

C Programming

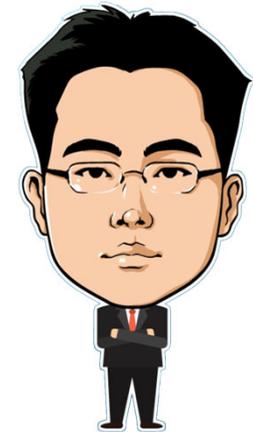
C 프로그램 구조 (C Program Structure)



Seo, Doo-Ok

Clickseo.com

clickseo@gmail.com



목 차



백문이불여일타(百聞而不如一打)

- 수식과 연산자
- 우선 순위와 결합성
- 비트 연산자



수식과 연산자



백문이불여일타(百聞而不如一打)

- 수식과 연산자

- 논리적 데이터와 연산

- 우선 순위와 결합성

- 비트 연산자



수 식 (1/11)

- 수식(Expression)

- 단일 값으로 변환하기 위한 피연산자와 연산자의 나열

- 연산자(Operator)
- 피연산자(Operand)

“수식들은 항상 단일 값으로 변형된다.”

- 수식의 종류

- 일차식(Primary Expression)
- 이진식(Binary Expression)
- 배정식(Assignment Expression)
- 후위식(Postfix Expression)
- 단항식(Unary Expression)
- 콤마식(Comma Expression)
- 삼원식(Ternary Expression)

수 식 (2/11)

- 일차식(Primary Expression)



- 이름(Name)

nValue	w100	Clickseo	MAX_SIZE	size
--------	------	----------	----------	------

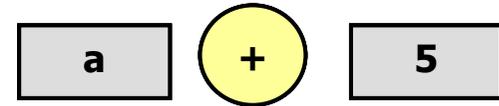
- 상수(Constant)

5	123.98	'A'	"Hi~ Clickseo"
---	--------	-----	----------------

- 괄호식

(10 * 5 + 20)	(a = 10 + b * 5)
---------------	------------------

수 식 (3/11)



● 이진식(Binary Expression)

피연산자 연산자 피연산자

○ 곱셈식(Multiplicative Expression)

종류	내 용
*	두 피연산자들의 대수 곱셈
/	첫 번째 피연산자를 두 번째 피연산자로 나누는 대수 나눗셈 - 두 피연산자가 정수 → 정수 몫 - 둘 중 어느 한 쪽의 피연산자가 부동소수점이면 부동소수점 몫
%	첫 번째 피연산자를 두 번째 피연산자로 나눈 나머지 (Remainder)

정 수	부동소수점 수
몫 : $10 / 3 \rightarrow 3$	나눗셈: $10.0 / 3.0 \rightarrow 3.333333$
나머지: $10 \% 3 \rightarrow 1$	

“모듈러스(Modulus) 연산자의 두 피연산자들은 반드시 정수 형태가 되어야 한다.”

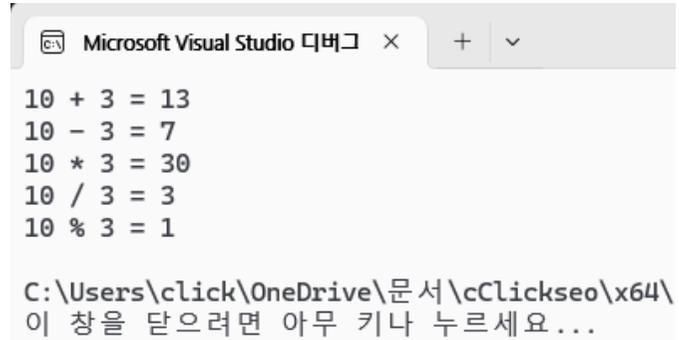
수 식 (4/11)

예제 2-1: 이진식

```
#include <stdio.h>
int main(void)
{
    int a = 10;
    int b = 3;

    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);

    return 0;
}
```



Microsoft Visual Studio 디버그

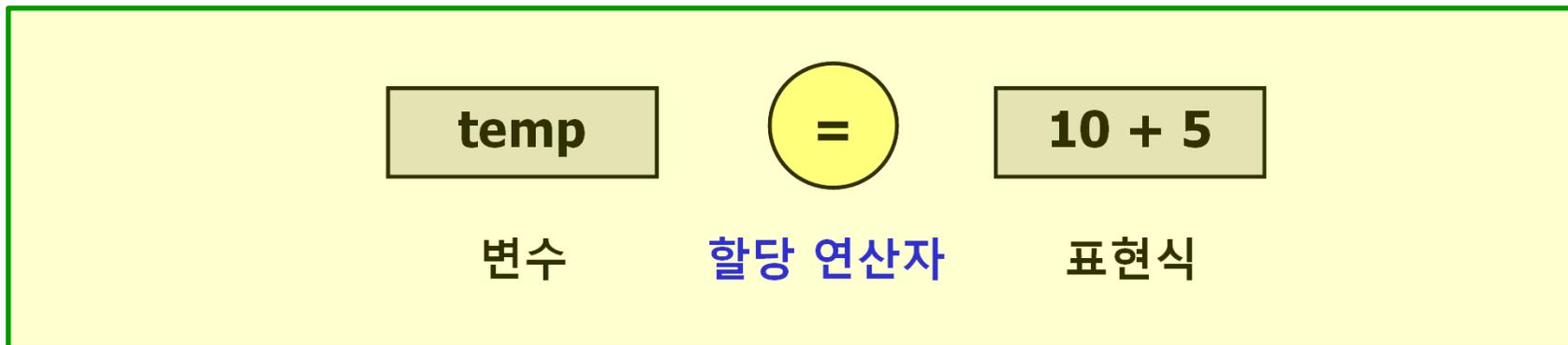
```
10 + 3 = 13
10 - 3 = 7
10 * 3 = 30
10 / 3 = 3
10 % 3 = 1
```

C:\Users\click\OneDrive\문서\cClickseo\x64\
이 창을 닫으려면 아무 키나 누르세요...

수 식 (5/11)

- **배정식**(Assignment Expression)

- 할당 연산자(=)의 오른쪽 피연산자를 평가하고, 왼쪽 변수에 대입
 - 하나의 값과 하나의 결과
- 두 가지 형태
 - 단일배정(Simple Assignment)
 - 복합배정(Compound Assignment)



수 식 (6/11)

- **배정식: 복합 배정**

- 복합배정(Compound Assignment) : 단일배정에 대한 약칭 표기

복 합 식	등 가 식
$a += b$	$a = a + b$
$a -= b$	$a = a - b$
$a *= b$	$a = a * b$
$a /= b$	$a = a / b$
$a \% = b$	$a = a \% b$

- 만약 복합배정이 이진식과 함께 사용된다면 이진식이 먼저 평가된다.

$a *= b + 3$ --> $a = a * (b + 3)$

수 식 (7/11)

예제 2-2: 복합 배정

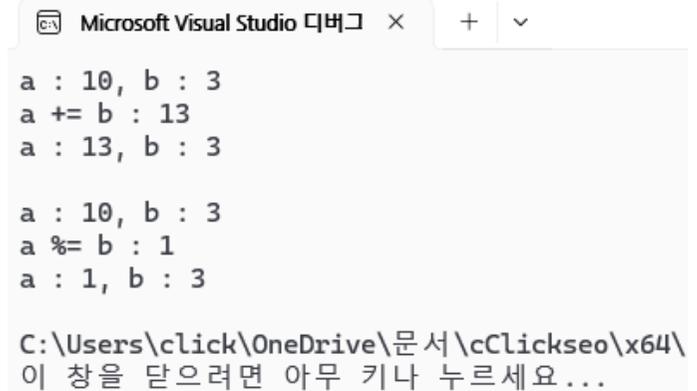
```
#include <stdio.h>
int main(void)
{
    int    a = 10, b = 3;

    printf("a : %d, b : %d \n", a, b);
    printf("a += b : %d \n", a += b );
    printf("a : %d, b : %d \n\n", a, b);

    a = 10; b = 3;

    printf("a : %d, b : %d \n", a, b);
    printf("a %= b : %d \n", a %= b );
    printf("a : %d, b : %d \n", a, b);

    return 0;
}
```



Microsoft Visual Studio 디버그

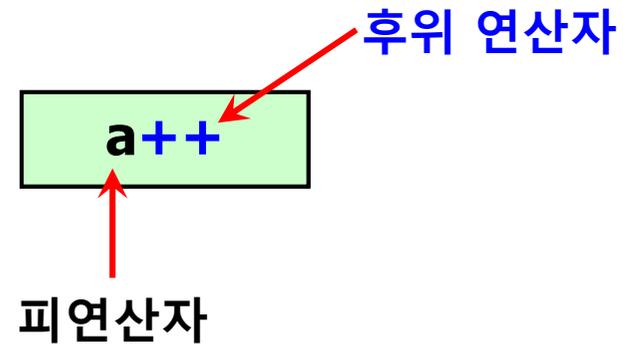
```
a : 10, b : 3
a += b : 13
a : 13, b : 3

a : 10, b : 3
a %= b : 1
a : 1, b : 3
```

C:\Users\click\OneDrive\문서\cClickseo\x64\
이 창을 닫으려면 아무 키나 누르세요...

수 식 (8/11)

- 후위식(Postfix Expression)



- 함수호출

- 함수 이름은 피연산자 이고 연산자는 이름 뒤에 오는 괄호이다.

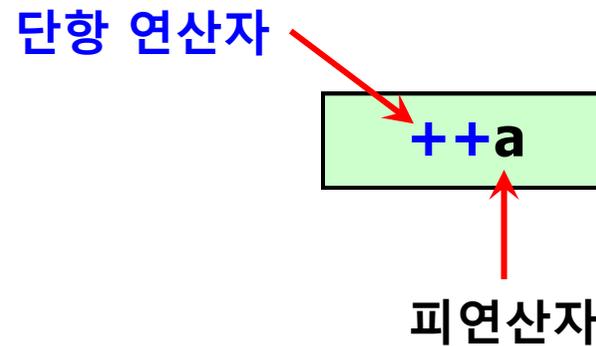
- 후위증가(Postfix Increment)와 후위감소(Postfix Decrement)

수 식	등가식
a++	a = a + 1
a--	a = a - 1

수 식 (9/11)

- 단항식(Unary Expression)

- 하나의 연산자와 하나의 피연산자로 구성



- 전위증가(Prefix Increment)와 전위감소(Prefix Decrement)

수 식	등가식
++a	$a = a + 1$
--a	$a = a - 1$

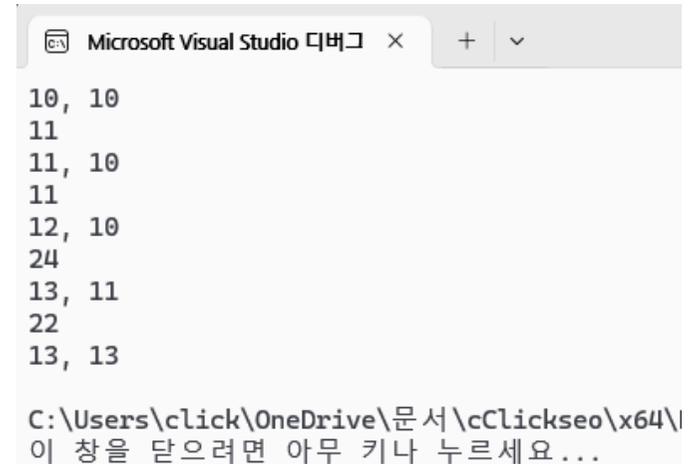
수 식 (10/11)

예제 2-3: 전위 증가와 후위 증가

```
#include <stdio.h>
int main(void)
{
    int a = 10, b = 10;

    printf("%d, %d\n", a, b);
    printf("%d\n", ++a);
    printf("%d, %d\n", a, b);
    printf("%d\n", a++);
    printf("%d, %d\n", a, b);
    printf("%d\n", ++a + ++b);
    printf("%d, %d\n", a, b);
    printf("%d\n", b++ + b++);
    printf("%d, %d\n", a, b);

    return 0;
}
```



```
Microsoft Visual Studio 디버그 × + ▾
10, 10
11
11, 10
11
12, 10
24
13, 11
22
13, 13
C:\Users\cLick\OneDrive\문서\cClickseo\x64\
이 창을 닫으려면 아무 키나 누르세요...
```

수 식 (11/11)

- 단항식: 부호와 sizeof 연산자

- sizeof 연산자

```
sizeof (int) // 할당 받는 메모리 공간의 크기를 바이트 단위로 반환
```

```
res = sizeof (int) // sizeof 연산자의 결과(정수형의 값)를 저장
```

```
sizeof (-345.23) // 실수형 상수가 할당 받는 메모리 공간의 크기를 확인
```

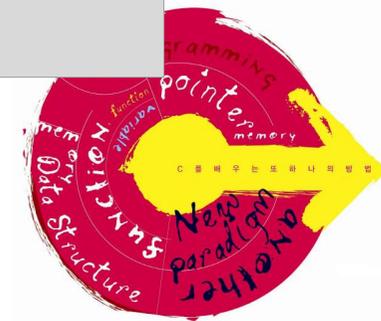
- 단항 플러스(Unary Plus)와 단항 마이너스(Unary Minus)

수 식	수식 평가 전/후 a의 내용	수식 값
+a	10	+10
-a	10	-10
+a	-10	-10
-a	-10	+10



수식과 연산자

논리적인 데이터와 연산



논리적인 데이터와 연산자 (1/4)

- 논리 연산자(Logical Operators)

- 논리적 데이터(Logical Data)

- (명제) 실생활에서 “예-아니요” 와 같은 데이터
- 컴퓨터 과학에서는 참(true) 과 거짓(false) 을 사용한다.

- not 연산자 : !

- 15의 선행을 갖는 단항 연산자(Unary Operator)
- 참 값(Nonzero)을 거짓(Zero)으로 바꾸고, 거짓 값(0)을 참(1)으로 바꾼다.

- and 연산자 : &&

- 5의 선행을 갖는 이진 연산자(Binary Operator)

- or 연산자 : ||

- 4의 선행을 갖는 이진 연산자

“만약 값이 0 이면, 논리 값이 거짓으로 사용된다.
만약 값이 0 이 아니면, 논리 값은 참으로 사용된다.”

논리적인 데이터와 연산자 (2/4)

예제 2-4: 논리식

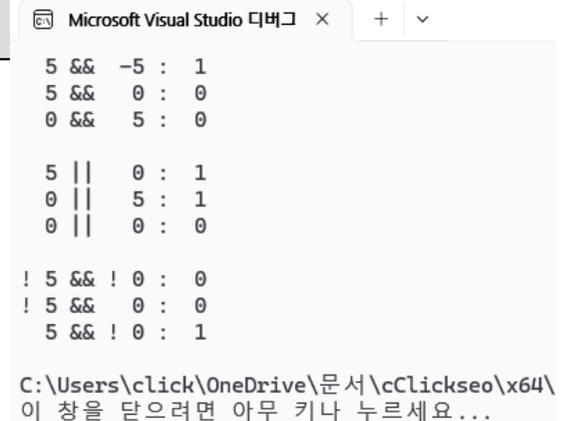
```
#include <stdio.h>
int main(void)
{
    int a = 5;
    int b = -5;
    int c = 0;

    printf(" %2d && %2d : %2d\n", a, b, a && b );
    printf(" %2d && %2d : %2d\n", a, c, a && c );
    printf(" %2d && %2d : %2d\n\n", c, a, c && a );

    printf(" %2d || %2d : %2d\n", a, c, a || c );
    printf(" %2d || %2d : %2d\n", c, a, c || a );
    printf(" %2d || %2d : %2d\n\n", c, c, c || c );

    printf(" !%2d && !%2d : %2d\n", a, c, !a && !c );
    printf(" !%2d && %2d : %2d\n", a, c, !a && c );
    printf(" %2d && !%2d : %2d\n", a, c, a && !c );

    return 0;
}
```



```
Microsoft Visual Studio 디버그 x + v
5 && -5 : 1
5 && 0 : 0
0 && 5 : 0

5 || 0 : 1
0 || 5 : 1
0 || 0 : 0

! 5 && ! 0 : 0
! 5 && 0 : 0
5 && ! 0 : 1

C:\Users\click\OneDrive\문서\cClickseo\x64\
이 창을 닫으려면 아무 키나 누르세요...
```

논리적인 데이터와 연산자 (3/4)

- **관계 연산자**(Relational Operators)

- 두 개의 피연산자를 받아서 서로 비교하는 이진 연산자
 - 결과는 논리적 데이터 값: **항상 참(1)과 거짓(0)**

관계연산자	의 미	우선순위
>	greater than	10
>=	greater than or equal	
<	less than	
<=	less than or equal	
==	equal	9
!=	not equal	

논리적인 데이터와 연산자 (4/4)

예제 2-5: 관계 연산자

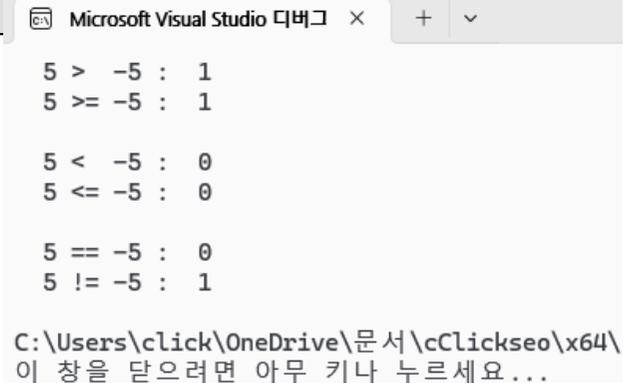
```
#include <stdio.h>
int main(void)
{
    int a = 5;
    int b = -5;

    printf(" %2d > %2d : %2d\n", a, b, a > b );
    printf(" %2d >= %2d : %2d\n\n", a, b, a >= b );

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

    printf(" %2d == %2d : %2d\n", a, b, a == b );
    printf(" %2d != %2d : %2d\n", a, b, a != b );

    return 0;
}
```



```
Microsoft Visual Studio 디버그 x + v
5 > -5 : 1
5 >= -5 : 1

5 < -5 : 0
5 <= -5 : 0

5 == -5 : 0
5 != -5 : 1

C:\Users\click\OneDrive\문서\cClickseo\x64\
이 창을 닫으려면 아무 키나 누르세요...
```

우선 순위와 결합성

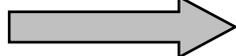


백문이불여일타(百聞而不如一打)

- 수식과 연산자
- 우선 순위와 결합성
 - 데이터 형 변환
- 비트 연산자



우선 순위와 결합성 (1/2)

기능별 분류	연산자	결합성	우선순위	
일차연산자	() [] -> .		1	
단항 연산자	+ - ++ -- ! ~ * & sizeof (datatype)		2	
산술연산자	* / %		3	
	+ -		4	
시프트 연산자	<< >>		5	
비교 연산자	< <= > >=		6	
등가 연산자	== !=		7	
비트논리연산자	&		8	
	^		9	
			10	
논리 연산자	&&		11	
			12	
조건 연산자	? :			13
대입 연산자	= += -= *= /= %=			14
	<<= >>= &= ^= =			
coma 연산자	,			15

우선 순위와 결합성 (2/2)

- **결합성**(Associativity)

- **왼쪽 결합성**(Left Associativity)

- 왼쪽에서 시작하여 오른쪽으로 이동해가면서 수식을 평가하는 것

$3 * 8 / 4 \% 4 * 5 \quad \text{-->} \quad (((3 * 8) / 4) \% 4) * 5$

- **오른쪽 결합성**(Right Associativity)

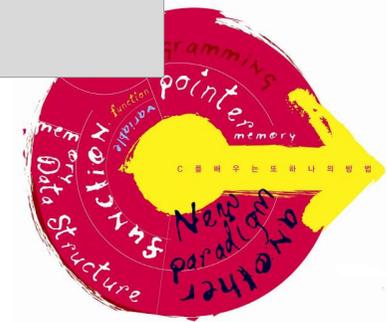
- 오른쪽에서 왼쪽으로 진행해 가면서 수식을 평가하는 것
 - 3가지 형식 : 단항식, 조건부 삼원식, 그리고 배정식

$a += b *= c -= 5 \quad \text{-->} \quad (a += (b *= (c -= 5)))$



우선 순위와 결합성

데이터 형 변환



데이터 형 변환 (1/7)

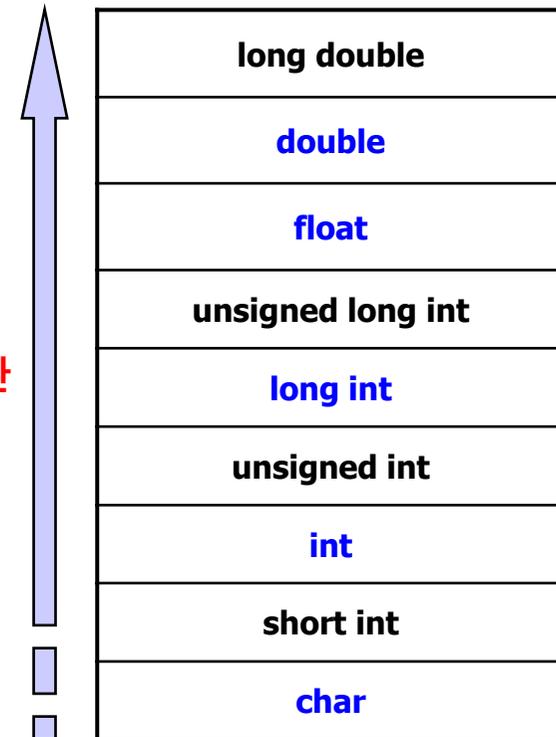
- **묵시적 형 변환**(Implicit Type Conversion)

- C 컴파일러가 자동으로 데이터의 형식을 변형시키는 것

“C 언어는 내부적으로 정해진 자동 변환 규칙에 따라 혼합식을 처리”

수 식	중간 형식
<code>char + int</code>	<code>int</code>
<code>int - long</code>	<code>long</code>
<code>int * double</code>	<code>double</code>
<code>float / double</code>	<code>double</code>
<code>(char + int) / double</code>	<code>double</code>

자동 형 변환



데이터 형 변환 (2/7)

예제 2-6: 묵시적 형 변환

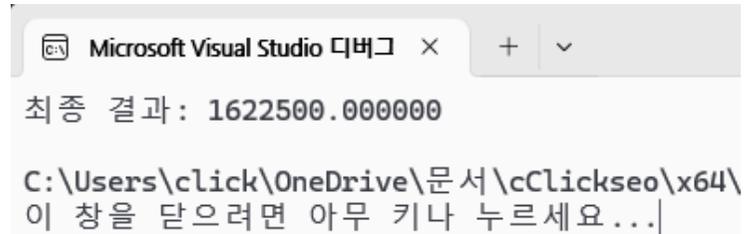
```
#include <stdio.h>
int main(void)
{
    char    ch;
    int     i;
    float   f;
    double  d;

    ch = 35;
    i  = 35000;
    f  = 1.3;      // warning C4305: '=' : truncation from 'const double' to 'float'

    d = ch * i * f + 30000;

    printf("최종 결과: %1f \n", d);

    return 0;
}
```



데이터 형 변환 (3/7)

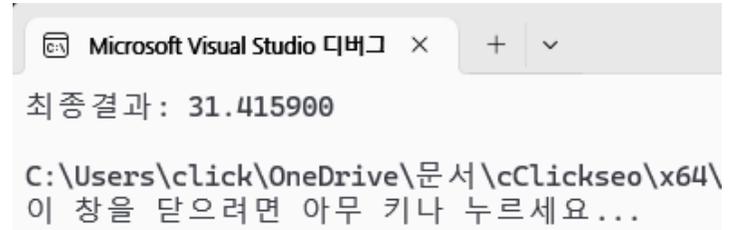
예제 2-7: 묵시적 형 변환

```
#include <stdio.h>
int main(void)
{
    int          i;
    double       d;

    i = 10;
    d = 3.14159;

    /* i * d 의 연산 도중 컴파일러가 double 형의 메모리 공간을 할당하고
       그 값을 출력할 뿐이지 변수 i 의 할당된 메모리 크기가 실제로 변하는 것은 아니다. */
    printf("최종결과: %lf \n", i * d);

    return 0;
}
```



Microsoft Visual Studio 디버그 × + ▾

최종결과: 31.415900

C:\Users\click\OneDrive\문서\cClickseo\x64\
이 창을 닫으려면 아무 키나 누르세요...

데이터 형 변환 (4/7)

예제 2-8: 데이터 손실

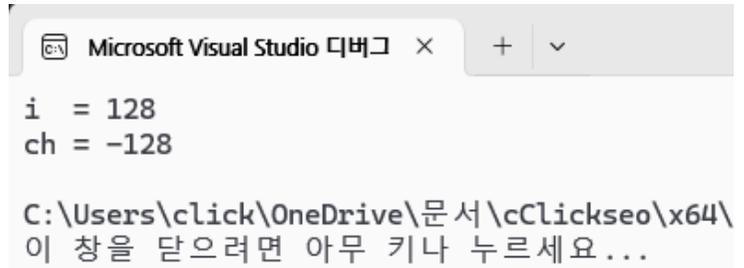
```
#include <stdio.h>
int main(void)
{
    int        i;
    char       ch;

    i = 128;

    // warning C4244: '=' : conversion from 'int' to 'char', possible loss of data
    ch = i;      // 데이터 손실 발생

    printf("i = %d \n", i);
    printf("ch = %d \n", ch);

    return 0;
}
```



```
Microsoft Visual Studio 디버그 x + v
i = 128
ch = -128
C:\Users\click\OneDrive\문서\cClickseo\x64\
이 창을 닫으려면 아무 키나 누르세요...
```

데이터 형 변환 (5/7)

예제 2-9: 데이터 손실

```
#include <stdio.h>
int main(void)
{
    int        i;
    double     d;
```

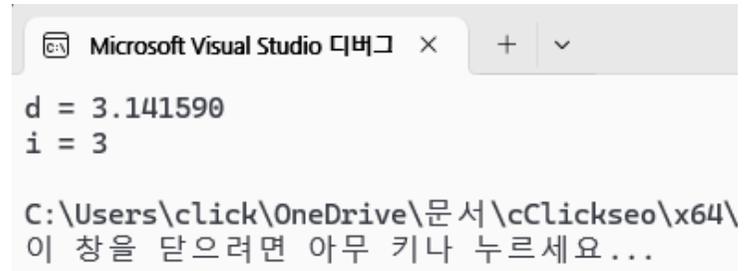
```
    d = 3.14159;
```

```
    // warning C4244: '=' : conversion from 'double' to 'int', possible loss of data
    i = d;           // 소수점 이하가 모두 잘려 나간다(데이터 손실 발생).
```

```
    printf("d = %lf \n", d);
    printf("i = %d \n", i);
```

```
    return 0;
```

```
}
```



```
Microsoft Visual Studio 디버그 × + ▾
d = 3.141590
i = 3
C:\Users\click\OneDrive\문서\cClickseo\x64\
이 창을 닫으려면 아무 키나 누르세요...
```

데이터 형 변환 (6/7)

- 명시적 형 변환(Explicit Type Conversion)

- 임의로 어떤 형식에서 다른 형식으로 데이터를 변환시킨다.
 - cast 수식 연산자(Cast Expression Operator)

```
(double) a; // 피연산자는 단항식이어야 한다.  
(double) (a + b); // 두 개의 정수 합을 하나의 부동소수로 변형
```

```
ave = (double) tot / num;
```

```
(double) (a / 10); // a가 3이라면 결과값은 0.0  
(double) a / 10; // 부동소수 결과는 숫자들 중 하나를 명시적 형 변환
```

데이터 형 변환 (7/7)

예제 2-10: 명시적 형 변환과 묵시적 형 변환

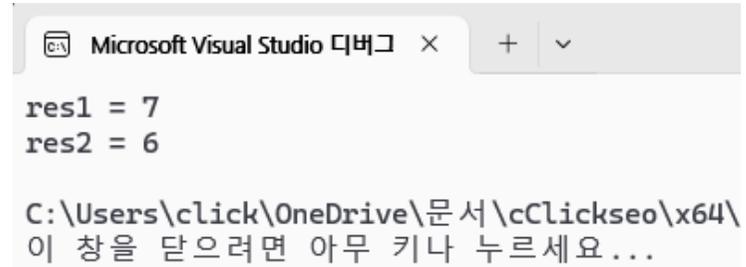
```
#include <stdio.h>
int main(void)
{
    double a, b;
    int    res1, res2;

    a = 3.4;
    b = 2.1;

    // warning C4244: '=' : conversion from 'double' to 'int', possible loss of data
    res1 = a * b;           // 묵시적(자동) 형 변환
    res2 = (int) a * (int) b; // 명시적(수동) 형 변환

    printf("res1 = %d \n", res1);
    printf("res2 = %d \n", res2);

    return 0;
}
```



Microsoft Visual Studio 디버그 × + ▾

```
res1 = 7
res2 = 6
```

C:\Users\click\OneDrive\문서\cClickseo\x64\
이 창을 닫으려면 아무 키나 누르세요...

비트 연산자



백문이불여일타(百聞而不如一打)

- 수식과 연산자
- 우선 순위와 결합성
- 비트 연산자
 - 비트 단위 논리 연산
 - 시프트 연산



비트 연산자

● 비트 연산자

○ 비트 단위의 정밀한 데이터 제어를 위해 사용

- 단, 피연산자로 `double`이나 `float`과 같은 부동형을 사용할 수 없다.

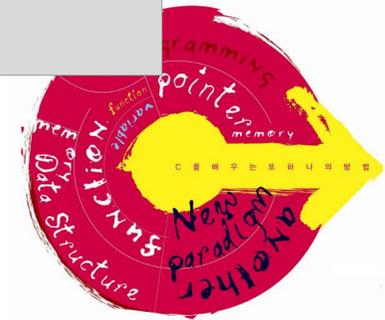
“char 를 포함한 정수 계산이 가능한 정수형 데이터에서만 동작”

구분	연산자	의미
비트 단위 논리 연산	<code>~a</code>	1의 보수 연산
	<code>a & b</code>	a와 b의 비트 단위 논리곱(AND)
	<code>a b</code>	a와 b의 비트 단위 논리합(OR)
	<code>a ^ b</code>	a와 b의 비트 단위 배타적 논리합(XOR)
시프트 연산	<code>a << n</code>	a의 각 비트를 왼쪽으로 n 만큼 이동
	<code>a >> n</code>	a의 각 비트를 오른쪽으로 n 만큼 이동



비트 연산자

비트 단위 논리 연산



비트 단위 논리 연산 (1/3)

- **~** : 1의 보수 연산자

- 1's Complement Operator

- 단항 연산자로 각각의 데이터 비트를 반대로 바꾼다.
- 1은 0으로 0은 1로 바꾼다(부호 비트도 반전).

- **&** : 논리곱

- 비트 단위 논리곱(AND): 특정 비트 값을 0으로 만들기 위해 주로 사용

```
a = a & 0x00FF;
```

```
00110100 01000001 : a
```

```
00000000 11111111 : 0x00FF
```

```
00000000 01000001 : a & 0x00FF
```

비트 단위 논리 연산 (2/3)

- | : 논리합

- 비트 단위 논리합(OR): 특정 비트 값을 1로 만들기 위해 주로 사용

```
a = a | 0xFF00;
```

```
00110100 01000001 : a
```

```
11111111 00000000 : 0xFF00
```

```
11111111 01000001 : a | 0xFF00
```

- ^ : 배타적 논리합

- 비트 단위 배타적 논리합(XOR): 특정 비트를 반전 시키기 위해 사용
 - 각각의 비트가 서로 다를 때 1이 된다.

```
a = a ^ 0xFF00;
```

```
00110100 01000001 : a
```

```
11111111 00000000 : 0xFF00
```

```
11001011 01000001 : a ^ 0xFF00
```

비트 단위 논리 연산 (3/3)

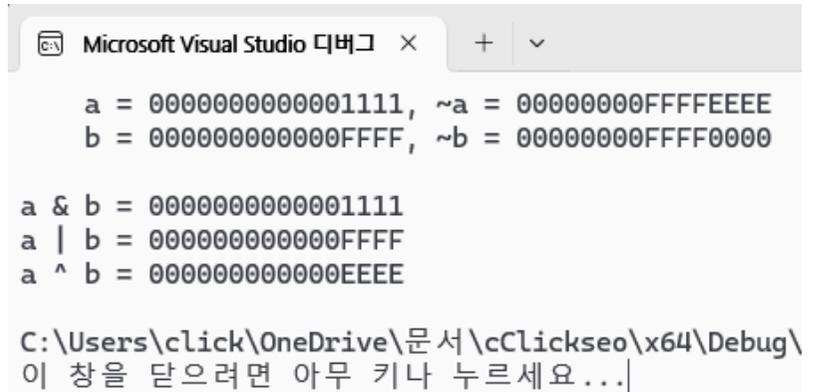
예제 2-11: 비트 단위 논리 연산자

```
#include <stdio.h>
int main(void)
{
    int    a = 0x00001111, b = 0x0000FFFF;

    printf("    a = %p, ~a = %p \n", a, ~a );
    printf("    b = %p, ~b = %p \n\n", b, ~b );

    printf("a & b = %p\n", a & b );
    printf("a | b = %p\n", a | b );
    printf("a ^ b = %p\n", a ^ b );

    return 0;
}
```



```
Microsoft Visual Studio 디버그 x + v
a = 0000000000001111, ~a = 00000000FFFFFFEE
b = 000000000000FFFF, ~b = 00000000FFFF0000

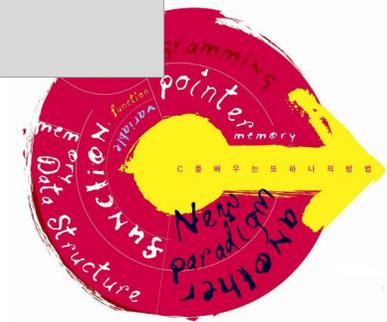
a & b = 0000000000001111
a | b = 000000000000FFFF
a ^ b = 000000000000EEEE

C:\Users\click\OneDrive\문서\cClickseo\x64\Debug\
이 창을 닫으려면 아무 키나 누르세요...
```



비트 연산자

시프트 연산



시프트 연산 (1/4)

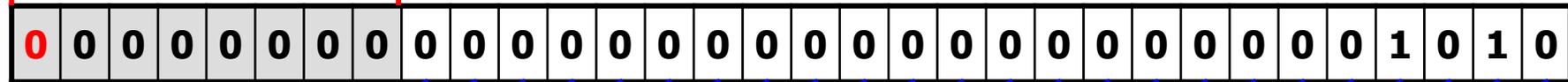
● 시프트 연산: 좌 시프트 연산자

○ 좌 시프트 연산자: <<

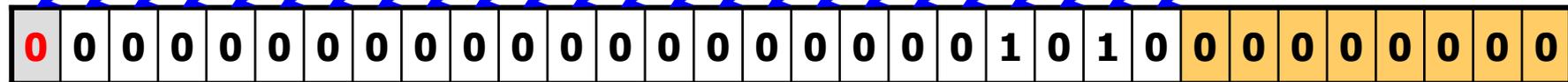
- 주어진 자릿수만큼 왼쪽으로 비트를 이동 시킬 수 있는 연산자
- 비트 연산자의 대상: 정수(즉, float 형이나 double 형에 비트 연산을 할 수 없다.)

“원래 있던 것은 잘려나간다”

```
int i = 10;
```



```
i = i << 8;
```



“이동된 자리는 0으로 채워진다”

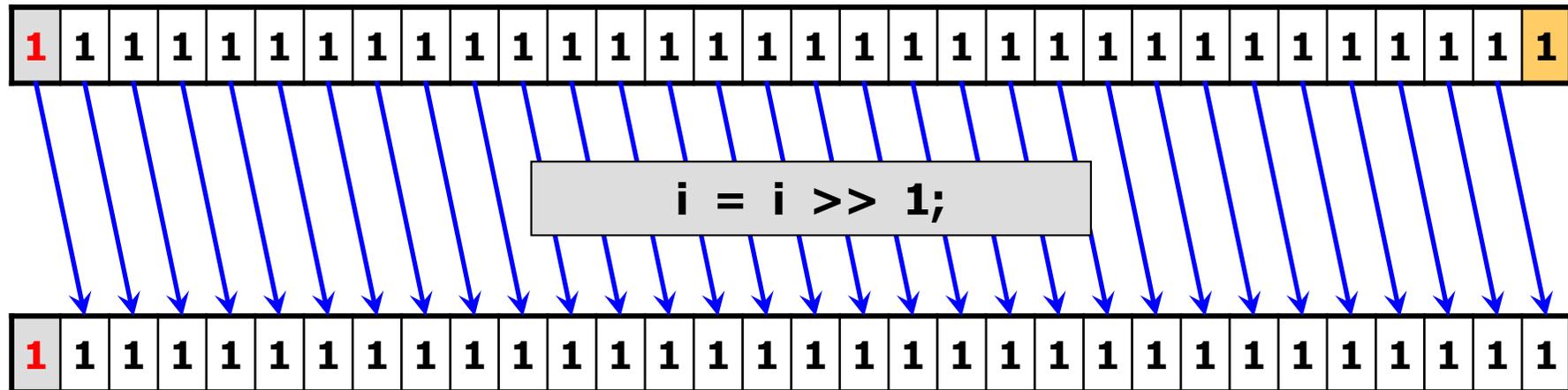
시프트 연산 (2/4)

- 시프트 연산: 우 시프트 연산자

- 우 시프트 연산자: \gg

- 주어진 자릿수만큼 오른쪽으로 비트를 이동시킬 수 있는 연산자
- 비트 연산자의 대상: 정수(float 형이나 double 형에 비트 연산을 할 수 없다.)

```
int i = -1;
```



“부호 비트가 이동하면
빈 자리는 부호 비트로 복사된다”

시프트 연산 (3/4)

예제 2-12: 비트 연산자 -- 좌 시프트 연산자

```
#include <stdio.h>
int main(void)
{
    int    a = 10;

    // 왼쪽으로 1비트 씩 이동할 때마다 값은 2배가 된다.
    printf("a        = %3d\n", a );
    printf("a << 1 = %3d\n", a << 1 );
    printf("a << 2 = %3d\n", a << 2 );
    printf("a << 3 = %3d\n", a << 3 );
    printf("a << 4 = %3d\n", a << 4 );

    return 0;
}
```

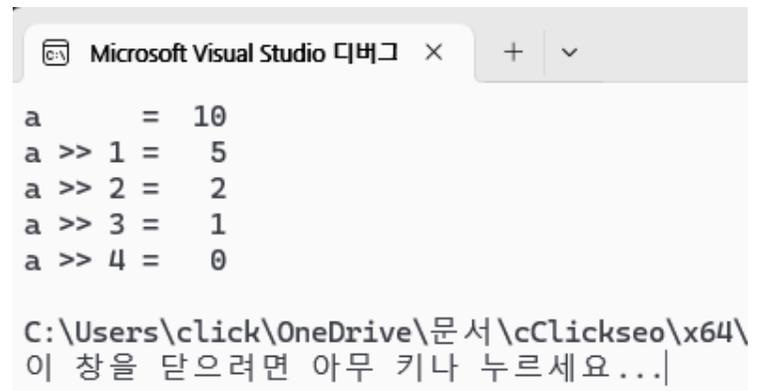
시프트 연산 (4/4)

예제 2-13: 비트 연산자 -- 우 시프트 연산자

```
#include <stdio.h>
int main(void)
{
    int    a = 10;

    // 오른쪽으로 1비트 씩 이동 할 때마다 값은 1/2이 된다.
    printf("a      = %3d\n", a );
    printf("a >> 1 = %3d\n", a >> 1 );
    printf("a >> 2 = %3d\n", a >> 2 );
    printf("a >> 3 = %3d\n", a >> 3 );
    printf("a >> 4 = %3d\n", a >> 4 );

    return 0;
}
```

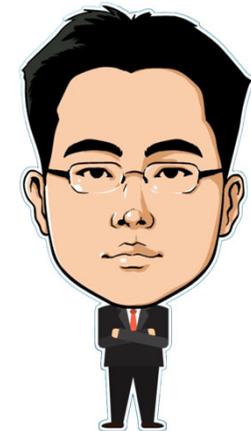


```
Microsoft Visual Studio 디버그 x + v
a      = 10
a >> 1 = 5
a >> 2 = 2
a >> 3 = 1
a >> 4 = 0

C:\Users\click\OneDrive\문서\cClickseo\x64\
이 창을 닫으려면 아무 키나 누르세요...
```

참고문헌

- [1] 서현우, "혼자 공부하는 C 언어: 1:1 과외 하듯 배우는 프로그래밍 자습서", 한빛미디어, 2023.
- [2] Paul Deitel, Harvey Deitel, "C How to Program", Global Edition, 8/E, Pearson, 2016.
- [3] Kamran Amini, 박지윤 번역, "전문가를 위한 C : 동시성, OOP부터 최신 C, 고급 기능까지!", 한빛미디어, 2022.
- [4] 서두옥, "(열혈강의) 또 하나의 C : 프로그래밍은 셀프입니다", 프리렉, 2012.
- [5] Behrouz A. Forouzan, Richard F. Gilberg, 김진 외 7인 공역, "구조적 프로그래밍 기법을 위한 C", 도서출판 인터비전, 2004.
- [6] Brian W. Kernighan, Dennis M. Ritchie, 김석환 외 2인 공역, "The C Programming Language", 2/E, 대영사, 2004.
- [7] "C reference", cppreference.com, 2024 of viewing the site, <https://en.cppreference.com/w/c>.



이 강의자료는 저작권법에 따라 보호받는 저작물이므로 무단 전제와 무단 복제를 금지하며, 내용의 전부 또는 일부를 이용하려면 반드시 저작권자의 서면 동의를 받아야 합니다.

Copyright © Clickseo.com. All rights reserved.