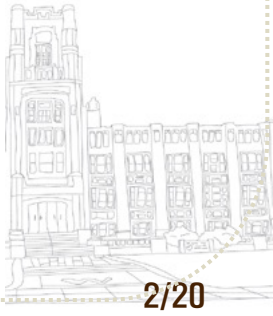
데이터를 가공하는 연산자

이것이 C#이다

Contents

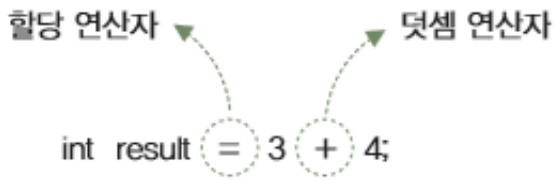


4.1 C#에서 제공하는 연산자

❖ 연산자들 중 대부분은 각각 특정 형식에 대해 사용 가능

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

❖ 연산자의 사용 예



4.2 산술 연산자

❖ 수치 형식의 데이터를 다루는 연산자

- 정수 형식과 부동 소수점 형식, decimal 형식

❖ 산술 연산자의 종류와 기능

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

❖ 두 개의 피연산자 필요 → 이항 연산자

```
왼쪽_피연산자  연산자  오른쪽_피연산자
```

❖ 데모 예제 – ArithmaticOperators

4.3 증가 연산자와 감소 연산자

- 증가 연산자 ++
  - 피연산자의 값 1 증가
- 감소 연산자 --
  - 피연산자의 값 1 감소

```
int a = 10;
a = a + 1;  // a는 11
a = a - 1;  // a는 10
```

- 연산자의 위치에 따라
  - 전위 증감 연산자
    - 변수의 값을 변경한 후 해당 구문 실행
  - 후위 증감 연산자
    - 해당 구문을 실행한 후 변수의 값 변경

```
int a = 10;
Console.WriteLine( a-- ); // 9가 아닌, 10을 출력. 출력 후에 a는 9로 감소
Console.WriteLine( --a ); // 8을 출력
```

- 데모 예제 - IncDecOperator

4.4 문자열 결합 연산자

❖ + 연산자

- int result = 123 + 456; ← **산술 연산자**
- string result = "123" + "456"; ← **문자열 결합 연산자**

❖ 데모 예제 – StringConcatenate

4.5 관계 연산자

❖ 두 피연산자 사이의 관계 확인

- 결과는 논리 형식(bool), 참 또는 거짓

❖ 관계 연산자의 종류

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.

❖ 데모 예제 - RelationalOperator

4.6 논리 연산자

- 참과 거짓을 피연산자로 사용
- C#의 논리 연산자

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

A	!A
참	거짓
거짓	참

- 분기문과 반복문에 자주 사용
- 데모 예제 - LogicalOperator

4.7 조건 연산자

❖ 조건 연산자의 형식

- 조건식 ? 참일_때의_값 : 거짓일_때의_값
- 조건식: 결과가 논리 값

❖ 사용 예

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

조건식 | 참일_때의_값 | 거짓일_때의_값

❖ 데모 예제 - ConditionalOperator

4.8 널 조건부 연산자

- C# 6.0에서 도입
- 객체의 멤버에 접근하기전에 널인지 검사
  - 객체 ?. 반환멤버
  - 객체 ?[*배열(컬렉션)의 인덱스*]

== 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

- 데모 예제 – NullConditionalOperator

4.9 비트 연산자

- 데이터 형식의 크기는 주로 바이트 단위 사용
- 더 작은 단위로 데이터를 가공해야 할 경우
- 1 바이트 = 8비트
- 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

- 비트를 왼쪽이나 오른쪽으로 이동
- 왼쪽 시프트 연산

- 데모 예제- ShiftOperator

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
```

❖ 데모 예제 - BitwiseOperator

4.10 할당 연산자

- 변수 또는 상수에 피연산자 데이터를 할당하는 기능
- 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.

- 데모 예제 - AssignmentOperator

4.11 Null 병합 연산자 ??

- 필요한 변수/객체의 null 검사 를 간결하게 만들어주는 역할
- 형식: OP1 ?? OP2
  - OP1이 Null인지 여부에 따라.

```
int? a = null;
Console.WriteLine($"{a ?? 0}");

a = 99;
Console.WriteLine($"{a ?? 0}");
```

a는 null이므로 0이 출력됩니다.

a는 null이므로 99이 출력됩니다.

- 데모 예제 - NullCoalescing

4.12 연산자 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	= *= /= %= += -= <<= >>= &= ^= |=

Thank You !

이것이 C#이다