3강

C_PROGRAMMING

산술연산자

❖ 두 개의 피 연산자 간의 산술연산을 하기 위해 사용

연산자	연산자의 기능	결합 방향
=	연산자의 오른쪽에 있는 값을 연산자의 왼쪽에 있는 변수에 대입한다. 예) num = 20;	←
+	두 피 연산자의 값을 더한다. 예) num = 4 + 3;	→
-	왼쪽의 피연산자 값에서 오른쪽 피연산자 값을 뺀다 예) num = 4 - 3;	→
*	두 피연산자의 값을 곱한다. 예) num = 4 * 3;	→
/	왼쪽의 피연산자 값을 오른쪽의 피연산자 값으로 나눈다. 예) num = 7 / 3;	→
%	왼쪽의 피연산자 값을 오른쪽의 피연산자 값으로 나눴을 때 얻게 되는 나머지를 변환한다. 예) num = 7 % 3;	→

❖ %의 경의 짝,홀수 구분과 배수 구분 시 사용

'%'연산의 사용예

- 짝, 홀수 구분
  - 10 % 2 => 0으로 짝수
  - 15 % 2 => 1로 홀수
- 배수 구분
  - 123 % 3 => 0으로 3의 배수
- 숫자의 자리수 구분하기
  - 156 %10 => 6
  - 156 / 10 => 15
  - 15 %10 => 5
  - 15 / 10 => 1

예제

```
#include <stdio.h>
int main(void){
   int su1 = 20, su2 = 3;

   printf("%d + %d = %d₩n",  su1, su2, su1 + su2);
   printf("%d - %d = %d₩n",  su1, su2, su1 - su2);
   printf("%d * %d = %d₩n",  su1, su2, su1 * su2);
   printf("%d / %d = %d₩n",  su1, su2, su1 / su2);
   printf("%d %% %d = %d₩n", su1, su2, su1 % su2);

   return 0;
}
```

관계연산자

- 두 개의 피 연산자 간의 대소관계를 비교하기 위하여 사용
- 비교값이 거짓이면 결과 값은 0, 참이면 결과값은 1
- 값이 거짓 0이외의 모든 수는 참을 의미

연산자	의 미	사용 예
<	…보다 작다	if(a<10)~
>	…보다 크다	if(a>10)~
<=	…보다 작거나 같다	if(a<=10)~
>=	…보다 크거나 같다	if(a>=10)~
==	…와 같다	if(a==10)~
!=	…와 같지 않다	if(a!=10)~

예제

```
#include <stdio.h>
int main( ){
   float su1 = 3.01, su2 = 3.0;

   printf("변수 su1과 su2의 크기 비교 결과 : %d₩n", su1 <= su2);
   printf("변수 su1과 su2의 크기 비교 결과 : %d₩n", su1 >= su2);
   printf("변수 su1과 su2의 크기 비교 결과 : %d₩n", su1 == su2);
   printf("변수 su1과 su2의 크기 비교 결과 : %d₩n", su1 != su2);

   return 0;
}
```

대입연산자

- ❖ 정의
  - ▪ 우측에 수행한 결과를 좌측에 지정된 변수로 대입
- ❖ 복합대입연산자
  - ▪ 대입연산자를 다른 연산자와 결합하여 사용

복합 대입 연산자	사용예	의 미
+=	a+=b	a=a+b
-=	a-=b	a=a-b
*=	a*=b	a=a*b
/=	a/=b	a=a/b
%=	a%=b	a=a%b

예제

```
#include <stdio.h>
int main(  ){
   int su1, su2;

   su1 = su2 = 5;
   printf("su1 + 1 = %d₩n", su1 += 1);
   printf("su1 - 1 = %d₩n", su1 -= 1);
   printf("su1 * su2 = %d₩n", su1 *= su2);
   printf("su1 / su2 = %d₩n", su1 /= su2);
   printf("su1 %% su2 = %d₩n", su1 %= su2);

   return 0;
}
```

```
su1 = su2 = 5;
①printf("su1 + 1 = %d₩n", su1 += 1);          //su1 = su1 + 1
②printf("su1 - 1 = %d₩n", su1 -= 1);          //su1 = su1 - 1
③printf("su1 * su2 = %d₩n", su1 *= su2);      //su1 = su1 * su2
④printf("su1 / su2 = %d₩n", su1 /= su2);      //su1 = su1 / su2
⑤printf("su1 %% su2 = %d₩n", su1 %= su2);     //su1 = su1 % su2
```

	su1	Su2
default	5	5
①	6	5
②	5	5
③	25	5
④	5	5
⑤	0	5

논리연산자

❖ 참과 거짓을 판별하는 연산

연산자	연산자의 기능	결합방향
||	예) A || B A와 B 둘 중 하나라도 '참'이면 연산결과로 '참'을 반환(논리OR)	
&&	예) A && B A와 B 모두 '참'이면 연산결과를 '참'을 반환(논리AND)	
!	예) !A A가 '참'이면 '거짓', A가 '거짓'이면 '참'을 반환(논리NOT)	

A	B	||(OR, +, 합집합)	&&(AND, *, 교집합)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

예제

```
#include <stdio.h>
int main( ){
    int  num1 = 10;
    int  num2 = 20;
    int  result1, result2, result3;

    result1 = (num1==10 && num2==12);
    result2 = (num1<10 || num2>12);
    result3 = (!num1);

    printf("result1 : %d ₩n", result1);
    printf("result2 : %d ₩n", result2);
    printf("result3 : %d ₩n", result3);
    return 0;
}
```

증감연산자

❖ 피연산자를 1씩 증가 혹은 감소하는 기능

연산자	연산자의 기능	결합방향
++num	값을 1증가 후, 속한 문장의 나머지를 진행(선 증가, 후 연산) 예) val = ++num;	⬅
num++	속한 문장을 먼저 진행한 후, 값을 1 증가(선 연산, 후 증가) 예) val = num++;	⬅
--num	값을 1감소 후, 속한 문장의 나머지를 진행(선 감소, 후 연산) 예) val = --num;	⬅
num--	속한 문장을 먼저 진행한 후, 값을 1 감소(선 연산, 후 감소) 예) val = num--;	⬅

❖ 전치와 후치에 따른 연산자 비교

- 전치 : ++a로 표기하며 a=a+1을 먼저 처리한다
- 후치 : a++로 표기하며 a의 데이터를 사용한 후 a=a+1을 처리한다.

예제

```
#include <stdio.h>
int main(void){
    int  num1 = 12;
    int  num2 = 12;

    printf("num1 : %d ₩n", num1);
    printf("num1++ : %d ₩n", num1++);          //후위 증가
    printf("num1 : %d ₩n₩n", num1);

    printf("num2 : %d ₩n", num2);
    printf("++num2: %d ₩n",  ++num2);          //전위 증가
    printf("num2 : %d ₩n₩n", num2);
    return 0;
}
```

예제

```
#include <stdio.h>
int main(void) {
    int  num1 = 10;
    int  num2 = (num1--) + 2;

    printf("num1: %d ₩n", num1);
    printf("num2: %d ₩n", num2);

    return 0;
}
```

비트연산자

❖ 10진수를 2진수로 변환하여 각 비트별로 논리/이동 연산을 한다.

비트연산자	의미
|	비트 단위 논리합(OR)
&	비트 단위 논리곱(AND)
^	비트 단위 배타적 논리합(XOR)
~	비트 부정(NOT)
<<	비트 좌측 이동(Left Shift)
>>	비트 우측 이동(Right Shift)

A	B	||(OR, +, 합집합)	&&(AND, *, 교집합)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

예제

```
#include <stdio.h>

int main(void)
{
    int  num1 =12;
    int  num2 = 7;
    int  result = num1 | num2;

    printf("su1과 su2의 논리합 : %d₩n", result);

    return 0;
}
```

	1	1	0	0
|	0	1	1	1
	1	1	1	1

예제

```
#include <stdio.h>
int main(void)
{
    int  su1 = 15;
    su1 = ~su1;
    printf("su1의 비트 부정 : %d₩n", su1);

    return 0;
}
```

~ 0000 0000 0000 0000 0000 0000 0000 1111

1111 1111 1111 1111 1111 1111 1111 0000

-16

예제

```
#include <stdio.h>
int main(void)
{
   int op = 30, result;
   result = op << 3;
   printf("30을 좌측으로 3비트 이동시킨 결과 = %d₩n", result);

   return 0;
}
```

```
<< 3      0000 0000   0000 0000   0000 0000   0001 1110
          0000 0000   0000 0000   0000 0000   1111 0000
```

기타연산자

❖ sizeof 연산자 : 크기를 바이트 단위로 표기

❖ 예제

```
#include <stdio.h>
int main(void)
{
        int a=0, b=0, c=0;
        printf("정수의 크기는 %d입니다. ₩n", sizeof(int));

        return 0;
}
```

예제

```
#include <stdio.h>

main(){
   int su1 = 123;
   char ch = 'a';
   float su2 = 12.345;

   printf("변수 su1의 크기          : %d byte₩n",sizeof(su1));
   printf("문자형 자료형의 크기    : %d byte₩n",sizeof(char));
   printf("수식(su1+su2)의 크기    : %d byte₩n", sizeof(su1+su2));
   printf("실수 1.23456 의 크기    : %d byte₩n",sizeof(1.23456));
   printf("문자 ch의 크기 : %d byte₩n",sizeof(ch));

}
```

연산자 우선순위

연산자	연산순서	우선순위	비고
(), [], ->, .(점)	좌에서 우		
sizeof, (type), &, *, -(단항), +(단항), --, ++, ~, !	좌에서 우		단항
*(곱셈), / , %, +, /	좌에서 우		산술
<<, >>	좌에서 우		비트
<, <=, >, >=, ==, !=	좌에서 우		비교
&, ^, |	좌에서 우		비트
&&, ||	좌에서 우		논리
? :	우에서 좌		삼항
%=, /=, *=, -=, +=, =	좌에서 우		대입
,	좌에서 우		콤마