

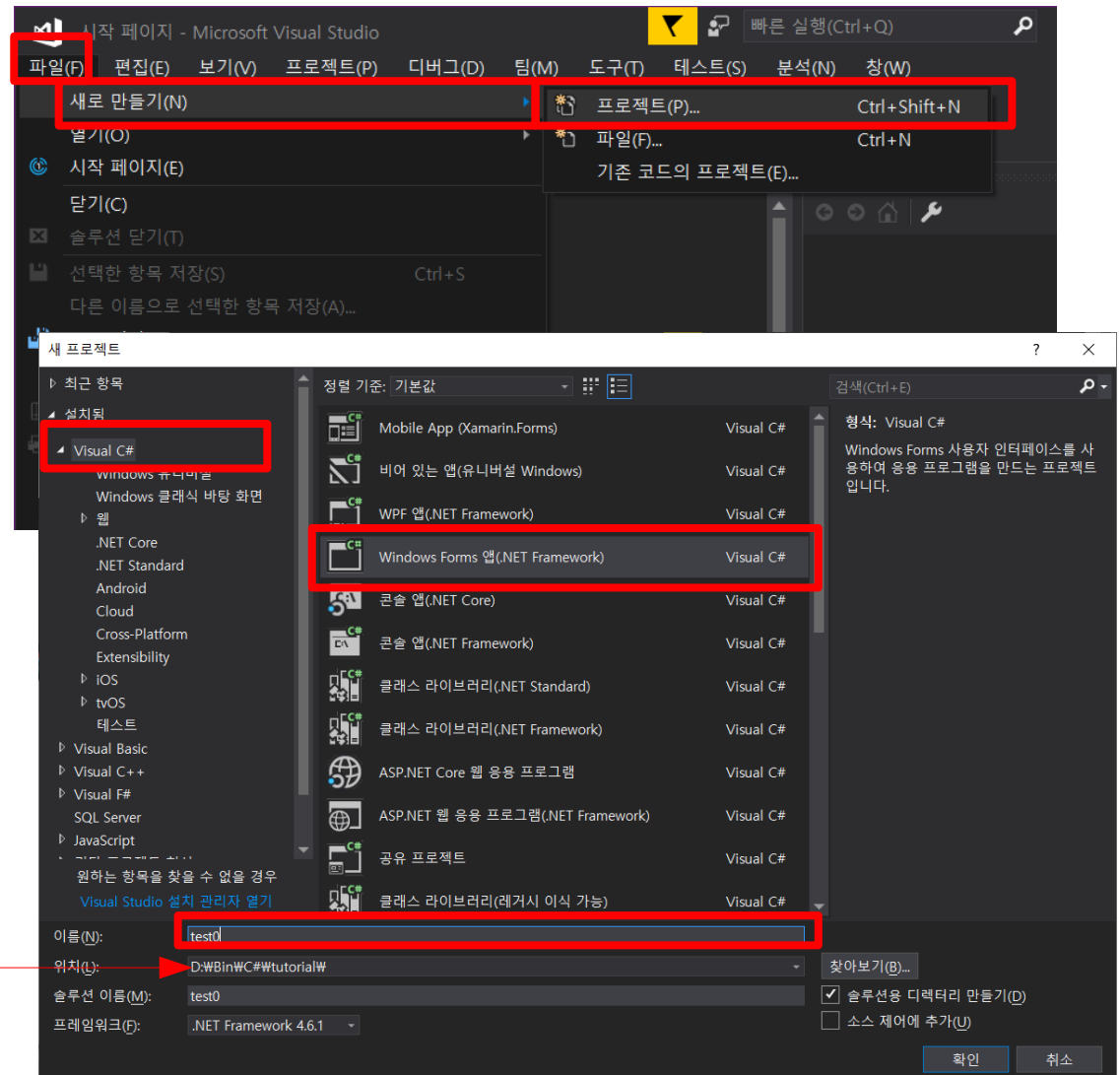
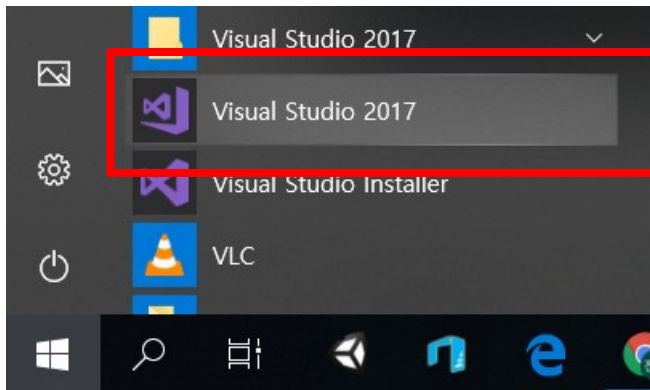


# C# & Robot

1) 2) 로봇 ...

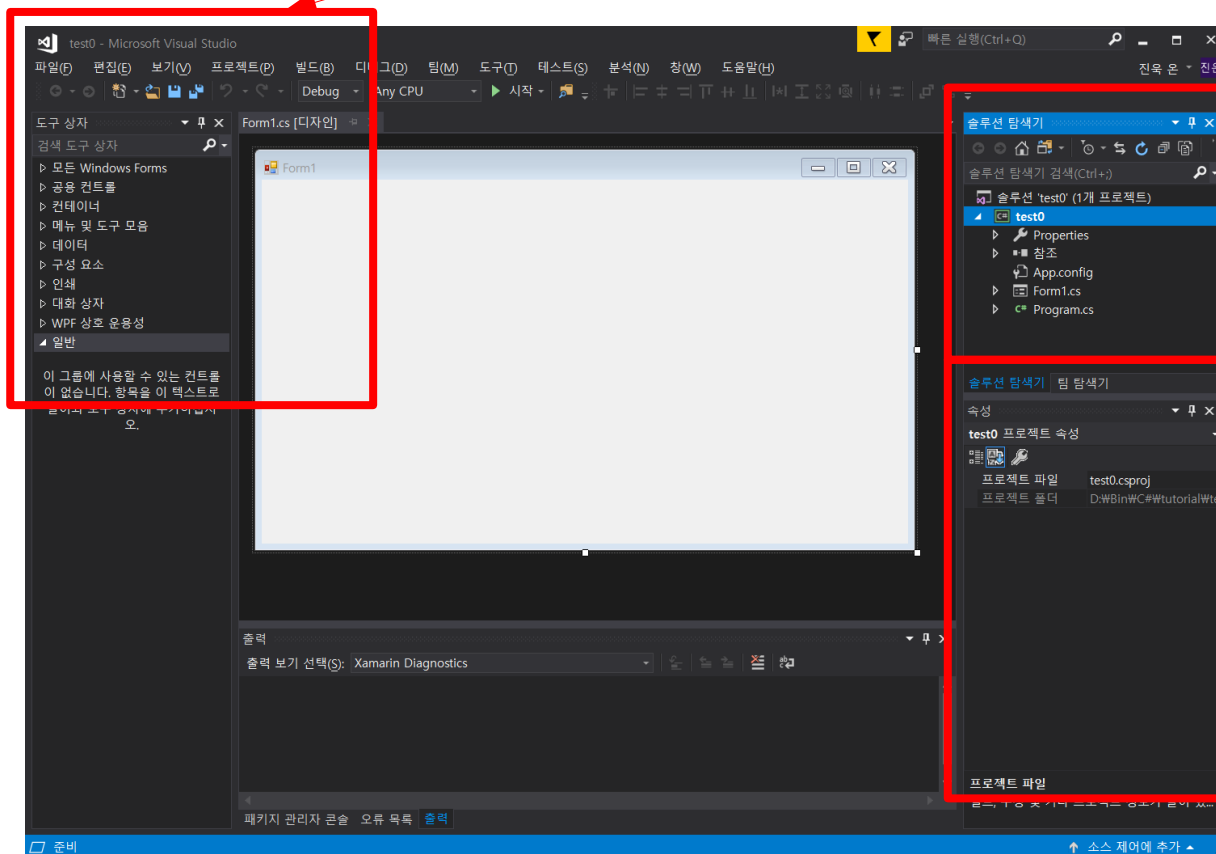
# 프로젝트의 시작

- 프로젝트 만들기



여기 프로젝트가 만들어질 공간

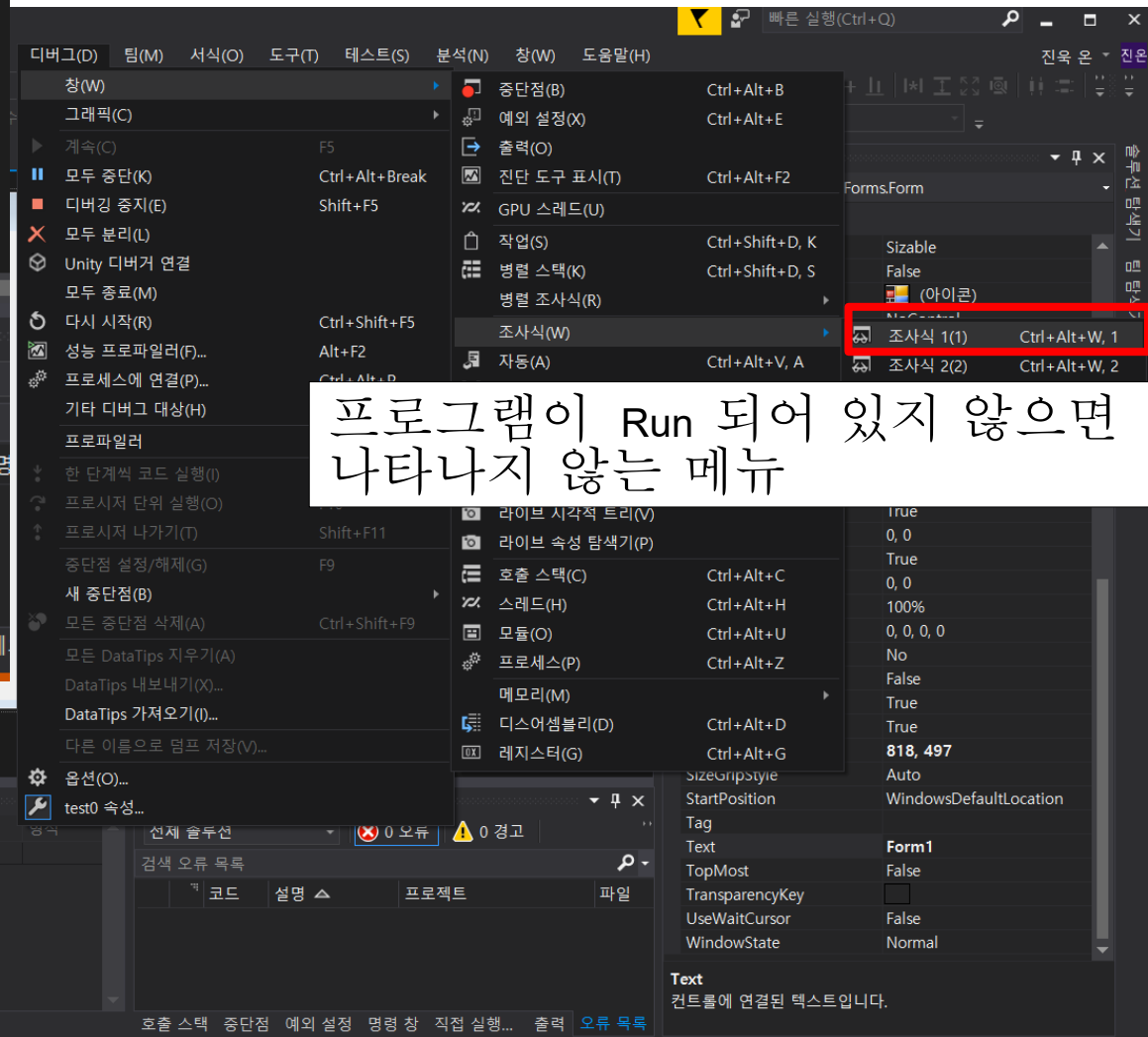
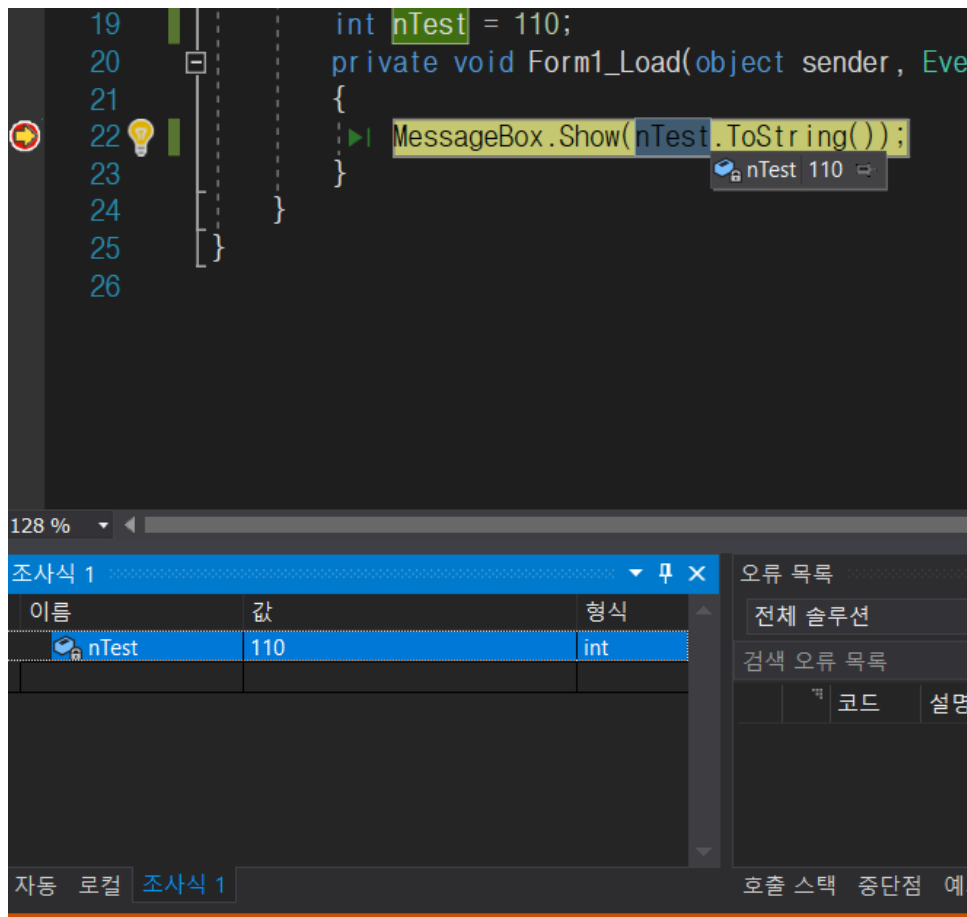
# 필요한 창



보기(V)	프로젝트(P)	빌드(B)	디버그(D)	팀(M)
<> 코드(C)			F7	
디자이너(D)			Shift+F7	
열기(O)				
다른 프로그램으로 열기(N)...				
솔루션 탐색기(P)			Ctrl+Alt+L	
팀 탐색기(M)			Ctrl+W, Ctrl+M	
서버 탐색기(V)			Ctrl+Alt+S	
Cloud Explorer			Ctrl+W, Ctrl+X	
SQL Server 개체 탐색기(S)			Ctrl+W, Ctrl+S	
책갈피 창(B)			Ctrl+K, Ctrl+W	
호출 계층 구조(H)			Ctrl+Alt+K	
클래스 뷰(A)			Ctrl+Shift+C	
코드 정의 창(D)			Ctrl+W, D	
개체 브라우저(J)			Ctrl+Alt+J	
오류 목록(I)			Ctrl+W, E	
출력(O)			Ctrl+Alt+O	
작업 목록(K)			Ctrl+W, T	
도구 상자(X)			Ctrl+Alt+X	
알림(N)			Ctrl+W, N	
찾기 결과(N)				
다른 창(E)				
도구 모음(T)				
전체 화면(U)			Shift+Alt+Enter	
모든 창(L)			Shift+Alt+M	
탭 순서(B)				
뒤로 탐색(Z)			Ctrl+-	
앞으로 탐색(F)			Ctrl+Shift+-	
다음 작업(Q)				
이전 작업(R)				
속성 창(W)			F4	

## 디버깅 도중의 필요 창

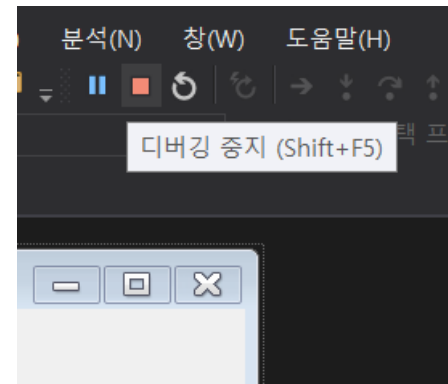
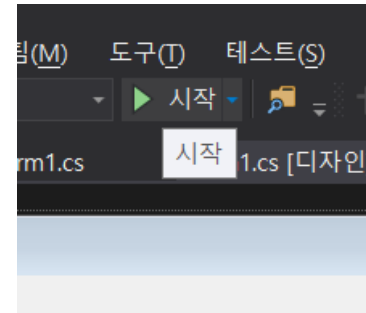
- 프로그램의 실행 도중 변수의 값을 확인해 볼 수 있다.



프로그램이 Run 되어 있지 않으면 나타나지 않는 메뉴

## 프로그램의 실행 및 종료

- 실행 (F5)
  - 혹은 “디버그” - “디버깅 시작”
  - 혹은 “시작” 버튼 클릭
- 정지 (Shift + F5)
  - 혹은 “디버그” - “디버깅 중지”
  - 혹은 “디버깅 중지” 버튼 클릭



## 구조

- Using

- DLL 등을 사용하기 위해서는 여기에 사용 선언을 해야 한다.

- Namespace

- 같은 네임스페이스
- 다른 네임스페이스간 불러오기
- 파일이 서로 달라도 같은 파일처럼 ... (partial)

- Class

- 변수나 기타 함수등은 여기에 선언한다.
  - 두번째 { } 안쪽에 선언

```
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;
10
11  namespace test0
12  {
13      public partial class Form1 : Form
14      {
15          public Form1()
16          {
17              InitializeComponent();
18          }
19
20          private void Form1_Load(object sender, EventArgs e)
21          {
22              //
23          }
24      }
25  }
26
```

두번째 { }

첫번째 { }

# 자주 쓰이는 컴포넌트

- 컴포넌트
  - 자주 쓰이는 컴포넌트

화면 표시 (글자)	명령	입력 받는 용	묶어 주는 용	반복 작업
• Label	• Button	• TextBox	• CheckBox	• RadioButton
		• ComboBox	• GroupBox	• Timer
			• Panel	

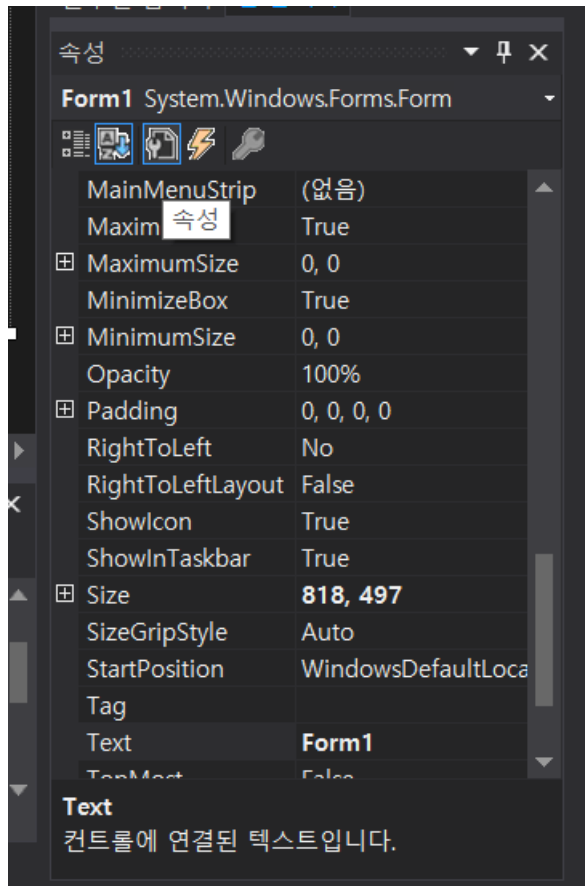
The screenshot shows the Visual Studio Windows Forms Designer. On the left is a 'Toolbox' containing various Windows Forms controls. On the right is a 'Form1' window with several controls placed on it. Red lines connect the component names in the table to the corresponding controls in the image:

- Label:** A line connects 'Label' to 'label1' on the form.
- Button:** A line connects 'Button' to 'button1' on the form.
- TextBox:** A line connects 'TextBox' to a text input box on the form.
- CheckBox:** A line connects 'CheckBox' to 'checkBox1' on the form.
- RadioButton:** A line connects 'RadioButton' to 'radioButton1' on the form.
- ComboBox:** A line connects 'ComboBox' to a dropdown menu on the form.
- GroupBox:** A line connects 'GroupBox' to 'groupBox1' on the form.
- Panel:** A line connects 'Panel' to a rectangular area on the form.
- Timer:** A line connects 'Timer' to 'timer1' in the 'Solution Explorer' on the right.

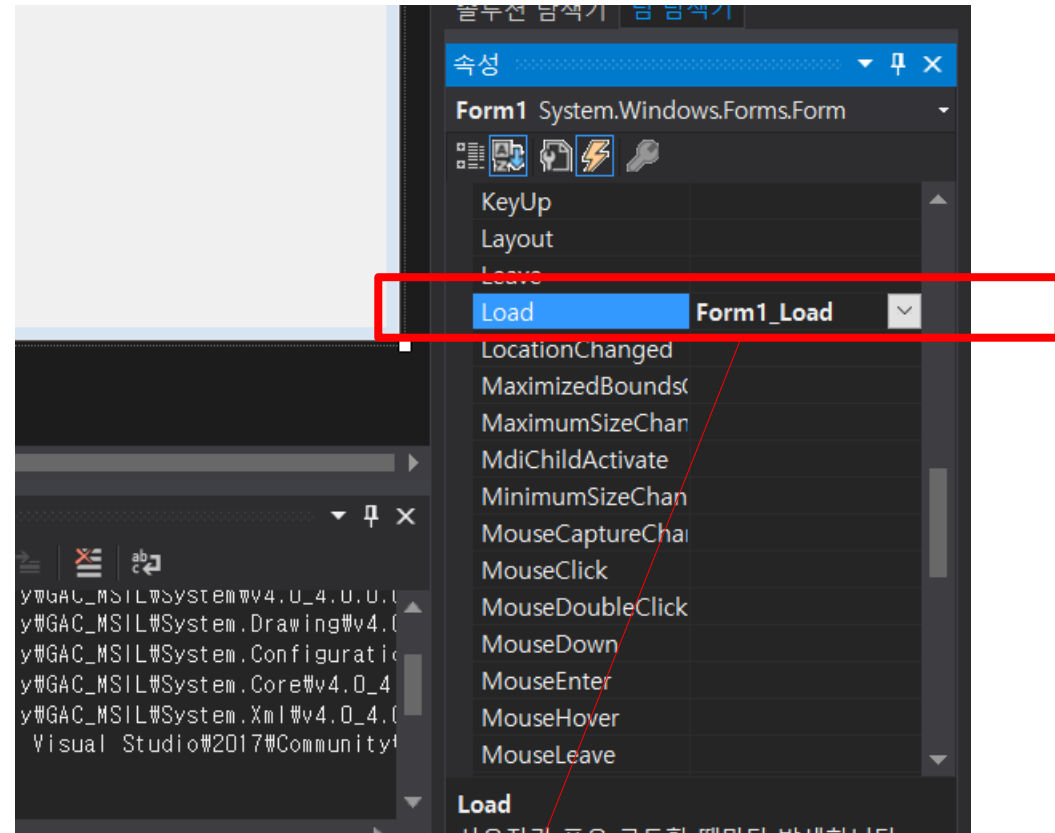
# 속성, 이벤트

- 속성

- [사전순] 정렬
  - 되도록 이걸로 ...
- [항목별] 정렬



- 이벤트

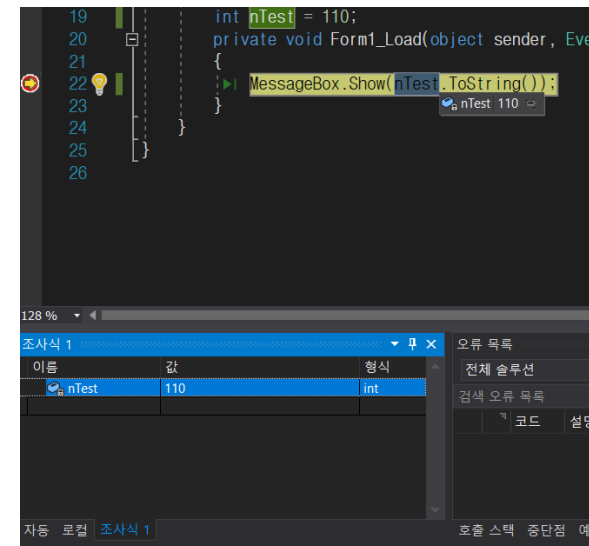


```
private void Form1_Load(object sender, EventArgs e)
{
    //
}
```



# 디버깅

- 중단점 (F9)
- 메시지의 출력
  - `System.Diagnostics.Debug.WriteLine("test");`
- 조사식 창에서의 변수값 검사
- 중단점에 걸린 이후는 ...
  - F5 : 그냥 갈래
  - F10 : 한 줄씩 가 볼래
  - F11 : 한 줄씩 가 볼래, 근데 더 자세하게 ...

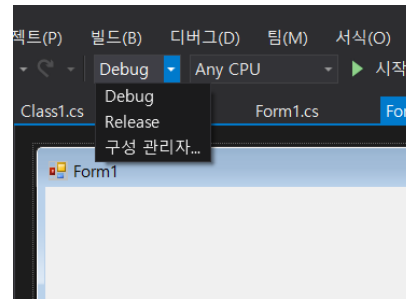


# 프로그램을 만들고 난 후 ...

- 실행

- Debug

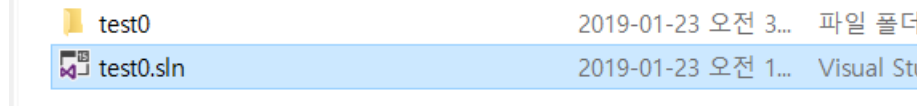
- Release : 실제 배포할 땐 이걸로 ...



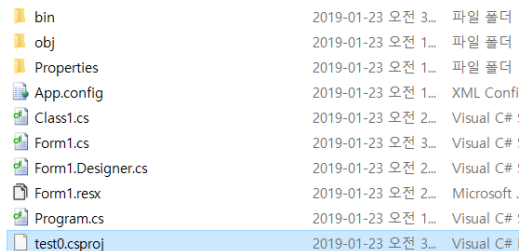
- 프로젝트의 저장

- 프로그램 다시 불러오기

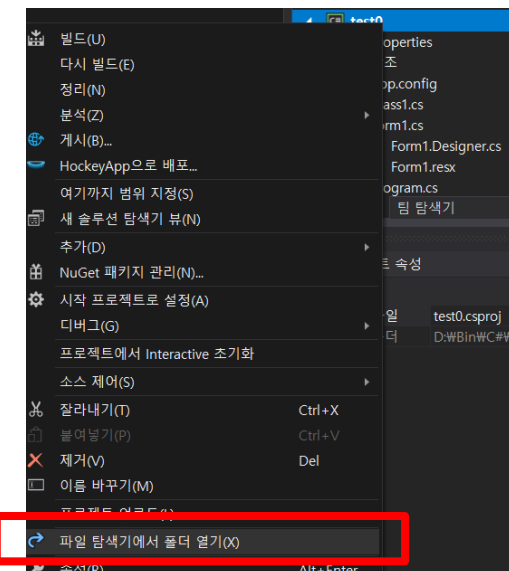
- 이걸 불러오는게 제일 좋지만 ...(.sln)



- 문제가 생긴다면 이걸로 불러오자 (.csproj)

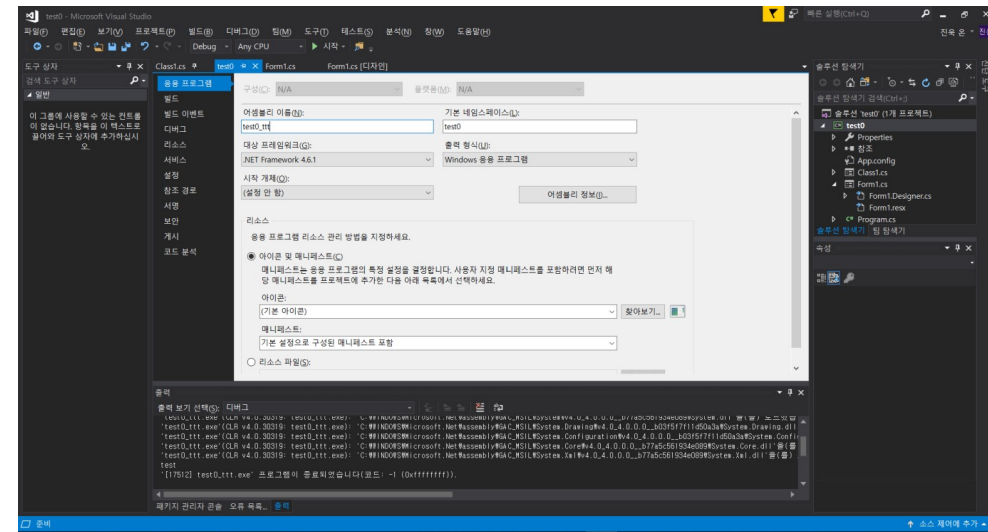
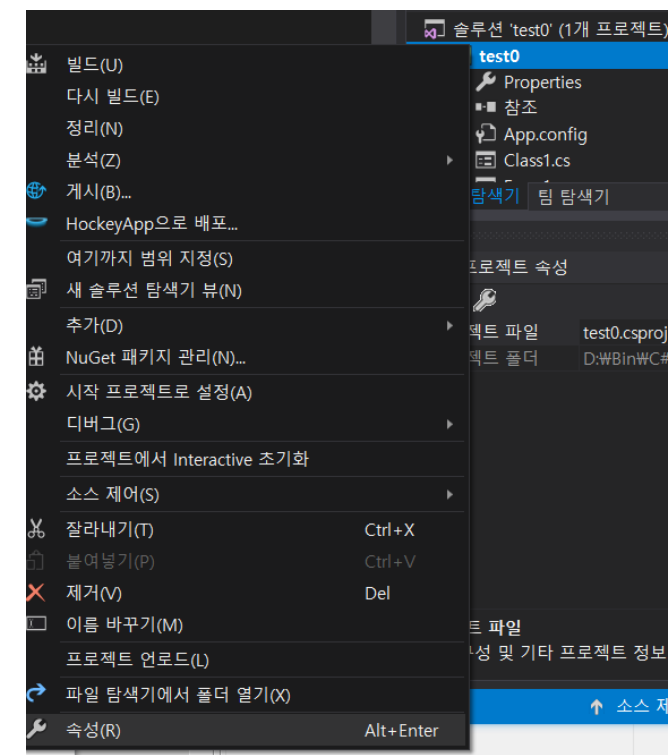


- 내 프로그램의 실행 위치 찾기



# 내가 만든 프로그램 꾸미기 ...

- 아이콘
  - 실행 후 작업 표시줄에 나타날 아이콘
  - Exe 실행 파일의 아이콘
- 프로그램 이름 바꿔보기
  - “내꺼 .exe” 라는 식으로 이름을 바꿔보자



# 변수

- 변수 ( 더 많이 있지만 ... 이것만 알아두자 )
  - 숫자
    - byte(Byte) : 0 ~ 255( 0xff 가 최대값 )
      - sbyte : -128 ~ 127
    - Short : -32768 ~ 32767 ( 2 bytes )
    - int(Int32) : -2,147,483,648 ~ 2,147,483,647 ( 4 bytes )
    - float(Single) :  $\pm 1.5 \times 10^{-45} \sim \pm 3.4 \times 10^{38}$ 
      - 6-9 개 자릿수 ( 4 bytes )
    - Double :  $\pm 5.0 \times 10^{-324} \sim \pm 1.7 \times 10^{308}$ 
      - 15-17 개 자릿수 ( 8 bytes )
  - 문자
    - string(String)
    - char(Char)
  - 참 / 거짓
    - bool(Boolean) : true, false 의 2 개의 값만 가짐
  - 그외 ...
    - var : 이걸 안 쓰는 걸로 ...
    - dynamic : 나중에 개인의 발전을 위해 되도록 안쓰셨으면 ...

ㄴ

## 변수명 ...

- 규칙
  - 변수이름은 숫자나 특수문자로 시작하면 안된다.
  - C, C++, C#의 언어는 변수 선언시 대소문자를 구분한다.
  - 내부에서 사용하고 있는 함수명이나 기타 정의된 것들은 변수명으로 사용할 수 없다.
- 변수명 표기(개인적으로 헝가리안 표기법, 카멜 표기법을 섞어쓰고 있음 - 요즈음은 이게 안좋다고 하는데... 습관들어서 난 이게 좋음...)
  - 헝가리안 표기법
    - 변수명 앞에는 변수의 특징을 알아볼수 있는 문자를 소문자로 넣어준다.
      - 예) `int nTest; float fTest; String strTest; Button btnTest;`
      - `bool bTest, short sTest, unsinged int unTest, byte byteTest;`
      - `int [] anTest; bool [] abTest;`
  - 카멜표기법
    - 낙타 등 ...
      - `bool bGoodMorning;`

## 상수 ...

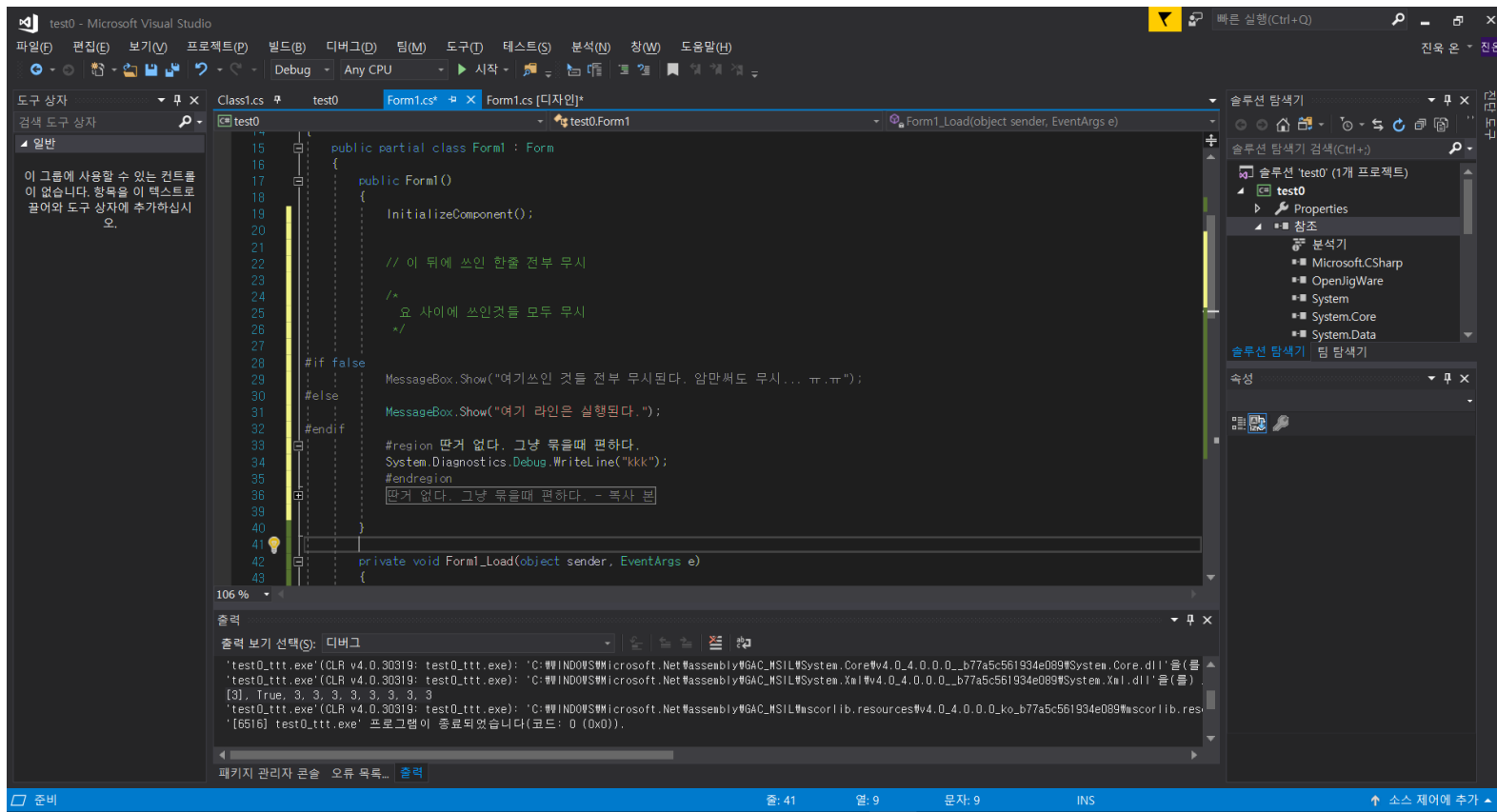
- 상수 : 한번 정하면 변하지 않는값 - 숫자만 말하는게 아니다.
  - 규칙
    - 초기값을 반드시 넣어주어야 한다.
    - 예) `const int _VALUE = 3;`
  - 관례
    - 상수는 모든 문자를 대문자로 표현해 준다.
    - 맨 앞 글자를 \_로 시작한다.(모든 프로그래머가 다 그런것은 아니다.)
    - 한번 정한 값은 중간에 바꿀수 없다.

# 문자를 숫자로, 숫자를 문자로

- 숫자를 문자로
  - 가장 간단한 방법
    - .ToString()
- 문자를 숫자로
  - `float fData = float.Parse("3");`
  - `float fData = Convert.Single("3");`
  - `int nData = Int.Parse("3");`
  - `int nData = Convert.Int32("3");`

# 알아두면 편리한 것들 ...

- `MessageBox.Show(" 문자 ");`
- 주석
  - 한줄 주석 : `//` 이 뒤로 쓰인것 전부 무시
  - 여러줄 주석 : `/*` 이 사이에 쓰인것들
- `#if ~ endif`
- `#region ~ #endregion`





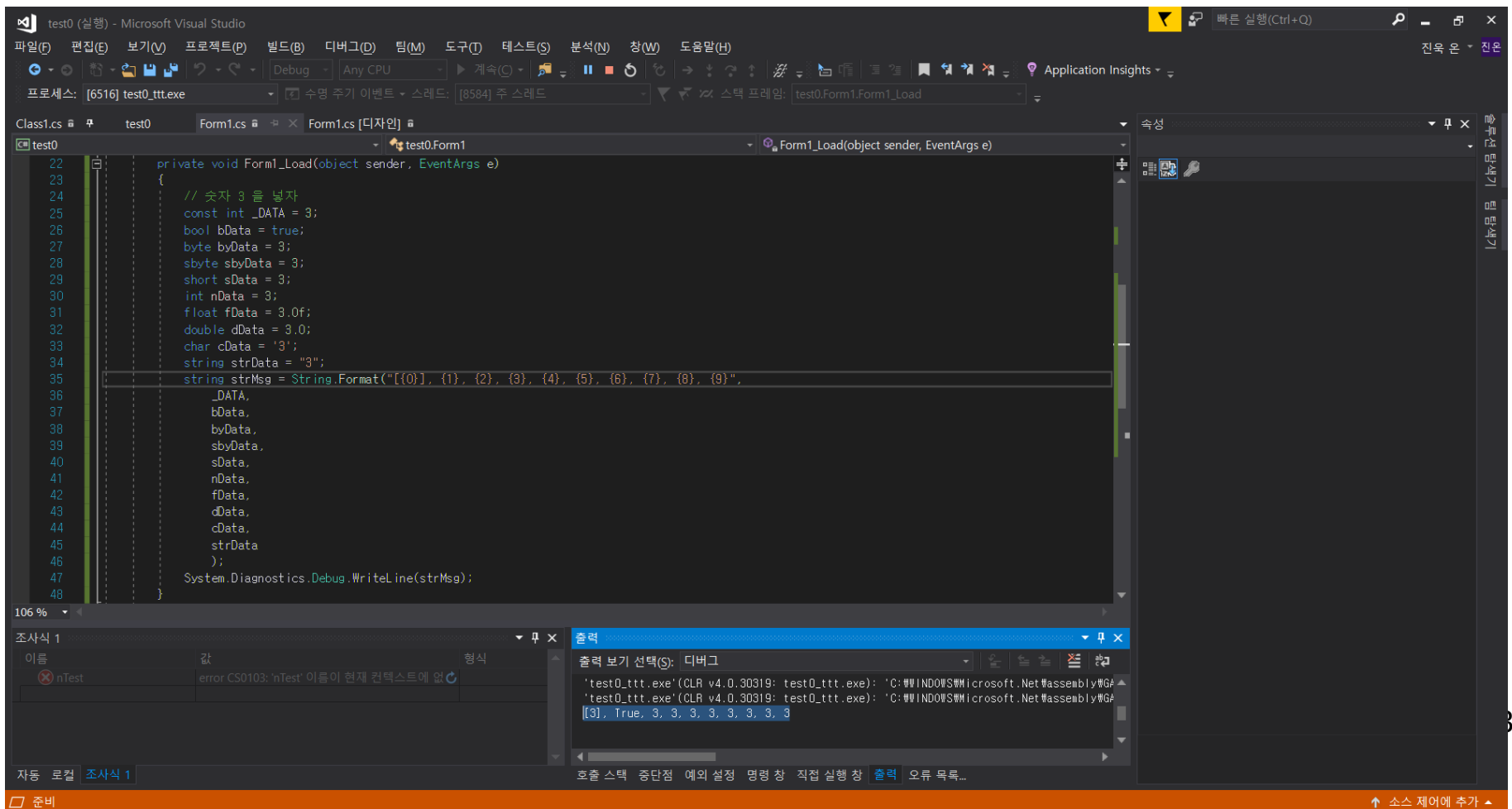


알아두면 편한 것들 ■■■

- `MessageBox.Show(“글자”);`
- 한 줄 주석 : `//, /* */`
- `#if ~ #end`

# 변수선언, 상수선언, 초기값 ...

- 디버그 출력 창에 변수의 값들을 출력해 보자.
  - 팁 1 : System.Diagnostics.Debug.WriteLine() 함수 사용
  - 팁 2 : 좀 짧게 쓰자 ...
  - 팁 3 ; 자동으로 using 추가
  - 팁 4 ; 자동완성 -> Ctrl+Space

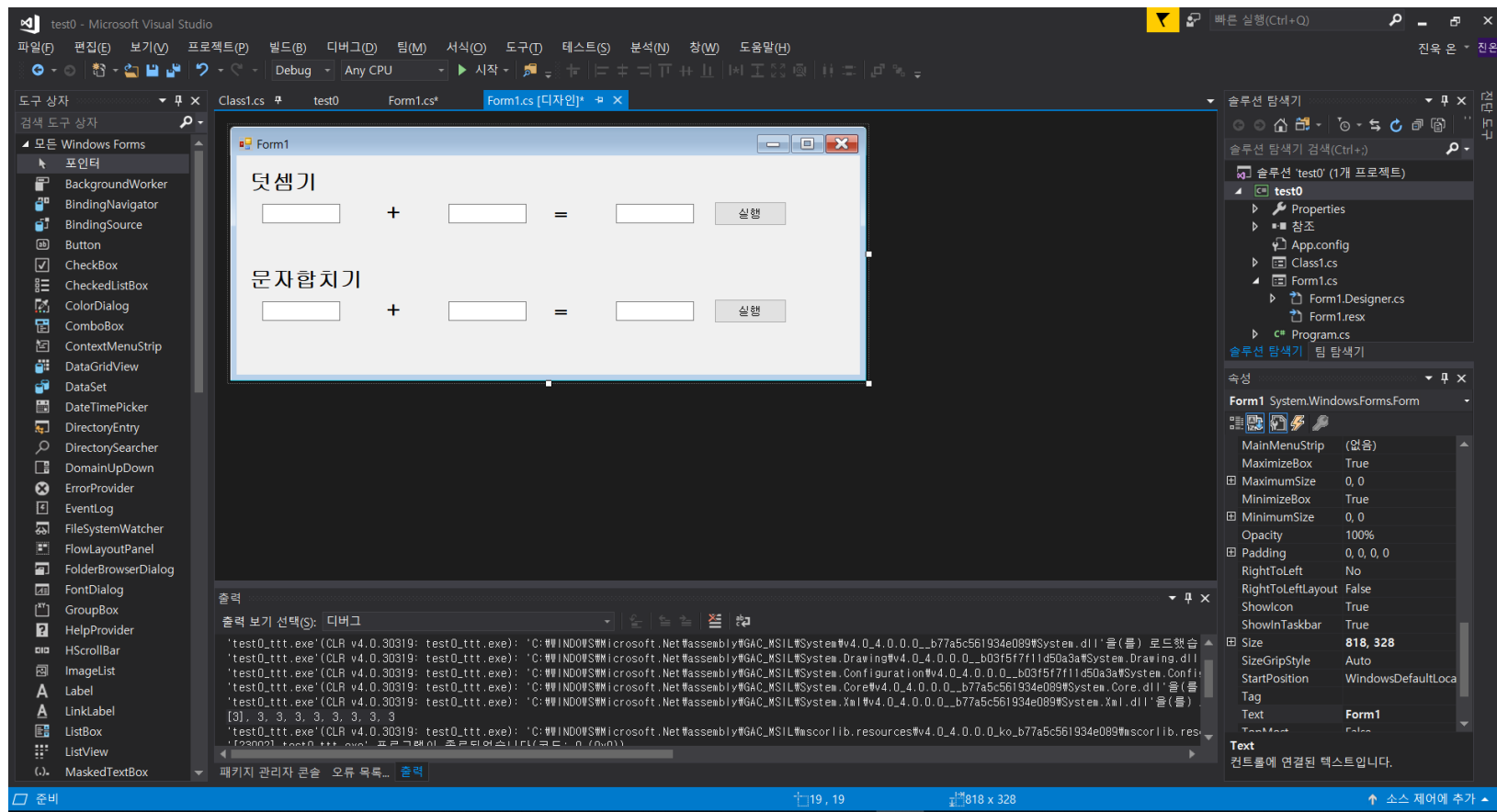


# 연산자

- 단항연산자 : +, -, ++, --
- 이항연산자 : +, -, \*, /, %
- 관계연산자 : >, >=, <, <=, ==, !=
- 논리연산자 &&, ||, &, |, !
- 비트연산자 : <<, >>, &, |, ^, ~

# 실습

- 간단한 덧셈기 만들기



# 조건문, 반복문

- 조건문
  - If
  - If ~ else
  - If ~ else if
  - If ~ else if ~ else
  - Switch ~ case
- 반복문
- for( 초기 ; 조건 ; 연산 )  
{  
반복  
}
- While, do ~ while, foreach, Break, continue, goto~

```
44 #if false
45 #else
69 #endif
70 for (int i = 0; i < 3; i++)
71 {
72     System.Diagnostics.Debug.WriteLine("* " + i.ToString());
73 }
74 #endif
75
76
77 private void Test()
78 {
79     System.Diagnostics.Debug.WriteLine("test");
80 }
81
82 private void button3_Click(object sender, EventArgs e)
83 {
84 }
85
86
87
88
```

106 %

조사식 1

이름	값	형식
nTest	error CS0103: 'nTest' 이름이 현재 컨텍스트에 없	

출력 보기 선택

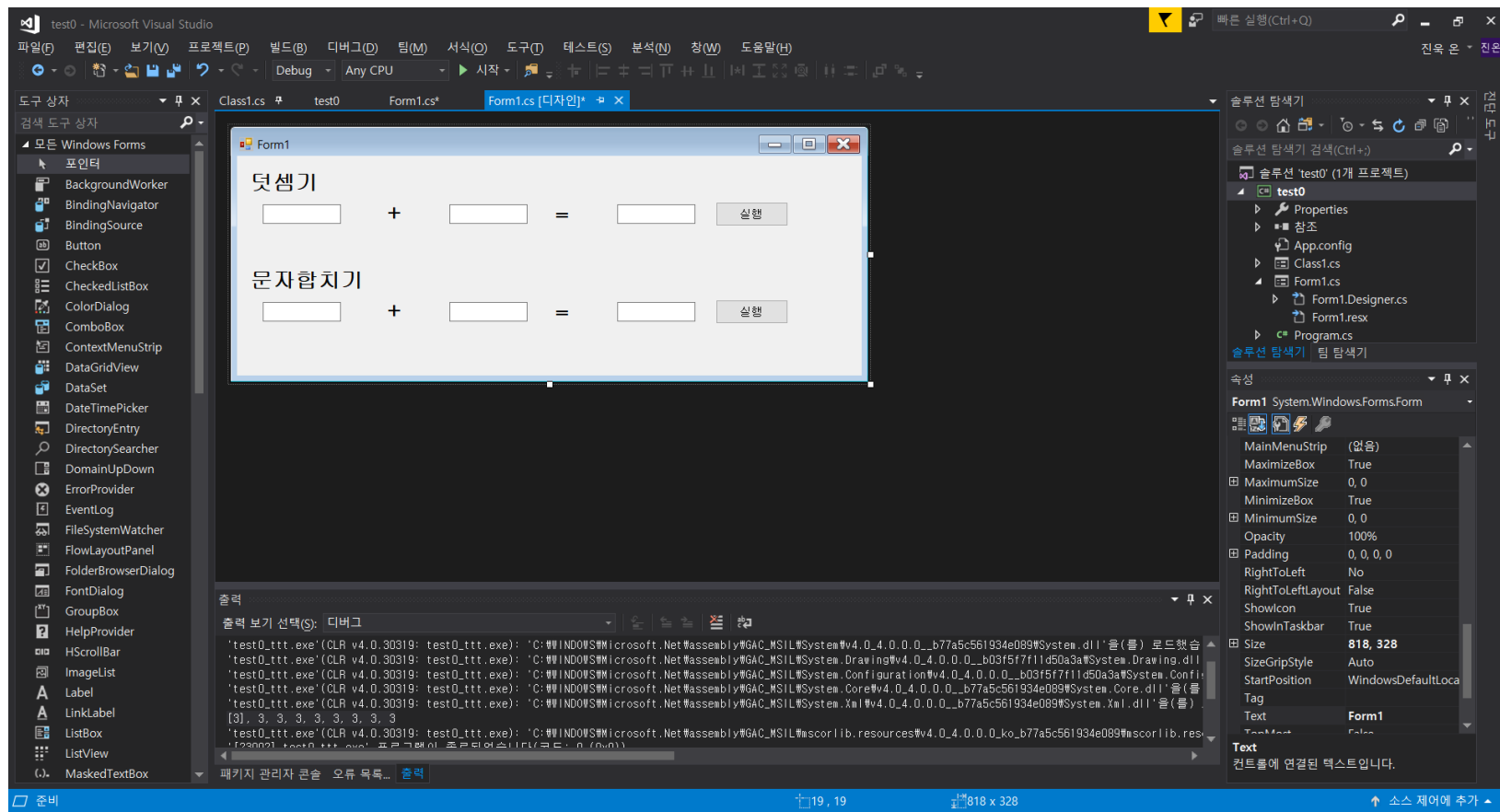
```
kkk
* 0
* 1
* 2
```

자동 로컬 조사식 1

준비

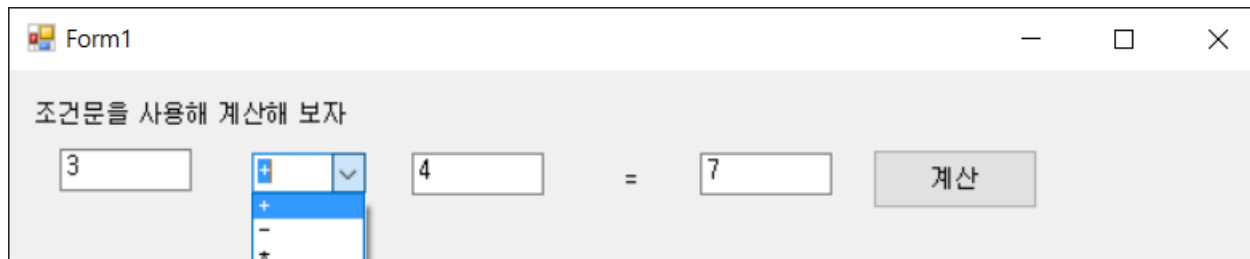
# 실습

- 간단한 덧셈기 만들기



## 실습 심화

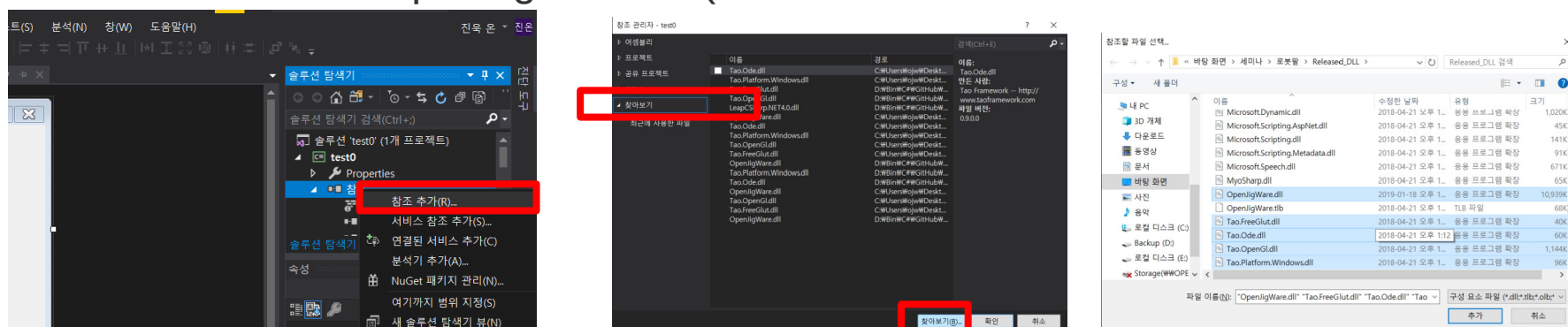
- 조건문을 활용한 (+,-,\*,/ 선택) 간단한 계산기 만들기



The screenshot shows a Windows-style application window titled "Form1". Inside the window, the text "조건문을 사용해 계산해 보자" (Let's try calculating using conditional statements) is displayed. Below this text, there is a simple calculator interface. It consists of two text input fields: the first contains the number "3" and the second contains the number "4". Between these fields is a dropdown menu with a blue header and a downward arrow; the dropdown is open, showing three options: "+", "-", and "\*". To the right of the dropdown is an equals sign "=" followed by a third text input field containing the number "7". Further to the right is a button labeled "계산" (Calculate).

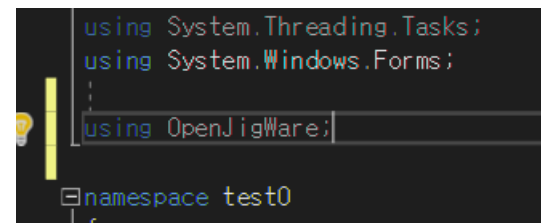
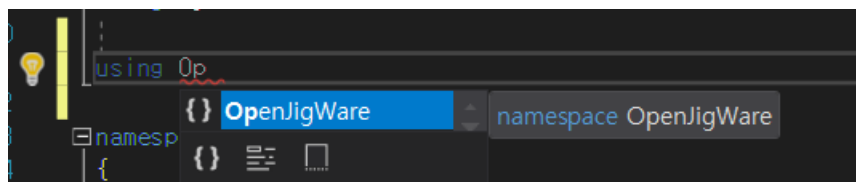
# 오픈지그웨어

- Release\_DLL 을 받아 저장
- 참조추가 - 파일찾기 - OpenJigWare.dll ( 3D 를 사용할 경우 Tao..... 라는 dll 4 개도 같이 ... )



3D 를 사용안한다면 OpenJigWare.dll 만 로드해도 된다.

- using OpenJigWare;



- 나중에 ...
  - 오픈지그웨어 함수내의 소스코드 살펴보기
  - 오픈지그웨어 안에 자신의 코드를 넣고 “내 DLL” 로 만들어 사용하는 방법 (COjw\_xx\_User.cs 를 활용)



# printf, scanf 준비

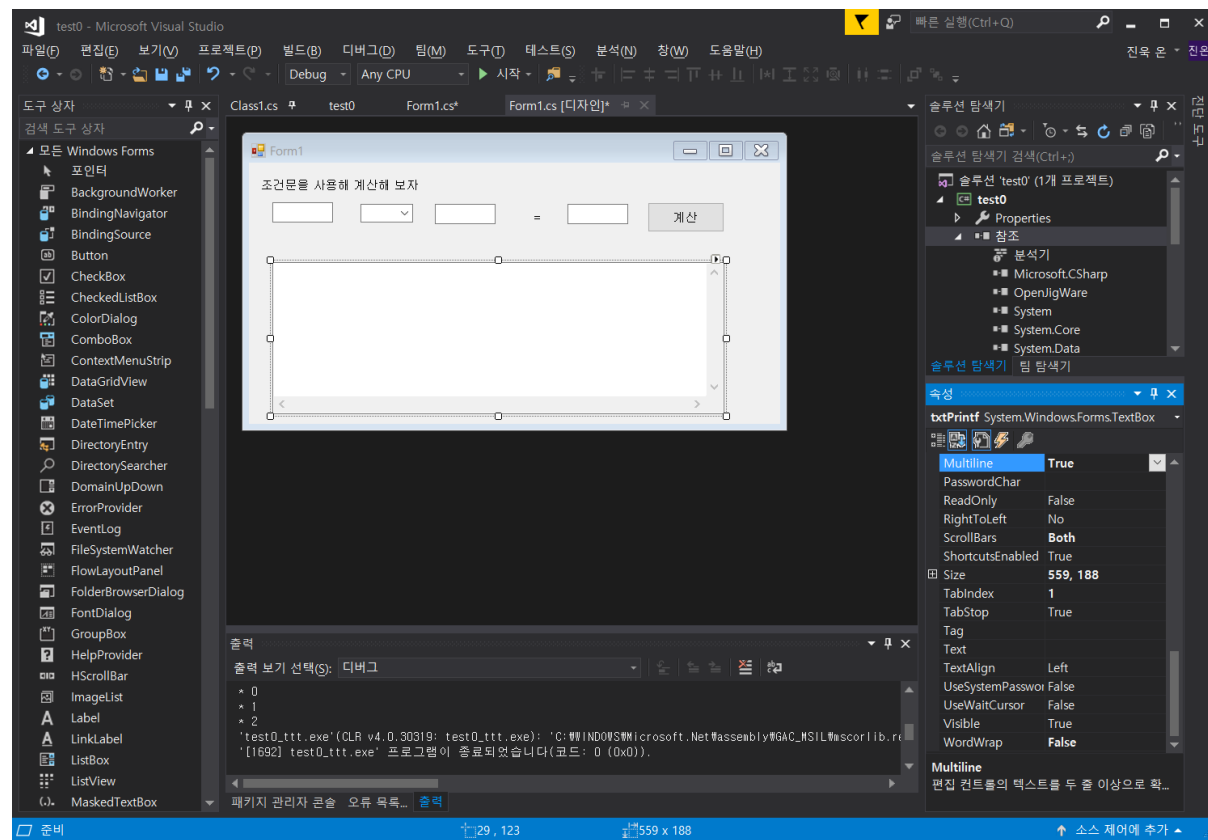
- TextBox

- Name : txtPrintf
- Multiline : True
- ScrollBars : Both
  - ( 안해도 상관없음 )
- WordWrap : false
- Ojw.printf\_Init(txtPrintf); // txtPrintf 에 출력할 거라고 ...

```
private void Form1_Load(object sender, EventArgs e)
{
    Ojw.printf_Init(txtPrintf);
}
```

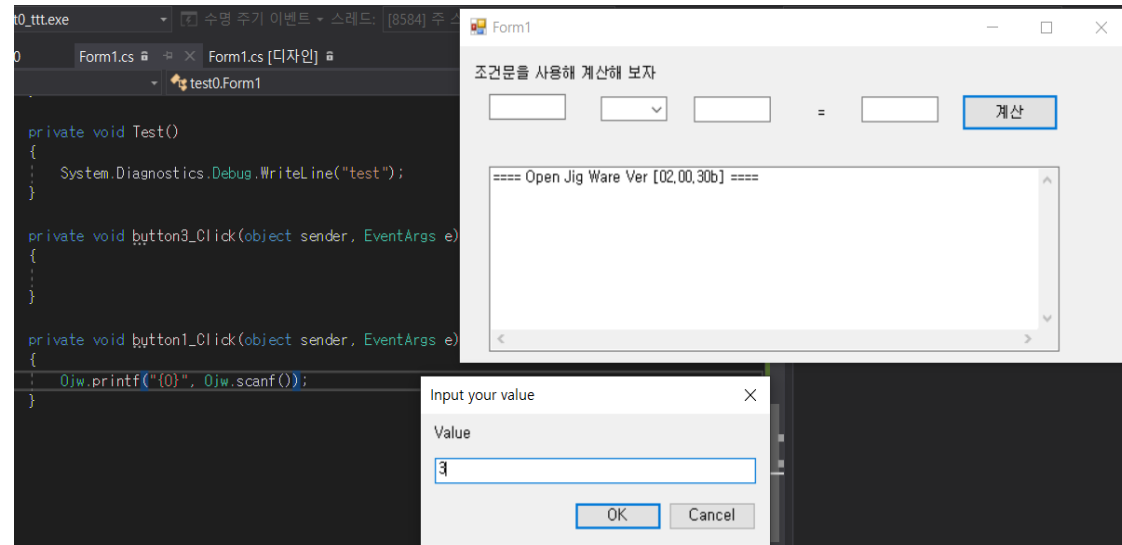
- 실습

- scanf() 를 이용해 문자를 입력받고 이를 출력하자.



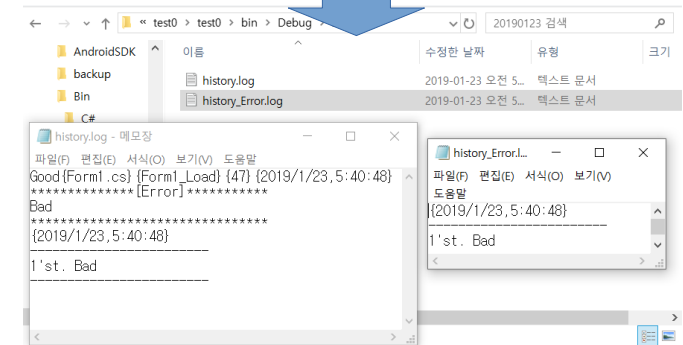
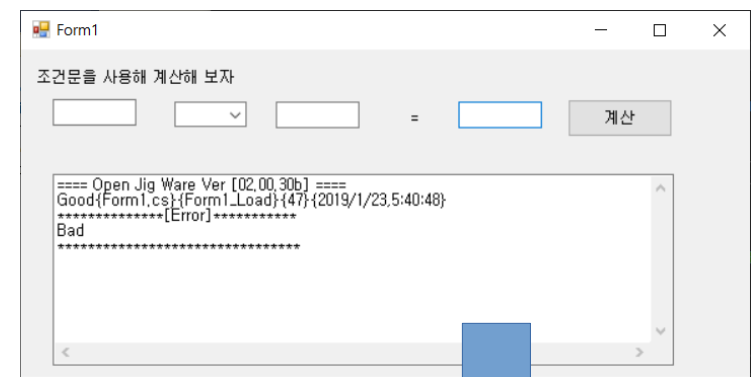
## 실습

- `scanf()` 로 값을 받아 `printf()` 로 출력하자
- `scanf()` 시 출력되는 메시지를 출력 안하게 ...
  - `scanf_SetAnswer(false);`



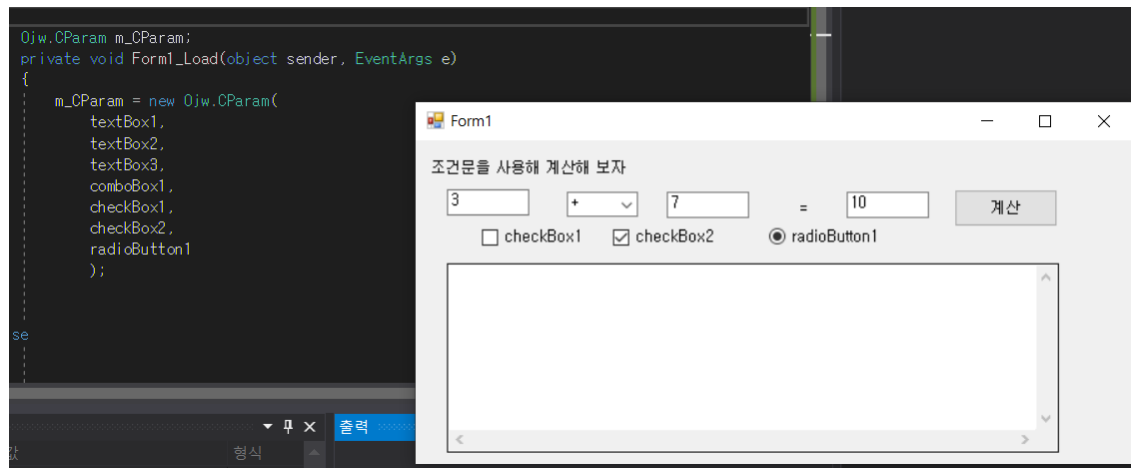
## printf 참고

- `Ojw.printf_SetDetail(true);` 를 하면 ...
- `Ojw.printf_Init_Error(txtError);` // 에러만 따로 표시
- `Ojw.printf_Error("에러메세지");` // 에러가 출력된다.
  - 에러 카운트도 올라가고, 에러 히스토리에 기록
- `Ojw.printf_File(true);` // 파일 기록을 시작한다.
  - 그날 그날의 날짜별로 새로 생성 (**Error** 를 발생하면 에러만 따로 관리하는 파일도 생성된다.)
- `Ojw.LogErr_Count();` → 현재 발생한 에러의 갯수를 가져옴
- `Ojw.LogErr_ClearMsg();` → 에러 히스토리 삭제
- `Ojw.Log_GetList();` → 발생한 에러메세지 전부를 가져옴
- `Ojw.Log_GetLastErrorMessage();` → 가장 최근에 발생한 에러 메세지를 가져옴



## Ojw.CParam - 컴포넌트의 값을 유지해 보자

- 각 컴포넌트의 값을 프로그램을 재시작해도 유지되게 해 보자
  - 변수선언 후 컴포넌트들의 이름을 적어 넣으면 된다.
  - Textbox, combobox, radiobutton, checkbox 의 4 종류 가능
  - 프로그램이 종료 될 때 **Save** 함수를 실행하게 한다.



- `FormClosing` 이벤트에 넣어준다. (취향따라 다르게 해도 상관없음)

```
private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    m_CParam.Param_Save();
}
```



Timer Class 부터는 다음 시간에 ...

**오랫동안 꿈을 그리는 사람은**

**마침내 그 꿈을 닮아간다 .**

**- 앙드레 말로**

**감사합니다 .**