

# 컴퓨터공학 All in One

---

C/C++ 문법, 자료구조 및 심화 프로젝트 (나동빈)  
제 4강 - 연산자

# 학습 목표

## 연산자

- 1) C언어에서 사용되는 기본적인 연산자 문법을 공부합니다.
- 2) 연산자의 우선순위를 이해하여 다양한 연산을 수행합니다.

# 연산자

## 연산자와 피연산자

- 1) 연산자(Operator)란 연산을 수행하는 기호를 의미합니다.
- 2) 피연산자(Operand)란 연산에 포함되는 변수나 상수를 의미합니다.
- 3)  $A + B$ 에서 A와 B는 피연산자에 해당하며 +는 연산자에 해당합니다.

피연산자	연산자	피연산자
A	+	B

# 연산자

## 연산자의 종류

1) C언어에는 다양한 연산자가 존재합니다.

대입 연산자	=
산술 연산자	+, -, *, /, %
관계 연산자	==, !=, >, <, >=, <=
논리 연산자	!, &&,
증감 연산자	++, --
삼항 연산자	? :
비트 연산자	!, ~, &, ^, >>, <<

# 연산자

## 대입 연산자

- 1) '=' 등호(Equals Sign)를 이용해서 우변항을 좌변항에 넣을 수 있습니다.
- 2) 자료형에 부합하는 값을 좌변항에 있는 변수에 넣습니다.

# 연산자

## 사칙연산

- 1) C언어에서는 기본적인 사칙 연산을 사용할 수 있습니다.
- 2) 나머지를 구하기 위해 모듈러(Modular) 연산을 사용합니다.

+	더하기
-	빼기
*	곱하기
/	나누기
%	나머지

# 연산자

## 사칙연산

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main(void) {
    int a, b;
    scanf("%d %d", &a, &b);
    printf("%d + %d = %d\n", a, b, a + b);
    printf("%d - %d = %d\n", a, b, a - b);
    printf("%d * %d = %d\n", a, b, a * b);
    printf("%d / %d = %d\n", a, b, a / b);
    printf("%d %% %d = %d\n", a, b, a % b);
    system("pause");
}
```

# 연산자

## 이스케이프 시퀀스(Escape Sequence)

1) C언어에서 특정한 표현을 출력하기 위해 사용하는 문법입니다.

<code>\n</code>	줄 바꾸기
<code>\t</code>	수평 탭 넣기
<code>\\</code>	백슬래시 넣기
<code>\"</code>	큰 따옴표 넣기
<code>\b</code>	백 스페이스 넣기



# 연산자

## 이스케이프 시퀀스(Escape Sequence)

```
#include <stdio.h>

int main(void) {
    printf("\tA\tB\tC\tD\t\n");
    printf("\tA\tB\tC\tD\t\n");
    printf("\tA\tB\tC\tD\t\n");
    system("pause");
}
```

# 연산자

## 관계 연산자

>	크다
<	작다
==	같다
!=	다르다
>=	크거나 같다
<=	작거나 같다

# 연산자

## 관계 연산자

- 1) 컴퓨터는 0을 거짓(False)로 받아들이며, 그 외의 숫자를 참(True)으로 받아들입니다.
- 2) 컴퓨터가 거짓(False)을 출력할 때는 0을, 참(True)을 출력할 때는 1을 출력합니다.

# 연산자

## 관계 연산자

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main(void) {
    int a, b;
    scanf("%d %d", &a, &b);
    printf("%d\n", a > b);
    system("pause");
}
```

# 연산자

## 논리 연산자

!	부정
&&	그리고
	또는

# 연산자

## 논리 연산자

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main(void) {
    int a, b, c;
    scanf("%d %d %d", &a, &b, &c);
    printf("%d\n", !a);
    printf("%d\n", a && b);
    printf("%d\n", (a > b) && (b > c));
    system("pause");
    return 0;
}
```

# 연산자

## 증감 연산자

++(변수)

i의 값을 1 증가시킨 후에 증가된 값을 반환

(변수)++

i의 값을 1 증가시킨 후에 증가되기 전의 값을 반환

--(변수)

i의 값을 1 감소시킨 후에 감소된 값을 반환

(변수)--

i의 값을 1 감소시킨 후에 감소되기 전의 값을 반환

# 연산자

## 삼항 연산자

- 1) 세 개의 항을 이용해서 조건문을 수행할 수 있는 연산자입니다.



# 연산자

## 삼항 연산자

```
#include <stdio.h>

int main(void) {
    int a = 7, b = 7;
    printf("%d\n", (a == b) ? 100 : -100);
    system("pause");
    return 0;
}
```

# 연산자

## 비트 연산자

1) 비트 단위의 연산을 수행할 수 있습니다.

~

부정:  $\sim(11000011)_2 = (00111100)_2$

&

그리고:  $(00001101)_2 \& (00000011)_2 = (00000001)_2$

|

또는:  $(11001100)_2 | (00110000)_2 = (11111100)_2$

^

배타적:  $(11001111)_2 ^ (00000011)_2 = (11001100)_2$

<<

왼쪽 시프트:  $(00001111)_2 \ll 3 = (01111000)_2$

# 연산자

## 비트 연산자

- 1) 시프트(Shift) 연산자는 2의 배수를 처리하고자 할 때 효과적입니다.
- 2) 9를 표현하면 다음과 같습니다.

0	0	0	0	1	0	0	1
---	---	---	---	---	---	---	---

# 연산자

## 비트 연산자

- 1) 왼쪽 시프트(Shift)를 수행하면 9에서 18로 2배가 증가합니다.

0	0	0	1	0	0	1	0
---	---	---	---	---	---	---	---

# 연산자

## 비트 연산자

- 1) 오른쪽 시프트(Shift)를 수행하면 2로 나눈 값이 반환됩니다.

# 연산자

## 연산자 우선순위

- 1) C언어의 연산자 우선순위는 기본적으로 수학에서의 우선순위와 흡사합니다.

# 연산자

## 연산자 우선순위

우선순위	연산자
1	++, --
2	!, ~
3	*, /, %
4	+, -
5	<<, >>
6	<, <=, >, >=
7	==, !=
8	비트, 논리, 삼항 연산자
9	삼항 연산자

# 배운 내용 정리하기

## 연산자

대입 연산자	=
산술 연산자	+, -, *, /, %
관계 연산자	==, !=, >, <, >=, <=
논리 연산자	!, &&,
증감 연산자	++, --
삼항 연산자	? :
비트 연산자	!, ~, &, ^, >>, <<