4. 데이터를 가공하는 연산자

4.1 C#에서 제공하는 연산자

분류	연산자
산술 연산자	+, −, *, /, %
증가/감소 연산자	++, −−
관계 연산자	〈, 〉, ==, !=, 〈=, 〉=
조건 연산자	?:
Null 조건부 연산자	?., ?[ ]
논리 연산자	&&, ||, !
비트 연산자	〈〈, 〉〉, &, |, ^, ~
할당 연산자	=, +=, −=, *=, /=, %=, &=, |=, ^=, 〈〈=, 〉〉=
Null 병합 연산자	??

4.2 산술 연산자

- 수치 형식의 데이터를 다루는 연산자 : 정수, 부동 소수점, decimal 형식

- 두 개의 피연산자 필요한 이항 연산자 : 피연산자(을) **연산자** 피연산자(으로)

연산자	설명	지원 형식
+	양쪽 피연산자를 더합니다.	모든 수치 데이터 형식
−	왼쪽 피연산자에서 오른쪽 피연산자를 차감합니다.	모든 수치 데이터 형식
*	양쪽 피연산자를 곱합니다.	모든 수치 데이터 형식
/	왼쪽 연산자를 오른쪽 피연산자로 나눈 몫을 구합니다.	모든 수치 데이터 형식
%	왼쪽 연산자를 오른쪽 피연산자로 나눈 후의 나머지를 구합니다.	모든 수치 데이터 형식

4.3 증가 연산자와 감소 연산자

- 전위 증감 연산자, 후위 증감 연산자

4.5 문자열 결합 연산자

+ : 산술 연산자, 문자열 결합 연산자로 사용될 수 있음.

4.5 관계 연산자

- 두 피연산자 사이의 관계 확인 : 반환 결과 논리 형식(bool형)

연산자	설명	지원 형식
〈	왼쪽 피연산자가 오른쪽 피연산자보다 작으면 참, 아니면 거짓	모든 수치 형식과 열거 형식
〉	왼쪽 피연산자가 오른쪽 피연산자보다 크면 참, 아니면 거짓	모든 수치 형식과 열거 형식
〈=	왼쪽 피연산자가 오른쪽 피연산자보다 작거나 같으면 참, 아니면 거짓	모든 수치 형식과 열거 형식
〉=	왼쪽 피연산자가 오른쪽 피연산자보다 크거나 같으면 참, 아니면 거짓	모든 수치 형식과 열거 형식
==	왼쪽 피연산자가 오른쪽 피연산자와 같으면 참, 아니면 거짓	모든 데이터 형식에 대해 사용 가능
!=	왼쪽 피연산자가 오른쪽 피연산자와 다르면 참, 아니면 거짓	모든 데이터 형식에 대해 사용 가능. string와 object 형식에 대해서도 사용이 가능합니다.

4.6 논리 연산자

- 참과 거짓을 피연산자로 사용

A	B	A || B
참	참	참
참	거짓	참
거짓	거짓	거짓
거짓	참	참

A	!A
참	거짓
거짓	참

4.7 조건 연산자

- 조건 연산자의 형식

```
조건식 ? 참일때_값 : 거짓일때_값
```

```
int a = 30;
string result = a == 30 ? "삼십" : "삼십아님"; // result는 "삼십"
```

조건식 | 참일_때의_값 | 거짓일_때의_값

4.8 널 조건부 연산자

- 객체의 멤버에 접근하기 전에 널인지 검사해줌

- 원래는 ==연산자를 이용해 코드를 짜야함.

== 연산자를 이용한 코드

```
class Foo
{
    public int member;
}

Foo foo = null;

int? bar;
if (foo == null)
    bar = null;
else
    bar = foo.member;
```

?. 연산자를 이용한 코드

```
class Foo
{
    public int member;
}

Foo foo = null;

int? bar;
bar = foo?.member;
```

foo 객체가 null이 아니면 member 필드에 접근하게 해줌

- 널 조건부 연산자의 형식

```

객체 **?.** 반환 멤버

```

```

객체 **? [**배열(컬렉션)의 인덱스**]**

```

: 객체가 null이 아니면 접근하게 해줌

4.9 비트 연산자

데이터 형식의 크기는 주로 바이트 단위 (1바이트 = 8비트) 를 사용하는데 더 작은 단위로 데이터를 가공해야 할 경우.

- 비트 연산자의 종류

연산자	이름
《	왼쪽 시프트 연산자
》	오른쪽 시프트 연산자
&	논리곱(AND) 연산자
|	논리합(OR) 연산자
^	배타적 논리합(XOR) 연산자
~	보수(NOT) 연산자

4.9.1 시프트 연산자

- **시프트 연산자** : 비트를 왼쪽이나 오른쪽으로 이동

- 왼쪽 시프트 연산

원본데이터 << 옮길 비트의 수

- 오른쪽 시프트 연산

원본데이터 >> 옮길 비트의 수

4.9.2 비트 논리 연산자

- 각 비트에 대해 수행하는 논리 연산

: Bool 형식과 정수 계열 형식, 각 비트에 대해 1은 True, 0은 False

- 보수 연산자 : ~

```
int a = 255;
int result = ~a; //  result는 -256
```

4.10 할당 연산자

- 변수 또는 상수에 피연산자의 데이터를 할당하는 기능

: 왼쪽 피연산자와 오른쪽 피연산자를 한 차례 가공한 후 다시 왼쪽 피연산자에 할당.

- 할당 연산자의 종류

=	할당 연산자	오른쪽 피연산자를 왼쪽 피연산자에게 할당합니다.
+=	덧셈 할당 연산자	a += b;는 a = a + b;와 같습니다.
-=	뺄셈 할당 연산자	a -= b;는 a = a - b;와 같습니다.
*=	곱셈 할당 연산자	a *= b;는 a = a * b; 와 같습니다.
/=	나눗셈 할당 연산자	a /= b;는 a = a / b;와 같습니다.
%=	나머지 할당 연산자	a %= b;는 a = a % b;와 같습니다.
&=	논리곱 할당 연산자	a &= b;는 a = a & b;와 같습니다.
|=	논리합 할당 연산자	a |= b;는 a = a | b;와 같습니다.
^=	배타적 논리합 할당 연산자	a ^= b;는 a = a ^ b;와 같습니다.
〈〈=	왼쪽 시프트 할당 연산자	a 〈〈= b;는 a = a 〈〈 b;와 같습니다.
〉〉=	오른쪽 시프트 할당 연산자	a 〉〉= b;는 a = a 〉〉 b;와 같습니다.

4.11 Null 병합 연산자

- ?? 연산자 : 두 개의 피연산자를 받아들이고 왼쪽 피연산자가 null인지 평가한 뒤 null이 아니면 왼쪽 피연산자를, null이면 오른쪽 피연산자를 반환.

- null 병합 연산자의 형식

```
왼쪽_피연산자 ?? 오른쪽_피연산자
```

4.12 연산자의 우선순위

우선순위	종류	연산자
1	증가/감소 연산자 및 Null 조건부 연산자	후위 ++/-- 연산자, ?., ?[ ]
2	증가/감소 연산자	전위 ++/-- 연산자
3	산술 연산자	* / %
4	산술 연산자	+ -
5	시프트 연산자	<< >>
6	관계 연산자	< > <= >= is as (is와 as 연산자는 뒷부분에서 설명합니다)
7	관계 연산자	== !=
8	비트 논리 연산자	&
9	비트 논리 연산자	^
10	비트 논리 연산자	|
11	논리 연산자	&&
12	논리 연산자	||
13	Null 병합 연산자	??
14	조건 연산자	?:
15	할당 연산자	= *= /= %= += -= <<= >>= &= ^= |=