Chapter 02 C# 기본

Section 01 기본 용어(1)

■ 표현식

- 값을 만들어 내는 간단한 코드
- 예

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

■ 문장

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

```
• 273;
• 10 + 20 + 30 + 2;
• var name = "윤" + "인" + "성"
• Console.Write("Hello C# Programming");
```

Section 01 기본 용어(2)

■ 키워드(1)

- 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

Section 01 기본 용어(2)

■ 키워드(1)

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

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)

표 2-2 컨텍스트 키워드(또는 문맥 키워드)

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

Section 01 기본 용어(4)

■ 식별자(1)

- C#에서 변수와 메서드 이름 식별자 규칙
  - 키워드를 사용하면 안 됨
  - 특수 문자는 _만 허용
  - 숫자로 시작하면 안 됨
  - 공백은 입력하면 안 됨

바른 예	바르지 않은 예
alpha	break
alpha10	273alpha
_alpha	has space
AlPha	
ALPHA	

- 전 세계의 언어를 모두 사용할 수 있지만 알파벳 사용이 관례

Section 01 기본 용어(4)

■ 식별자(2)

- 식별자 의미를 더 명확하게 하기 위한 사용 규칙
  - 클래스, 속성, 메서드, 네임스페이스의 이름은 항상 대문자로 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 매개변수Parameter

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

Section 01 기본 용어(3)

■ 주석

- 프로그램의 진행에 전혀 영향을 주지 않는 코드, 프로그램 설명에 사용

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 02 출력

■ 출력 방법

- 방법1 : Console 클래스의 WriteLine ( ) 메서드 사용

```
Console.WriteLine(출력하고_싶은_대상);
```

그림 2-1 Console.WriteLine() 메서드의 형태

- 방법2 : Write ( ) 메서드 사용
  - WriteLine ( ) 메서드를 사용하면 출력 후 개행, Write ( ) 메서드는 출력 후 개행되지 않음
- **기본예제 2-1** C# 기본 출력 익히기(교재 62p) /2장/Output

코드 2-1 Hello World 예제

```
01  static void Main(string[] args)
02  {
03      Console.WriteLine("Hello C# Programming .. !");
04  }
```

실행 결과

```
Hello C# Programming .. !
```

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 생성 /2장/DefaultData

```
01 static void Main(string[] args)
02 {
03     Console.WriteLine(52);
04 }
```

Section 03 기본 자료형(1)

■ 정수(2)

- 사칙 연산자와 나머지 연산자로 연산 가능

표 2-4 기본적인 사칙 연산자

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

표 2-5 나머지 연산자

연산자	설명
%	나머지 연산자

- 예 1

코드 2-3 정수 덧셈 연산자 /2장/DefaultData

```
01 static void Main(string[] args)
02 {
03     // 325를 출력합니다.
04     Console.WriteLine(52 + 273);
05 }
```

Section 03 기본 자료형(2)

■ 정수(3)

- 예2

코드 2-4 연산자 우선순위 /2장/DefaultData

```
01 static void Main(string[] args)
02 {
03     // 결과를 예측해봅시다.
04     Console.WriteLine(5 + 3 * 2);
05 }
```

- 예3

코드 2-5 나머지 연산자 /2장/DefaultData

```
01 static void Main(string[] args)
02 {
03     Console.WriteLine(10 % 5);
04     Console.WriteLine(7 % 3);
05 }
```

■ 정수(4)

- 정수 연산 주의 사항
  - 정수 연산 결과는 정수
  - 예를 들어 10/4는 2.5가 아니라 2
- **기본예제 2-2** 정수와 연산자(교재 66p) /2장/IntegerBasic

코드 2-6 정수와 연산자

```
01  static void Main(string[] args)
02  {
03      Console.WriteLine(1 + 2);
04      Console.WriteLine(1 - 2);
05      Console.WriteLine(1 * 2);
06      Console.WriteLine(1 / 2);
07      Console.WriteLine(1 % 2);
08  }
```

실행 결과

```
3
-1
2
0
1
```

■ 정수(5)

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

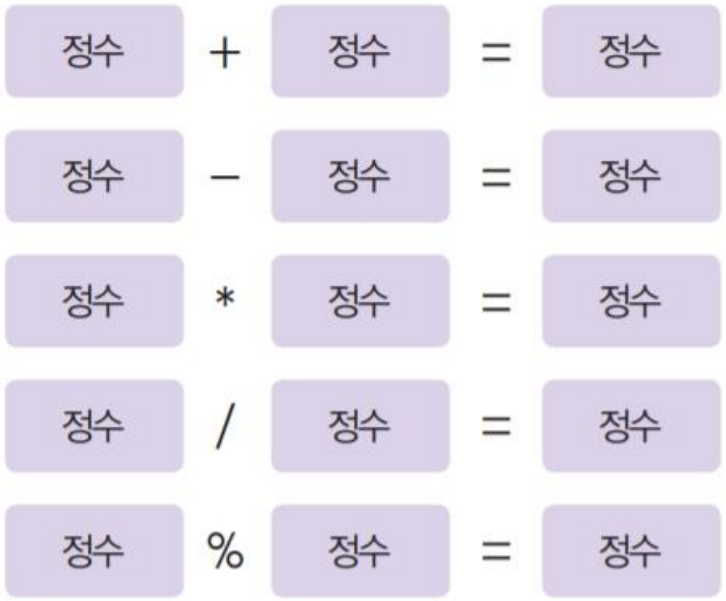

그림 2-2 정수 연산 결과

■ 나머지 연산자와 부호

- 나머지 연산자의 부호는 왼쪽 피연산자의 부호를 따름
- 예

코드 2-7 음수와 나머지 연산자 /2장/DefaultData

```
01 static void Main(string[] args)
02 {
03     Console.WriteLine(4 % 3);
04     Console.WriteLine(-4 % 3);
05     Console.WriteLine(4 % -3);
06     Console.WriteLine(-4 % -3);
07 }
```

실행 결과

```
1
-1
1
-1
```

Section 03 기본 자료형(3)

■ 실수

- 실수를 만들려면 다음과 같이 소수점(.) 사용
- 예

코드 2-8 실수 /2장/DefaultData

```
01 static void Main(string[] args)
02 {
03     Console.WriteLine(52.273);
04 }
```

코드 2-9 정수와 실수 /2장/DefaultData

```
01 static void Main(string[] args)
02 {
03     Console.WriteLine(0);      // 정수입니다.
04     Console.WriteLine(0.0);    // 실수입니다.
05 }
```

- 연산자(+, -, *, /)로 사칙연산,
- %(나머지 연산자) : 사용은 가능하나 결과 예측이 어려워 비추천

Section 03 기본 자료형(4)

- **기본예제 2-3** 실수와 사칙 연산자(교재 68p) /2장/RealNumberBasic

코드 2-10 실수와 사칙 연산자

```
01 static void Main(string[] args)
02 {
03     Console.WriteLine(1.0 + 2.0);
04     Console.WriteLine(1.0 - 2.0);
05     Console.WriteLine(1.0 * 2.0);
06     Console.WriteLine(1.0 / 2.0);
07 }
```

실행 결과

```
3
-1
2
0.5
```

Section 03 기본 자료형(5)

■ 문자

- 알파벳뿐만 아니라 모든 문자 표현 가능

'a'

그림 2-3 문자 표현

- **기본예제 2-4** 문자(교재 69p) /2장/CharacterBasic

코드 2-12 문자

```
01  static void Main(string[] args)
02  {
03      Console.WriteLine('A');
04      Console.WriteLine('가');
05      Console.WriteLine('나');
06  }
```

실행 결과

```
A
가
나
```

Section 03 기본 자료형(6)

■ 문자열

- 문자의 집합
- 예

코드 2-13 문자열 /2장/DefaultData

```
01 static void Main(string[] args)
02 {
03     Console.WriteLine("안녕하세요");
04 }
```

Section 03 기본 자료형(7)

- **기본예제 2-5** 이스케이프 문자(교재 70p) /2장/EscapeCharacter

코드 2-14 이스케이프 문자

```
01 static void Main(string[] args)
02 {
03     Console.WriteLine("한빛\t아카데미");
04     Console.WriteLine("한빛\n아카데미");
05     Console.WriteLine("\"\"\"");
06 }
```

실행 결과

```
한빛    아카데미
한빛
아카데미
"""
```

표 2-6 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 71p) /2장/StringConnection

코드 2-15 문자열 연결 연산자

```
01  static void Main(string[] args)
02  {
03      Console.WriteLine("가나다" + "라마" + "바사아" + "자차카타" + "파하");
04  }
```

실행 결과

```
가나다라마바사아자차카타파하
```

표 2-7 문자열 연결 연산자

연산자	설명
+	문자열 연결 연산자

Section 03 기본 자료형(9)

- **기본예제 2-7** 문자 선택(교재 71p) /2장/StringSelector

코드 2-16 문자 선택

```
01 static void Main(string[] args)
02 {
03     Console.WriteLine("안녕하세요"[0]);
04     Console.WriteLine("안녕하세요"[1]);
05     Console.WriteLine("안녕하세요"[3]);
06 }
```

실행 결과

```
안
녕
세
```

표 2-8 문자 선택 괄호

연산자	설명
문자열[숫자]	문자 선택 괄호

■ 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 2-4

예외발생(디버그 모드)

그림 2-4 예외 발생(디버그 모드)

■ 예외

- 그림 2-5 예외 발생(릴리즈 모드)

그림 2-5 예외 발생(릴리즈 모드)

■ 문자 덧셈 연산

- 문자열은 + 연산자로 연결 가능, 문자는 불가능

코드 2-18 문자 덧셈 /2장/DefaultData

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

Section 03 기본 자료형(6)

■ 불

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

코드 2-19 불 /2장/DefaultData

```
01 static void Main(string[] args)
02 {
03     Console.WriteLine(true);
04     Console.WriteLine(false);
05 }
```

Section 03 기본 자료형(7)

- **기본예제 2-8** 불과 비교 연산자(교재 74p) /2장/BoolBasic

코드 2-20 불과 비교 연산자

```
01  static void Main(string[] args)
02  {
03      Console.WriteLine(52 < 273);
04      Console.WriteLine(52 > 273);
05  }
```

실행 결과

```
True
False
```

표 2-9 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 75p) /2장/LogicalNot

코드 2-21 논리 부정 연산자

```
01  static void Main(string[] args)
02  {
03      Console.WriteLine(!true);
04      Console.WriteLine(!false);
05      Console.WriteLine(!(52 < 273));
06      Console.WriteLine(!(52 > 273));
07  }
```

실행 결과

```
False
True
False
True
```

표 2-10 논리 연산자

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

- 논리 부정 연산자는 숫자의 부호를 반대로 만드는 - 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 2-11 논리합 연산자

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

- 논리곱 연산자(and)

표 2-12 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 78p) /2장/LogicalOperator

코드 2-22 불과 논리 연산자

```
01 static void Main(string[] args)
02 {
03     Console.WriteLine(DateTime.Now.Hour < 3 || 8 < DateTime.Now.Hour);
04     Console.WriteLine(3 < DateTime.Now.Hour && DateTime.Now.Hour < 8);
05 }
```

실행 결과

```
False
True
```

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

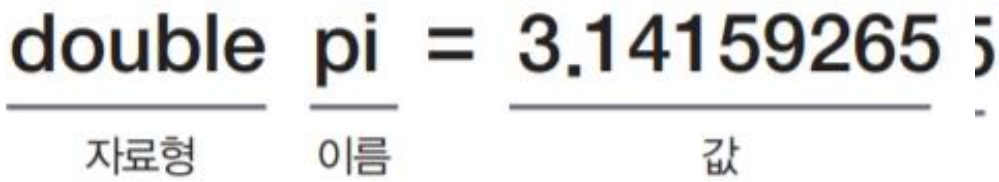

그림 2-9 변수 선언 예

■ 정수 자료형

표 2-13 정수 자료형

키워드	설명
int	4바이트의 정수
long	8바이트의 정수

```
int a = 10
long b = 20
```

그림 2-10 정수 자료형 선언 예

- **기본예제 2-11** 정수 변수 생성(교재 80p) /2장/IntegerVariable

코드 2-23 정수 변수 생성

```
01 static void Main(string[] args)
02 {
03     int a = 273;
04     int b = 52;
05
06     Console.WriteLine(a + b);
07     Console.WriteLine(a - b);
08     Console.WriteLine(a * b);
09     Console.WriteLine(a / b);
10     Console.WriteLine(a % b);
11 }
```

실행 결과

```
325
221
14196
5
13
```

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

```
01 static void Main(string[] args)
02 {
03     int a = 2147483640;
04     int b = 52273;
05
06     Console.WriteLine(a + b);
07 }
```

그림 2-13 int 자료형의 범위

Section 04 변수(4)

- **기본예제 2-12** 오버플로우(교재 82p) /2장/Overflow

코드 2-25 오버플로우

```
01 static void Main(string[] args)
02 {
03     int a = 2000000000;
04     int b = 1000000000;
05     Console.WriteLine(a + b);
06 }
```

실행 결과

```
-1294967296
```

- 오버플로우 문제 해결 방법 : 자료형 변환
- 예

코드 2-26 자료형 변환을 사용한 해결 방법 3가지

```
01 static void Main(string[] args)
02 {
03     int a = 2000000000;
04     int b = 1000000000;
05     Console.WriteLine(a + b);
06 }
```

NOTE(1)

■ unsigned 자료형(부호가 없는 자료형)

- 양수 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 = 112233720368547758O8;

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

■ 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);
}
```

■ 실수 자료형

표 2-14 실수 자료형

키워드	설명
float	4바이트의 실수
double	8바이트의 실수

- **기본예제 2-13** 실수 변수 생성(교재 84p) /2장/RealNumberVariable

코드 2-30 실수 변수 생성

```
01 static void Main(string[] args)
02 {
03     double a = 52.273;
04     double b = 103.32;
05
06     Console.WriteLine(a + b);
07     Console.WriteLine(a - b);
08     Console.WriteLine(a * b);
09     Console.WriteLine(a / b);
10 }
```

실행 결과

```
155.593
-51.047
5400.84636
0.50593302361595
```

Section 04 변수(6)

■ 문자 자료형

표 2-15 문자 자료형

키워드	설명
char	문자

▪ **기본예제 2-14** 문자 변수 생성(교재 85p) /2장/CharacterVariable

코드 2-31 문자 변수 생성

```
01 static void Main(string[] args)
02 {
03     char a = 'a';
04     Console.WriteLine(a);
05 }
```

실행 결과

```
a
```

■ 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));
}
```

■ 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);
}
```

■ 문자열 자료형

표 2-16 문자열 자료형

키워드	설명
string	문자열 자료형

▪ **기본예제 2-15** 문자열 변수 생성(교재 87p) /2장/StringVariable

코드 2-34 문자열 변수 생성

```
01 static void Main(string[] args)
02 {
03     string message = "안녕하세요";
04
05     Console.WriteLine(message + "!");
06     Console.WriteLine(message[0]);
07     Console.WriteLine(message[1]);
08     Console.WriteLine(message[2]);
09 }
```

실행 결과

```
안녕하세요!
안
녕
하
```

■ 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		

■ 불 자료형

표 2-18 불 자료형

키워드	설명
bool	불 자료형

▪ **기본예제 2-16** 불 변수 생성(교재 89p) /2장/BoolVariable

코드 2-36 불 변수 생성

```
01 static void Main(string[] args)
02 {
03     bool one = 10 < 0;
04     bool other = 20 > 100;
05
06     Console.WriteLine(one);
07     Console.WriteLine(other);
08 }
```

실행 결과

```
False
False
```

Section 05 복합 대입 연산자(1)

■ 복합 대입 연산자

- 자료형에 적용하는 기본 연산자와 = 연산자를 함께 사용
  - a+=10 은 a=a+10 을 뜻함
- **기본예제 2-17** 숫자와 관련된 복합 대입 연산자(교재 91p) /2장 /AssignmentOperator

표 2-19 숫자에 적용하는 복합 대입 연산자

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

코드 2-37 숫자와 관련된 복합 대입 연산자

```
01 static void Main(string[] args)
02 {
03     int output = 0;
04     output += 52;
05     output += 273;
06     output += 103;
07
08     Console.WriteLine(output);
09 }
```

실행 결과

```
428
```

Section 05 복합 대입 연산자(2)

■ 복합 대입 연산자

표 2-20 문자열에 적용하는 복합 대입 연산자

연산자	설명
+=	문자열 연결 후 대입 연산자

- **기본예제 2-18** 문자와 관련된 복합 대입 연산자(교재 92p) /2장/StringAssignmentOperator

코드 2-39 문자열과 관련된 복합 대입 연산자

```
01 static void Main(string[] args)
02 {
03     string output = "hello ";
04     output += "world ";
05     output += "!";
06
07     Console.WriteLine(output);
08 }
```

실행 결과

```
hello world !
```

Section 06 증감 연산자(1)

■ 증감 연산자 사용

- 단항 연산자로 변수 앞과 뒤에 ++ 기호와 -- 기호 붙여 만듬

표 2-21 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 93p) /2장/IncrementOperator

코드 2-41 증감 연산자

```
01 static void Main(string[] args)
02 {
03     int number = 10;
04     number++;
05     Console.WriteLine(number);
06     number--;
07     Console.WriteLine(number);
08 }
```

실행 결과

```
11
10
```

Section 06 증감 연산자(2)

■ 증감 연산자 사용

- **기본예제 2-20** 증감 연산자의 전위와 후위 (교재 94p) /2장/IncrementOperatorPosition
- 전위 : 해당 문장을 실행하기 전에 값을 변경
- 후위 : 문장을 실행한 이후에 값을 변경

코드 2-42(1) 증감 연산자의 후위 형태

```
01 static void Main(string[] args)
02 {
03     int number = 10;
04     Console.WriteLine(number);
05     Console.WriteLine(number++);
06     Console.WriteLine(number--);
07     Console.WriteLine(number);
08 }
```

실행 결과

```
10
10
11
10
```

코드 2-42(2) 증감 연산자의 전위 형태

```
01 static void Main(string[] args)
02 {
03     int number = 10;
04     Console.WriteLine(number);
05     Console.WriteLine(++number);
06     Console.WriteLine(--number);
07     Console.WriteLine(number);
08 }
```

실행 결과

```
10
11
10
10
```

■ 자료형 검사 방법

- 방법1 : 마우스 가져다 대기

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

(지역 변수) int number

그림 2-16 마우스를 사용한 자료형 확인

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

표 2-22 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 98p) /2장/TypeCheck

실행 결과

```
System.Int32
System.Int64
System.Single
System.Double
System.Char
System.String
```

- **기본예제 2-22** 직접적인 **GetType( ) 메서드 활용**(교재 93p) /2장/DirectTypeCheck

실행 결과

```
System.Int32
System.Int64
System.Single
System.Double
System.Char
System.String
```

Section 08 var 키워드(1)

■ var 키워드 사용

표 2-23 var 키워드

var 키워드	설명
var	자료형을 자동으로 지정합니다.

코드 2-48 var 키워드 /2장/VarKeyword

```
01 static void Main(string[] args)
02 {
03     var number = 100;
04 }
```

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

코드 2-49 var 키워드의 제약 /2장/VarKeyword

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

Section 08 var 키워드(2)

■ var 키워드 추가 사용 조건 (2가지)

▪ (1) 지역 변수로 선언

- 지역 변수 : 메서드 내부에 선언되어 있는 변수

코드 2-50 지역 변수 /2장/VarKeyword

```
01 class Program
02 {
03     var global = 52;──────────인스턴스 변수(var 키워드 사용 불가)
04
05     static void Main(string[] args)
06     {
07         var local = 273;──────지역 변수(var 키워드 사용 가능)
08     }
09 }
```

▪ (2) 변수를 선언과 동시에 초기화

코드 2-51 var 키워드의 선언과 초기화 동시 수행 /2장/VarKeyword

```
01 static void Main(string[] args)
02 {
03     var number = 20;
04 }
```

■ 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(**대문자 사용**)
- 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
}
```

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);
}
```

■ 입력

표 2-24 입력 메서드

메서드 이름	설명
Console.ReadLine()	사용자로부터 한 줄의 문자열을 입력 받습니다.

- **기본예제 2-23** 문자열 입력과 출력 (교재 105p) /2장/Input

코드 2-55 문자열 입력과 출력

```
01 static void Main(string[] args)
02 {
03     string input = Console.ReadLine();
04     Console.WriteLine("input: " + input);
05 }
```

- **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

```
01 static void Main(string[] args)
02 {
03     // long 자료형을 int 자료형으로 변환합니다.
04     long longNumber = 2147483647L + 2147483647L;
05     int intNumber = longNumber;
06     Console.WriteLine(intNumber);
07 }
```

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.WriteL (
}
```

오류

(지역 변수) long longNumber

CS0266: 암시적으로 'long' 형식을 'int' 형식으로 변환할 수 없습니다. 명시적 변환이 있습니다. 캐스트가 있는지 확인하세요.

잠재적 수정 사항 표시 (Alt+Enter 또는 Ctrl+.)

그림 2-23 자료형 변환 실패

- 예

코드 2-57 강제 자료형 변환 /2장/Casts

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

Section 10 자료형 변환(3)

■ 강제 자료형 변환

- **기본예제 2-24** 강제 자료형 변환 (교재 107p) /2장/ExplicitConversion

코드 2-58 강제 자료형 변환

```
01 static void Main(string[] args)
02 {
03     // long 자료형을 int 자료형으로 변환합니다.
04     long longNumber = 2147483647L + 2147483647L;
05     int intNumber = (int)longNumber;
06     Console.WriteLine(intNumber);
07 }
```

- 강제 자료형 변환 데이터 손실 발생하지 않는 예

코드 2-59 강제 자료형 변환의 데이터 손실 미발생 /2장/Casts

```
01 static void Main(string[] args)
02 {
03     // long 자료형을 int 자료형으로 변환합니다.
04     long longNumber = 52273;
05     int intNumber = (int)longNumber;
06     Console.WriteLine(intNumber);
07 }
```

Section 10 자료형 변환(4)

■ 자동 자료형 변경

- **기본예제 2-25** 숫자 손상(교재 108p) /2장/NumberLost

표 2-25 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 109p) /2장/ImplicitConversion

Section 10 자료형 변환(5)

■ 다른 자료형을 숫자로 변환

표 2-26 문자열을 숫자로 변환하는 메서드

메서드 이름	설명
int.Parse()	다른 자료형을 int 자료형으로 변경합니다.
long.Parse()	다른 자료형을 long 자료형으로 변경합니다.
float.Parse()	다른 자료형을 float 자료형으로 변경합니다.
double.Parse()	다른 자료형을 double 자료형으로 변경합니다.

- **기본예제 2-27** 문자열을 숫자로 변환(교재 111p) /2장/StringTo

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"));
```

Section 10 자료형 변환(6)

■ 다른 자료형을 문자열로 변환

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

표 2-27 문자열로 변환하는 메서드

메서드	설명
ToString()	문자열로 변환합니다.

- **기본예제 2-28** 문기본 자료형을 문자열로 변환(교재 112p) /2장/ToStringBasic

Section 10 자료형 변환(7)

- **기본예제 2-29** 소숫점 제거(교재 113p) /2장/DoubleToString

코드 2-66 소수점 제거

```
01 static void Main(string[] args)
02 {
03     double number = 52.273103;
04     Console.WriteLine(number.ToString("0.0"));
05     Console.WriteLine(number.ToString("0.00"));
06     Console.WriteLine(number.ToString("0.000"));
07     Console.WriteLine(number.ToString("0.0000"));
08 }
```

숫자 3 대신에 6입력 후 출력 (반올림 확인!)

- **기본예제 2-30** 숫자와 문자열 덧셈(교재 114p) /2장/StringPlusNumber

코드 2-67 숫자와 문자열 덧셈

```
01 static void Main(string[] args)
02 {
03     Console.WriteLine(52 + 273);
04     Console.WriteLine("52" + 273);
05     Console.WriteLine(52 + "273");
06     Console.WriteLine("52" + "273");
07 }
```

■ 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);
```

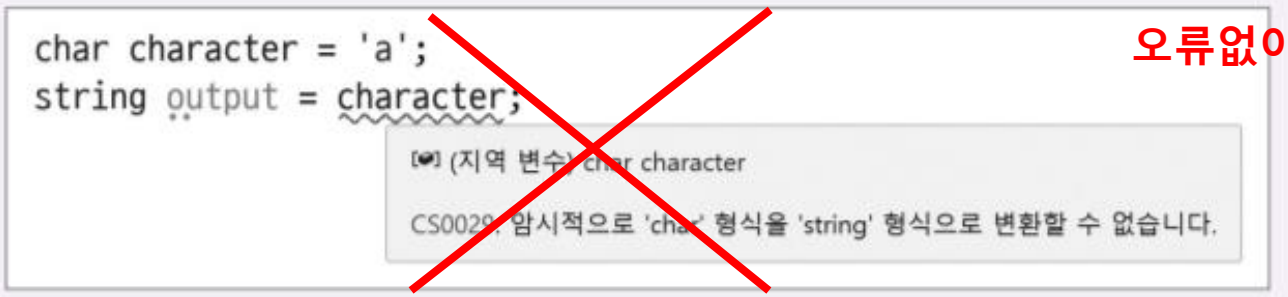

그림 2-24 문자 문자열 변환 오류

Section 10 자료형 변환(8)

■ 다른 자료형을 불로 변환

표 2-28 문자열을 불로 변환하는 메서드

메서드	설명
bool.Parse()	문자열을 불 자료형으로 변환합니다.

- **기본예제 2-31** 문자열을 불로 전환(교재 116p) /2장/StringToBool

코드 2-69 문자열을 불로 변환

```
01 static void Main(string[] args)
02 {
03     Console.WriteLine(bool.Parse("True"));
04     Console.WriteLine(bool.Parse("true"));
05     Console.WriteLine(bool.Parse("False"));
06     Console.WriteLine(bool.Parse("false"));
07 }
```

실행 결과

```
True
True
False
False
```

■ 음수밖에 없는 숫자

- 예

코드 2-70 int 자료형 최솟값의 음수

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

```
static void Main(string[] args)
{
    Console.WriteLine(-(-2147483648));
}
```

readonly struct System.Int32
부호 있는 32비트 정수를 나타냅니다.

CS0220: checked 모드에서 컴파일하면 작업이 오버플로됩니다.

그림 2-25 숫자를 직접 입력하면 오류 발생

NOTE

■ Read()와 ReadLine()의 차이

- int Console.Read();
  - 한개의 문자만 리턴된다. 입력받은 값이 'abcde' 이던 'aaaaa' 이던 제일 앞의 'a'만 리턴
  - 정수형 아스키코드로 리턴
  - Convert.ToChar() 메소드를 사용하여 문자로 변환

- string Console.ReadLine();
  - 문자열로 리턴
  - 정수를 입력했을때 int형 변수에 저장 할 수 없는 문제가 발생

    → int.Parse() 나 Convert 메소드로 해결

NOTE

■ 소수점 이하 자리수 출력

- 고정 소수점 표기법 사용하기

```
double num = 3.141592653589793238 46;

Console.WriteLine(num.ToString("0.00"));
Console.WriteLine(num.ToString("F2"));      // 2자리까지 출력
Console.WriteLine(num.ToString("F5"));      // 5자리까지 출력
```

- 지수 표기법 사용하기

```
double num = 123456789.123456789;
Console.WriteLine(num.ToString("E2"));   // 지수 형태로 2자리까지 출력
```

실습하기

(1) 단위 변환 프로그램

- 콘솔 창에서 1개의 정수 값을 입력 받아 Inch 단위를 cm 단위로 계산 후 출력
- 1 inch = 2.54cm

(2) 사칙연산 프로그램

- 콘솔 창에서 2개의 정수 값을 입력 받아 사칙 연산 수행 후 결과 값 출력
- 나눗셈 연산은 소수점 2자리까지 출력
- 실행 결과 예)

1. 첫번째 정수 입력 : 3

2. 두번째 정수 입력 : 2

3 + 2 = 5

3 - 2 = 1

3 * 2 = 6

3 / 2 = 1.5

실습하기

(3) 원의 공식

- **콘솔 창에서 원의 반지름을 입력 받아 ①원의 둘레와 ②넓이를 구하는 코드 작성**
  - 둘레 = 2*PI*반지름
  - 넓이 = PI * 반지름*반지름
  - PI=3.14159 (상수로 정의함)
  - 둘레와 넓이 모두 소수점 이하 3자리까지 반올림하여 구하기!

실습하기

(4) 문자 계산 프로그램

- 콘솔 창에서 한 문자를 입력 받아 정수 5를 더한 후 값을 출력하되,
  아래 실행 결과처럼 출력!
- Console.Read() 사용, Convert.ToChar()

- 실행 결과 예)

  문자 입력 : A

  1. 연산 수행 전

  - 아스키코드 값 : 65

  - 변환된 문자 출력 : A

  2. 연산 수행 후

  - 아스키코드 값 : 70

  - 변환된 문자 출력 : F