4) 연산자와 문자열 처리

매 강의 강의자료 시작에 PDF파일을 올려두었어요!

▼ PDF 파일

[수업 목표]

- 산술, 관계, 논리 연산자의 사용법을 이해한다.
- 복합 대입 연산자와 증감 연산자를 활용하는 방법을 이해한다.
- 문자열 처리 기능과 문자열 메서드의 사용법을 이해한다.

[목차]

01. 산술, 관계, 논리 연산자 사용법

02. 비트 연산자

03. 복합 대입 연산자와 증감 연산자 활용

04. 연산자 우선순위

05. 문자열 처리 기능 및 메서드

모든 토글을 열고 닫는 단축키
Windows : `Ctrl` + `alt` + `t`
Mac : `⌘` + `⌥` + `t`

---

01. 산술, 관계, 논리 연산자 사용법

✔️ C# 에서는 다양한 연산자를 제공해줍니다

▼ 1) 산술연산자

산술 연산자는 숫자를 대상으로 사용됩니다.

연산자	설명
+	덧셈
-	뺄셈
*	곱셈
/	나눗셈
%	나머지

▼ 2) 관계연산자

관계 연산자는 두 값을 비교하여 참(True) 또는 거짓(False) 값을 반환합니다.

연산자	설명
==	같음
!=	다름
>	큼
<	작음
>=	크거나 같음
<=	작거나 같음

▼ 3) 논리연산자

논리 연산자는 참(True) 또는 거짓(False) 값을 대상으로 사용됩니다.

연산자	설명
&&	논리곱(AND)
||	논리합(OR)
!	논리부정(NOT)

02. 비트 연산자

✔️ 비트 단위로 데이터를 조작하고, 이진수 연산을 해보자

▼ 1) 비트연산자

비트 연산자는 데이터의 비트(bit) 단위로 연산을 수행하는 연산자입니다.

연산자	설명
& (AND)	두 비트 값이 모두 1일 때 1을 반환
| (OR)	두 비트 값 중 하나라도 1일 때 1을 반환
^ (XOR)	두 비트 값이 서로 다를 때 1을 반환
~ (NOT)	비트 값의 보수(complement)를 반환
<< (왼쪽 시프트)	비트를 왼쪽으로 이동
>> (오른쪽 시프트)	비트를 오른쪽으로 이동

```
int a = 0b1100; // 12 (2진수)
int b = 0b1010; // 10 (2진수)

int and = a & b; // 0b1000 (8)
int or = a | b; // 0b1110 (14)
int xor = a ^ b; // 0b0110 (6)

int c = 0b1011; // 11 (2진수)
int leftShift = c << 2; // 0b101100 (44)
int rightShift = c >> 1; // 0b0101 (5)

int d = 0b1100; // 12 (2진수)
int bit3 = (d >> 2) & 0b1; // 0 (3번째 비트)
d |= 0b1000; // 0b1100 | 0b1000 = 0b1100 (12)
```

03. 복합 대입 연산자와 증감 연산자 활용

✔️ 변수를 더 편리하게 조작하고 값을 증감시킬 수 있어요

▼ 1) 복합 대입 연산자

C#에서는 변수에 값을 할당하는 대입 연산자(=) 외에도, 다양한 복합 대입 연산자를 제공합니다. 복합 대입 연산자는 변수에 연산을 수행한 결과를 저장하는 연산자입니다.

연산자	예시	설명
+=	x += y;	x = x + y;
-=	x -= y;	x = x - y;
*=	x *= y;	x = x * y;
/=	x /= y;	x = x / y;
%=	x %= y;	x = x % y;

▼ 2) 증감 연산자

증감 연산자는 변수의 값을 1 증가시키거나 감소시키는 연산자입니다.

연산자	설명
++	1 증가
--	1 감소

04. 연산자 우선순위

✔️ 연산자 우선순위를 생각하면서 연산해주세요~

▼ 1) 연산자 우선순위란?

- 연산자 우선순위는 수식 내에서 연산자가 수행되는 순서를 결정합니다.
- 연산자 우선순위에 따라 연산의 결과가 달라질 수 있으므로 중요한 개념입니다.

▼ 2) C#의 주요 연산자 우선순위

연산자들은 다양한 우선순위를 가지고 있으며, 높은 우선순위의 연산자가 먼저 수행됩니다.

아래는 C#에서 주로 사용되는 연산자들의 우선순위를 나열한 것입니다.

1. 괄호 (): 괄호로 감싸진 부분은 가장 높은 우선순위로 먼저 계산됩니다.
2. 단항 연산자: 단항 연산자들(++, --, +, -, ! 등)은 괄호 다음으로 높은 우선순위를 가집니다.
3. 산술 연산자: 산술 연산자들(*, /, %, +, -)은 단항 연산자보다 우선순위가 낮습니다.
4. 시프트 연산자: 시프트 연산자(<<, >>)는 산술 연산자보다 우선순위가 낮습니다.

5. 관계 연산자: 관계 연산자들(<, >, <=, >=, ==, !=)는 시프트 연산자보다 우선순위가 낮습니다.
6. 논리 연산자: 논리 연산자들(&&, ||)는 관계 연산자보다 우선순위가 낮습니다.
7. 할당 연산자: 할당 연산자들(=, +=, -=, *=, /= 등)는 논리 연산자보다 우선순위가 낮습니다.

05. 문자열 처리 기능 및 메서드

✔️ C#에서는 문자열을 처리하는 다양한 기능과 메서드를 제공합니다.

▼ 1) 문자열 생성

```
string str1 = "Hello, World!"; // 리터럴 문자열 사용
string str2 = new string('H', 5); // 문자 'H'를 5개로 구성된 문자열 생성
```

▼ 2) 연결

```
string str1 = "Hello";
string str2 = "World";
string str3 = str1 + " " + str2;
```

이 코드는 str1 문자열과 str2 문자열을 공백으로 구분하여 연결한 새로운 문자열 str3을 생성합니다.

▼ 3) 분할

```
string str = "Hello, World!";
string[] words = str.Split(',');
```

이 코드는 str 문자열을 쉼표(,)로 구분하여 분할한 문자열 배열 words를 생성합니다.

▼ 4) 검색

```
string str = "Hello, World!";
int index = str.IndexOf("World");
```

이 코드는 str 문자열에서 "World" 문자열의 첫 번째 인덱스를 찾아 index 변수에 저장합니다.

▼ 5) 대체

```
string str = "Hello, World!";
string newStr = str.Replace("World", "Universe");
```

이 코드는 str 문자열에서 "World" 문자열을 "Universe" 문자열로 대체한 새로운 문자열 newStr을 생성합니다.

▼ 6) 변환

문자열을 숫자로 변환

```
string str = "123";
int num = int.Parse(str);
```

이 코드는 문자열 str을 정수형 숫자로 변환한 후, num 변수에 저장합니다.

숫자를 문자열로 변환

```
int num = 123;
string str = num.ToString();
```

이 코드는 정수형 숫자 num을 문자열로 변환한 후, str 변수에 저장합니다.

▼ 7) 비교

문자열 비교

C#에서 문자열을 비교하는 방법은 다음과 같습니다.

문자열 값 비교

```
string str1 = "Hello";
string str2 = "World";
bool isEqual = str1 == str2;
```

이 코드는 str1 문자열과 str2 문자열을 비교한 후, isEqual 변수에 그 결과를 저장합니다.

문자열 대소 비교

```
string str1 = "Apple";
string str2 = "Banana";
int compare = string.Compare(str1, str2);
```

이 코드는 str1 문자열과 str2 문자열을 대소 비교한 후, compare 변수에 그 결과를 저장합니다. compare 변수는 0보다 작으면 str1이 str2보다 작고, 0이면 str1과 str2가 같으며, 0보다 크면 str1이 str2보다 큽니다.

▼ 8) 포멧팅

문자열 포맷팅

C#에서 문자열 포맷팅을 하는 방법은 다음과 같습니다.

문자열 형식화

```
string name = "John";
int age = 30;
string message = string.Format("My name is {0} and I'm {1} years old.", name, ag
e);
```

이 코드는 문자열 형식 문자열을 사용하여 name 변수와 age 변수의 값을 문자열 message에 삽입합니다.

문자열 보간

```
string name = "John";
int age = 30;
string message = $"My name is {name} and I'm {age} years old.";
```

이 코드는 문자열 보간 기능을 사용하여 name 변수와 age 변수의 값을 문자열 message에 삽입합니다.

Tip

- 연산자의 우선순위는 연산자가 수행되는 순서를 결정합니다. 연산자 우선순위는 수학에서 사용하는 연산자 우선순위와 유사하며, 괄호를 사용하여 우선순위를 변경할 수 있습니다.
- 연산자를 사용할 때에는 연산자 우선순위를 항상 고려하여 코드를 작성하는 것이 중요합니다.

이전 강의

📕 3) 변수와 자료형

다음 강의

📕 1) 조건문과 반복문