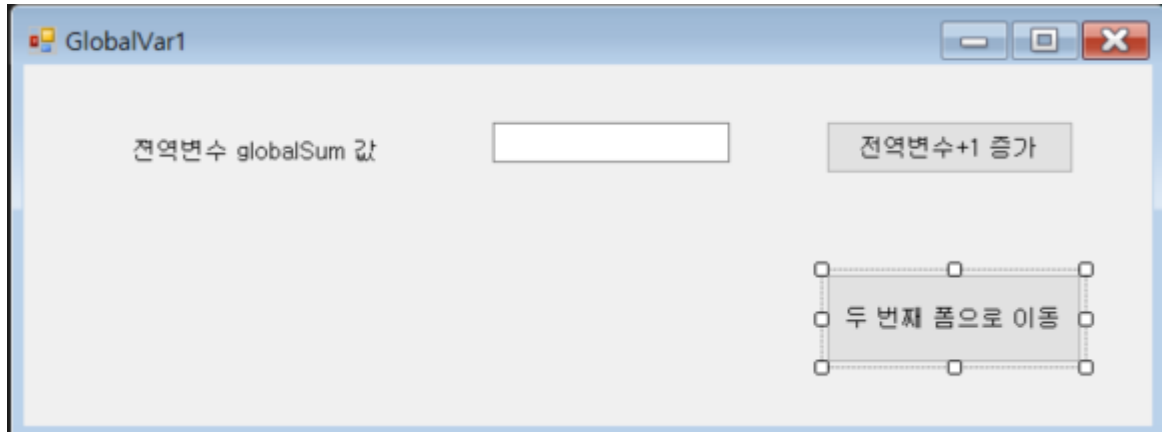


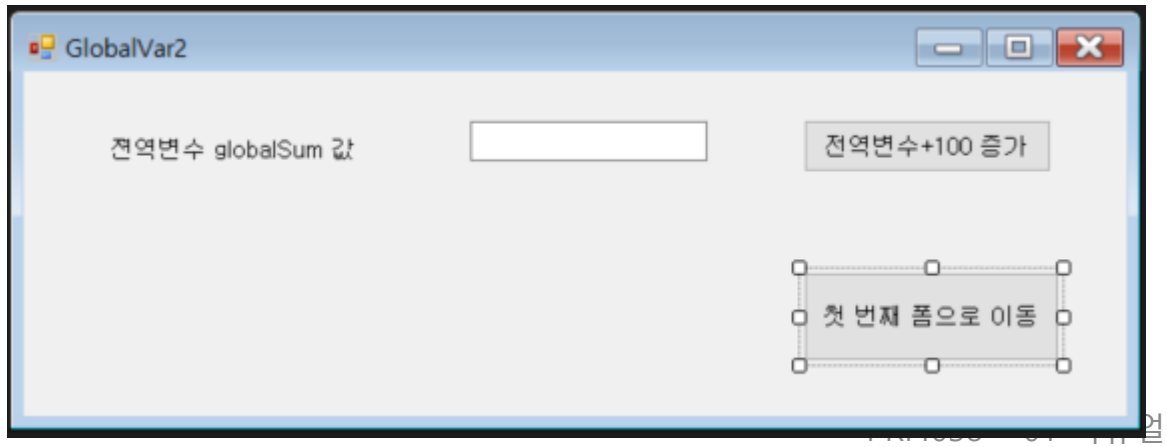


변수와 데이터형

- 실습 – 전역변수 사용하기

1) 윈도우 폼 디자인





2) 속성 설정

폼	컨트롤	속성	속성값
폼1	Button1	Text	전역변수+1 증가
	Button2	Text	Form2 보기
	Label1	Text	전역변수 Pub_sum 값
	TextBox1	Text	(빈칸)
폼2	Button1	Text	전역변수+100 증가
	Button2	Text	Form1 보기
	Label1	Text	전역변수 Pub_sum 값
	TextBox1	Text	(빈칸)

변수와 데이터형

3) 코드 작성 – 첫 번째 폼

```
namespace Chapter04
{
    public partial class GlobalVar1 : Form
    {
        public static int globalSum = 0;

        public GlobalVar1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            globalSum = globalSum + 1;
            textBox1.Text = globalSum.ToString();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            GlobalVar2 globalVar2 = new GlobalVar2();
            globalVar2.Show();

        }
    }
}
```

변수와 데이터형

4) 코드 작성 – 두 번째 폼

```
namespace Chapter04
{
    public partial class GlobalVar2 : Form
    {
        public GlobalVar2()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            GlobalVar1.globalSum = GlobalVar1.globalSum + 100;
            textBox1.Text = GlobalVar1.globalSum.ToString();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            GlobalVar1 globalVar1 = new GlobalVar1();
            globalVar1.Show();
        }
    }
}
```

변수와 데이터형

5) 결과 확인