



# 처음 만드는 C# 프로그램

이것이 C#이다



# Contents

---

- ❖ Hello, World!
- ❖ 첫 번째 프로그램 뜯어보기
- ❖ CLR에 대하여



## 2.1 Hello, World!

- ❖ “Hello, World!” 출력 콘솔 애플리케이션 제작
- ❖ Step1: Visual Studio 2017 실행
- ❖ Step2: [새 프로젝트] 대화 상자 실행
- ❖ Step3: HelloWorld 프로젝트 생성
- ❖ Step4: 코드 편집기와 솔루션 탐색기
- ❖ Step5: HelloWorld.cs 이름 변경
- ❖ Step6: 소스 코드 작성
- ❖ Step7: 컴파일
- ❖ Step8: 컴파일 결과 확인
- ❖ Step9: 명령창에서 실행 파일 위치로 이동
- ❖ Step10: HelloWorld 애플리케이션 실행



## HelloWorld 애플리케이션 만들기

```
01  using System;
02  using static System.Console;
03
04  namespace BrainCSharp
05  {
06      class HelloWorld
07      {
08          // 프로그램 실행이 시작되는 곳
09          static void Main(string[] args)
10          {
11              if (args.Length == 0)
12              {
13                  Console.WriteLine("사용법 : HelloWorld.exe <이름>");
14                  return;
15              }
16
17              WriteLine("Hello, {0}!", args[0]);
18          }
19      }
20  }
```



## 2.2 첫 번째 프로그램 뜯어보기

- ❖ `using System;`
- ❖ `using static System.Console;`
- ❖ `namespace BrainCSharp`
- ❖ `class HelloWorld{ }`
- ❖ `//프로그램 실행이 시작되는 곳`
- ❖ `static void Main(string[] args){ }`
- ❖ `if(args.Length==0){ }`



## 2.2.1 using System;

### ❖ using

- C# 키워드

### ❖ System

- 기본 클래스의 네임스페이스

### ❖ 세미콜론(;)

- 문장의 끝

```
Text = "나는 자랑스러운 태극기 앞에 "  
+ "자유롭고 정의로운 대한민국의 영광을 위하여 "  
+ "몸과 마음을 바쳐 충성을 다할 것을 "  
+ "굳게 다짐합니다." ;
```

```
alpha = 1.2; beta = 4.0; gamma = 7.22;
```

- Visual Basic - 줄 바꿈

```
Text = "나는 자랑스러운 태극기 앞에 " _  
+ "자유롭고 정의로운 대한민국의 무궁한 영광을 위하여 " _  
+ "몸과 마음을 바쳐 충성을 다할 것을 " _  
+ "굳게 다짐합니다."
```



## 2.2.2 using static System.Console;

### ❖ using static

- 데이터 형식의 정적 멤버 사용

### ❖ Console 클래스의 대표적인 정적 멤버

- Write(),WriteLine(),Read(),ReadLine()



## 2.2.3 namespace BrainCSharp{ }

### ❖ 네임스페이스

- 성격, 하는 일이 비슷한 형식을 하나의 이름으로 그룹화
- .NET 프레임워크 라이브러리
  - System.IO
  - System.Printing

### ❖ 네임스페이스 만들기

- Namespace 키워드
- 형식

```
namespace 네임스페이스_이름
{
    // 클래스
    // 구조체
    // 인터페이스 등...
}
```





## 2.3.4 class HelloWorld{ }

### ❖ 클래스

- C# 프로그램을 구성하는 기본 단위
- 데이터와 데이터 처리 기능(메소드)
- `class class_name { }`

### ❖ C#의 ‘{ ‘와 ‘}’

- 코드 블록



## 2.3.5 // 프로그램 실행이 시작되는 곳

### ❖ C#의 주석

- 소스 코드 안에 기록하는 메모
- 컴파일러는 주석을 처리하지 않음
- // - 한 줄 주석
- /\* \*/ - 여러 줄 주석

```
// 프로그램  
// 실행이  
// 시작되는 곳  
static void Main(string[] args)  
{  
  
}
```

```
/* 프로그램  
   실행이  
   시작되는 곳 */  
static void Main(string[] args)  
{  
  
}
```



## 2.2.6 static void Main(string[] args){ }

### ❖ 메소드

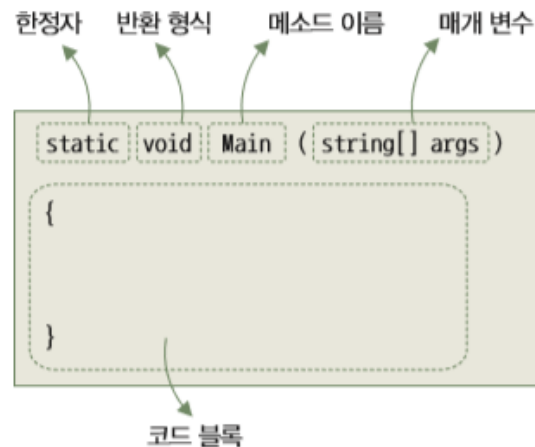
- C 프로그래밍 언어 → 함수
- 객체 지향 프로그래밍 → 메소드
- 입력(객체) → 계산 → 출력 (객체)

### ❖ 진입점(Entry Point)

- 특별한 메소드, Main
- 프로그램 시작 시 실행
- Main 메소드 종료 시 프로그램 종료
- 프로그램 실행 시 매개 변수 입력

>HelloWorld.exe C#

```
09      static void Main(string[] args)
10      {
11          if (args.Length == 0)
12          {
13              Console.WriteLine("사용법 : He
14              return;
15          }
16
17          WriteLine("Hello, {0}!", args[0]);
18      }
```



## 2.3.7 if(args.Length==0){ }

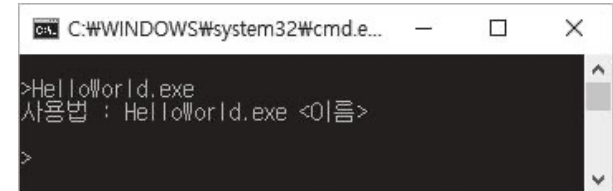
### ❖ 매개변수 입력이 필요한 프로그램

### ❖ if 문

- 조건을 평가해 프로그램의 흐름 제어
- 목록의 길이 검사 (args.Length==0)

### ❖ Return

- 호출자에게 메서드 실행 결과 반환
- Main 메서드 종료



```
C:\WINDOWS\system32\cmd.e...
>HelloWorld.exe
사용법 : HelloWorld.exe <이름>
>
```



## 2.3 CLR에 대하여

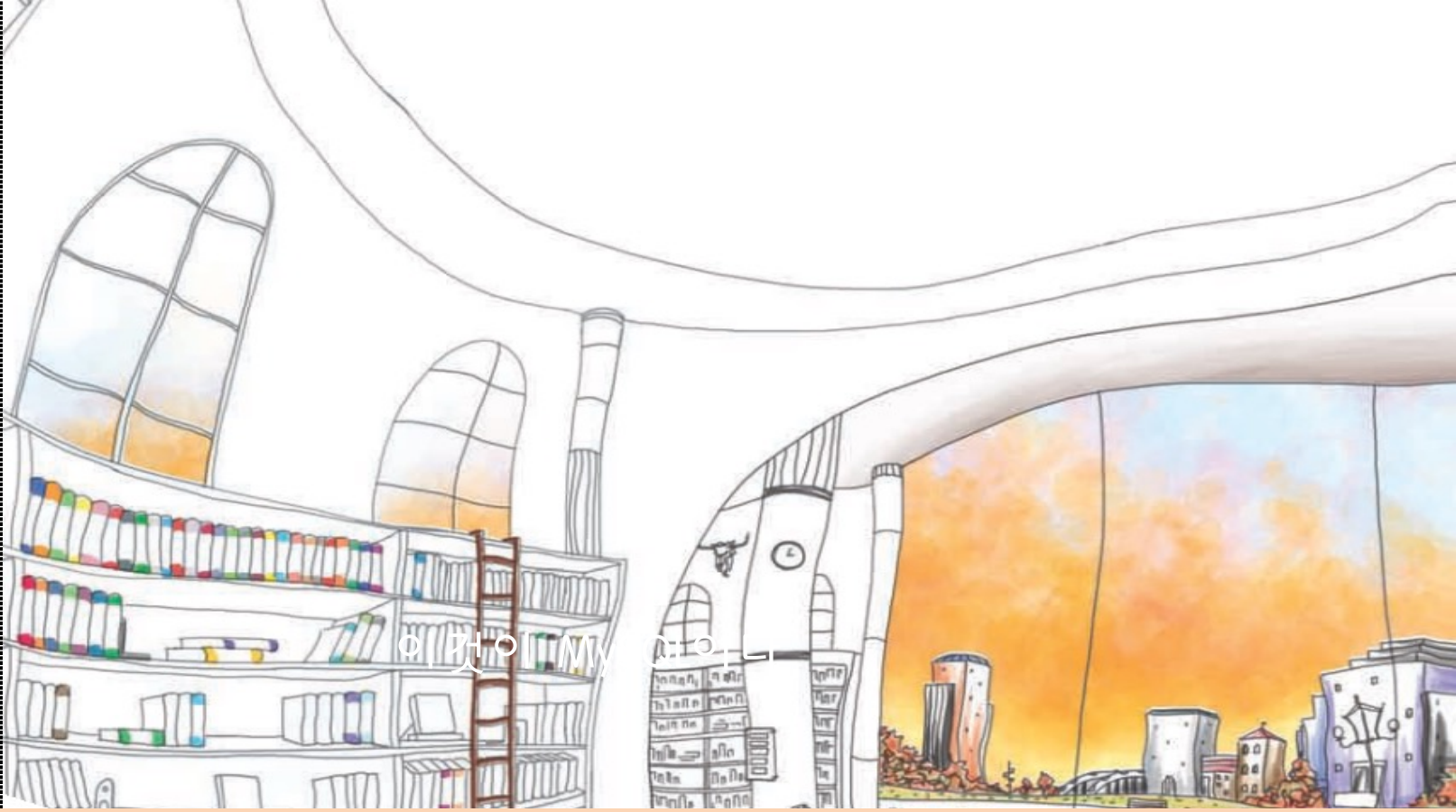
### ❖ Common Language Runtime

- C#으로 만든 프로그램의 실행 환경
- 중간 언어를 통한 다중 언어 지원
- 플랫폼 최적화된 코드 생성

### ❖ C# 코드의 여정

- →C# 컴파일러
- →IL(Intermediate Language) 파일
- →JIT(Just In Time) 컴파일
  - CLR의 IL 코드→네이티브 코드 컴파일→실행





# Thank You !

이것이 C#이다

