

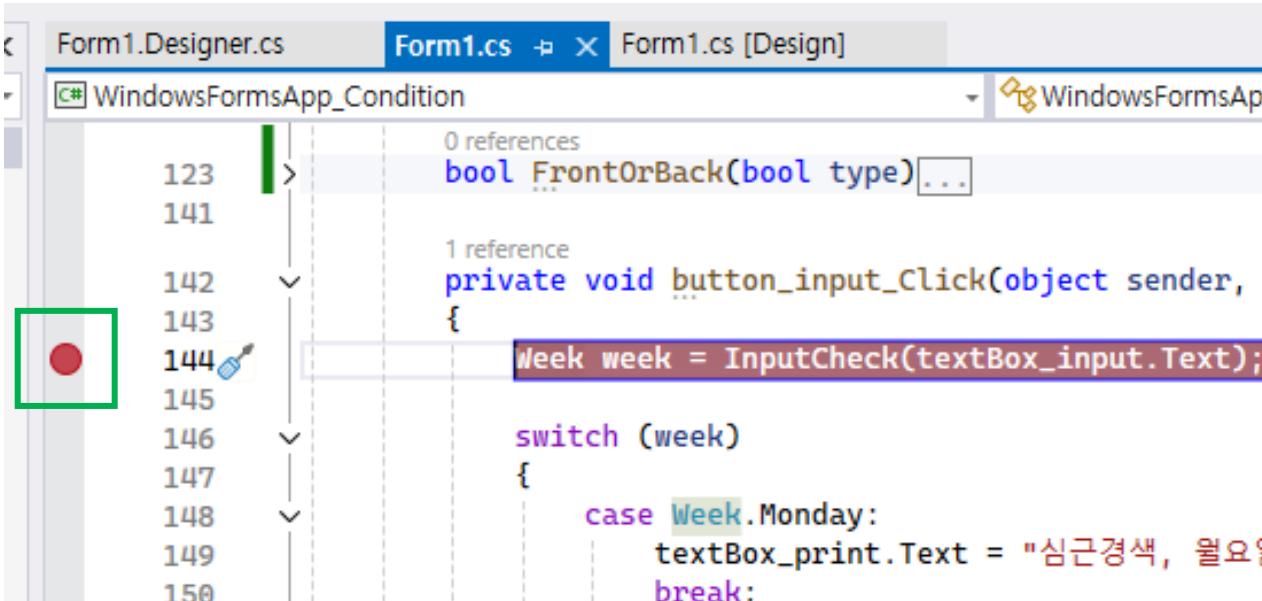
기타

디버깅

- 오류를 해결하고, 코드를 테스트하는데 가장 강력한 방법   
- 소스코드를 한 줄 단위로 실행하면서 변수에 담겨진 값의 변화를 추적
- F5로 디버그 모드 시작
- Shift + F5로 디버그 모드 종료
- F9로 브레이크 포인트를 설정하여 어느 줄에서 코드 실행을 멈출지 선택
- F10으로 한 줄 단위로 실행
- 함수를 만났을 때, F11로 함수 내부로 들어가는 것이 가능

디버깅

- 코드 줄 수 왼편에 회색 지점을 클릭하여 브레이크 포인트 생성
- 또는 단축키 F9 사용

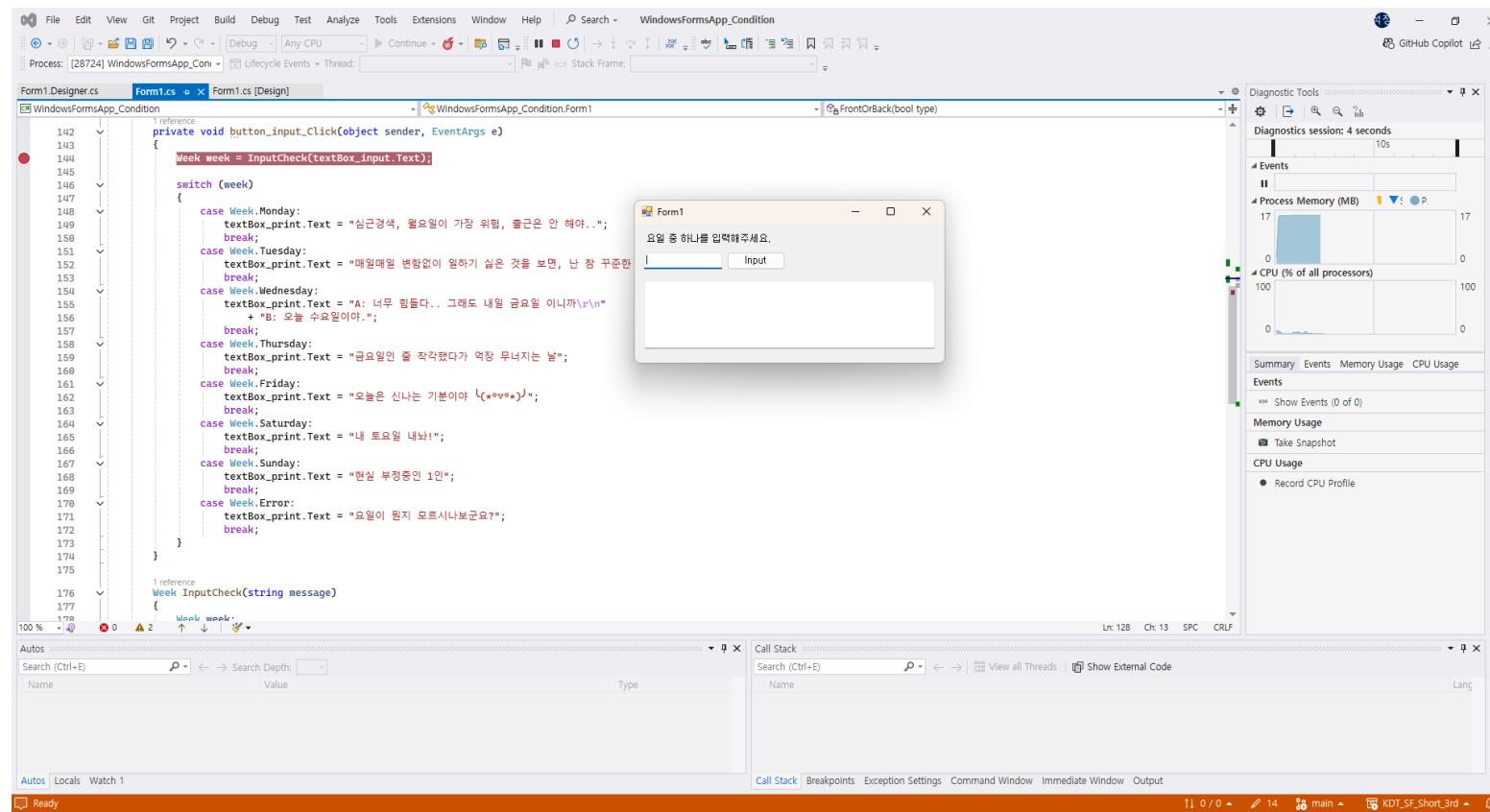


The screenshot shows the Visual Studio IDE with the code editor open. The tab bar at the top has 'Form1.Designer.cs', 'Form1.cs', and 'Form1.cs [Design]' tabs, with 'Form1.cs' being the active tab. The code editor displays C# code. A green square box highlights the line number 144, which contains the line 'week week = InputCheck(textBox_input.Text);'. A red circle, representing a breakpoint, is placed on the left margin next to the line number 144. The code itself includes a method definition 'bool FrontOrBack(bool type)' and a switch statement handling days of the week.

```
123 > 0 references  
141  
142 < 1 reference  
143  
144 bool FrontOrBack(bool type) ...  
private void button_input_Click(object sender,  
{  
    Week week = InputCheck(textBox_input.Text);  
    switch (week)  
    {  
        case Week.Monday:  
            textBox_print.Text = "심근경색, 월요일";  
            break;  
    }  
}
```

디버깅

- 디버그 모드가 시작되면 창 하단이 주황색으로 바뀜



디버깅

- 코드가 실행되면 노랑 화살표가 생기며 브레이크 포인트에서 멈춤
- F10 또는 F11 버튼으로 코드를 한 줄 단위로 실행하거나 함수 진입 가능

The screenshot shows the Visual Studio IDE with two tabs: Form1.Designer.cs and Form1.cs. The Form1.cs tab is active, displaying C# code. A red circular breakpoint is set at line 144. A yellow arrow cursor is positioned over the code at line 149. The code is as follows:

```
private void button_input_Click(object sender, EventArgs e)
{
    Week week = InputCheck(textBox_input.Text);
    switch (week)
    {
        case Week.Monday:
            textBox_print.Text = "Monday";
            break;
        case Week.Tuesday:
            textBox_print.Text = "Tuesday";
            break;
    }
}
```

Two green boxes highlight specific elements: one around the breakpoint icon at line 144, and another around the yellow arrow cursor at line 149.

디버깅

- 브레이크 포인트가 여러 개 일 경우 F5 키로 다음 브레이크 포인까지 코드 실행 가능

The screenshot shows a debugger interface with a vertical toolbar on the left containing three red circular breakpoints. A green square highlights the second breakpoint from the top, which has a yellow arrow icon indicating it is the current active point. The code editor on the right displays the following C# code:

```
78 string animal = "Cat";
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
```

switch (animal)
{
 case "Dog":
 // animal == "Dog" 일때 실행되는 코드
 break;

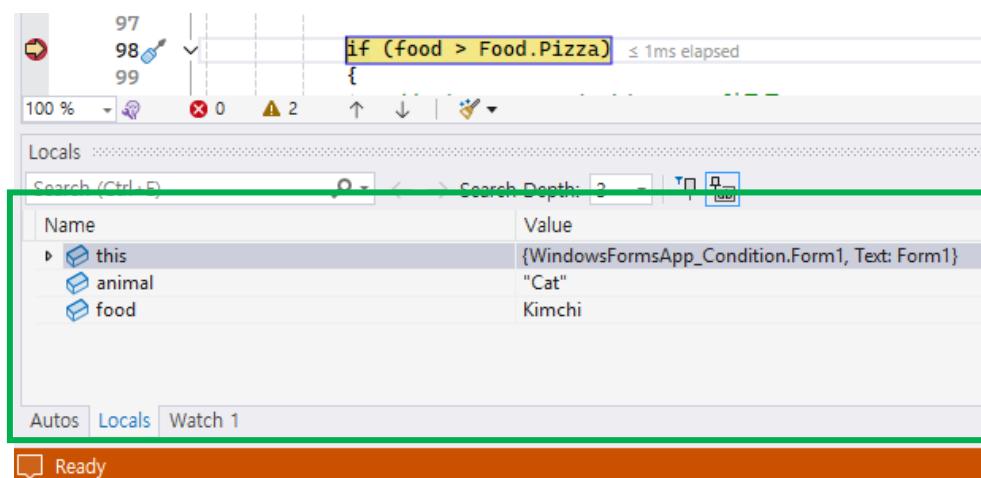
 case "Cat":
 // animal == "Cat" 일때 실행되는 코드
 break; ≤ 1ms elapsed

 default:
 // 위 case들에 모두 해당되지 않는다면 실행
 break;
}

A horizontal timeline at the bottom of the code editor shows the execution flow. The cursor is positioned over the second 'break' statement at line 89. A tooltip indicates that the time elapsed since the previous instruction was less than 1 millisecond.

디버깅

- 코드 실행이 멈춘 시점에서 같은 스코프에 있는 변수들의 값을 확인 가능
- Autos: 최근에 변화된 변수의 값
- Local: 같은 스코프에 있는 변수의 값
- Watch 1: 직접 Watch로 등록한 변수의 값



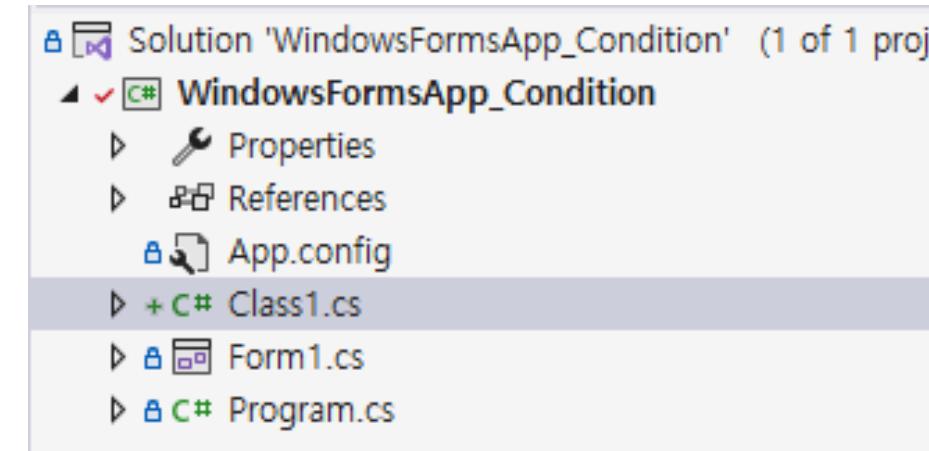
적극적으로 이용해주세요!!



소스 파일 활용

- 새로운 파일을 생성하면 파일 이름으로 클래스가 생성됨
- 클래스는 “인스터스”라는 것을 만들어서 클래스에 작성한 변수 및 함수를 동작 시킬 수 있음
 - 자세한 설명은 이후 클래스를 배우면서 진행

```
namespace WindowsFormsApp_Condition
{
    0 references
    internal class Class1
    {
    }
}
```



소스 파일 활용

- 클래스에 함수를 작성하고 클래스의 인스턴스를 통해 함수를 사용 가능

0 references

```
internal class Class1
{
    0 references
    public int Sum(int num1, int num2)
    { public을 불여줘야 사용가능
        return num1 + num2;
    }
}
```

}

1 reference

```
public Form1()
{
    InitializeComponent();

    Class1 myFunctions = new Class1();
    Class 이름, 인스턴스 이름 -> 인스턴스 생성
    int result = myFunctions.Sum(100, 300);
    인스턴스 이름.클래스 내부에 선언된 함수
}
```

소스 파일 활용

- 클래스에 함수를 작성하고 클래스의 인스턴스를 통해 함수를 사용 가능

2 references

```
internal class Class1
{
    public double d_value = 0.0;
}
```

1 reference

```
public int Sum(int num1, int num2)
{
    return num1 + num2;
}
```

}

1 reference

```
public Form1()
{
    InitializeComponent();
    Class1 myFunctions = new Class1();
    int result = myFunctions.Sum(100, 300);
    double value = myFunctions.d_value;
}
```

public을 붙여주면 변수도 사용 가능

미니 프로젝트. 윈도우 계산기 따라서 만들기

1. 팀장님의 Remote Repo. 및 솔루션 만들기
 - Fork 또는 Collaborator 편한 방법을 사용하되 어느 쪽이든 Pull Request는 사용
2. 버튼 위치, 숫자 표기 방식, 내역 보기 등등 여러가지 기능 중 현재 팀 멤버로 구현 가능한 부분과 불가능한 부분을 추리기
 - 구현이 어려운 기능은 다른 형태로 타협 가능 (예, 계산 내역은 새창으로 띄우기)
 - 참고로 표준, 공학용, 그래프, 프로그래머, 날짜 계산 등등 여러 기능이 있음
3. 추려진 기능 목록을 갖고 역할을 나누기
4. 충돌이 지나치게 발생하지 않도록 각자 어떤 파일을 어떤 이름으로 작성할지 미리 협의하기
5. 문서 작성이 완료되면 리더님께 전달 후, 개발을 시작하기

