

C# 교실

CLASS 1

- Introduction to C#
- C# Basic (Part 1)

고주형

2019/4/12

Introduction to C#

간단한 프로그램을 만들어보고 문자열에 대해서
알아보며 프로그래밍에 대한 감을 얻어봅시다!

특히, 간단한 프로그램에 대해 무엇을 할지 생각해봅시다!

Hello World

```
Console.WriteLine("Hello World!");
```

- `string`인 “Hello World!” 메시지를 출력하는 간단한 프로그램입니다!
- `Console`은 콘솔 창을 나타내는 형식!
- `WriteLine`은 `text`를 해당 콘솔에 출력하는 `Console`형식의 메서드!
- `;`은 문장의 끝을 알려준다! 세미콜론 없이는 못 살아~

String ?

“Hello World!” == string

- “Hello World!” – 이런 것이 string!! 문자열!!
- 콘솔형식과 마찬가지로 string형식에도 메서드가 있다!
- string 메서드? Text와 관련된 작업을 수행한다!

String ?

```
string CIEN_Programmer = "Hello World!"
```

- 메시지를 더 많이 출력하고 싶으면?
- `string CIEN_Programmer = "고주형";`
- `Console.WriteLine(CIEN_Programmer);`
- `CIEN_Programmer = "김수환";`
- `Console.WriteLine(CIEN_Programmer);`

String ?

```
string CIEN_Programmer = "Hello " + "World";
```

- Text앞에 “안녕”을 붙여주고 싶다.
- `string CIEN_Programmer = “고주형”;`
- `Console.WriteLine(“안녕, ” + CIEN_Programmer);`
- `+`를 사용해서 변수 및 상수 문자열에서 문자열을 빌드했다!

String ? 또 다른 방법이 있다!

```
Console.WriteLine($"안녕, {Programmer}");
```

- Text앞에 “안녕”을 붙여주고 싶다.
- `string CIEN_Programmer = “고주형”;`
- `Console.WriteLine($"안녕, {CIEN_Programmer}");`
- 문자열을 여는 따옴표 앞에 \$를 사용하면 중괄호 안에 변수를 넣을 수 있다!

String ?

```
Console.WriteLine($"안녕, {CIEN_Programmer} 답변: {groot}");
```

- 변수 여러개도 가능!!
- `string CIEN_Programmer = "고주형";`
- `string groot = "I am groot";`
- `Console.WriteLine($"안녕, {CIEN_Programmer} 답변: {groot}");`

String의 Property

문자열의 속성 **Length** – 문자열의 문자 수 반환

- 문자열의 속성인 Length를 사용하면 문자열의 길이를 알 수 있다.
- `string korean = “가나다라마”;`
- `Console.WriteLine(“문자열 {korean}는 {korean.Length}자이다”);`
- 외울게 너무 많나여?? 걱정 L L
- IntelliSense기능이 있습니다! – 수행할 수 있는 작업에 대한 제안을 제공해 줘!
.을 눌러서 뒤에 사용할 수 있는 속성 및 메서드에 대한 제안 목록을 확인해 봅시다!

String TMI

.ToUpper() .ToLower

.To까지 입력하면 힌트를 제공하는 것-intelliSense확인하기

- 아 소문자가 싫다! ToUpper 메서드!
- `string groot = "I am Groot";`
- `string upperGroot = groot.ToUpper();`
- 갑자기 대문자가 싫다! ToLower 메서드!
- `string lowerGroot = groot.ToLower();`

String TMI

.Trim() .TrimStart() .TrimEnd()
.Replace()

- 아 문자열의 양 옆의 공백이 싫다!
- `string groot` = “ I am Groot ”;
- `string trimmedGroot` = `groot.Trim()`;
- 아 그루트가 싫다!
- `string juhyeong` = `groot.Replace(“Groot”, “juhyeong”)`;

Search String

.Contains(“이거 있?”)

- 문자열에 하위 문자열이 있는지 검색해보기! Contains 메서드!
- `string groot = “ I am Groot ”;`
- `Console.WriteLine(Groot.Contains(“Groot”));`
- 예상 되는 답=> 있다(`true`) | 없다(`false`)
- Contains 메서드는 부울 값을 반환한다!
- 부울? `bool`은 `true` 또는 `false` 값을 저장합니다!

Search에 대해서 더 궁금하다 ...

- 문자열의 시작하는 부분을 체크하는 메서드는 있을까?
- `StartsWith("이걸로 시작?")`
- 그럼 끝은?
- `EndsWith("이걸로 끝남?")`
- 반환값은 뭡까요??

C# Programming

이제 본격적으로 C#에 대해서 꼼꼼히
(매우 빠른 속도로) 알아보시다

(매우 빠른 속도로) 알아보시다

먼저 우리 프로그램을 다시 한번 살펴보자

using System; <- 시스템에 관한 것 쓸래!!

참조 0개

```
class Program <- 프로그램이라는 클래스!  
{  
    일단 외웁시다 ππ
```

참조 0개

```
static void Main() <- Main함수!  
{  
    여기가 바로 프로그램의 시작점(Entry Point)!  
    Console.WriteLine("Hello World!");  
}  
}
```

오늘 할 것

- 자료형(int, float, string, bool...)
- float의 정확성 (부동소수점)
- 형 변환(암시적 vs 명시적)
- 키워드(예약어), 리터럴
- 변수(값형만)
- 상수(Const)
- 산술 연산자(+ - * / %)
- 문장 부호들(;, {}, (), ...)

자료형

요리 재료를 각각 담아두면 정리도 되고 필요할 때 편하게 써먹을 수 있겠죠? ^^
이것이 **자료형**입니다.

bool
byte
sbyte
int
float
double
char
string
object

참, 거짓으로 나뉘는 그릇. true, false
1바이트 만을 담는 그릇. 양수만 가능 0~255
1바이트 만을 담는 그릇. 음수 양수 가능 -127~128
정수형(숫자)를 담는 그릇. 음수 양수 가능
소수점 7자리까지 숫자를 담는 그릇
소수점 16자리까지 숫자를 담는 그릇
문자 하나를 담는 그릇 'A'
문자열을 담는 그릇 "Hello" (문자 여러개)
object형 을 담는 그릇. 모든 그릇이 올 수 있다.

자료형 - Object?

object	object형 을 담는 그릇. 모든 그릇이 올 수 있다.
--------	---------------------------------

다른 건 다 이해가 되는데 오브젝트형은 뭐종??

단순하게 이야기 하자면 Object는 모든 자료형이 담길 수 있는 그릇입니다.

```
1 int val = 10;
```

```
2 Object obj = (Object)val;
```

이것을 박싱(Boxing)이라고 합니다. 박스에 담는다 처럼요.

(반대는 언박싱 UnBoxing 이겠죠?)

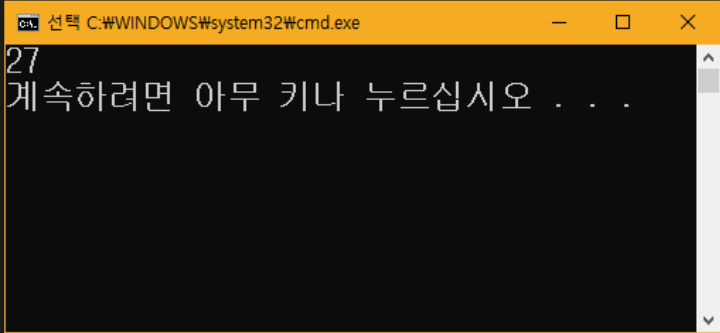
이것에 대해서는 나중에(Class에 대해서 배울 때) 더 이야기 해볼게요.

float의 정확성 (부동소수점)

- float는 정확하지 않아요!! $\pi\pi$



```
using System;

참조 0개
class Program
{
    참조 0개
    static void Main()
    {
        float f1 = 2.8f;
        float f2 = 10f;
        int i = (int)(f1*f2);
        Console.WriteLine(i);
    }
}
```



float의 정확성 (부동소수점)

부동 소수점 형식 표(C# 참조)

2018. 08. 20. • 읽는 데 2분 • 참가자    

다음 표는 부동 소수점 형식의 자릿수와 근사 범위를 보여 줍니다.

형식	근사 범위	전체 자릿수
<code>float</code>	$\pm 1.5 \times 10^{-45} \sim \pm 3.4 \times 10^{38}$	~6-9개 자릿수
<code>double</code>	$\pm 5.0 \times 10^{-324} \sim \pm 1.7 \times 10^{308}$	~15-17개 자릿수
<code>decimal</code>	$\pm 1.0 \times 10^{-28} \sim \pm 7.9228 \times 10^{28}$	28-29개의 자릿수

<https://docs.microsoft.com/ko-kr/dotnet/csharp/language-reference/keywords/floating-point-types-table>

형 변환(암시적)

- 암시적으로 이루어지는 형 변환
- 더 큰 값으로 이동할 때는 알아서 해줌!

```
int num = 2147483647;  
long bigNum = num;
```

형 변환(명시적)

- 명시적으로 이루어지는 형 변환
- 정보 손실의 위험이 있는 경우에는 컴파일러에서 캐스트라는 명시적 변환을 수행!

```
double x = 1234.7;  
int a = (int)x;
```

키워드(예약어)

- 키워드는 컴파일러에 대해 특별한 의미를 갖는, 미리 정의되어 있는 예약된 식별자입니다.
- 키워드는 프로그램에서 식별자로 사용되려면 접두어로 @을 포함해야 합니다.
 - @int (가능)
 - int (불가능)

키워드(예약어)

- 키워드는 컴파일러에 대해 특별한 의미를 갖는, 미리 정의되어 있는 예약된 식별자입니다.
- 키워드는 프로그램에서 식별자로 사용되려면 접두어로 @을 포함해야 합니다.
 - @int (가능)
 - int (불가능)

키워드들 구경

abstract	as	base	bool
break	byte	case	catch
char	checked	class	const
continue	decimal	default	delegate
do	double	else	enum
event	explicit	extern	false
finally	fixed	float	for
foreach	goto	if	implicit
in	int	interface	internal
is	lock	long	namespace
new	null	object	operator
out	override	params	private
protected	public	readonly	ref
return	sbyte	sealed	short
sizeof	stackalloc	static	string
struct	switch	this	throw
true	try	typeof	uint
ulong	unchecked	unsafe	ushort
using	Static 사용	virtual	void
volatile	while		

리터럴

리터럴(literal)? "문자 그대로의"

- 단어의 의미처럼 값 그대로라는 뜻.
- 소스 코드의 대입하는 문자나 값들!!
- 변하지 않는 값이므로 상수라고도 말한다.
- 일단 정수형과 실수형만 살펴봅시다

리터럴(정수형)

16진수: 0x를 앞에!

숫자만 적으면 값 크기+부호에 따라 결정 됨!

2,147,483,647 이하는 int

2,147,483,648 이상은 uint

9223372036854775807 이하는 long

9223372036854775808 이상은 ulong

숫자에 접미사를 붙여서 우리가 지정할 수도 있다

예를 들어서 10 뒤에 L을 붙여서

L은 Long형으로 취급 가능!

리터럴(실수형) 중요!!

C#은 소수점을 포함하는 숫자(리터럴)을 기본적으로는 **Double**형으로 인식한다!

우리가 원하는게 **double**이 아닌 경우 **f(float)**나 **m(decimal)** 등의 접미사를 붙여서 다른 타입의 리터럴로 지정해줘야 된다!

소수점이 없어도 접미사로 다른 타입의 리터럴로 인식하게 할 수도 있다.

1은 **int**형 상수

1f는 1.0의 **float**형 상수

변수 VS 상수

변수? 변하는 수! VS 상수? 일정한 수!

변수 1

변수? 재료를 담는 그릇이다.
변수를 만들어 보자!

```
1 int          // (나는 숫자를 담고 싶어.)
2 int val      // (나는 숫자를 val 그릇에 담고 싶어.)
3 int val = 10 // (나는 숫자를 val 그릇에 10만큼 담고 싶어.)
4 int val = 10; // (나는 숫자를 val 그릇에 10만큼 담고 싶어. 담아줘!)
```

변수 2

변수? 변하는 수.
변수에 다른 값을 넣어보자!

```
1 int val = 10; // 새 그릇에 숫자 10을 담는다  
2 val = 20;    // 내가 쓰던 val 그릇에 숫자 20을 다시 담는다.
```

상수 1

상수? 변하지 않는 수.
상수를 만들어 보자!

```
1 const int VAL = 20; // 숫자 20이 들어 있는 상수 VAL 입니다.
```

실수로 값을 바꾸는 것을 예방한다! (주로 문자열에서 많이 씀!)

```
1 const string SEOUL = "서울"; // SEOUL을 상수로 선언!  
2 SEOUL = "부산"; // 못 바꾸는 값인데? ㄱ 에러 !!!
```


산술연산자

+, -, *, / : 다 우리가 아는 사칙 연산자들!!

% (모듈리 연산자) : 새로운 연산자!!

Quiz)

```
int a = 3; int b = 4; int c = a/b;
```

```
Console.WriteLine(b);
```

다음의 출력 결과는 ?

산술연산자 - %

A % B (모듈러 연산자) : A/B의 나머지를 반환해 준다!
다른 말로 나머지 연산자!

예를 들어서

$$3 \% 2 \Rightarrow 1$$

$$4 \% 2 \Rightarrow 0$$

$$5 \% 10 \Rightarrow 5$$

$$9 \% 4 \Rightarrow 1$$

산술연산자 - % 활용하기

- 모듈러 연산
number가 x 의 배수? $== \text{number} \% x == 0$
- 짝수? $x \% 2 == 0$

문장 부호들 - ;, {}

;- 한 문장(명령)의 끝을 알려줌.

{ } - 명령어들의 묶음!

() - 메소드에서 사용, 인자를 여기에.

Quiz

프로그램이 실행되고 가장 처음 실행되는 함수는?



•출처

- 여기에 있는 내용들은 마이크로소프트 C# 튜토리얼 문서와 이고잉님의 오픈튜토리얼을 참고하여 만들었음을 알립니다!!!
- <https://docs.microsoft.com/ko-kr/dotnet/csharp/tutorials/>
- <https://opentutorials.org/module/3366/19780>