3장 연산자

한림대학교 소프트웨어융합대학 양은샘.

3장 연산자

- 안녕하세요? 여러분!

- 이번 장에서는 C의 연산자를 학습하도록 하겠습니다.
- 기본적인 연산자는 대부분의 프로그래밍 언어에서 사용하는 연산자와 비슷합니다.

- C 프로그램에서 자주 사용하는 연산자와
- 다른 프로그래밍 언어에서 사용하지 않았던 연산자를 위주로 학습합니다.

- 지난 시간에 학습한 내용을 리뷰한 후 학습을 시작하도록 하겠습니다.

지난 시간 Review

- 자료형
- 변수(variable)
- 상수(constant)
- 출력 함수 printf()
- 입력 함수 scanf(), scanf_s()
- 식에서의 형변환

학습 목차

3.1 연산자의 종류와 우선순위
3.2 형변환 연산자
3.3 단항 연산자
3.4 산술 연산자
3.5 이동(Shift) 연산자
3.6 비교 연산자
3.7 비트 연산자
3.8 논리 연산자
3.9 조건 연산자
3.10 대입 연산자
3.11 나열 연산자
3.12 연산에 사용되는 함수들
3.13 연산자 응용
□ 개념 확인 학습
□ 적용 확인 학습
□ 응용 프로그래밍

학습 목표

- C 프로그램에서 사용하는 연산자의 종류 및 우선순위를 안다.
- 연산자를 사용하여 원하는 조건을 완성할 수 있다.
- 같은 조건을 서로 다른 연산자를 이용하여 구현 할 수 있다.

- 개념 확인 학습으로 배운 내용을 정리한다.
- 적용 확인 학습으로 개념 습득 여부를 확인한다.
- 응용 프로그래밍으로 문제해결력을 키운다.

연산자의 종류와 우선순위

[표 3.1] 연산자의 종류 및 우선순위

우선순위	구 분		연산자
고	cast, 구조체, 공용체		( ) [ ] . –>
	단항 연산자		! ~ – ++ –– (자료형) * &
	이항 연산자	산술연산	* / %
			+ –
		이 동(Shift)	《 》
		비 교	〈 〈= 〉 〉=
			== !=
		비트 AND	&
		비트 XOR	^
		비트 OR	
		논리 AND	&&
저		논리 OR	
	조건 연산자		? :
	대입 연산자		= += *= /= %= 〉〉= 〈〈= &= ^= |=
	나열 연산자		,

[예제 3.1] 연산에서의 연산자 우선순위

```
#include <stdio.h>
int main()
{
    int result, a=8, b=6, c=4, d=2;

    result = a + b * c - d;
    printf("%d + %d * %d - %d = %d₩n", a, b, c, d, result);

    result = a + b * (c - d);
    printf("%d + %d * (%d - %d) = %d₩n", a, b, c, d, result);
    return 0;
}
```

```
8 + 6 * 4 - 2 = 30
8 + 6 * (4 - 2) = 20
```

형변환 연산자

- 자료형의 우선순위
  - char < short < int < long < unsigned < float < double 순으로 높다.

cast 연산자의 형식

```
(데이터형) 수식;
(데이터형) 변수;
```

cast 연산자 예

```
int a = (int) 10.5 + (int) 3.14;
int b = 10;
double c = (double)b / a
```

단항 연산자

[표 3.2] 단항 연산자의 표현식과 의미

연산자	기능	사용 예	의미
++	1 증가(increment)	a++; ++a;	a = a + 1;
--	1 감소(decrement)	a--; --a;	a = a - 1;
-	부호반전(reverse sign)	b = -a;	

[표 3.3] 대입 연산자와 증감 연산자

연산식	의미	사용 예 (a=3 경우)	
		a	b
b = ++a;	a=a+1; b=a;	4	4
b = a++;	b=a; a=a+1;	4	3
b = --a;	a=a-1; b=a;	2	2
b = a--;	b=a; a=a-1;	2	3

[예제 3.2] 대입 연산자와 증감 연산자

```
#include <stdio.h>                                    a=8, b=9, c=4, d=6
int main()
{
    int a, b, c = 3, d = 5;

    a = c++ + d;
    b = c + d++;
    printf("a=%d, b=%d, c=%d, d=%d₩n", a, b, c, d);
    return 0;
}
```

이동(Shift) 연산자

<< 경우

- 왼쪽(MSB 방향)으로 이동 할 때 생기는 공백 비트에는 0이 채워짐.
- 부호(MSB, 양수는 0, 음수는 1) 비트는 변경 없음.

a = 15	0	0	0	0	1	1	1	1	
a << 2	0	0	1	1	1	1	0	0	//32+16+8+4 = 60

b = −15	1	1	1	1	0	0	0	1	
b << 2	1	1	0	0	0	1	0	0	//−128+64+4 = −60

>> 경우

- 부호(MSB, 양수는 0, 음수는 1) 비트로 채워짐

a = 15	0	0	0	0	1	1	1	1	
a >> 2	0	0	0	0	0	0	1	1	//2+1 = 3

b = −15	1	1	1	1	0	0	0	1	
b >> 2	1	1	1	1	1	1	0	0	//−128+64+32+16+8+4 = −4

[예제 3.3] 이동(Shift) 연산자

```
#include <stdio.h>
int main()
{
    int a=15, b=-15;

    printf("a=%d, a<<2=%d₩n", a, a<<2);
    printf("b=%d, b<<2=%d₩n", b, b<<2);

    printf("a=%d, a>>2=%d₩n", a, a>>2);
    printf("b=%d, b>>2=%d₩n", b, b>>2);
    return 0;
}
```

```
a=15, a<<2=60
b=-15, b<<2=-60
a=15, a>>2=3
b=-15, b>>2=-4
```

비교 연산자

C 언어에서의 참과 거짓의 의미

C 언어에서 데이터에 대한 참의 의미는 0이 아닌 수를 말하며, 0은 거짓으로 간주합니다. 문자형일 경우 NULL('₩0')은 거짓이고 그 이외의 문자는 참입니다. 참과 거짓에 관련된 연산의 반환 값은 참인 경우에는 1, 거짓인 경우에는 0을 반환합니다.

[표 3.6] 비교 연산자의 표현식과 의미

연산자	의미	표현식	y=? (a=3; b=2; 경우)
〉	왼쪽 값이 오른쪽 값보다 크다.	y = (a 〉 b);	1
〈	왼쪽 값이 오른쪽 값보다 작다.	y = (a 〈 b);	0
〉=	왼쪽 값이 오른쪽 값보다 크거나 같다.	y = (a 〉= b);	1
〈=	왼쪽 값이 오른쪽 값보다 작거나 같다.	y = (a 〈= b);	0
==	왼쪽 값과 오른쪽 값이 같다.	y = (a == b);	0
!=	왼쪽 값과 오른쪽 값이 같지 않다.	y = (a != b);	1

비트 연산자

[표 3.7] 비트 연산자

비트 연산자			&	|	^	~
의미			비트 곱(AND)	비트 합(OR)	배타적 논리합(XOR)	비트 반전
비트연산	x_0	y_0	x_0 & y_0	x_0 | y_0	x_0 ^ y_0	$\sim x_0$;
	0	0	0	0	0	1
	0	1	0	1	1	1
	1	0	0	1	1	0
	1	1	1	1	0	0
표현식			z = x & y;	z = x | y;	z = x ^ y;	z = ~x;
x=7, y=5일 경우 표현식의 결과 z			z=5	z=7	z=2	z=-8

[예제 3.4] 비트 단위 논리 연산자

```
#include <stdio.h>
int main()
{
    int x = 7, y = 5;

    printf("%d & %d = %d₩n", x, y, x&y);
    printf("%d | %d = %d₩n", x, y, x|y);
    printf("%d ^ %d = %d₩n", x, y, x^y);
    printf("~%d = %d₩n", x, ~x);
    return 0;
}
```

```
7 & 5 = 5
7 | 5 = 7
7 ^ 5 = 2
~7 = -8
```

	b7	b6	b5	b4	b3	b2	b1	b0
x = 5	0	0	0	0	0	1	0	1
y = 7	0	0	0	0	0	1	1	1
x & y	0	0	0	0	0	1	0	1
x | y	0	0	0	0	0	1	1	1
x ^ y	0	0	0	0	0	0	1	0
~x	1	1	1	1	1	0	1	0

논리 연산자

[표 3.8] 논리 연산자

논리 연산자	&&	||	!
의미	논리 곱(AND)	논리 합(OR)	논리 반전
표현식	z = x && y;	z = x || y;	z = !x;
x=7, y=0일 경우 표현식의 결과 z	z=0	z=1	z=0

[예제 3.5] 논리 연산자

```
#include <stdio.h>
int main()
{
    int x = 7, y = 0;
    printf("%d && %d = %d₩n", x, y, x && y);
    printf("%d || %d = %d₩n", x, y, x || y);
    printf("!%d = %d₩n", x, !x);
    return 0;
}
```

```
7 && 0 = 0
7 || 0 = 1
!7 = 0
```

조건 연산자

조건 연산자 형식

조건식 ? 수식1 : 수식2;

반환값 = 조건식 ? 수식1 : 수식2;

[예제 3.6] 조건 연산자

```
#include <stdio.h>
int main()
{
    int apple = 10, person = 3;
    //int apple = 10, person = 5;
    int share, remain;

    share = (apple >= person) ? apple / person : 0;
    remain = share? apple % person : apple;

    printf("apple=%d, person=%d, share=%d, remain=%d₩n", apple, person,
    share, remain);
    remain? printf("사과 드실 분?₩n") : printf("맛있게 드세요.");
    return 0;
}
```

```
apple=10, person=3, share=3, remain=1
사과 드실 분?
```

```
apple=10, person=5, share=2, remain=0
맛있게 드세요.
```

산술 & 대입 연산자

[표 3.4] 산술 연산자

연산자	기능	사용 예	의미
+	덧셈	y = a + b;	a와 b를 더해 그 결과를 y에 대입
−	뺄셈	y = a − 3;	a에서 3을 빼 그 결과를 y에 대입
*	곱셈	y = a * b;	a와 b를 곱해 그 결과를 y에 대입
/	나눗셈	y = a / 3;	a를 3으로 나누어 그 결과를 y에 대입
%	나머지	y = a % b	a를 b로 나누어 그 나머지 값을 y에 대입

[표 3.9] 혼합 대입연산자의 의미

표현식	의미
a += b;	a = a + b;
a −= 3;	a = a − 3;
a *= b;	a = a * b;
a /= 3;	a = a / 3;
a %=b;	a = a % b;
a &= 3;	a = a & 3;
a |= b;	a = a | b;

나열 연산자

나열 연산자 형식

연산식1, 연산식2, 연산식3, ...;

[예제 3.7] 나열 연산자

```
#include <stdio.h>
int main()
{
        int a, b, c, d_bro, d_brx;

        a = 2;
        d_bro = (a++, b = a * 2, c = b - 2);

        a = 2;
        d_brx = a++, b = a * 2, c = b - 2;

        printf("a=%d, b=%d, c=%d, d_bro=%d\n", a, b, c, d_bro);
        printf("a=%d, b=%d, c=%d, d_brx=%d\n", a, b, c, d_brx);
        return 0;
}
```

```
a=3, b=6, c=4, d_bro=4
a=3, b=6, c=4, d_brx=2
```

연산에 사용되는 함수들

지수승, 제곱근, 올림, 내림 함수

```
#include <math.h>
double pow(double x, double y);      //x의 y지수승 값을 반환
double sqrt(double x);               //x의 제곱근을 반환
double ceil(double x);               //x보다 큰 최소 정수를 반환
double floor(double x);              //x보다 작은 최대 정수를 반환
```

[예제 3.8] 연산에 사용되는 함수들

```
#include <stdio.h>
#include <math.h>
#define PI 3.141592
int main()
{
    printf("pow(2,4) = %f ₩n", pow(2,4));
    printf("sqrt(16) = %lf ₩n", sqrt(16));
    printf("ceil(%lf) = %lf ₩n", PI, ceil(PI));
    printf("floor(%lf) = %lf ₩n", PI, floor(PI));
    return 0;
}
```

```
pow(2,4) = 16.000000
sqrt(16) = 4.000000
ceil(3.141592) = 4.000000
floor(3.141592) = 3.000000
```

연산자 응용

[예제 3.9] 연산자 응용

```
#include <stdio.h>
#include <conio.h>
int main()
{
      char cha, chb;
      int ina, inb;

      printf("₩n0~9까지의 숫자 입력 : ");
      cha = getche();

      printf("₩n0~9까지의 숫자 입력 : ");
      chb = getche();

      ina = (cha >= '0' && cha <= '9') ? cha - '0' : 0; //'5'->5 //'5'-'0'= 5
      inb = (chb >= '0' && chb <= '9') ? chb - '0' : 0;

      printf("₩n합=%d ₩n", ina + inb);
      return 0;
}
```

```
0~9까지의 숫자 입력 : 5
0~9까지의 숫자 입력 : 7
합=12
```

```
0~9까지의 숫자 입력 : 5
0~9까지의 숫자 입력 : a
합=5
```

개념 확인학습 & 적용 확인학습 & 응용 프로그래밍

- 다음 파일에 있는 문제들의 해답을 스스로 작성 해 보신 후 개념 & 적용 확인 학습 영상을 학습 하시기 바랍니다.
  - c_03장_연산자_ex.pdf
- 퀴즈와 과제가 출제되었습니다.
  - 응용 프로그래밍 영상을 학습하신 후 과제와 퀴즈를 수행 하시기 바랍니다.

Q & A

- “연산자”에 대한 학습이 모두 끝났습니다.

- 모든 내용을 이해 하셨나요?
- 아직 이해가 안되는 내용이 있다면 다시 한번 복습하시기 바랍니다.
- 질문은 한림 SmartLEAD 쪽지 또는 e-mail 또는 전화상담을 이용하시기 바랍니다.

- 퀴즈와 과제가 출제되었습니다. 마감시간에 늦지 않도록 주의해 주세요.

- 다음 시간에는 “선택과 반복 제어”에 대해 알아보겠습니다.

- 수고하셨습니다.^^