

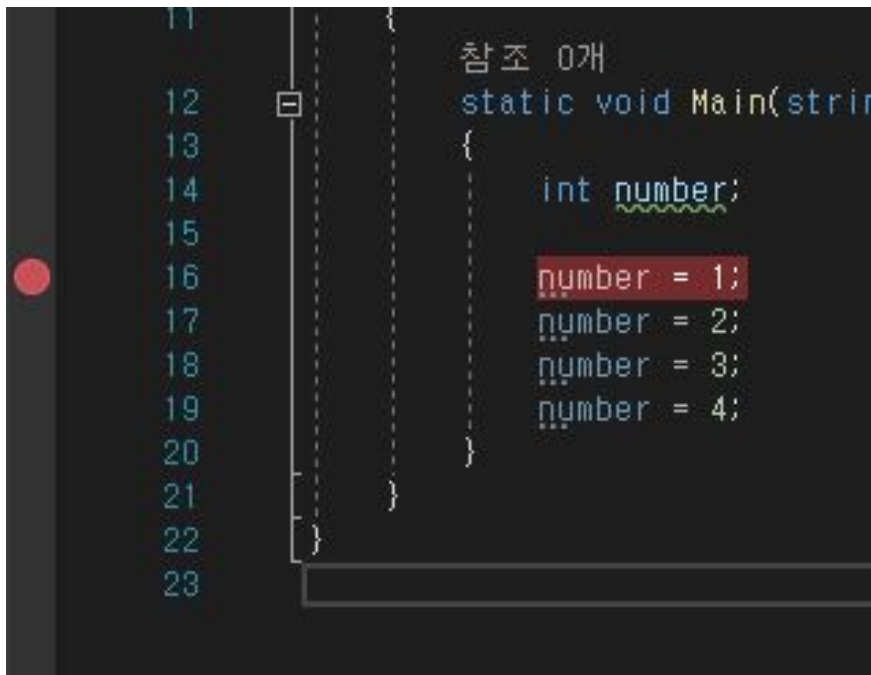
추가 내용

디버깅

null, static과 main

프로젝트 만들기

Branch

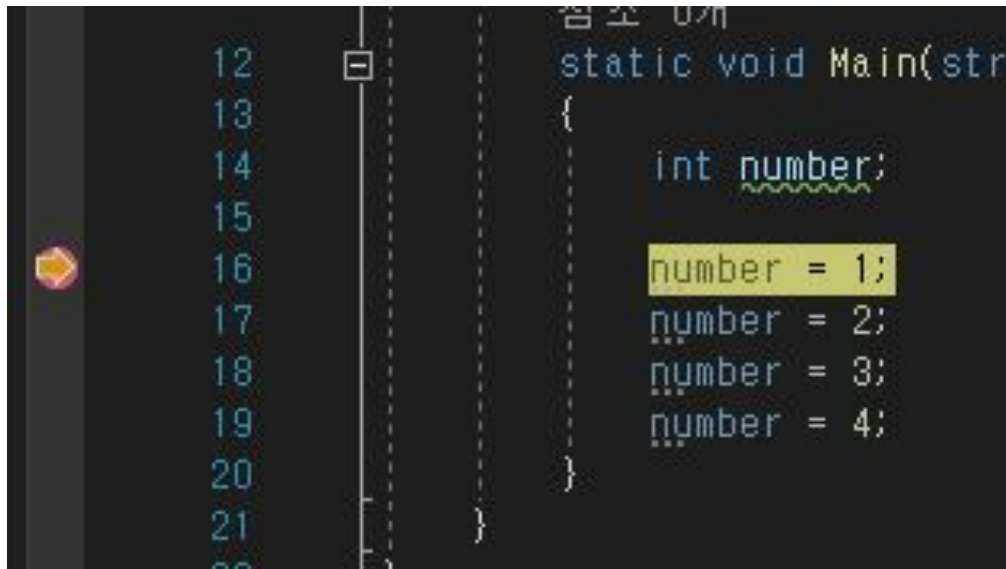


## 디버깅

개발자가 개발을 편하게 하기 위해

오류가 있는지 쉽게 찾아보기 위한 도구

줄 넘버가 적혀있는 곳 왼쪽을 클릭하거나,  
줄에서 **F9**를 누르면 빨간 구슬이 뜬다.



```
12 static void Main(str  
13 {  
14     int number;  
15     number = 1;  
16     number = 2;  
17     number = 3;  
18     number = 4;  
19 }  
20  
21  
22
```

이때 **ctr + F5** 가 아닌

**F5**를 눌러서 실행한다  
(컴파일한다)

값(number)위에 마우스를 올려놓으면  
지금 현재 값이 나온다.

이때, 값은 해당 줄(구문)을 실행하기 전  
데이터이다.

즉, 16번째 줄에 number위에 마우스를  
올려놓으면 0이라고 뜬다.

```
12  
13  
14     int number;  
15  
16     number = 1;  
17     number = 2;  
18     number = 3;  
19     number = 4;  
20  
21  
22
```

## F10

F10을 누르면 해당 줄을 읽는다.

F5를 누르면 계속 진행하게 된다.

# Null

값이 비어있다는 것을 보여준다.

예) `string newString = null;`



0



null



undefined

```

namespace ConsoleApp1
{
    참조 0개
    internal class Program
    {
        참조 0개
        static void Main(string[] args)
        {
        }
    }
}

```

## Static - 정적변수

어디서든지 사용할 수 있다.

**static**으로 선언된 함수, 클래스 안에서 모두 **static**으로 선언한 것들만 사용해야 한다.

여기서 **void**는 공허한, 비었다 라는 것을 뜻한다.

```
{
    참조 0개
    internal class Program
    {
        참조 0개
        static void Main(string[] args)
        {
        }
    }
}
```

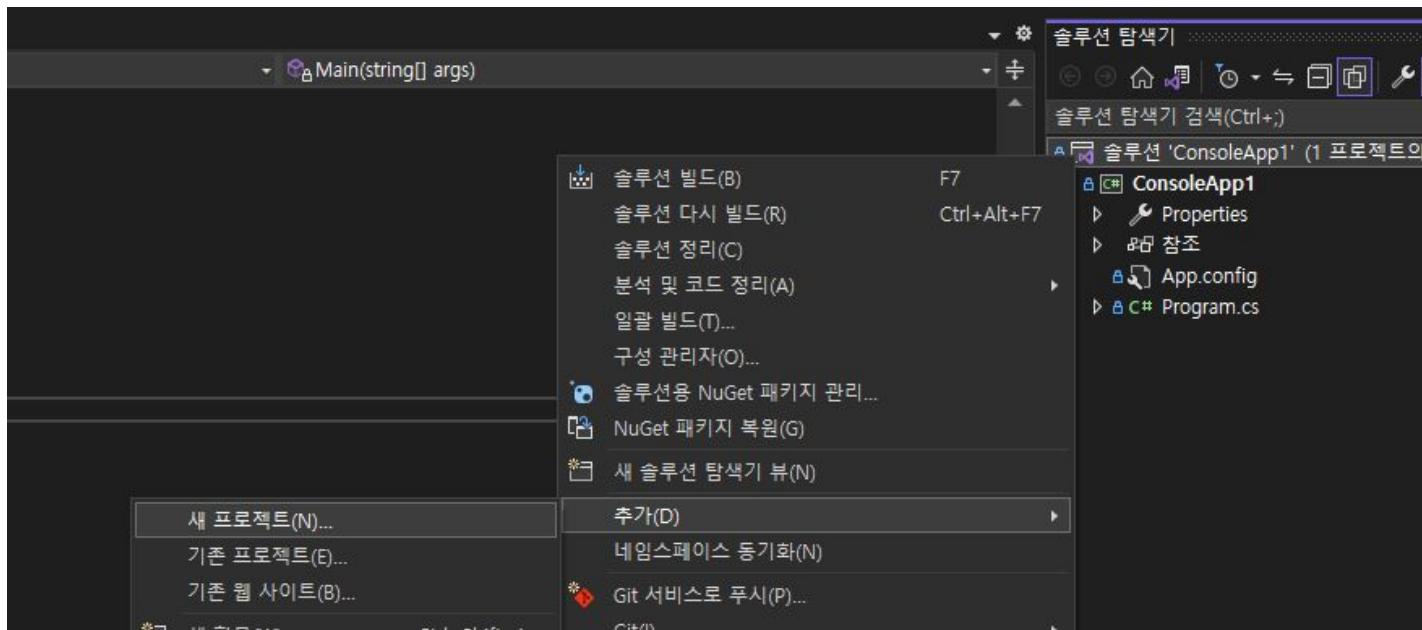
## Main

C# 콘솔 앱 프로젝트에서

Main만 실행된다.

즉, 모든 코드는 Main에서 실행되어야 한다.

Main은 반드시 위에서부터 아래 순서로 실행된다.



## 새로운 프로젝트 만들기

솔루션이란 파일 안에 여러개의 프로젝트를 만들 수 있다.

솔루션 탐색기 안에서 맨 위의 솔루션을 오른쪽 클릭해서 위 그림과 같이 새프로젝트 만든다.



## 프로젝트 추가

### 근 프로젝트 템플릿(R)

콘솔 앱(.NET Framework)

C#

콘솔 앱

C#

콘솔 앱

모두 지우기(C)

모든 언어(L)

모든 플랫폼(P)

모든 프로젝트 형식(T)

C# 콘솔 앱

Windows, Linux 및 macOS의 .NET에서 실행할 수 있는 명령줄 응용 프로그램을 만들기 위한 프로젝트

C#

Linux

macOS

Windows

콘솔

VB 콘솔 앱

Windows, Linux 및 macOS의 .NET에서 실행할 수 있는 명령줄 응용 프로그램을 만들기 위한 프로젝트

Visual Basic

Linux

macOS

Windows

콘솔

C++ 콘솔 앱

Windows 터미널에서 코드를 실행합니다. 기본적으로 "Hello World"를 출력합니다.

C++

Windows

콘솔

C# 콘솔 앱(.NET Framework)

명령줄 애플리케이션을 만드는 프로젝트입니다.

C#

Windows

콘솔

VB 콘솔 앱(.NET Framework)

명령줄 애플리케이션을 만드는 프로젝트입니다.

Visual Basic

Windows

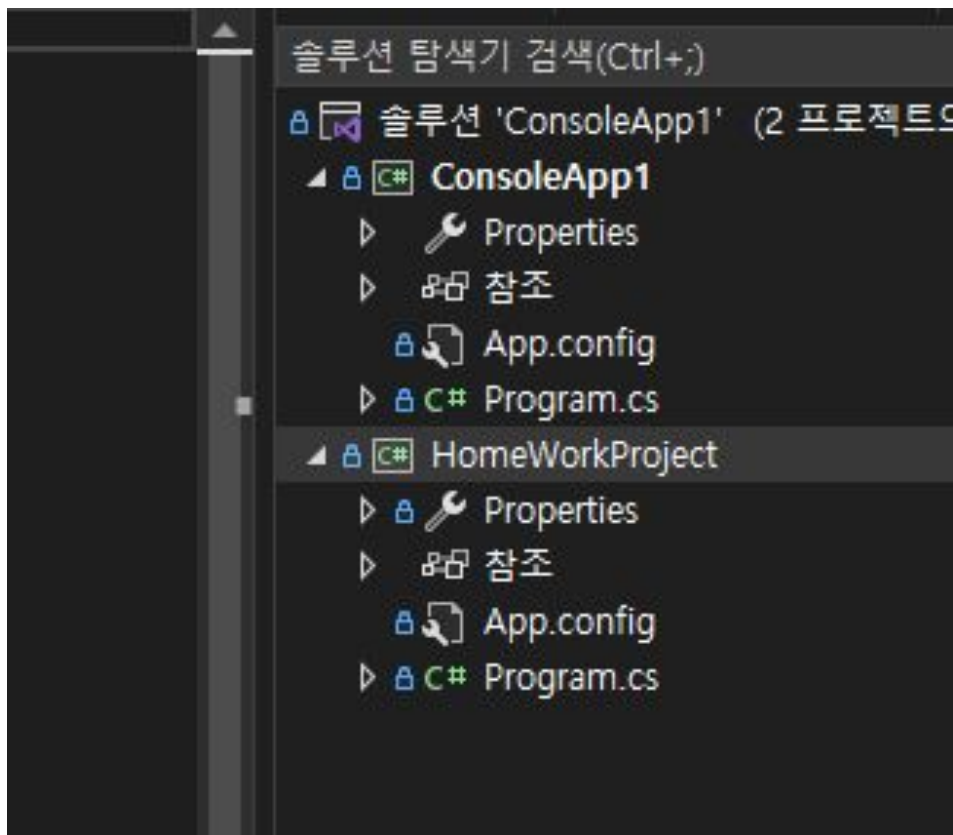
콘솔

F# 콘솔 앱

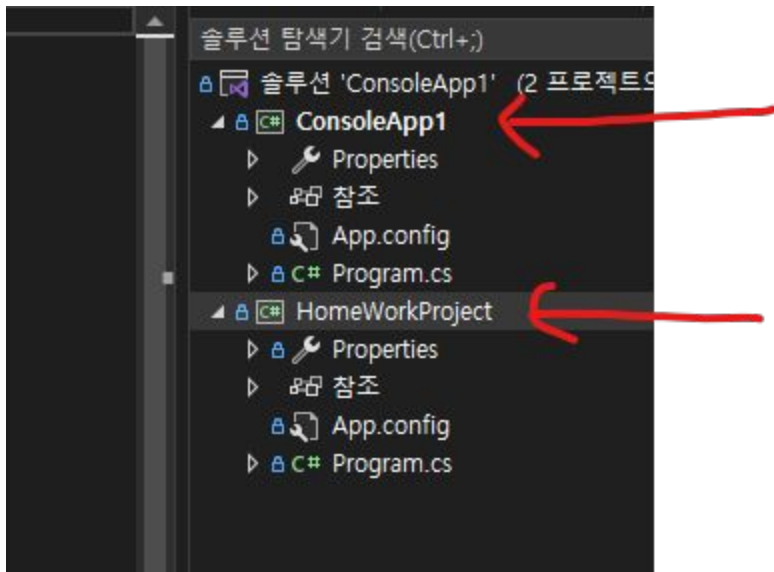
Windows, Linux 및 macOS의 .NET에서 실행할 수 있는 명령줄 응용 프로그램을 만

다음(N)

# 콘솔앱 .net 으로 만들기



프로젝트가 추가된다.



## 시작 프로젝트 지정

왼쪽 위 그림을 보면

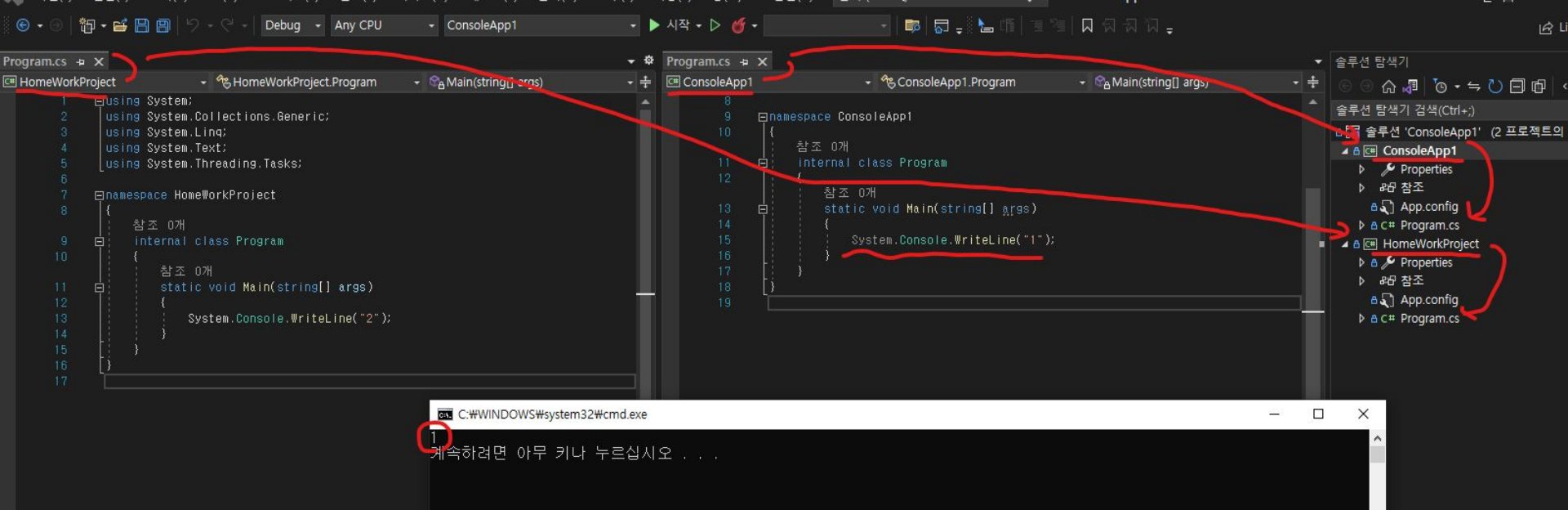
위 프로젝트는 두꺼운 글씨이고,  
두번째 프로젝트는 얇다.

ctr + f5를 누르면 컴파일(실행)하게 되는데,  
두꺼운 글씨의 프로젝트"만" 실행이 된다.

따라서, 이 두꺼운 글씨를 바꿔줘야 하는데  
이는 시작 프로젝트 라고 한다.

실행을 하고자 원하는 프로젝트  
우클릭누르고 시작 프로젝트 설정을 하면  
수정할 수 있다.

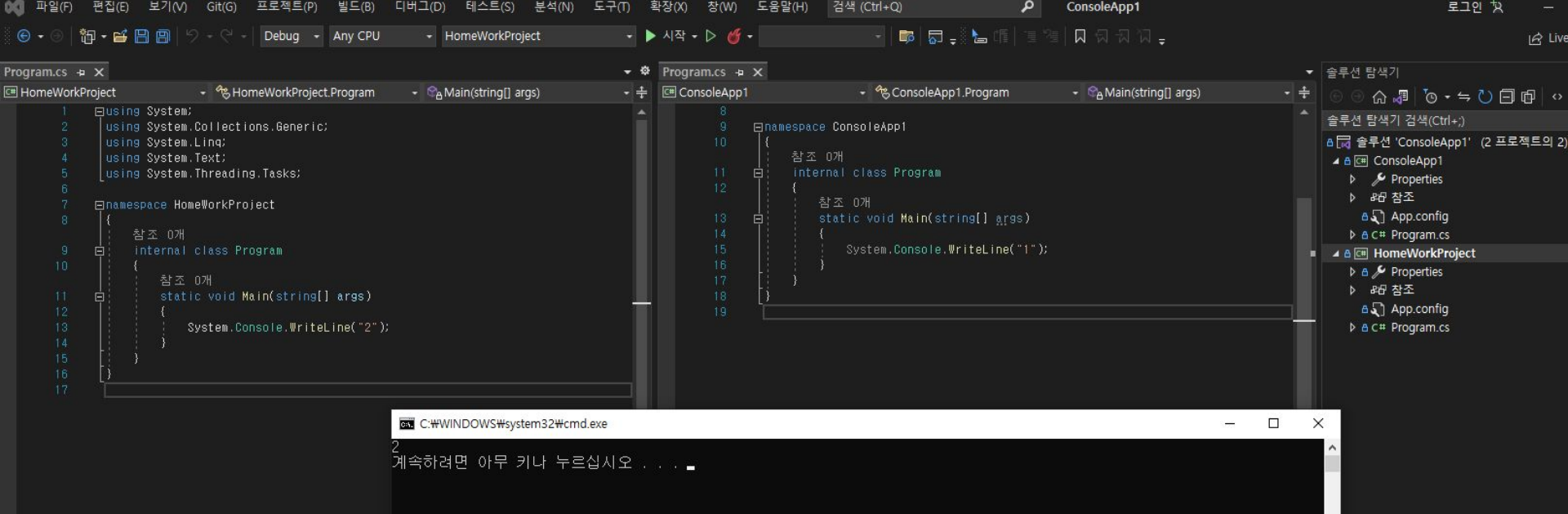




## 시작 프로젝트 지정

왼쪽은 HomeWork 프로젝트의 스크립트이고,  
오른쪽은 ConsoleApp1 프로젝트의 스크립트이다.

시작프로젝트가 ConsoleApp1이라서 실행하면 오른쪽 프로젝트가 시작된다.

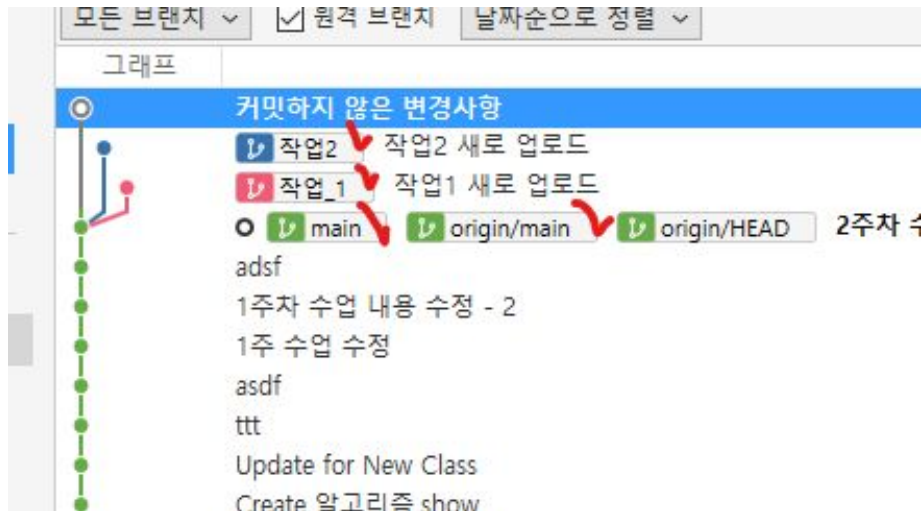


## 시작 프로젝트 변경

시작 프로젝트를 변경하고 다시 컴파일(실행)하니까 결과가 수정되었다.

# 브랜치

Github에서 사용하는 독립적으로 어떤 작업을 진행할때 분류되는 개념.



작업 1을 하고자 한다면  
거기에 맞는 브랜치로  
들어가서 작업을 하고, 같은  
방식으로 **Commit, Push** 하면  
된다.

브랜치는 세이브 파일 같은  
느낌이다.

각각의 **Commit**들은 세이브  
포인트 같은 느낌이다.

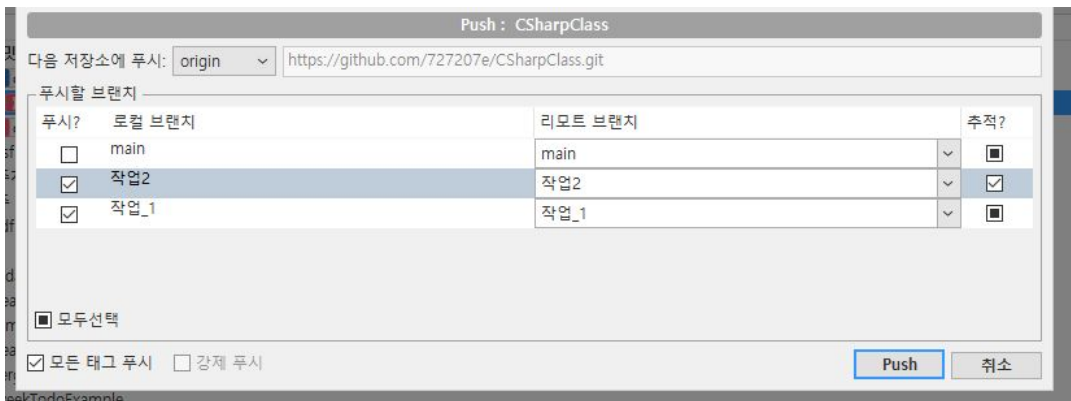
# 브랜치

로컬에서 브랜치를 만들었기 때문에 리모트(서버) 브랜치로 만들어서 연결시켜 주어야 한다.

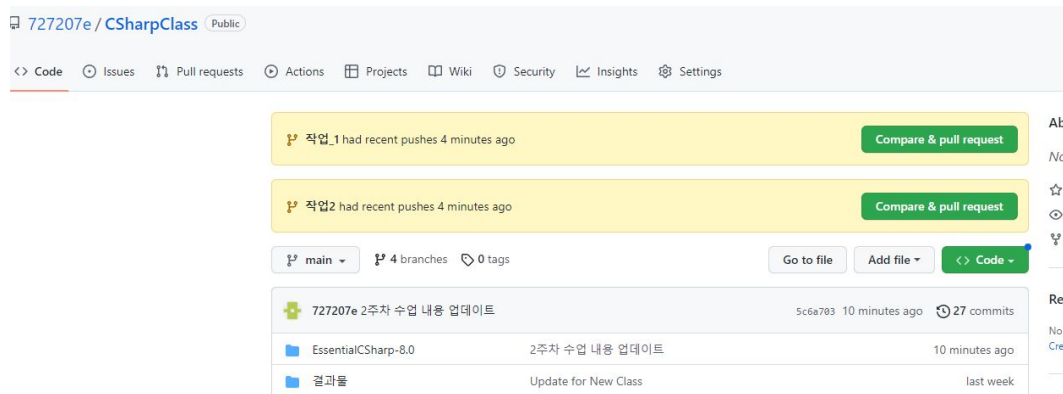
서버에도 같은 브랜치를 만든다는 느낌.

서버 브랜치는 브랜치 이름 앞에 **origin/** 이 붙는다.

이 중에서 **main**브랜치가 서버에서 가지고 있는 진짜 데이터.



# 브랜치



Github로 들어가면 노란색  
Compare & pull request 라는  
문구가 뜬다.

이건 해당 github의 주인에게  
내가 작업한 내용을  
합쳐서 서버에 저장해주세요  
라고 요청하는 내용이다.



# 브랜치

## Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

↺

base: main

←

compare: 작업2

✓ Able to merge. These branches can be automatically merged.

+

작업2 새로 업로드

Write

Preview

H B I ≡ < > 🔗 ≡ ≡ ≡ @ ↻ ↶

작업 2 진행했습니다

Attach files by dragging & dropping, selecting or pasting them.

📎

Create pull request

▼

① Remember, contributions to this repository should follow our [GitHub Community Guidelines](#).

Reviewer  
No review  
Assignee  
No one  
Labels  
None yet  
Projects  
None yet  
Milestone  
No miles  
Develop  
Use Closi  
automati

버튼 눌러서

작업한 내용 설명적이고

Create Pull Request를 누른다.

# 브랜치

727207e / CSharpClass Public

<> Code Issues **Pull requests** 2 Actions Projects Wiki Security

Filters is:pr is:open

☐ 2 Open ✓ 1 Closed



☐ 작업2 새로 업로드  
#5 opened now by 727207e

☐ 작업1 새로 업로드  
#4 opened now by 727207e



Pul Request에 새로 생긴게 보인다.

# 브랜치



## 작업1 새로 업로드 #4

 Open 727207e wants to merge 1 commit into `main` from `작업1` 



Conversation 0 Commits 1 Checks 0 Files changed 1


 727207e commented 2 minutes ago Owner 


작업1 테스트

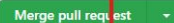

  작업1 새로 업로드 409213a

































Add more commits by pushing to the `작업1` branch on 727207e/CSharpClass.

  **Approval not required**  
This pull request may be merged without approvals. [Learn more.](#)

 **Continuous integration has not been set up**  
[GitHub Actions](#) and [several other apps](#) can be used to automatically catch bugs and enforce style.

 **This branch has no conflicts with the base branch**  
Merging can be performed automatically.

 Merge pull request  You can also open [this in GitHub Desktop](#) or view [command line instructions](#).

 Write Preview H B I                               

Leave a comment

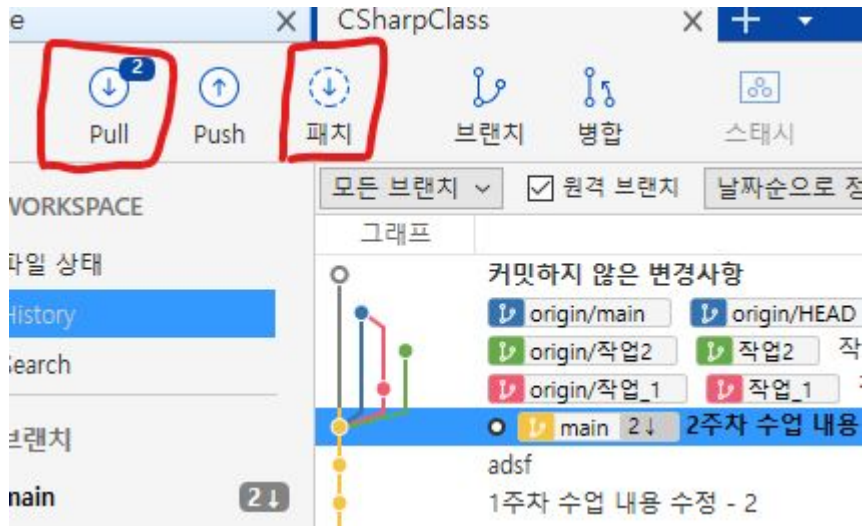
Merge 버튼으로 프로젝트를  
합친다.

# 브랜치

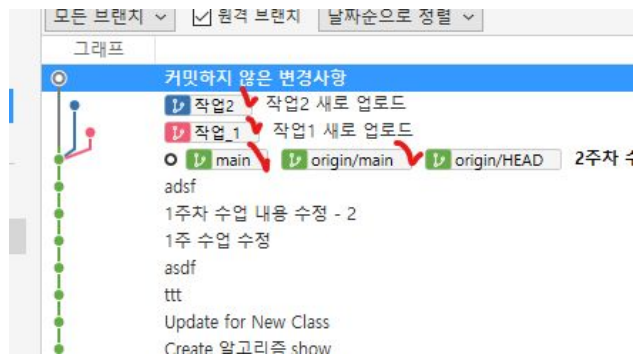
패치 버튼은  
서버에서 생긴 변경사항을  
로컬로 가져온다는 내용이다.

Merge를 하면 왼쪽 그림처럼  
브랜치가 main 그래프에 붙는  
그림이 된다.

로컬 main은 서버 main과  
다르게 데이터가 수정되기  
전임으로 pull을 통해 서버  
main에 데이터를 가져온다.



후↑      전→



# 4장

연산자.

if, for, while, switch

```

static void Main(string[] args)
{
    int number1 = 5;
    int number2 = 16;

    int plus = number1 + number2;
    System.Console.WriteLine($"P : {plus}");

    int minus;
    minus = 5 - number2;
    System.Console.WriteLine($"Min : {minus}");

    int mult = number1 * 16;
    System.Console.WriteLine($"Mul : {mult}");

    int quotient;
    quotient = 16 / 5;
    System.Console.WriteLine($"Quo : {quotient}");

    int remainder = number2 % 5;
    System.Console.WriteLine($"Rem : {remainder}");
}

```

## 산술 이항 연술자

더하기

빼기

곱하기

나누기(몫)

나누기(나머지)

```

P : 21
Min : -11
Mul : 80
Quo : 3
Rem : 1

```

```
int x = 123;
```

```
x += 2;
```

```
Console.WriteLine($"+= : {x}");
```

```
x -= 2;
```

```
Console.WriteLine($"-= : {x}");
```

```
x /= 2;
```

```
Console.WriteLine($"/= : {x}");
```

```
x *= 2;
```

```
Console.WriteLine($"*= : {x}");
```

```
x %= 2;
```

```
Console.WriteLine($"%= : {x}");
```

C:\WINDOWS\system32

```
+= : 125
```

```
-= : 123
```

```
/= : 61
```

```
*= : 122
```

```
%= : 0
```

계속하려면 아무 키를 누르세요

## 할당 연산자

x 값에 연산식으로 계산되어 다시 x에 할당된다.

더하기 +=

빼기 -=

곱하기 \*=

나누기(몫) /=

나누기(나머지) %=

```
int value = 123;
```

```
Console.WriteLine($"1 : {++value}");
```

```
Console.WriteLine($"2 : {++value}");
```

```
Console.WriteLine($"3 : {value++}");
```

```
Console.WriteLine($"4 : {value++}");
```

```
Console.WriteLine($"5 : {value++}");
```

C:\ C:\WINDOWS

1 : 124

2 : 125

3 : 125

4 : 126

5 : 127

계속하려면

## 증가 연산자

`++value` 전위 증가 연산자 -> 해당 구문이 끝나기 전에 값이 수정됨

`value++` 후위 증가 연산자 -> 해당 구문이 끝나면 값이 수정됨

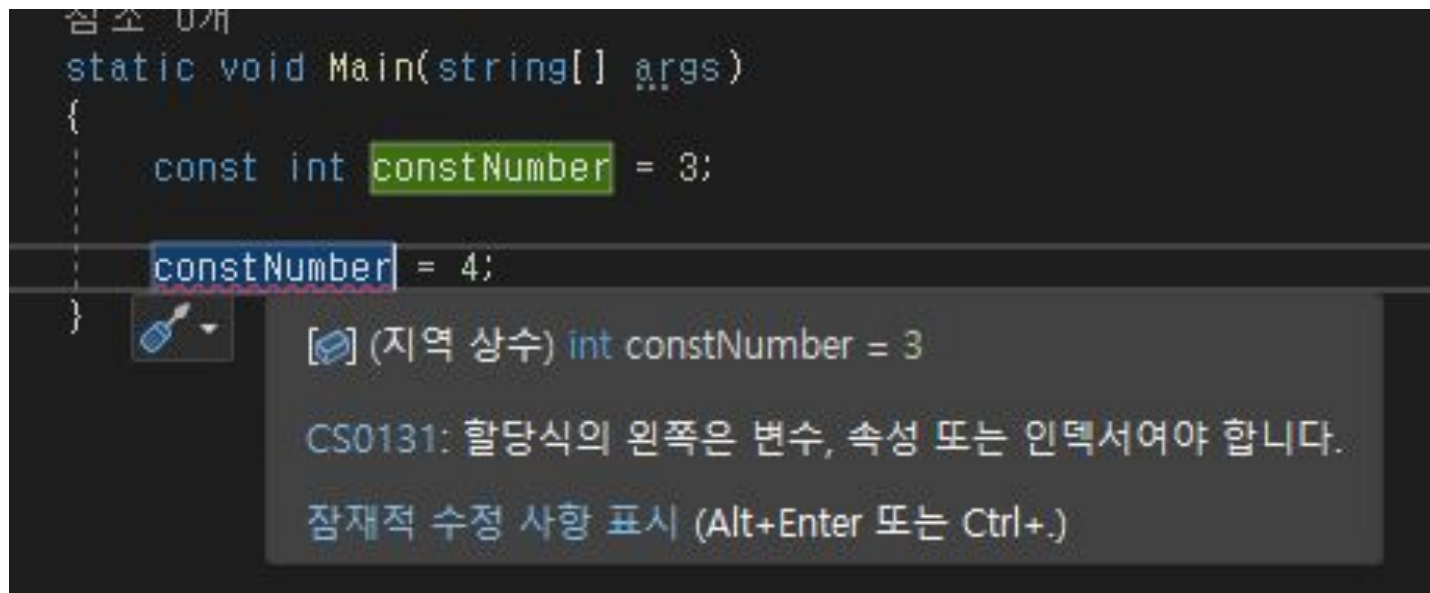


숫자 연산하기

덧셈 , 뺄셈, 나눗셈, 곱셈 **5**번씩

할당 연산자 **5**번씩

증가 연산자 **5**번씩



## 상수식 Const

const는 값을 한번 할당하면 다시 값을 수정할 수 없다.

사용 예제 : 두번다시 값을 수정하지 않을 절대값들(예, 파이값(3.141592...))

# Bool

bool 은 참 / 거짓만 담을 수 있는 변수형

```
bool isMyNameUmin = true;  
bool isThereanyQuestion = false;
```

Bool 값 앞에 ! 가 붙으면 값이 반전된다.

```
bool isTheValue = false;  
bool isTheValue2 = !isTheValue;
```

```
System.Console.WriteLine(isTheValue);  
System.Console.WriteLine(isTheValue2);
```



C:\WINDOWS

```
False  
True
```

종류	기호	문법	의미
비교 연산자	==	a==3	a와 3이 같은가?
	!=	a!=3	a와 3이 다른가?
	>, <	a>3, a<3	a가 3보다 큰(작은)가?
	>=, <=	a>=3, a<=3	a가 3보다 크거나(작거나)나 같은가?
논리 연산자	&&	a&& b	a와 b가 모두 참일 경우에만 참
		a   b	a와 b가 모두 거짓일 경우에만 거짓
	!	!a	a가 참이면 거짓, 거짓이면 참

```

bool isValueTrue = (3 == 3);
bool isValueFalse = (true && false);

System.Console.WriteLine(isValueTrue);
System.Console.WriteLine(isValueFalse);

```

C:\#WINDC

True  
False

비교 연산자로 할 수 있는 모든 종류 해보기

And or 연산 해보기

# IF 문 - 1

if(조건식)

조건식이 참인 경우(true) 안에 내용을 실행한다.

```
if(true)
{
    System.Console.WriteLine("if문 실행함");
}

if(false)
{
    System.Console.WriteLine("if문 실행안함");
}
```

C:\WINDOWS\system32\cmd.exe

if문 실행함  
계속하려면 아무 키나 누르십시오 . . .

```
if(false)
{
    System.Console.WriteLine("내용 1");
}

else if(false)
{
    System.Console.WriteLine("내용 2");
}

else if(true)
{
    System.Console.WriteLine("내용 3");
}

else
{
    System.Console.WriteLine("내용 4");
}
```

C:\WINDOWS\system

내용3  
계속하려면 아무 키

```
int mynumber = 10;
if(mynumber < 0)
{
    System.Console.WriteLine("내용 1");
}

else if(mynumber < 10)
{
    System.Console.WriteLine("내용 2");
}

else if(mynumber < 20)
{
    System.Console.WriteLine("내용 3");
}
else
{
    System.Console.WriteLine("내용 4");
}
```

C:\ C:\#WINDOW

내용3  
계속하려면 0



```
int mynumber = 10;
int yourNumber = 150;
if(mynumber < 0 && yourNumber > 100)
{
    System.Console.WriteLine("내용 1");
}

else if(mynumber < 10 && yourNumber > 100)
{
    System.Console.WriteLine("내용 2");
}

else if(mynumber < 20 && yourNumber > 100)
{
    System.Console.WriteLine("내용 3");
}
else
{
    System.Console.WriteLine("내용 4");
}
```

C:\C#\WINDOWS

내용3  
계속하려면

```
string theStringValue = null;
if (theStringValue == null)
{
    System.Console.WriteLine("Null");
}
else if (theStringValue != null)
{
    System.Console.WriteLine("not Null");
}

theStringValue = "stringValue";
if (theStringValue == null)
{
    System.Console.WriteLine("Null");
}
else if (theStringValue != null)
{
    System.Console.WriteLine("not Null");
}
```

C:\ C:\#WINDOW

Null  
not Null  
계속하려면 0

## IF 문 - 2

if(조건식1)        - 조건식 1이 True라면 내용1실행

```
{  
    내용1  
}
```

else if(조건식2) - 조건식 1이 아니고, 조건식2가 True라면 내용 2 실행

```
{  
    내용2  
}
```

else if(조건식3) - 조건식 1이 아니고, 조건식 2가 아니고, 조건식3가 True라면 내용 3 실행

```
{  
    내용3  
}
```

else

```
{  
  
}
```

1. 숫자 하나를 입력받아서  
짝수, 홀수, 0 중 하나를 가려내기

2. 문자 하나를 입력받아서  
문자의 길이가 1~3, 4~6, 7~10, 10이상 가려내기

3. 숫자와 문자 입력해서 최종 결과 띄우기  
예) 짝수 이면서 1~3 입니다.

# While 문 - 1

while(조건식)

조건식이 해당된다면 안에 내용을 반복한다.

```
조 0개
atic void Main(string[] args)
{
    int numberValue = 0;

    System.Console.WriteLine("시작");

    while (numberValue < 10)
    {
        System.Console.WriteLine(numberValue);
        numberValue += 1;
    }

    System.Console.WriteLine("끝");
}
```

C:\C#\C#WW  
시작  
0  
1  
2  
3  
4  
5  
6  
7  
8  
9  
끝  
계속하러

numerValue	조건	내용
0	0 < 10 (참)	0 출력
1	1 < 10 (참)	1 출력
2	2 < 10 (참)	2 출력
3	3 < 10 (참)	3 출력
4	4 < 10 (참)	4 출력
5	5 < 10 (참)	5 출력
6	6 < 10 (참)	6 출력
7	7 < 10 (참)	7 출력
8	8 < 10 (참)	8 출력
9	9 < 10 (참)	9 출력
10	10 < 10 (거짓)	

## While 문 - 2

do ~ while(조건식)

while과 같지만 반드시 한번은 실행된다.

```
참조 0개
static void Main(string[] args)
{
    int numberValue = 0;

    System.Console.WriteLine("시작");

    do
    {
        System.Console.WriteLine(numberValue);

        numberValue += 1;
    } while(numberValue > 5);

    System.Console.WriteLine("끝");
}
```

C:\C:\WWI  
시작  
0  
끝  
계속하려

numervalue	내용	조건
0	0 출력	1 > 5 (거짓)

1. 숫자하나를 입력받는다.

문장 하나를 입력받는다.

입력 받은 숫자만큼 문장을 띄운다.

# For 문 - 1

for (초기식 ; 조건식 ; 증감식)

초기값이 조건식에 맞으면 내용 실행. 내용 끝나면  
증감식 적용

```
System.Console.WriteLine("시작");  
  
for(int num = 0; num < 5; num++)  
{  
    System.Console.WriteLine(num);  
}
```

```
System.Console.WriteLine("끝");
```

C:\C:\WW

시작  
0  
1  
2  
3  
4  
끝  
계속하러

num	조건	내용	증감
0	$0 < 5$ (참)	0 출력	num은 1
1	$1 < 5$ (참)	1 출력	num은 2
2	$2 < 5$ (참)	2 출력	num은 3
3	$3 < 5$ (참)	3 출력	num은 4
4	$4 < 5$ (참)	4 출력	num은 5
5	$5 < 5$ (거짓)		



## For 문 - 2

foreach ((변수형) 변수 in (변수형배열))

배열이 가지고 있는 각 원소들마다 반복함

참조 0개

```
static void Main(string[] args)
{
    System.Console.WriteLine("시작");

    string[] myFoods = new string[4] { "떡", "밥", "국", "반찬" };
    foreach(string Food in myFoods)
    {
        System.Console.WriteLine(Food);
    }
    System.Console.WriteLine("끝");
}
```

C# C:##W

시작

떡







밥

국

반찬

계속하러

# 1. 별그리기

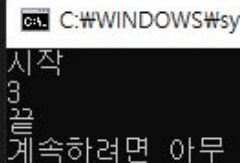
①	②	③	④	⑤	⑥
					

# switch 문

if문이 많아지면 생겼던 불편함을 해소하고자 나옴.  
각 조건에 맞으면 내용을 실행한다.

```
System.Console.WriteLine("시작");

int myNumber = 3;
switch(myNumber)
{
    case 0:
        System.Console.WriteLine("0");
        break;
    case 1:
        System.Console.WriteLine("1");
        break;
    case 2:
        System.Console.WriteLine("2");
        break;
    case 3:
        System.Console.WriteLine("3");
        break;
    case 4:
        System.Console.WriteLine("4");
        break;
    default:
        System.Console.WriteLine("default");
        break;
}
```



switch(비교하고자 하는 변수)

```
{
    case (비교할 대상값) :
        break;

    case (비교할 대상 값) :
        break;

    ...

    default :
        break;
}
```

default는 위가 모두 아닌 경우 실행됨.

# break 와 continue

if문이나 for, while문에서

break는 루프를 깨고 나온다.

continue는 다음 루프를 진행한다.

<오른쪽 그림 설명>

1. i가 0으로 시작된다.
2. i가 0이면 "0"출력, continue로 끝.
3. i가 1이면 "1"출력, "11"출력, continue로 끝.  
( "111"은 출력안됨 )
4. i가 2이면 "2"출력, break로 for문 끝

```
System.Console.WriteLine("시작");
```

```
for(int i = 0; i < 5; i++)
```

```
{
```

```
    if(i == 0)
```

```
    {
```

```
        Console.WriteLine("0");
```

```
        continue;
```

```
    }
```

```
    else if(i == 1)
```

```
    {
```

```
        Console.WriteLine("1");
```

```
        Console.WriteLine("11");
```

```
        continue;
```

```
        Console.WriteLine("111");
```

```
    }
```

```
    else if(i == 2)
```

```
    {
```

```
        Console.WriteLine("2");
```

```
        break;
```

```
    }
```

```
    else if(i == 3)
```

```
    {
```

```
        Console.WriteLine("3");
```

```
    }
```

```
}
```

```
System.Console.WriteLine("끝");
```

C:\WINDOWS\system32

시작

0

1

11

2

끝

계속하려면 아무 키나

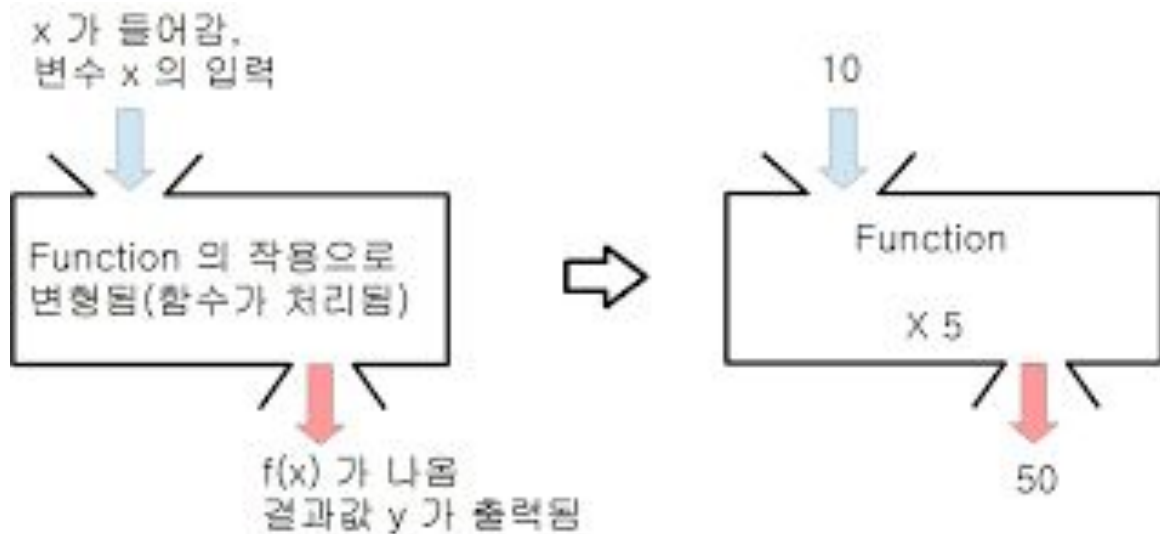
5장

메소드

# 메소드란

값을 넣어주면

값이 가공되어서 돌려주는 형태.



참조 0개

```
static void Main(string[] args)
{
    System.Console.WriteLine("시작");

    ShowUpNewSentence( );

    System.Console.WriteLine("끝");
}
```

참조 1개

```
private static void ShowUpNewSentence( )
{
    System.Console.WriteLine("새로운 문장");
}
```

ShowUpNewSentence()  
가 메소드

C:\WINDOWS#sy  
시작  
새로운 문장  
끝  
계속하려면 아무

```
// 메소드 선언
반환형식 메소드 정의(매개 변수)
{
    ....
    if(조건) return 값;
    ...
    return 값;
}
```

```
...
메소드 호출(매개변수);
...
```

참조 0개  
static void Main(string[] args)

```
{
    System.Console.WriteLine("시작");

    int myNumber = ShowUpNewSentence();

    System.Console.WriteLine(myNumber);

    System.Console.WriteLine("끝");
}
```

참조 1개

```
private static int ShowUpNewSentence()
{
    return 3;
}
```

C:\#WIN

시작  
3  
끝  
계속하려면

ShowUpNewSentence()에서

반환형식이 int형식.

return으로 값을 돌려준다.



참조 0개

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

```
{  
    System.Console.WriteLine("시작");
```

```
    string myName;
```

```
    myName = WhatIsYourNumber(1);
```

```
    System.Console.WriteLine(myName);
```

```
    System.Console.WriteLine("끝");  
}
```

참조 1개

```
private static string WhatIsYourNumber(int theNumber)
```

```
{  
    if(theNumber == 1)
```

```
    {  
        return "Umin";  
    }
```

```
    else if(theNumber == 2)
```

```
    {  
        return "SBS";  
    }
```

```
    else
```

```
    {  
        return "No";  
    }  
}
```

C:\C:\WINDOWS#sy

시작

Umin

끝

계속하려면 아무

WhatIsYourNumber 메소드에서

매개변수 **theNumber**는 메소드에서  
앞으로 사용될 값이다.

WhatIsYourNumber는  
**theNumber**라는 int값이 필요하다  
라고 미리 선언한 것이다.

위에서 WhatIsYourNumber를  
사용할땐 인수라는 값으로  
넣어주어야 한다.

참조 0개

```
static void Main(string[] args)
{
    System.Console.WriteLine("시작");

    int addRes;
    addRes = Add(5, 10);

    System.Console.WriteLine(addRes);

    System.Console.WriteLine("끝");
}
```

참조 1개

```
private static int Add(int firstNumber, int secondNumber)
{
    int resultNumber;
    resultNumber = firstNumber + secondNumber;

    return resultNumber;
}
```

C# C:\WINDOWS  
시작  
15  
끝  
계속하려면

매개변수의 갯수는 여러개 넣을 수 있다.

왼쪽 그림과 같이

2개의 숫자를 받아  
더한 결과를 돌려줄 수 있다.

1. 메소드 이용해서 숫자 2개를 입력받아  
덧셈, 뺄셈, 곱셈, 나눗셈 한번씩 띄우기

2. 메소드 이용해서 숫자, 글자 입력받아  
1~3, 4~6 조건 위에서 했던거 그대로 진행

3. 메소드 이용해서 숫자, 글자 입력받아  
숫자 만큼 문장 호출하기

4. 별짓기