

http://bit.ly/미림2019_cs02

Lecture #02

Newmedia Contents Programming Programming Basic

Teacher: Gihun Ham

평가 기준

- Git 개념 및 사용법을 이해하고 사용할 수 있다.
- Github와 Git을 연동하여 프로젝트를 관리할 수 있다.
- VisualStudio 2017에서 Git과 Github를 연동할 수 있다.
- C#에서 사용하는 기본 용어를 이해한다.
- 기본 출력 방법을 익힌다.
- 기본 자료형과 변수, 이와 관련된 연산자의 사용법을 익힌다.
- 자료형 변환 방법을 익힌다.

Document

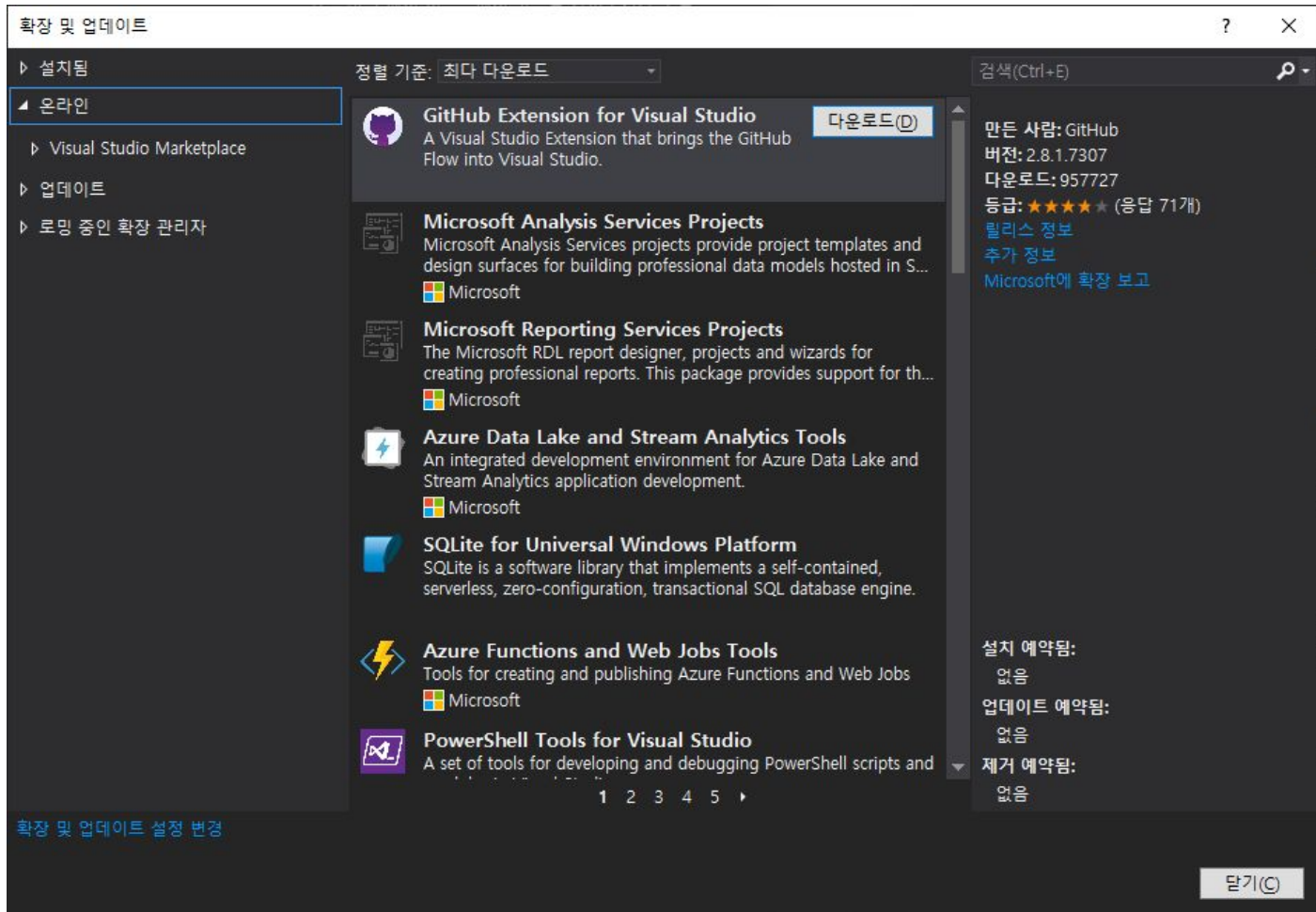
- [git - 간편 안내서](#)
- [완전 초보를 위한 깃허브](#)
- [누구나 쉽게 이용할 수 있는 Git 입문](#)
- [Progit\(무료 서적\)](#)

Lecture

- ▣ [생활코딩\(지옥에서 온 git/이고잉님\)](#)
- ▣ [생활코딩\(Githib/이고잉님\)](#)

VisualStudio2017 Github Plugin(1)

도구 -> 확장 및 업데이트 -> 온라인 -> Github Extension for visual Studio “다운로드” -> VS2017 종료(해야 설치됨)



VS2017에서 커밋, 푸시 사용하기

<https://blogs.msdn.microsoft.com/benjaminperkins/2017/04/04/setting-up-and-using-github-in-visual-studio-2017/>

Section 01 기본 용어(1)

■ 표현식(Expression)

- 값을 만들어 내는 간단한 코드
- 예

```
• 273  
• 10 + 20 + 30 * 2  
• "C# Programming"
```

■ 문장(Statement)

- 표현식의 모임, 마지막에는 종결의 의미로 세미콜론(;) 추가
- 예

```
• 273;  
• 10 + 20 + 30 + 2;  
• var name = "미" + "림" + "성";  
• Console.WriteLine("Hello C# Programming");
```

Section 01 기본 용어(2)

■ 키워드(1) - keyword

- 특별한 의미가 부여된 단어, C# 처음 만들어질때 정해짐
- 일반 키워드(표 2-1)와 컨텍스트(문맥) 키워드(표 2-2)가 있음

표 2-1 일반 키워드

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	virtual	void	volatile	while	

Section 01 기본 용어(3)

■ 키워드(2) - 특정 위치에서만 키워드로 동작

표 2-2 컨텍스트 키워드(또는 문맥 키워드)

add	alias	ascending	async	await	descending
dynamic	from	get	global	group	into
join	let	orderby	partial	remove	select
set	value	var	where	yield	

Section 01 기본 용어(4)

■ 식별자(1) - **identifier**

■ C#에서 변수와 메서드 이름 식별자 규칙

- 키워드를 사용하면 안 됨
- 특수 문자는 _만 허용
- 숫자로 시작하면 안 됨
- 공백은 입력하면 안 됨

alpha alpha10 _alpha AlPha ALPHA	break 273alpha has space
--	--------------------------------

바른 예

바르지 않은 예

- 전 세계의 언어를 모두 사용할 수 있지만 **알파벳 사용이 관례**

Section 01 기본 용어(4)

■ 식별자(2) - **identifier**

- 식별자 의미를 더 명확하게 하기 위한 사용 규칙
 - 클래스, 속성, 메서드, 네임스페이스의 이름은 항상 **대문자**로 시작
 - 지역 변수와 전역 변수의 이름은 항상 소문자 시작
 - 여러 단어로 이루어진 식별자는 각 단어의 첫 글자를 대문자로 시작

```
i love you → iLoveYou  
i am a boy → iAmABoy  
create server → createServer
```

- 괄호가 있는 식별자는 메서드, 이외의 것은 변수, 메서드 괄호 안에 넣는 것은 매개변수 **Parameter**
(vs. **Argument**)

```
Console.WriteLine("Hello C# Programming");  
Math.PI;  
Math.Floor(10.1);  
Console.BackgroundColor
```

Section 01 기본 용어(5)

■ 주석

- 프로그램의 진행에 전혀 영향을 주지 않는 코드, 프로그램 설명에 사용 cf) **/// XML주**

방법	표현
한 줄 주석 처리	// 주석
여러 줄 주석 처리	/* 주석 주석 */

- 예

```
// 주석은 코드의 실행에 영향을 주지 않습니다.  
/*  
Console.WriteLine("C# Programming");  
Console.WriteLine("C# Programming");  
Console.WriteLine("C# Programming");  
*/
```

Section 01 기본 용어(5)

■ XML 주석을 활용한 코드 문서화

- 사용자 정의 형식(클래스)이나 멤버의 정의 위에 추가되는 특수 주석입니다. 컴파일 시간에 XML 문서 파일을 생성하기 위해 컴파일러에서 처리될 수 있으므로 특별합니다. Visual Studio 및 기타 IDE가 IntelliSense를 사용하여 형식이나 멤버에 대한 빠른 정보를 표시할 수 있도록 컴파일러에서 생성된 XML 파일은 .NET 어셈블리와 함께 배포될 수 있습니다.

■ ///

내용 및 예제

<https://docs.microsoft.com/ko-kr/dotnet/csharp/codedoc>

Section 02 출력

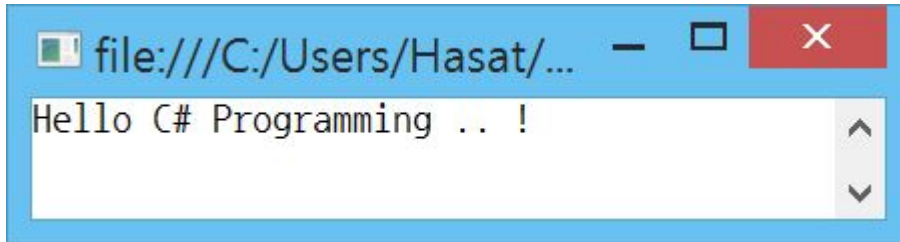
■ 출력 방법

- 방법1 : Console 클래스의 WriteLine () 메서드 사용

```
console.WriteLine("문자열");
```

그림 2-1 Console.WriteLine() 메서드 형태

- 방법2 : Write () 메서드 사용
 - WriteLine () 메서드를 사용하면 출력 후 개행, Write () 메서드는 출력 후 개행되지 않음
- 기본예제 2-1 C# 기본 출력 익히기



Section 03 기본 자료형(1)

■ 정수(1)

- 가장 기본적인 자료형(정수 : 273, 52, -103, 0처럼 하나하나 셀 수 있는 숫자)
- 정수 생성 예

코드 2-2 정수

/2장/DefaultData

```
static void Main(string[] args)
{
    Console.WriteLine(52);
}
```

Section 03 기본 자료형(1)

■ 정수(2)

- 사칙 연산자와 나머지 연산자로 연산 가능

표 2-3 기본적인 사칙 연산자

연산자	설명
+	덧셈 연산
-	뺄셈 연산
*	곱셈 연산
/	나눗셈 연산

표 2-4 나머지 연산자

연산자	설명
%	나머지 연산자

■ 예 1

코드 2-3 정수 덧셈 연산자

/2장/DefaultData

```
static void Main(string[] args)
{
    // 325를 출력합니다.
    Console.WriteLine(52 + 273);
}
```

Section 03 기본 자료형(2)

■ 정수(3)

■ 예2

코드 2-4 연산자 우선순위

/2장/DefaultData

```
static void Main(string[] args)
{
    // 결과를 예측해봅시다.
    Console.WriteLine(5 + 3 * 2);
}
```

■ 예3

코드 2-5 나머지 연산자

/2장/DefaultData

```
static void Main(string[] args)
{
    Console.WriteLine(10 % 5);
    Console.WriteLine(7 % 3);
}
```


Section 03 기본 자료형(2)

■ 정수(4)

- 정수 연산 주의 사항
 - 정수 연산 결과는 정수
 - 예를 들어 10/4는 2.5가 아니라 2

$$\text{정수} + \text{정수} = \text{정수}$$

$$\text{정수} - \text{정수} = \text{정수}$$

$$\text{정수} * \text{정수} = \text{정수}$$

$$\text{정수} / \text{정수} = \text{정수}$$

그림 2-2 정수 연산 결과

- 기본예제 2-2 정수와 연산자(교재 59p)

/2장/IntegerBasic



Section 03 기본 자료형(2)

■ 정수(5)

- 정수 연산 주의 사항
 - 정수 연산 결과는 정수
 - 예를 들어 $10/4$ 는 2.5가 아니라 2

$$\text{정수} + \text{정수} = \text{정수}$$

$$\text{정수} - \text{정수} = \text{정수}$$

$$\text{정수} * \text{정수} = \text{정수}$$

$$\text{정수} / \text{정수} = \text{정수}$$

그림 2-2 정수 연산 결과

■ 나머지 연산자와 부호

- 나머지 연산자의 부호는 왼쪽 피연산자의 부호를 따름
- 예

코드 2-7 음수와 나머지 연산자

/2장/DefaultData

```
static void Main(string[] args)
{
    Console.WriteLine(4 % 3);
    Console.WriteLine(-4 % 3);
    Console.WriteLine(4 % -3);
    Console.WriteLine(-4 % -3);
}
```



Section 03 기본 자료형(3)

■ 실수

- 실수를 만들려면 다음과 같이 소수점(.) 사용
- 예

코드 2-8 실수

/2장/DefaultData

```
static void Main(string[] args)
{
    Console.WriteLine(52.273);
}
```

코드 2-9 정수와 실수

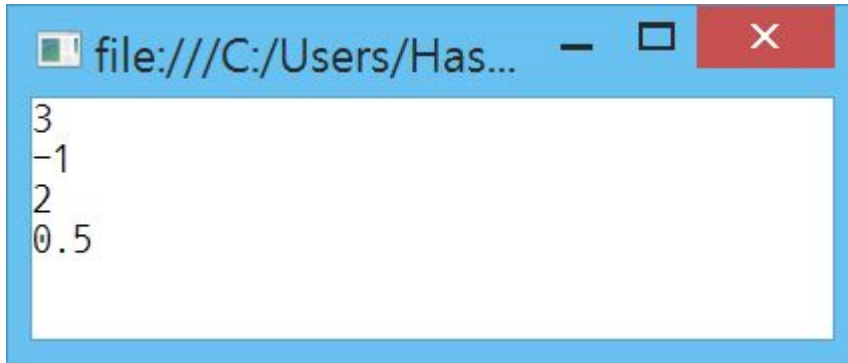
/2장/DefaultData

```
static void Main(string[] args)
{
    Console.WriteLine(0);           정수입니다.
    Console.WriteLine(0.0);        실수입니다.
}
```

- `간단식(1, 2, 3, 4) 또는 간단, 1, 2, 3, 4`
- `%(나머지 연산자)`: 사용은 가능하나 결과 예측이 어려워 비추천

Section 03 기본 자료형(4)

- 기본예제 2-3 실수와 사칙 연산자



Section 03 기본 자료형(5)

■ 문자

- 알파벳뿐만 아니라 모든 문자 표현 가능
- 그림 2-5 문자 표현



- 기본예제 2-4 문자



Section 03 기본 자료형(6)

■ 문자열

- 문자의 집합
- 예

코드 2-13 문자열

/2장/DefaultData

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

Section 03 기본 자료형(7)

■ 기본예제 2-5 이스케이프 문자(교재 64p)

/2장/EscapeCharacter



표 2-5 자주 사용되는 이스케이프 문자

이스케이프 문자	설명	이스케이프 문자	설명
\\t	수평 탭	\\\\	역 슬래시
\\n	행 바꿈	\\"	큰따옴표

Section 03 기본 자료형(8)

- 기본예제 2-6 문자열 연결 연산자(교재 65p)

/2장/StringConnection

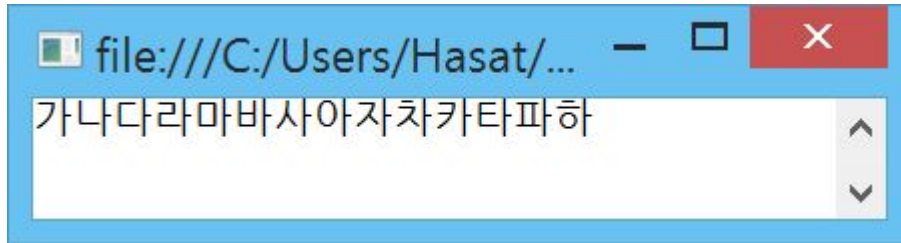


표 2-6 문자열 연결 연산자

연산자	설명
+	문자열 연결 연산자

Section 03 기본 자료형(9)

■ 기본예제 2-7 문자 선택(교재 65p)

/2장/StringSelector

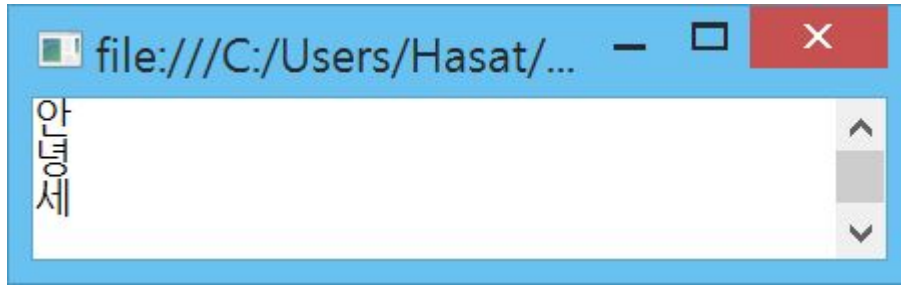


표 2-7 문자 선택 괄호

연산자	설명
문자열[숫자]	문자 선택 괄호

NOTE(1)

■ 예외

- 코드 실행 중 발생하는 오류(예외Exception, 런타임 에러Runtime Error)
- 예

코드 2-17 예외

/2장/DefaultData

```
static void Main(string[] args)
{
    Console.WriteLine("안녕하세요"[100]);
}
```

→ 그림 2-10

예외 발생(디버그 모드)

Microsoft Visual Studio Express 2015 for Windows Desktop



처리되지 않은 'System.IndexOutOfRangeException' 형식의 예외가 StringSelector.exe에서 발생했습니다.

추가 정보: 인덱스가 배열 범위를 벗어났습니다.

☐ 이 예외 형식이 throw되면 중단

[예외 설정 중단 및 열기\(S\)](#)

중단(B)

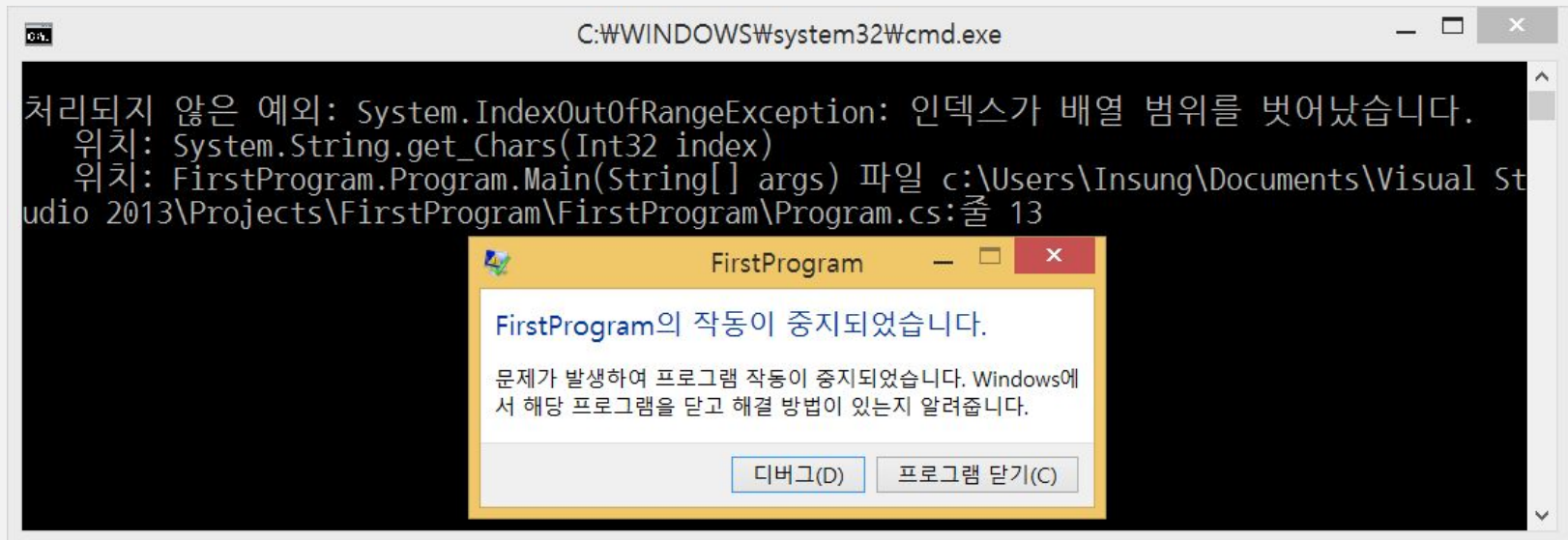
계속(C)

무시(I)

NOTE(1)

■ 예외

- 그림 2-7 예외 발생(릴리즈 모드)



NOTE(2)

■ 문자 덧셈 연산

- 문자열은 + 연산자로 연결 가능, 문자는 불가능

코드 2-18 문자 덧셈

/2장/DefaultData

```
static void Main(string[] args)
{
    Console.WriteLine('가' + '황');
}
```



Section 03 기본 자료형(6)

■ 불

- 참과 거짓의 표현(true와false 두 가지 값만 존재)
- 예

코드 2-19 불

/2장/DefaultData

```
static void Main(string[] args)
{
    Console.WriteLine(true);
    Console.WriteLine(false);
}
```

Section 03 기본 자료형(7)

■ 기본예제 2-8 불과 비교 연산자(교재 68p)

/2장/BoolBasic



표 2-8 비교 연산자

연산자	설명
==	같다
!=	다르다
>	왼쪽 피연산자가 크다
<	오른쪽 피연산자가 크다
>=	왼쪽 피연산자가 크거나 같다
<=	오른쪽 피연산자가 크거나 같다

Section 03 기본 자료형(8)

■ 기본예제 2-9 논리 부정 연산자(교재 69p)

/2장/LogicalNot

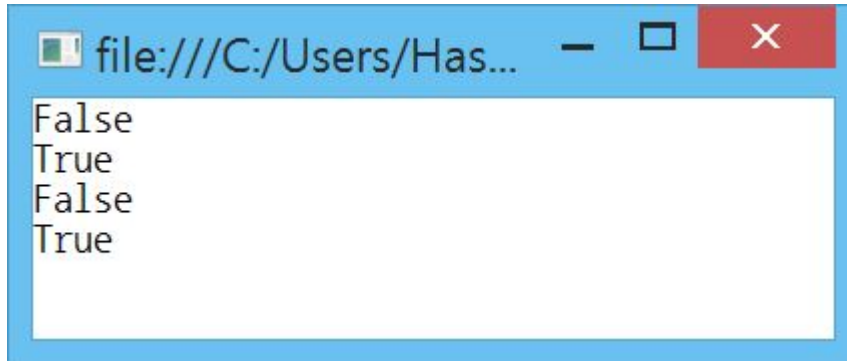


표 2-9 논리 연산자

연산자	설명
!	논리 부정 연산자
	논리합 연산자
&&	논리곱 연산자

- 논리 연산자는 소괄호로 묶어서 사용 가능 - 연산자와 같은 형태로 사용
- 논리 부정 연산자는 피연산자를 하나만 갖는 단항 연산자
- 피연산자의 개수에 따라 단항 연산자, 이항 연산자, 삼항 연산자라고 함

Section 03 기본 자료형(9)

- 논리합 연산자(or)

표 2-10 논리합 연산자

왼쪽 피연산자	오른쪽 피연산자	결과
true	true	true
true	false	true
false	true	true
false	false	false

- 논리곱 연산자(and)

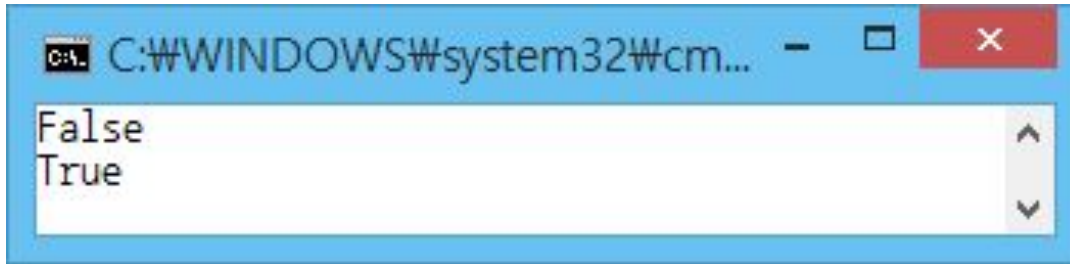
표 2-11 논리곱 연산자

왼쪽 피연산자	오른쪽 피연산자	결과
true	true	true
true	false	false
false	true	false
false	false	false

Section 03 기본 자료형(10)

- 기본예제 2-10 불과 논리 연산자(교재 72p)

/2장/LogicalOperator



Section 04 변수(1)

■ 변수

- 값을 저장할 때 사용하는 식별자
- 숫자뿐만 아니라 모든 자료형 저장
- 변수 사용 단계
 - 변수 선언(변수를 만드는 것)
 - 변수에 값 할당

double pi = 3.14159265

자료형 이름 값

그림 2-12 변수 선언 예

Section 04 변수(2)

■ 정수 자료형

표 2-12 정수 자료형

키워드	설명
int	4바이트의 정수
long	8바이트의 정수

int a = 10

long b = 20

그림 2-13 정수 자료형 선언 예

■ 기본예제 2-11 정수 변수 생성(교재 74p)

/2장/IntegerVariable



Section 04 변수(3)

▪ int 자료형

- 일반적으로 정수를 만들 때 사용
- 크기 : 4바이트(32비트), 범위 : 2^{32} 개의 숫자(-2,147,483,648~2,147,483,647) 나타냄
- 오버플로우 : int 자료형의 범위를 넘는 현상
- 예

코드 2-24 오버플로우

/2장/Variables

```
static void Main(string[] args)
{
    int a = 2147483640;
    int b = 52273;

    Console.WriteLine(a + b);
}
```

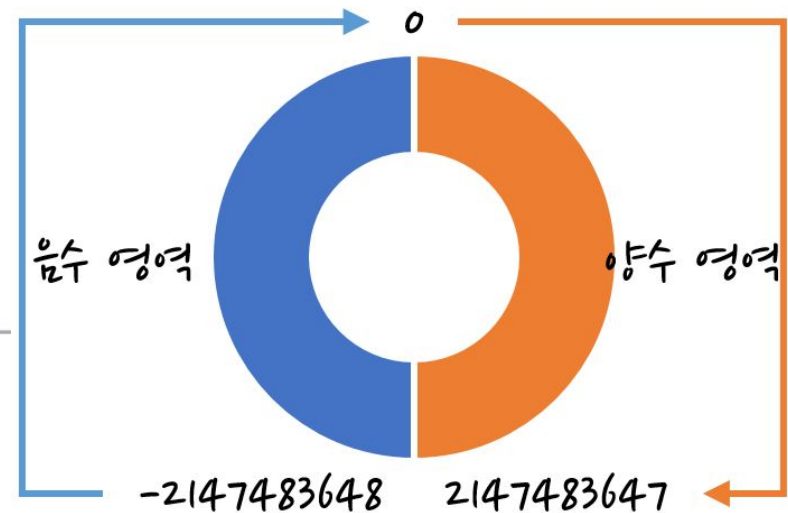
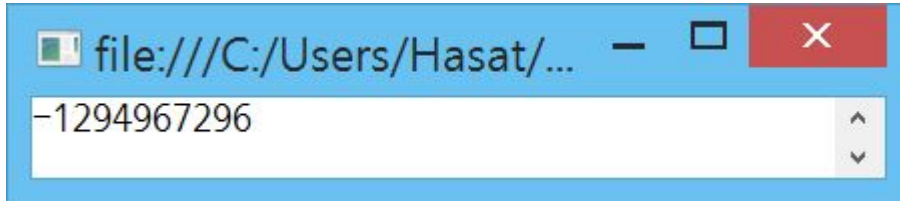


그림 2-17 int 자료형의 범위

Section 04 변수(4)

- 기본예제 2-12 오버플로우(교재 76p)

/2장/Overflow



- 오버플로우 문제 해결 방법 : 자료형 변환
- 예

코드 2-26 자료형 변환을 사용한 해결 방법 3가지

```
static void Main(string[] args)
{
    int a = 2000000000;
    int b = 1000000000;

    Console.WriteLine((long)a + b);
    Console.WriteLine(a + (long)b);
    Console.WriteLine((long)a + (long)b);
}
```

NOTE(1)

■ unsigned 자료형(부호가 없는 자료형)

- 음수 사용을 위한 자료형
- uint와 ulong 키워드 사용
- 예

코드 2-27 uint와 ulong 자료형

/2장/Variables

```
static void Main(string[] args)
{
    uint unsignedInt = 4147483647;
    ulong unsignedLong = 11223372036854775808;

    Console.WriteLine(unsignedInt);
    Console.WriteLine(unsignedLong);
}
```

NOTE(2)

■ MaxValue와 MinValue

■ 예

코드 2-28 int 자료형의 최댓값과 최솟값

/2장/Variables

```
static void Main(string[] args)
{
    Console.WriteLine(int.MaxValue);
    Console.WriteLine(int.MinValue);
}
```

코드 2-29 long 자료형의 최댓값과 최솟값

```
static void Main(string[] args)
{
    Console.WriteLine(long.MaxValue);
    Console.WriteLine(long.MinValue);
}
```


NOTE(3)

■ L

- 폰트에 따라 1과 소문자 l 혼동, 코드 작성 시 long 자료형 나타내는 대문자 L 사용
- 예

코드 2-30 long 자료형을 나타내는 기호: 소문자

```
static void Main(string[] args)
{
    Console.WriteLine(123456 + 65432l);
}
```

코드 2-31 long 자료형을 나타내는 기호: 대문자

```
static void Main(string[] args)
{
    Console.WriteLine(123456 + 65432L);
}
```

Section 04 변수(5)

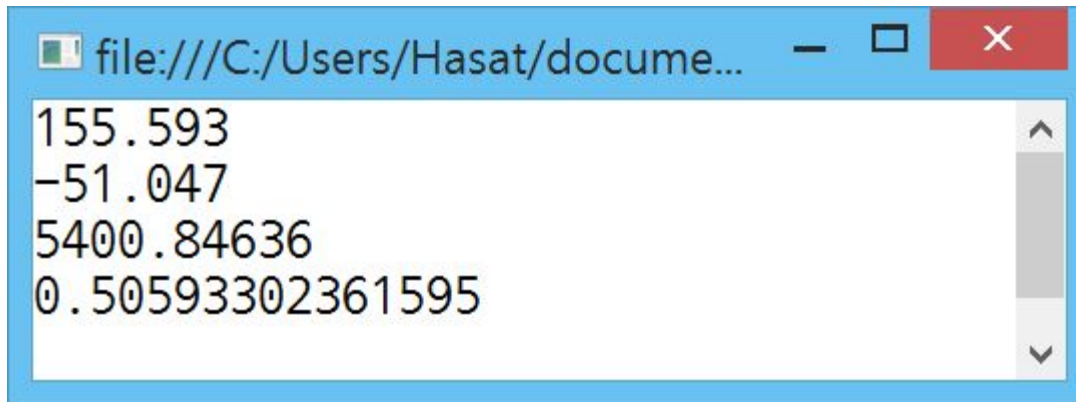
■ 실수 자료형

표 2-13 실수 자료형

키워드	설명
float	4바이트의 실수
double	8바이트의 실수

- 기본예제 2-13 실수 변수 생성(교재 80p)

/2장/RealNumberVariable



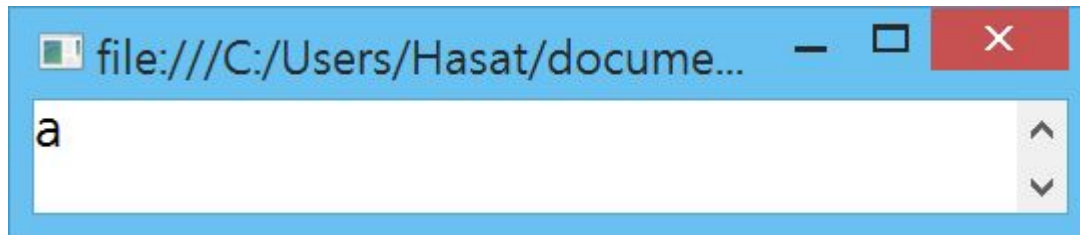
Section 04 변수(6)

■ 문자 자료형

표 2-14 문자 자료형

키워드	설명
char	문자

- 기본예제 2-14 문자 변수 생성(교재 81p) [/2장/CharacterVariable](#)



NOTE(1)

■ sizeof 연산자와 char 자료형의 크기

■ 예

코드 2-34 sizeof 연산자

/2장/Variables

```
static void Main(string[] args)
{
    Console.WriteLine("int: " + sizeof(int));
    Console.WriteLine("long: " + sizeof(long));
    Console.WriteLine("float: " + sizeof(float));
    Console.WriteLine("double: " + sizeof(double));
    Console.WriteLine("char: " + sizeof(char));
}
```

NOTE(2)

■ 문자 자료형과 연산자

- 문자 자료형은 문자열 자료형보다 정수에 가까움(연산가능)
- 예

코드 2-35 문자 자료형과 연산자

/2장/Variables

```
static void Main(string[] args)
{
    char a = 'a';
    char b = 'b';

    Console.WriteLine(a + b);
    Console.WriteLine(a - b);
    Console.WriteLine(a * b);
    Console.WriteLine(a / b);
    Console.WriteLine(a % b);
}
```

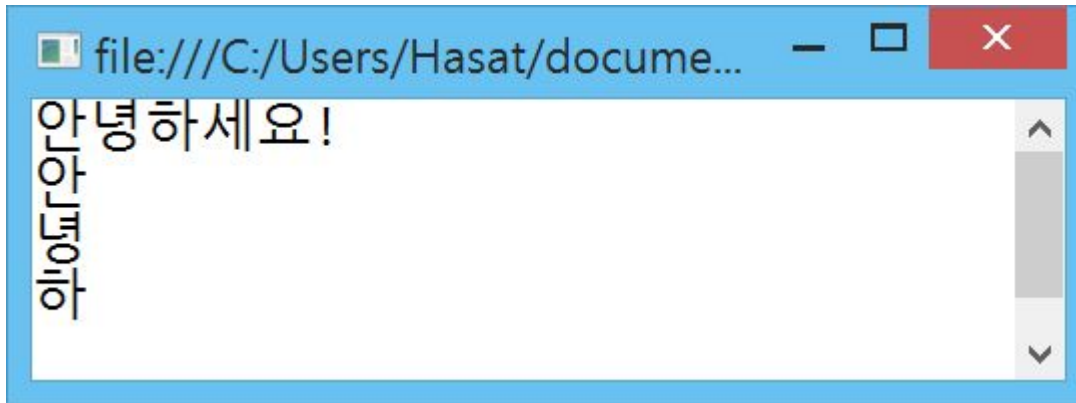
Section 04 변수(7)

■ 문자열 자료형

표 2-15 문자열 자료형

키워드	설명
string	문자열 자료형

- 기본예제 2-15 문자열 변수 생성(교재 83p) [/2장/StringVariable](#)



■ sizeof 연산자와 string 자료형

- string 자료형은 sizeof 연산자로 자료형의 크기를 구할 수 없음
 - string 자료형만 struct로 시작하지 않고 class로 시작
 - 예

코드 2-37 sizeof 연산자와 string 자료형

/2장/Variables

```
static void Main(string[] args)
{
    Console.WriteLine("string: " + sizeof(string));
}
```

```
int output = 0;
```

struct System.Int32
부호 있는 32비트 정수를 나타냅니다.

표 2-16 기본 자료형의 원형

기본 자료형	원형	기본 자료형	원형
int	struct System.Int32	float	struct System.Single
long	struct System.Int64	double	struct System.Double
char	struct System.Char	bool	struct System.Boolean
string	class System.String		

Section 04 변수(8)

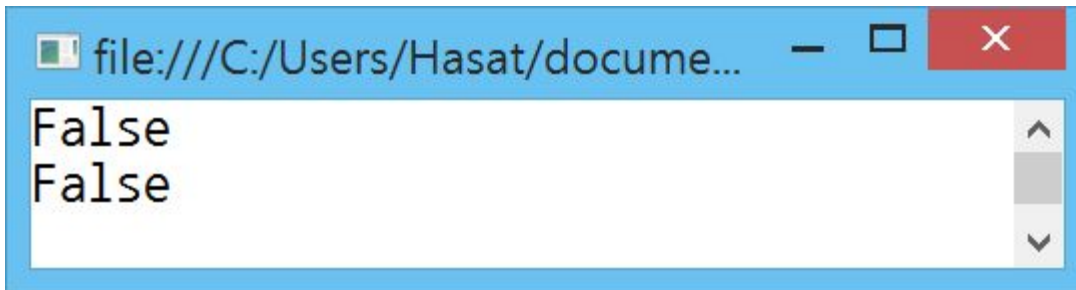
■ 불 자료형

표 2-17 불 자료형

키워드	설명
bool	불 자료형

- 기본예제 2-16 불 변수 생성(교재 84p)

/2장/BoolVariable



A screenshot of a text editor window with a blue title bar. The title bar contains a file path: `file:///C:/Users/Hasat/docume...` and standard window controls (minimize, maximize, close). The editor area has a white background and contains two lines of text, both reading `False`. A vertical scrollbar is visible on the right side of the text area.

Section 05 복합 대입 연산자(1)

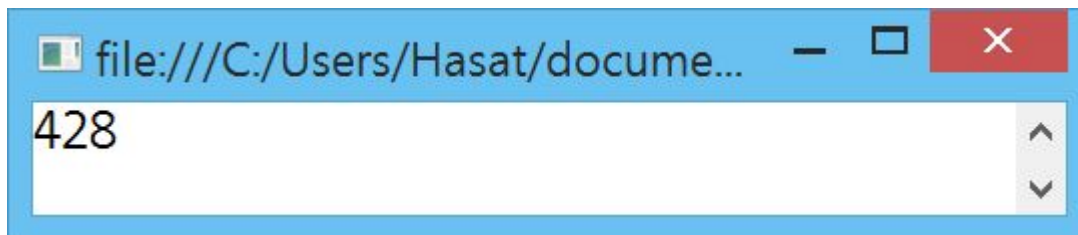
■ 복합 대입 연산자

- 자료형에 적용하는 기본 연산자와 = 연산자를 함께 사용
 - $a+=10$ 은 $a=a+10$ 을 뜻함

표 2-18 숫자에 적용하는 복합 대입 연산자

연산자 이름	설명
$+=$	숫자 덧셈 후 대입 연산자
$-=$	숫자 뺄셈 후 대입 연산자
$*=$	숫자 곱셈 후 대입 연산자
$/=$	숫자 나눗셈 후 대입 연산자

- **기본예제 2-17** 숫자와 관련된 복합 대입 연산자(교재 86p) [/2장/AssignmentOperator](#)



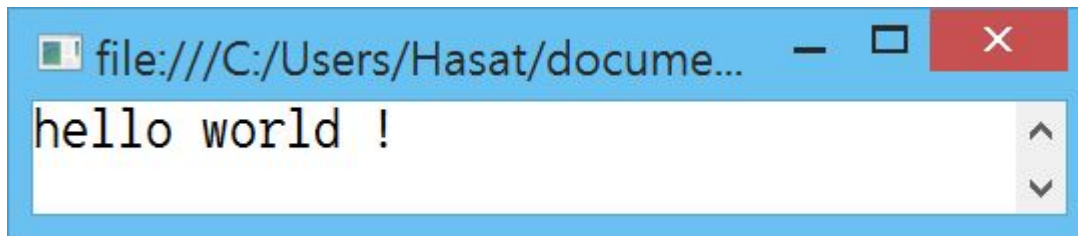
Section 05 복합 대입 연산자(2)

■ 복합 대입 연산자

표 2-19 문자열에 적용하는 복합 대입 연산자

연산자 이름	설명
<code>+=</code>	문자열 연결 후 대입 연산자

- **기본예제 2-18** 문자와 관련된 복합 대입 연산자(교재 87p) /2장/StringAssignmentOperator



Section 06 증감 연산자(1)

■ 증감 연산자 사용

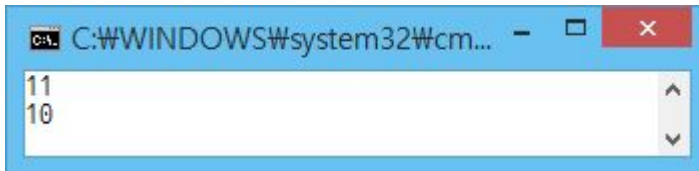
- 단항 연산자로 변수 앞과 뒤에 ++ 기호와 -- 기호 붙여 만듦

표 2-20 증감 연산자

연산자	설명
[변수]++	기존 변수의 값에 1을 더합니다(후위).
++[변수]	기존 변수의 값에 1을 더합니다(전위).
[변수]--	기존 변수의 값에서 1을 뺍니다(후위).
--[변수]	기존 변수의 값에서 1을 뺍니다(전위).

- 기본예제 2-19 증감 연산자(교재 88p)

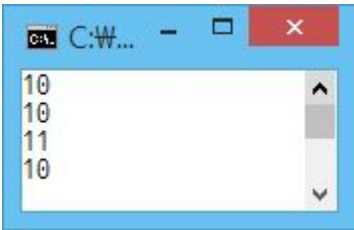
/2장/IncrementOperator



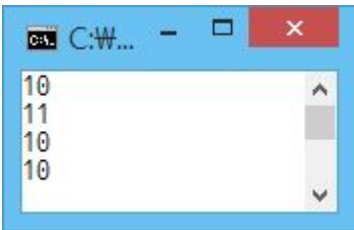
Section 06 증감 연산자(2)

■ 증감 연산자 사용

- 기본예제 2-20 증감 연산자의 전위와 후위 (교재 89p) /2장/IncrementOperatorPosition
- 전위 : 해당 문장을 실행하기 전에 값을 변경



- 후위 : 문장을 실행한 이후에 값을 변경



Section 07 자료형 검사(1)

■ 자료형 검사 방법

- 방법1 : 마우스 가져다 대기

```
int number = 10;  
Console.WriteLine(number);
```

(지역 변수) int number

그림 2-24 마우스를 사용한 자료형 확인

- 방법2 : GetType () 메서드(프로그램 내부에서 자료형 확인)

표 2-21 자료형 확인 메서드

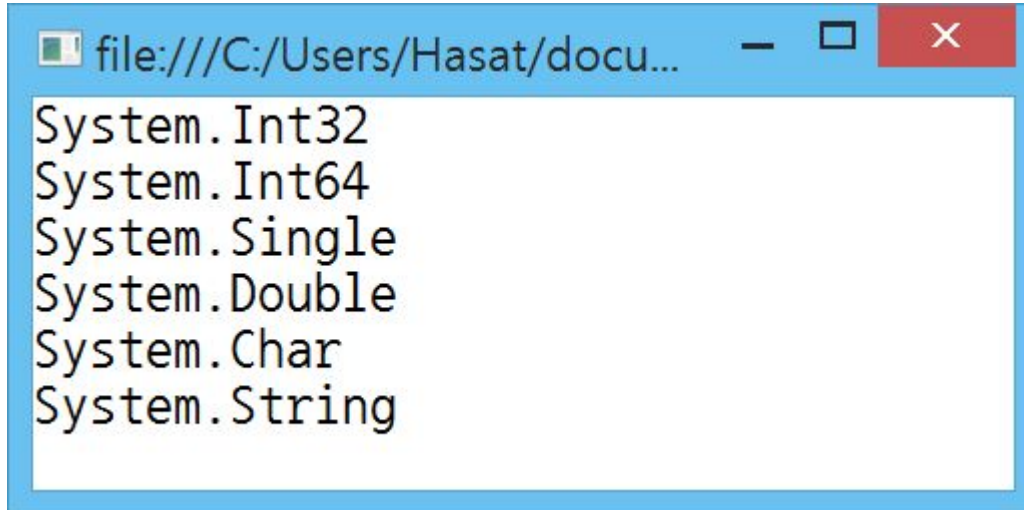
메서드	설명
GetType()	해당 변수의 자료형을 추출합니다.

- 변수뿐만 아니라 숫자 또는 문자열에 직접 적용 가능

Section 07 자료형 검사(2)

- 기본예제 2-21 GetType() 메서드 활용(교재 92p)

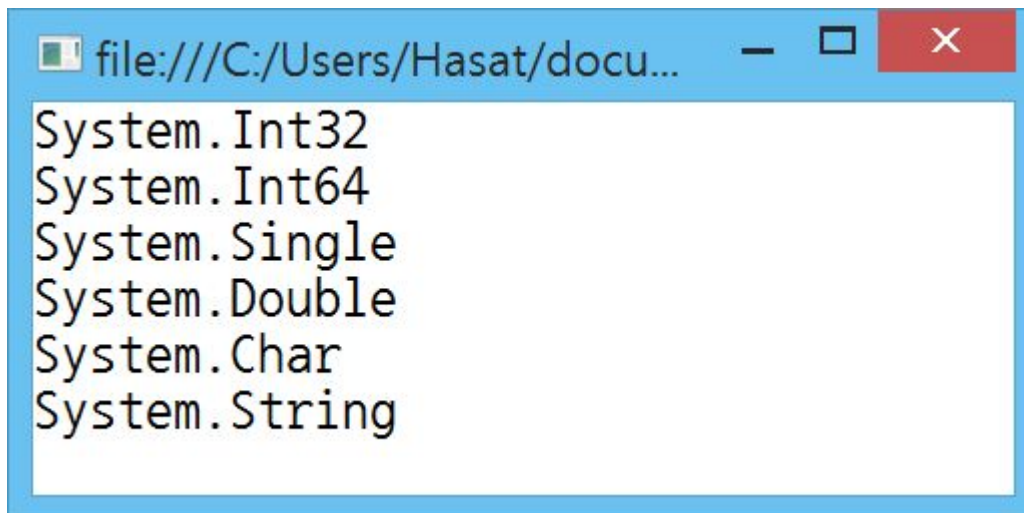
/2장/TypeCheck



A screenshot of a Windows file explorer window. The title bar shows the path 'file:///C:/Users/Hasat/docu...'. The main content area displays a list of system types: System.Int32, System.Int64, System.Single, System.Double, System.Char, and System.String.

- 기본예제 2-22 직접적인 GetType() 메서드 활용(교재 93p)

/2장/DirectTypeCheck



A screenshot of a Windows file explorer window. The title bar shows the path 'file:///C:/Users/Hasat/docu...'. The main content area displays a list of system types: System.Int32, System.Int64, System.Single, System.Double, System.Char, and System.String.

Section 08 var 키워드(1)

■ var 키워드 사용

표 2-22 var 키워드

var 키워드	설명
var	자료형을 자동으로 지정합니다.

코드 2-50 var 키워드

/2장/VarKeyword

```
static void Main(string[] args)
{
    var number = 100;
}
```

- 한 번 지정된 자료형은 계속 유지
- int 자료형으로 선언된 변수를 string 자료형으로 바꾸는 것은 불가능

코드 2-51 var 키워드의 제약

/2장/VarKeyword

```
var number = 10.2;
number = "변경";
```

Section 08 var 키워드(2)

■ var 키워드 추가 사용 조건

■ 지역 변수로 선언

- 지역 변수 : 메서드 내부에 선언되어 있는 변수
- 예

코드 2-52 지역 변수

/2장/VarKeyword

```
class Program
{
    var global = 52; // 인스턴스 변수(var 키워드 사용 불가)

    static void Main(string[] args)
    {
        var local = 273; // 지역 변수(var 키워드 사용 가능)
    }
}
```

■ 변수를 선언할 동시에 초기화

- 예

코드 2-53 var 키워드의 선언과 초기화 동시 수행

/2장/VarKeyword

```
static void Main(string[] args)
{
    var number = 20;
}
```


■ var 키워드 선언

- 정수 선언 시 `var number = 100` 입력, `int` 자료형으로 선언
- 실수 선언 시 `var number = 10.0` 입력, `double` 자료형으로 선언
- `long` 자료형, `float` 자료형 선언 시, 숫자 뒤에 `L`, `F` 등 기호 붙여야 함
- 예

코드 2-54 var 키워드를 사용한 다양한 자료형 선언

/2장/VarKeyword

```
static void Main(string[] args)
{
    var numberA = 100L;    // long 자료형
    var numberB = 100.0;   // double 자료형
    var numberC = 100.0F;  // float 자료형
}
```

Section 09 입력

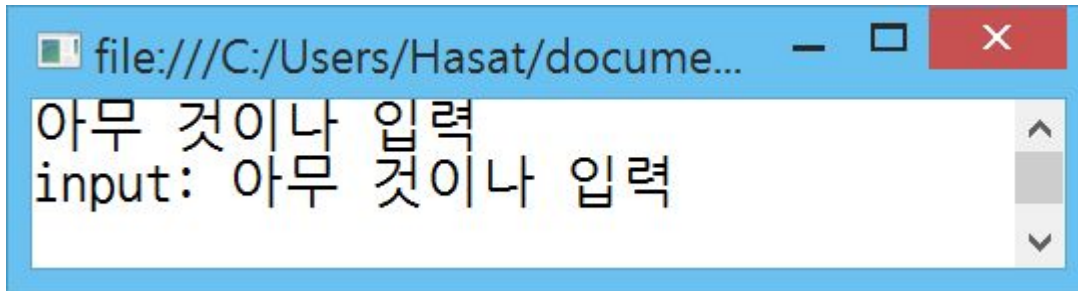
■ 입력

표 2-23 입력 메서드

메서드 이름	설명
<code>Console.ReadLine()</code>	사용자로부터 한 줄의 문자열을 입력 받습니다.

- 기본예제 2-23 문자열 입력과 출력 (교재 98p)

/2장/Input



- `Console.ReadLine()` 메서드는 문자열만 입력 가능
- 숫자를 입력 받아 더하거나 하는 프로그램을 만들려면 문자열을 숫자로 바꾸는 방법 필요

Section 10 자료형 변환(1)

■ 자료형 변환

- 한 자료형을 다른 자료형으로 바꾸는 것
- 예

코드 2-56 자료형 변환

/2장/Casts

```
static void Main(string[] args)
{
    // long 자료형을 int 자료형으로 변환합니다.
    long longNumber = 2147483647L + 2147483647L;
    int intNumber = longNumber;
    Console.WriteLine(intNumber);
}
```

Section 10 자료형 변환(2)

■ 강제 자료형 변환

```
static void Main(string[] args)
{
    // long 자료형을 int 자료형으로 변환합니다.
    long longNumber = 2147483647L + 2147483647L;
    int intNumber = longNumber;
    Console.WriteLine
}
```

(지역 변수) long longNumber

오류:

암시적으로 'long' 형식을 'int' 형식으로 변환할 수 없습니다. 명시적 변환이 있습니다. 캐스트가 있는지 확인하십시오.

그림 2-30 자료형 변환 실패

■ 예

코드 2-57 강제 자료형 변환

/2장/Casts

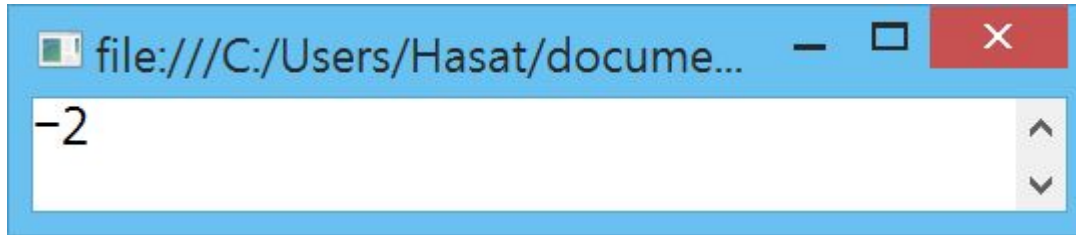
```
var a = (int)10.0;
var b = (float)10;
var c = (double)10;
```

Section 10 자료형 변환(3)

■ 강제 자료형 변환

- 기본예제 2-24 강제 자료형 변환 (교재 100p)

/2장/ExplicitConversion



- 강제 자료형 변환 데이터 손실 발생하지 않는 예

코드 2-59 강제 자료형 변환의 데이터 손실이 항상 일어나는 것은 아님

/2장/Casts

```
static void Main(string[] args)
{
    // long 자료형을 int 자료형으로 변환합니다.
    long longNumber = 52273;
    int intNumber = (int)longNumber;
    Console.WriteLine(intNumber);
}
```

Section 10 자료형 변환(4)

■ 자동 자료형 변경

- 기본예제 2-25 숫자 손상(교재 101p)

/2장/NumberLost

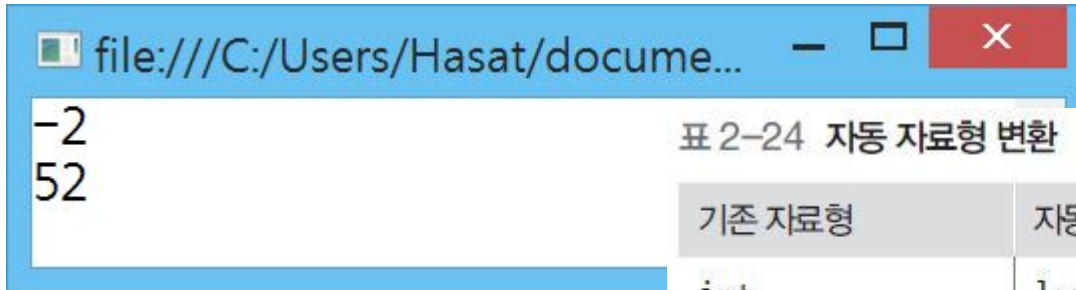
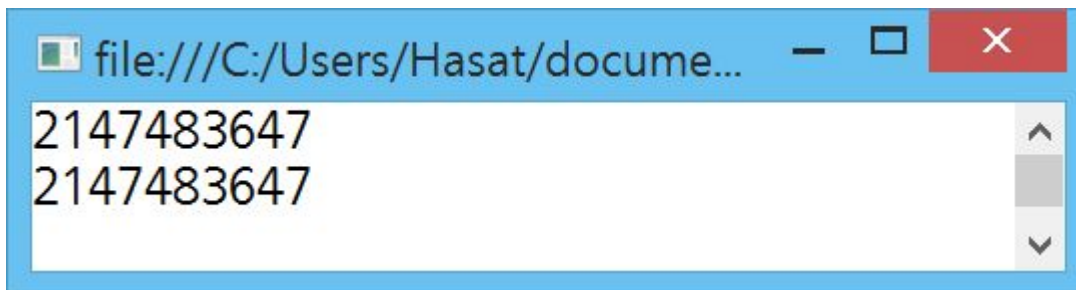


표 2-24 자동 자료형 변환

기존 자료형	자동 변환되는 자료형
int	long, float, double
long	float, double
char	int, long, float, double
float	double

- 기본예제 2-26 자동 자료형 변환(교재 102p)

/2장/ImplicitConversion



Section 10 자료형 변환(5)

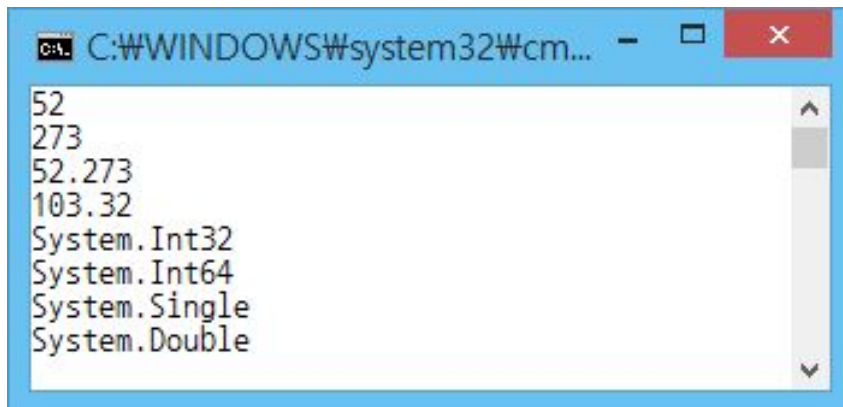
■ 다른 자료형을 숫자로 변환

표 2-25 문자열을 숫자로 변환하는 메서드

메서드 이름	설명
<code>int.Parse()</code>	다른 자료형을 <code>int</code> 자료형으로 변경합니다.
<code>long.Parse()</code>	다른 자료형을 <code>long</code> 자료형으로 변경합니다.
<code>float.Parse()</code>	다른 자료형을 <code>float</code> 자료형으로 변경합니다.
<code>double.Parse()</code>	다른 자료형을 <code>double</code> 자료형으로 변경합니다.

■ 기본예제 2-27 문자열을 숫자로 변환(교재 104p)

/2장/StringTo



```
C:\WINDOWS\system32\cmd...  
52  
273  
52.273  
103.32  
System.Int32  
System.Int64  
System.Single  
System.Double
```

■ FormatException 예외

■ 예

코드 2-64 숫자로 변환할 수 없는 문자열을 변환하는 경우

/2장/Casts

```
Console.WriteLine(int.Parse("52.273"));  
Console.WriteLine(int.Parse("abc"));
```

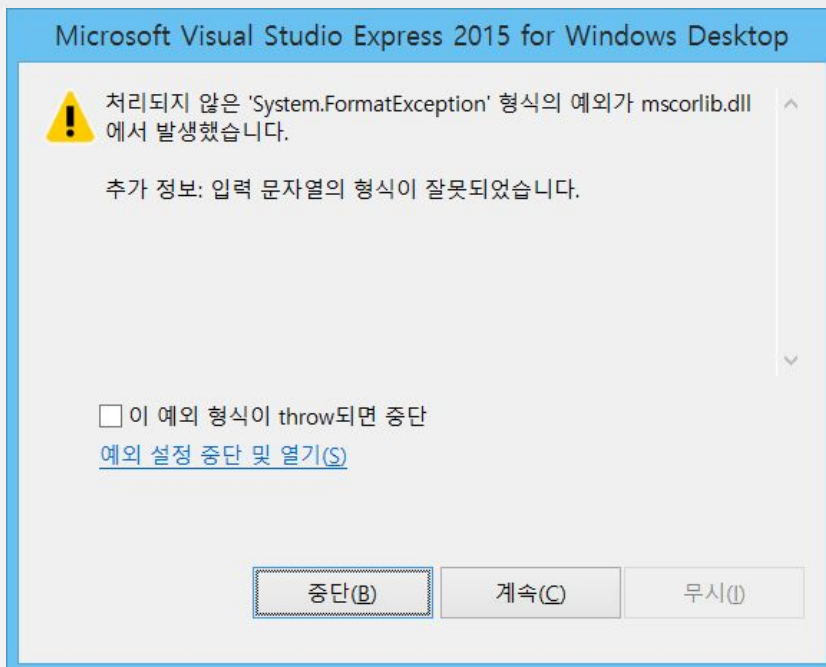


그림 2-31 FormatException 예외 발생

Section 10 자료형 변환(6)

■ 다른 자료형을 문자열로 변환

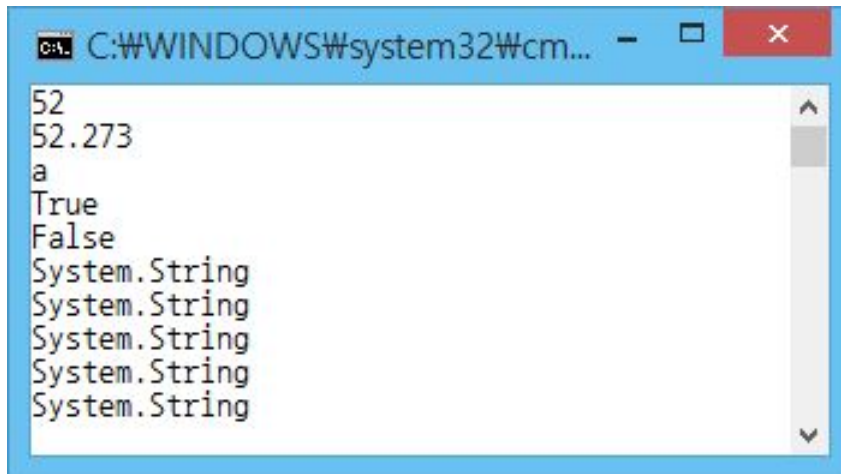
- C#의 모든 자료형은 ToString () 메서드를 가지고 있음

표 2-26 문자열로 변환하는 메서드

메서드	설명
ToString()	문자열로 변환합니다.

- **기본예제 2-28** 문기본 자료형을 문자열로 변환(교재 104p)

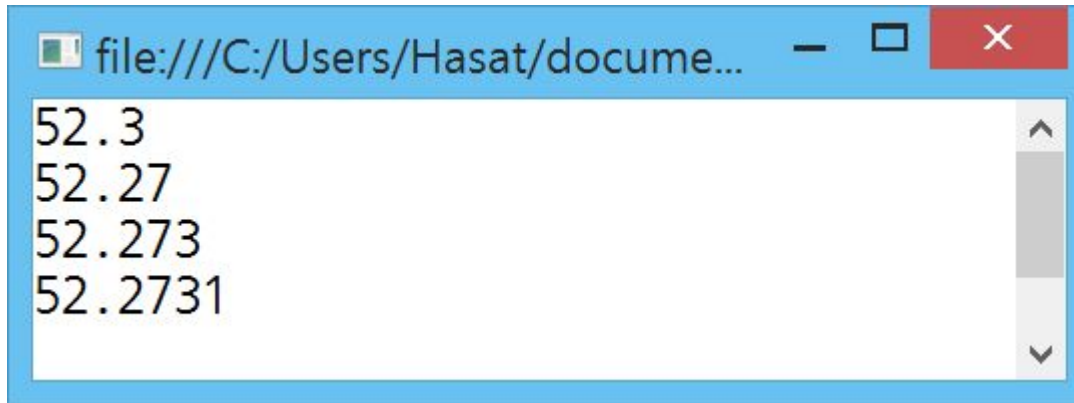
/2장/ToStringBasic



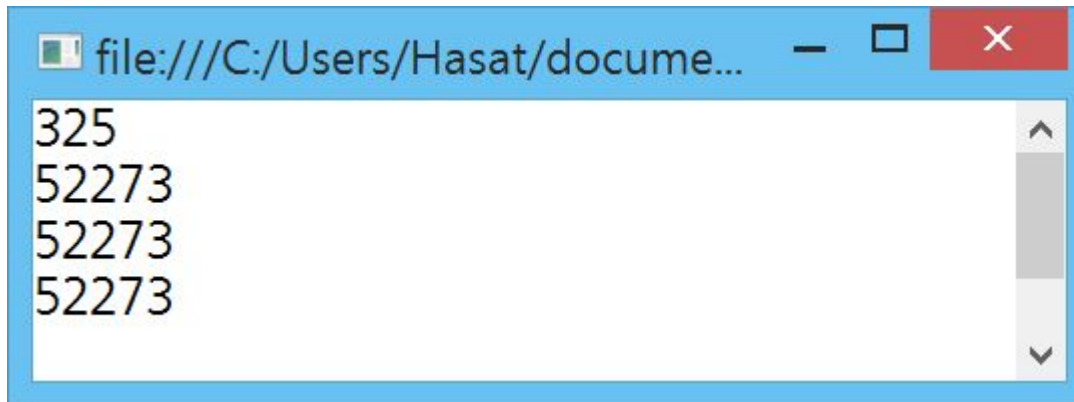
```
C:\WINDOWS\system32\cmd.exe
52
52.273
a
True
False
System.String
System.String
System.String
System.String
System.String
```

Section 10 자료형 변환(7)

- 기본예제 2-29 소숫점 제거(교재 107p) /2장/DoubleToString



- 기본예제 2-30 숫자와 문자열 덧셈(교재 108p) /2장/StringPlusNumber



■ 간단한 문자열 변환

■ 예

코드 2-68 간단한 문자열 변환

/2장/Casts

```
int number = 52273;
string outputA = number + "";
Console.WriteLine(outputA);

char character = 'a';
string outputB = character + "";
Console.WriteLine(outputB);
```

```
char character = 'a';
string output = character;
```

(지역 변수) char character

오류:

암시적으로 'char' 형식을 'string' 형식으로 변환할 수 없습니다.

그림 2-32 문자 문자열 변환 오류

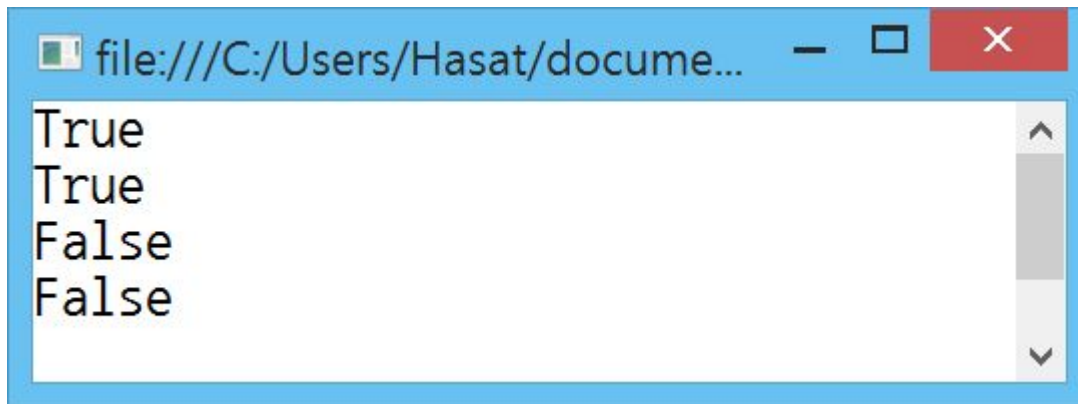
Section 10 자료형 변환(8)

■ 다른 자료형을 불로 변환

표 2-27 문자열을 불로 변환하는 메서드

메서드	설명
<code>bool.Parse()</code>	문자열을 불 자료형으로 변환합니다.

- 기본예제 2-31 문자열을 불로 전환(교재 109p) [/2장/StringToBool](#)



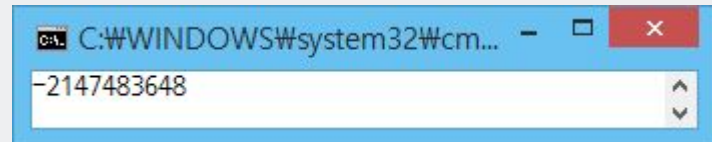
The image shows a Windows file explorer window with the title bar 'file:///C:/Users/Hasat/docume...'. The main content area displays a text file with the following lines: True, True, False, False. The window has a blue border and standard Windows window controls (minimize, maximize, close) in the top right corner.

■ 음수밖에 없는 숫자

■ 예

코드 2-70 int 자료형 최솟값의 음수

```
static void Main(string[] args)
{
    int output = int.MinValue;
    Console.WriteLine(-output);
}
```



```
Console.WriteLine(-(-2147483648));
```

struct System.Int32
부호 있는 32비트 정수를 나타냅니다.

오류:

checked 모드에서 컴파일하면 작업이 오버플로됩니다.

그림 2-34 숫자를 직접 입력하면 오류 발생



Thank You
