



# Chapter 02. Data Types and Operators

# 목차

1. Understanding the data types
2. Different kind of data types
3. Basic Operators
4. Bitwise Operators
5. Other Operators

# 학습목표

- Understand the concept of data types.
- Learn about the various data types which is used in C ++.
- Arithmetic, relationship, cook the use of logic, or decrease, the assignment operator.
- Bitwise operators and conditional operators, sizeof operator, cast operator for the study casts.

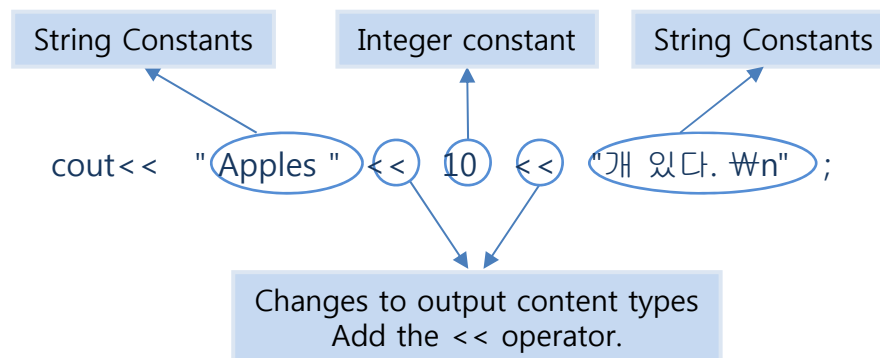
# 01. Understanding the data types

## ■ Concept of data types

- A unique value determined the type of data to create a program, which is called "data type".

## ■ The concept of constants and variables

- Integer constant: the un-change of the a unique value that is referred to as "constant (constant)".



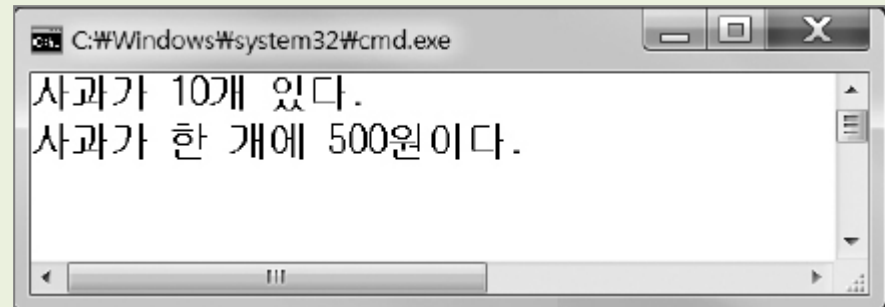
- Integer variables: variable may store integer values in memory space to store the value in the program execution. The variable declaration consists of a data type and the variable name, as follows:

Variable types

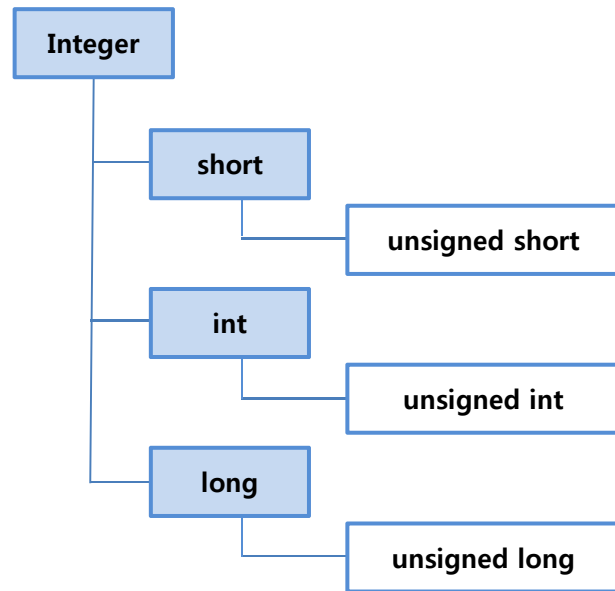
변수 선언 기본 형식

## Example 2-1. To an integer constant output (02\_01.cpp)

```
01 #include <iostream>
02 using namespace std;
03 void main()
04 {
05     cout<<"사과가 "<< 10 <<"개 있다.\n";
06     cout<<"사과가 한 개에 "<< 500 <<"원이다.\n";
07 }
```



# 01. Understanding the data types



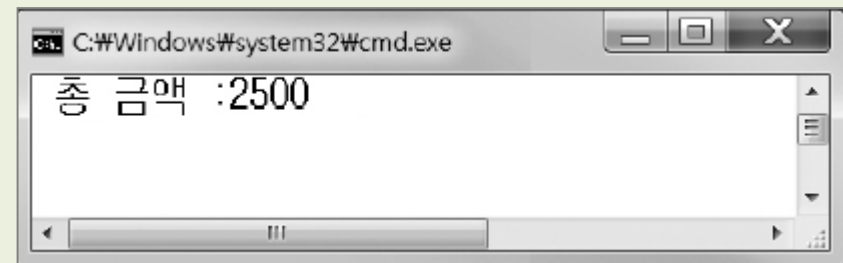
[Picture 2-1] Kind of data type that is used to declare an integer variable

- Identifiers defined by rules

- ① Alphabet (A ~ Z, a ~ z), numbers (0-9), made of a combination of the underscore character (\_).
- ② The first letter must begin with an alphabetic character or \_. We can not begin with a number.
- ③ Even if the identifier is equal to the spelling to be careful because it is case sensitive.
- ④ Reserved words that are used in C ++, can not be used as identifiers.
- ⑤ If the identifier is preferably given a name for the self-serve.

## Example 2-2. Using an integer variable (02\_02.cpp)

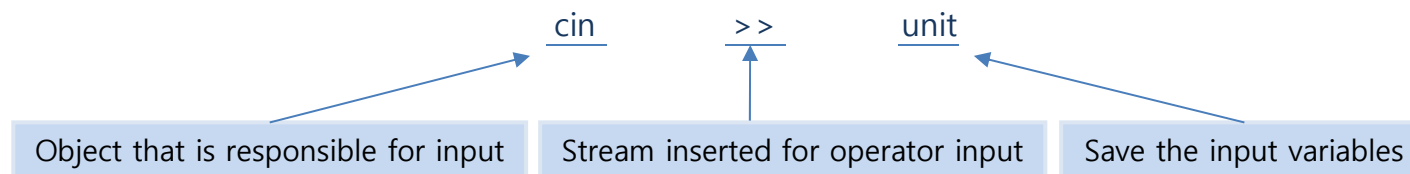
```
01 #include <iostream>
02 using namespace std;
03 void main()
04 {
05     int unit; // 변수 선언
06     int count;
07     int total;
08     unit=500; // 한 개에 500원짜리
09     count=5; // 5개를 구입한
10     total=unit*count; // 총 금액 구하기
11     cout<<"총 금액 : " << total <<"₩n";
12 }
```



# 01. Understanding the data types

## ■ cin enter the responsible object

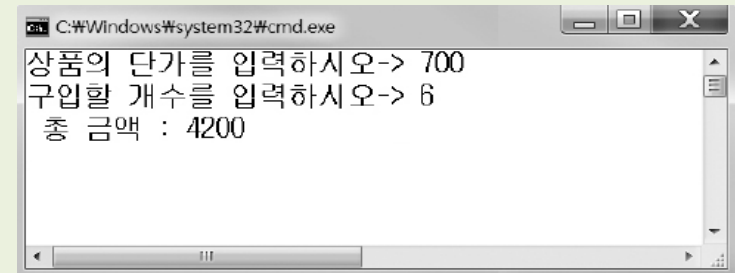
- cin object stores a value received from the keyboard to the variables described in the following operator >>.



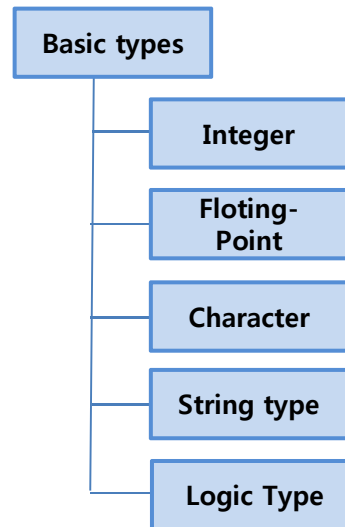


## Example 2-3. Getting used to the input variables sin (02\_03.cpp)

```
01 #include <iostream>
02 using namespace std;
03 void main()
04 {
05     int unit, count, total; // 변수 선언
06     cout<<"상품의 단가를 입력하시오-> ";
07     cin>>unit;
08     cout<<"구입할 개수를 입력하시오-> ";
09     cin>>count;
10     total=unit*count; // 키보드에서 입력받은 데이터로 총금액 구하기
11     cout<<" 총 금액 : " << total <<"\n";
12 }
```



## 02. Different kind of data types



[Picture 2-2] C ++ type of elementary data type

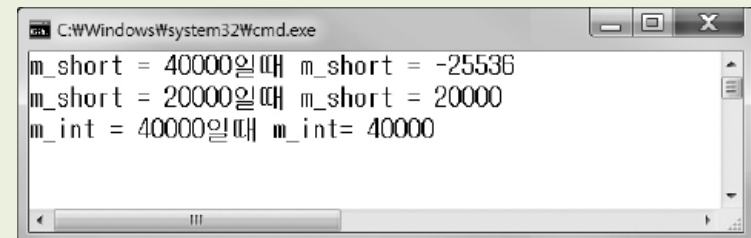
- **Integer** : Classified into short, int, long.

[Table 2-1] Type of integer data types

Types	size	The range of values that can be saved	
short	2byte(16bit)	$-2^{15} \sim 2^{15} - 1$	-32768 ~ 32767
unsigned short	2byte (16bit)	$0 \sim 2^{16} - 1$	0 ~ 65535
int	4byte (32bit)	$-2^{31} \sim 2^{31} - 1$	-2147483648 ~ 2147483647
unsigned int	4byte (32bit)	$0 \sim 2^{32} - 1$	0 ~ 4294967295
long	4byte (32bit)	$-2^{31} \sim 2^{31} - 1$	-2147483648 ~ 2147483647
unsigned long	4byte (32bit)	$0 \sim 2^{32} - 1$	0 ~ 4294967295

## Example 2-4. Integer overflow error occurs, look at the example (02\_04.cpp)

```
01 #include <iostream>
02 using namespace std;
03 void main()
04 {
05     short m_short = 40000;
06     cout << "m_short = 40000일때 m_short = " << m_short << "\n";
07     m_short = 20000;
08     cout << "m_short = 20000일때 m_short = " << m_short << "\n";
09     int m_int = 40000;
10     cout << "m_int = 40000일때 m_int= " << m_int << "\n";
11 }
```



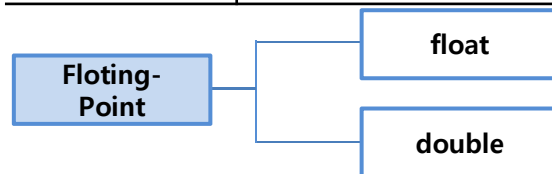
```
C:\Windows\system32\cmd.exe
m_short = 40000일때 m_short = -25536
m_short = 20000일때 m_short = 20000
m_int = 40000일때 m_int= 40000
```

## 02. Different kind of data types

- **Floting-Point:** It means a number with a decimal point.

[Table 2-2] Examples of real constant

Type of constant	Yes	meaning
Point type	1234.5 or 0.0000987	The most commonly used floating point data
Exponential	1.2345E3 or 0.987E-5	Preceded by hydrolysis unit, based on the letters A, also followed by a technical factor



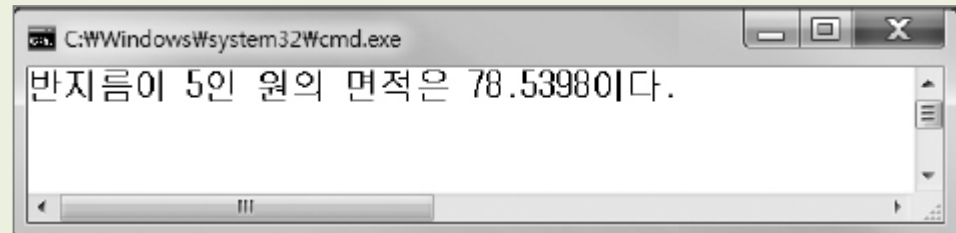
[Picture 2-3] Floting-Point type

[Table 2-3] Floting-Point type

type	size	Store range
float	4byte	$\pm 3.4 \times 10^{-38} \sim \pm 3.4 \times 10^3$
double	8byte	$\pm 1.7 \times 10^{-308} \sim \pm 1.7 \times 10^{30}$

## Example 2-6. Find the area of a circle with a floating point (02\_06.cpp)

```
01 #include <iostream>
02 using namespace std;
03 void main()
04 {
05     double pi=3.141592;
06     int r=5;
07     double area;
08
09     area=r*r*pi;
10
11     cout << "반지름이 "<<r<<"인 원의 면적은 "<<area<<"이다.\n";
12 }
```



## 02. Different kind of data types

### ■ Character

- Character (char) is a data type that represents a single characters, numbers, and special characters, such as characters are represented enclosed in single quotes ( ' ).

[Table 2-5] The effective range of the size and character

Type	Size	Store range
char	1byte (8bit)	-128 ~ 127
unsigned char	1byte (8bit)	0 ~ 255

#### capital letter

'A'	'B'	'C'	'D'	'E'	'F'	...	'X'	'Y'	'Z'
65	66	67	68	69	70	...	88	89	90

#### small letter

'a'	'b'	'c'	'd'	'e'	'f'	...	'x'	'y'	'z'
97	98	99	100	101	102	...	120	121	122

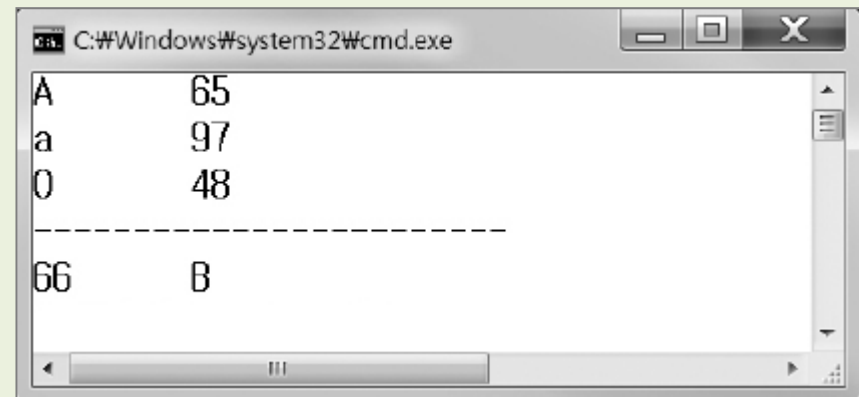
#### Number

'0'	'1'	'2'	'3'	'4'	'5'	'6'	'7'	'8'	'9'
48	49	50	51	52	53	54	55	56	57

[Picture 2-4] ASCII code (ASCII code) value

## Example 2-7. Learn about the relationship between a character constant and ASCII code (02\_07.cpp)

```
01 #include <iostream>
02 using namespace std;
03 void main()
04 {
05     cout << 'A' << " ";
06     cout << (int) 'A' << "\n";
07     cout << 'a' << " ";
08     cout << (int) 'a' << "\n";
09     cout << '0' << " ";
10     cout << (int) '0' << "\n";
11     cout << "-----\n";
12     cout << 'A'+1 << "\t";
13     cout << (char) ('A'+1) << "\n";
14 }
```



## 02. Different kind of data types

### ■ Expand special characters

[Table 2-4] Expand the type of special character W= \

Expand special characters	meaning
Wn	<Enter> 키의 기능을 하며 줄을 바꾼다 (New Line).
Wt	수평 탭으로 일정한 간격을 띄운다 (horizontal tab).
Wb	백스페이스 기능으로 뒤로 한 칸 후진한다 (backspace).
Wr	동일한 줄의 맨 앞으로 커서만 옮긴다 (carriage return).
Wf	출력 용지를 한 페이지 넘긴다 (form feed).
Wa	경고음을 낸다 (alert).
W \	\ 문자를 출력한다 (back slash).
W'	' 문자를 출력한다 (single quote).
W"	" 문자를 출력한다 (double quote).
W0	널 (Null) 문자다.



## 02. Different kind of data types

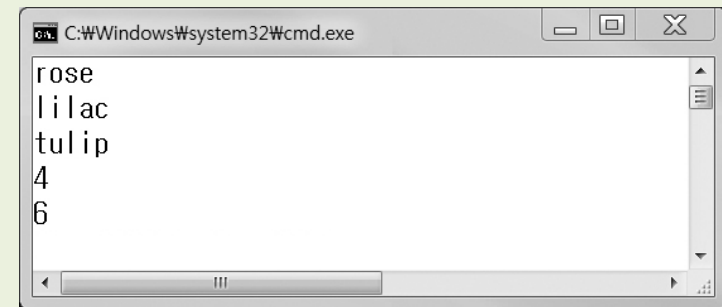
---

### ■ String type

- String should be constant and consists of a set of a series of statements enclosed in double quotes ( "").
- To use a string that represents the end of a null (NULL) characters, words should be added as the last character \0.

## Example 2-10. Save string (02\_10.cpp)

```
01 #include <iostream>
02 #include <cstring> // strlen() 함수를 사용하기 위한 헤더파일
03 using namespace std;
04 void main()
05 {
06     char flowers1[5] = "rose";           // 문자 배열 (문자열)
07     char flowers2[6] = {'l', 'i', 'l', 'a', 'c'}; // 문자 배열 (문자열)
08     char flowers3[] = "tulip";          // 문자 배열 (문자열)
09
10     cout << flowers1 << "\n";
11     cout << flowers2 << "\n";
12     cout << flowers3 << "\n";
13
14     cout << strlen(flowers1) << "\n"; // 문자열의 길이 (NULL 문자 제외)
15     cout << sizeof(flowers3) << "\n"; // 배열의 크기 (NULL 문자 포함)
16 }
```



## 02. Different kind of data types

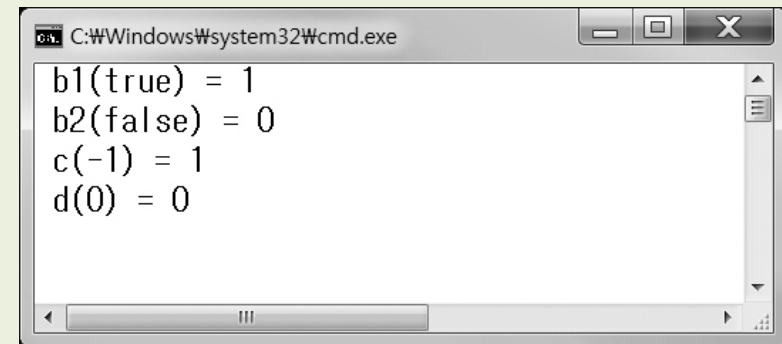
---

### ■ Logic Type

- Unlike C, C ++ language has a logical type that can represent the true and false with the true and false (bool).

## Example 2-11. Look at the form of logical storage type (02\_11.cpp)

```
01 #include <iostream>
02 using namespace std;
03 void main()
04 {
05     bool b1 = true, b2 = false;
06     cout << " b1(true) = " << b1 << "\n";
07     cout << " b2(false) = " << b2 << "\n";
08     bool c = -1, d = 0; // 논리형에서는 0이 아니면 true, 0이면 false 이며, 묵시적으로 true가 1로, false가 0으로 변환됨
09     cout << " c(-1) = " << c << "\n";
10     cout << " d(0) = " << d << "\n";
11 }
```



The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The output of the program is displayed as follows:

```
b1(true) = 1
b2(false) = 0
c(-1) = 1
d(0) = 0
```

# 03. Basic Operators

---

## ■ Operator

- Operator is the symbol for operating on the data of the operand (operand), such as variables and constants.
  - Basic operators of C ++
    - Arithmetic
    - relation
    - logic
    - increase
    - Substituting
  - other operators
    - bit
    - Condition
    - sizeof
    - Cast

## 03. Basic Operators

### ■ Arithmetic operators

- Arithmetic operators is the operator for addition, subtraction, multiplication, and division of the operands.

[Table 2–6] Type of arithmetic

division	Operator	meaning	Mathematical expression	C++ expression	C++ result
Unary operator	+	Positive	+3	+3	3
	-	negative	-2(-2)	-2(-2)	4
Binary operator	+	addition	3+2	3+2	5
	-	subtraction	10-2	10-2	8
	*	multiplication	xy or x×y	2*4	8
	/	Division	x/y or x÷y	8/2	4
	%	remainder	x mod y	9%2	1

### ■ Arithmetic operators Precautions

- ① The resulting value may be changed by the data type of the operands.
- ② % may be taken as a operands operator, one of the binary operators

## 03. Basic Operators

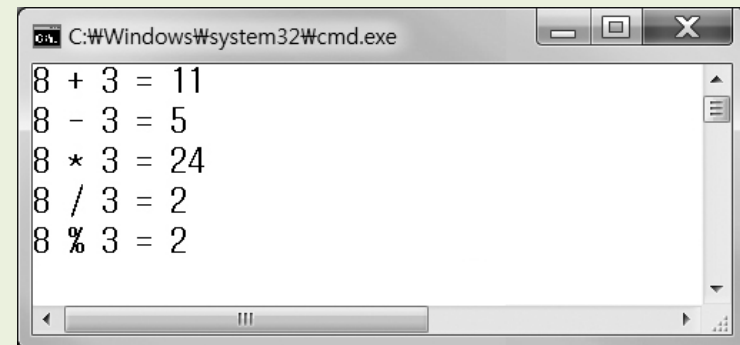
---

### ■ Priority of arithmetic operators

- ① The operator in parentheses are calculated first. If you have multiple dogs are nested parentheses, the innermost parentheses are evaluated first.
- ② \* (Multiplication), / (division), % (the rest) is calculated by the operator. In the formula \*, /, %, etc.
- ③ If there are multiple operators, the operation is performed from left to right. This is equal to the priority of the operator between the three.
- ④ + (Addition) - (subtraction) operator is calculated to the next. In Formula +, - if there are multiple operators are running operations from left to right. This is equal to the priority of the operator between the two.

## Example 2-12. Using the arithmetic operators (02\_12.cpp)

```
01 #include <iostream>
02 using namespace std;
03 void main()
04 {
05     int a = 8, b = 3;
06     cout<<a<<" + "<<b<<" = "<<a+b<<"\n";
07     cout<<a<<" - "<<b<<" = "<<a-b<<"\n";
08     cout<<a<<" * "<<b<<" = "<<a*b<<"\n";
09     cout<<a<<" / "<<b<<" = "<<a/b<<"\n";
10     cout<<a<<" % "<<b<<" = "<<a%b<<"\n";
11 }
```



A screenshot of a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The window displays the output of the C++ program, showing five arithmetic operations with their results:

```
8 + 3 = 11
8 - 3 = 5
8 * 3 = 24
8 / 3 = 2
8 % 3 = 2
```



## 03. Basic Operators

### ■ Relational operators

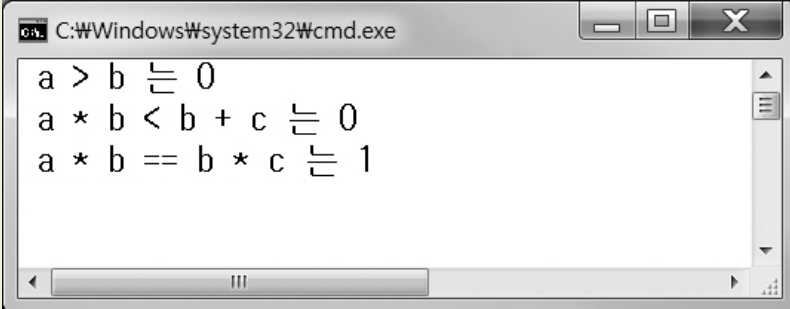
- Relational operators (relational operator) is the result of the operation to the operator to compare the two numbers is the always true (true) or false (false).

[Table 2-7] Types of relational operators

division	Operator	meaning	Mathematical expression	C++ expression	C++ result
Equality	==	Left as the right.	=	2 == 4	false
	!=	The left does not seem right.	≠	2 != 4	true
Comparison Operators	>	Left is greater than right.	>	3 > 2	true
	>=	The left is greater than or equal to the right.	≥	2 >= 3	false
	<	Left is less than right.	<	4 < 5	true
	<=	The right or the left is less than equal.	≤	4 <= 4	true

## Example 2-14. Using the relational operators (02\_14.cpp)

```
01 #include <iostream>
02 using namespace std;
03 void main()
04 {
05     int a = 3, b = 5, c = 3;
06     bool istrue;           // 논리형 변수
07     istrue = a > b;
08     cout<<" a > b 는 " << istrue << "\n";
09
10     istrue = a * b < b + c ;
11     cout<<" a * b < b + c 는 " << istrue << "\n";
12
13     istrue = a * b == b * c ;
14     cout<<" a * b == b * c 는 " << istrue << "\n";
15 }
```



The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The output of the program is displayed as follows:

```
a > b 는 0
a * b < b + c 는 0
a * b == b * c 는 1
```

## 03. Basic Operators

### ■ Logical operators

- Logical operators are used to distinguish cases that satisfy only some or all of the cases that satisfy a set of conditions.

[Table 2-8] Type of logical operator

Types	Explanation
&&	논리곱 연산자 (logical AND operator)
	논리합 연산자 (logical OR operator)
!	단항 논리부정 연산자 (logical NOT operator)

[The truth table of the logical operators && and ||

값		&&	
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

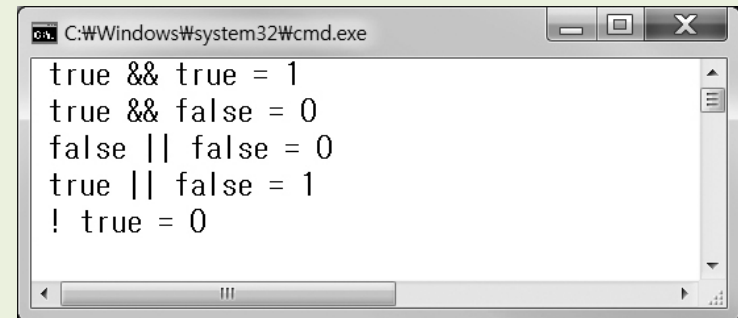
[The truth table of the logical operators]

값	1
0	1
1	0

[Picture 2-5] The truth table for the logic operations

## Example 2-15. To use logical operators in a logical value (02\_15.cpp)

```
01 #include <iostream>
02 using namespace std;
03 void main()
04 {
05     bool istrue;
06
07     istrue = true && true;
08     cout<< " true && true = " << istrue << "\n";
09
10     istrue = true && false;
11     cout<< " true && false = " << istrue << "\n";
12
13     istrue = false || false;
14     cout<< " false || false = " << istrue << "\n";
15
16     istrue = true || false;
17     cout<< " true || false = " << istrue << "\n";
18
19     istrue = ! true;
20     cout<< " ! true = " << istrue << "\n";
21 }
```



A screenshot of a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The window displays the output of the C++ program, showing five lines of results for logical operations: true && true = 1, true && false = 0, false || false = 0, true || false = 1, and ! true = 0.

```
true && true = 1
true && false = 0
false || false = 0
true || false = 1
! true = 0
```

## 03. Basic Operators

### ■ Increase or decrease operator

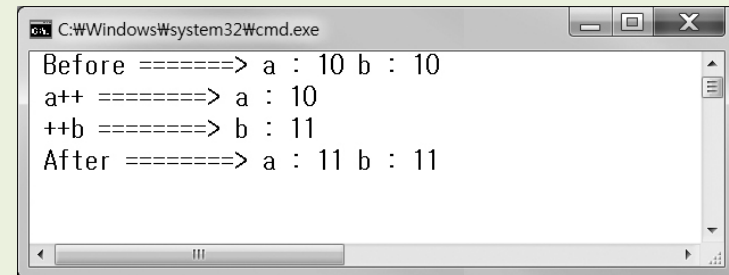
- When writing a program it happened that Variable Value increase 1 or reduced one. The operator using the operator of increase or decrease.

[Table 2-9] Type of increase or decrease operator

Operator	Form	designation
++	++a	Pre-incrementation operator
	a++	Post-incrementation operator
--	--a	Pre-decrementation operator
	a--	Post-decrementation operator

## Example 2-17. Use increase and decrease operator (02\_17.cpp)

```
01 #include <iostream>
02 using namespace std;
03 void main()
04 {
05     int a=10, b=10;
06     cout<<" Before =====> a : "<<a<<" b : "<<b<<"\n";
07     cout<<" a++ =====> a : "<<a++<<"\n";
08     cout<<" ++b =====> b : "<<++b<<"\n";
09     cout<<" After =====> a : "<<a<<" b : "<<b<<"\n";
10 }
```



The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The output of the program is displayed as follows:

```
Before =====> a : 10 b : 10
a++ =====> a : 10
++b =====> b : 11
After =====> a : 11 b : 11
```

## 03. Basic Operators

### ■ The assignment operator

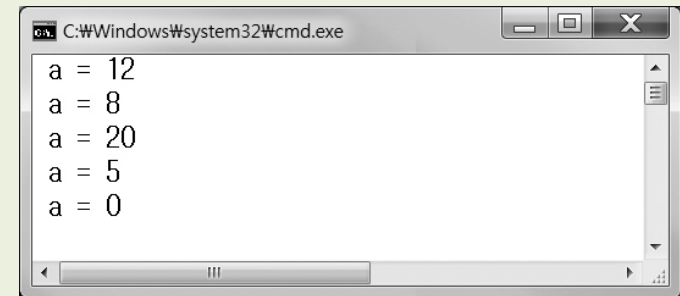
- The assignment is commonly used to save the equation by substituting the result of a technical assignment to a variable, the right to left assignment operator.

[Table 2–10] Type of assignment

Operator	Example	The formula in the same sense
<code>+=</code>	<code>x+=y</code>	<code>x=x+y</code>
<code>-=</code>	<code>x-=y</code>	<code>x=x-y</code>
<code>*=</code>	<code>x*=y</code>	<code>x=x*y</code>
<code>/=</code>	<code>x/=y</code>	<code>x=x/y</code>
<code>%=</code>	<code>x%=y</code>	<code>x=x%y</code>
<code>&amp;=</code>	<code>x&amp;=y</code>	<code>x=x&amp;y</code>
<code> =</code>	<code>x =y</code>	<code>x=x y</code>
<code>^=</code>	<code>x^=y</code>	<code>x=x^y</code>
<code>&gt;&gt;=</code>	<code>x&gt;&gt;=y</code>	<code>x=x&gt;&gt;y</code>
<code>&lt;&lt;=</code>	<code>x&lt;&lt;=y</code>	<code>x=x&lt;&lt;y</code>

## Example 2-18. Using the assignment of various types (02\_18.cpp)

```
01 #include<iostream>
02 using namespace std;
03 void main()
04 {
05     int a , b = 2;
06
07     a = 10; a += b; cout << " a = " << a<< "\n";
08     a = 10; a -= b; cout << " a = " << a<< "\n";
09     a = 10; a *= b; cout << " a = " << a<< "\n";
10     a = 10; a /= b; cout << " a = " << a << "\n";
11     a = 10; a %= b; cout << " a = " << a << "\n";
12 }
```





# 04 Bitwise Operators

- Bitwise operator used to operate directly the bits of the bit operators.

[Table 2-11] Type of operator bit

Operator	designation	meaning
&	비트 논리곱 (AND)	If both bits are 1, the resulting bit is 1.
	비트 논리합 (OR)	If at least one bit of the two bits are 1, the resulting bit is 1.
^	비트 배타적 논리합 (XOR)	When the two bits are different, the resulting bit is 1 bit.
<<	왼쪽 이동 (left shift)	It moves to the left side (shift).
>>	오른쪽 이동 (right shift)	It moves to the right (shift).
~	1의 보수 (one's complement)	all bits are 0 and 1 is 1.

# 04 Bitwise Operators

## ■ bit AND, bit OR, bit XOR, bitwise NOT

[Table 2-12] bit AND, bit OR, XOR result of bit

bit operands		Bit AND	Bit OR	Bit XOR
X	Y	(X&Y)	(X Y)	(X^Y)
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

[Table 2-13] 1Conservative (NOT) operation results

X bit operands	Bit NOT(~X)
0	1
1	0

# 04 Bitwise Operators

## ■ bit computing process

- short x = 10, y = 6;
- First, you'll need to convert the decimal to binary for bit operations, converting 10 into binary
  - This is  $10 = 8 + 2 = 2^3 + 2^1$ . And because the short-type display in a 2-byte (16 bits) as follows: 1 to  $2^3$  and  $2^1$  spot, and the rest is filled with zero.

10	$2^{15}$	$2^{14}$	$2^{13}$	$2^{12}$	$2^{11}$	$2^{10}$	$2^9$	2	2	$2^6$	2	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$
	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0

- And converting the respective 6 in a binary number is a  $6 = 4 + 2 = 2^2 + 2^1$  display 1 in the first, the  $2^2$   $2^1$  digit place, and the rest is filled with zero.

6	$2^{15}$	$2^{14}$	$2^{13}$	$2^{12}$	$2^{11}$	$2^{10}$	$2^9$	2	2	$2^6$	2	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$
	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0

# 04 Bitwise Operators

## ■ bit AND operation

- If the binary number 10 and 6 to perform a bitwise AND operation (&) is converted to the bit are both 1, the one corresponding to the following, if not zero. One of only two wins digit is 1, so  $2^1 = 2$ .

10	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
&	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

## ■ bit OR operation

- The binary number 10 and 6 bit OR operation on the converted into (|) Performing a one or more 1, the first of the two bits corresponding to the following, if both a 0 0. 23 wins digit, the digit of 22 wins, 1 win, because it is the one place in  $2 \cdot 2^3 + 2^2 + 2^1 = 8 + 4 + 2 = 14$ .

10	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0

# 04 Bitwise Operators

## ■ bit XOR operation

- When performing a binary 10 bit exclusive-OR operation on the 6 (^) is converted to the bit is 1 and if both 0 or both 1 0 result 0, corresponding to the following: 23 wins digit, so the square is one of the two digits is the  $2^3 + 2^2 = 8 + 4 = 12$ .

10	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0
6	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
^	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0

# 04 Bitwise Operators

## ■ NOT bit operator

- Performing a bitwise NOT operation ( $\sim$ ) in 10 has been converted to a binary number as follows.

10	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0
$\sim$	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	1

- Left shift filled with the sign bit is 1, as above, and a negative number for negative size is stored in the form of a two-complement.

### ① ~ The result of the operation

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

### ② ~ 1 for the maintenance of the operation result

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

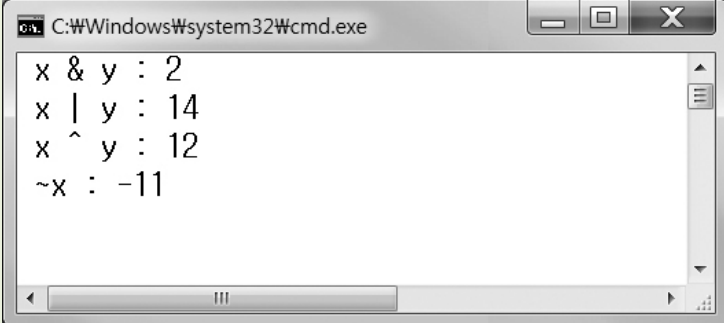
### ③ 2's complement of the result of the + 1 = 1 Maintenance

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

- finally,  $\sim 10_{10} = -11_{10}$

## Example 2-19. Using the Bitwise Operators (02\_19.cpp)

```
01 #include<iostream>
02 using namespace std;
03 void main()
04 {
05     short x = 10, y = 6 ;
06     cout << " x & y : " << (x & y) << "\n";
07     cout << " x | y : " << (x | y) << "\n";
08     cout << " x ^ y : " << (x ^ y) << "\n";
09     cout << " ~x : " << (~x) << "\n";
10 }
```



A screenshot of a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The window displays the output of the C++ program, showing the results of bitwise operations on x=10 and y=6. The output is as follows:

```
x & y : 2
x | y : 14
x ^ y : 12
~x : -11
```

# 04 Bitwise Operators

## ■ Shift operator

- Shift operator is operator to move the binary bit by bit. The shift operator moves to convert the operands to binary coming on the left operand by the number coming to the right.

[Table 2-14] Type of shift operators

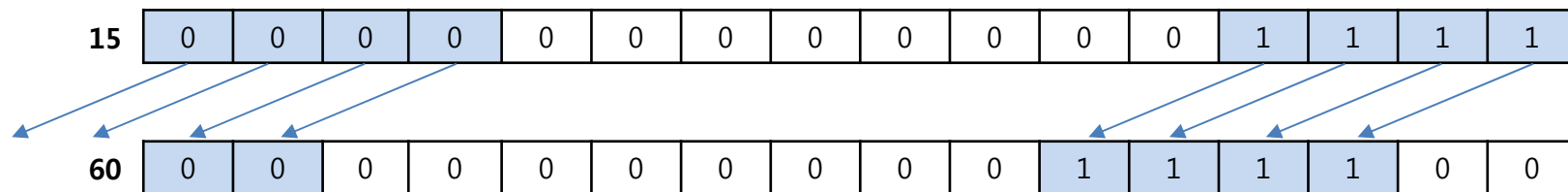
Operator	designation	Example	Meaning
<<	Left shift operator	a << 1	The value of a is moved to the left by one bit.
>>	Right shift	a >> 1	The value of a is moved to the right by one bit.



# 04 Bitwise Operators

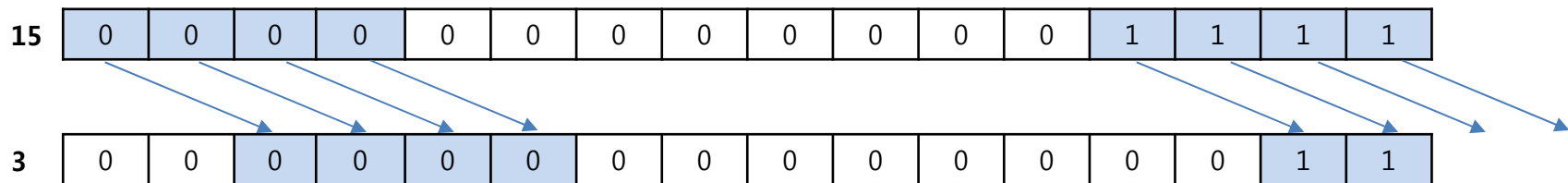
## ■ Left shift operator

- If the left shift operation is stored in a variable  $x$  to 15 ( $<<$ ) application, such as  $x << 2$  moves the 15 stored in the variable  $x$ , as follows: two bits in the left. And it left shift operator will fill the empty space to the right of zero, and if there is no space on the left abandoned. Therefore, the result is a 60 ( $15 * 2^2$ ).



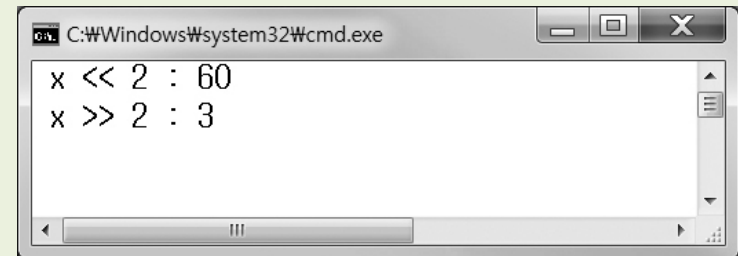
## ■ Right shift

- When the right-shift operation ( $>>$ ) application, such as  $2 x >> 15$  stored in the variable  $x$  stored in the 15 to move the variable  $x$ , as shown in the following figure to the right 2 bits. And where it is left blank, the first bit is 0, the amount of sleep, fill in negative, one, if there is no space on the right discarded.



## Example 2-20. Using the shift operators (02\_20.cpp)

```
01 #include <iostream>
02 using namespace std;
03 void main()
04 {
05     short int x = 15;
06
07     cout << "x << 2 : " << (x << 2) << "\n";
08     cout << "x >> 2 : " << (x >> 2) << "\n";
09 }
```



## 05 Other operators

### ■ Conditional operator

- The condition operator (conditional operator) is the ternary operator (ternary operator) with an operator and operand 3 to perform pick a sentence depending on the result of the condition is true or false. The result is true, the "formula 1" of the condition, and if false, perform the 'formula 2'.

The condition? Formula 1: Formula 2;

조건 연산자 기본 형식

### ■ sizeof Operator

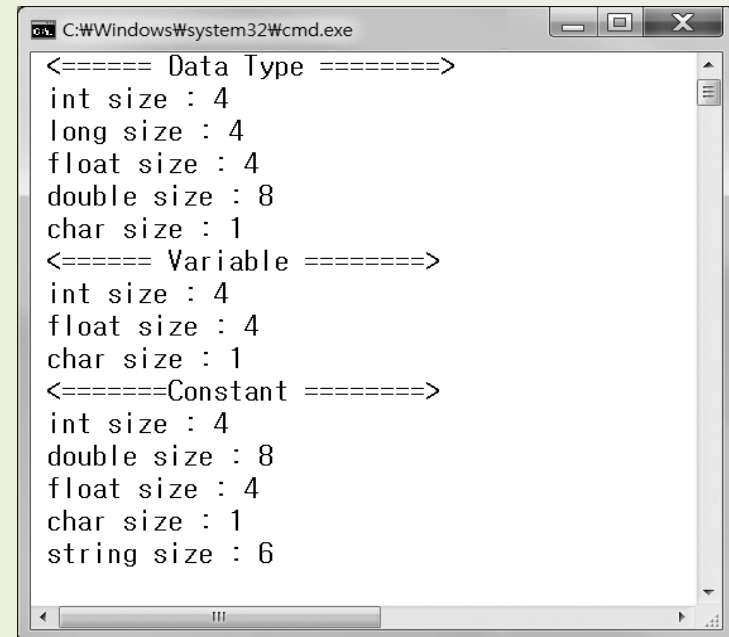
- sizeof operator variable names or types, or the operator for calculating the number of bytes occupied by the memory to apply to the constant value. And return value as the result of the operation is an integer.

sizeof (data type or variable or constant)

sizeof 연산자 기본 형식

## Example 2-22. sizeof operator to obtain the amount of memory (02\_22.cpp)

```
01 #include <iostream>
02 using namespace std;
03 void main()
04 {
05     int a=10;
06     float b=3.5f;
07     char c='A';
08     cout<<"\n <===== Data Type =====>";
09     cout<<"\n int size : "<<sizeof(int);
10     cout<<"\n long size : "<<sizeof(long);
11     cout<<"\n float size : "<<sizeof(float);
12     cout<<"\n double size : "<<sizeof(double);
13     cout<<"\n char size : "<<sizeof(char);
14
15     cout<<"\n <===== Variable =====>";
16     cout<<"\n int size : "<<sizeof(a);
17     cout<<"\n float size : "<<sizeof(b);
18     cout<<"\n char size : "<<sizeof(c);
19
20     cout<<"\n <===== Constant =====>";
21     cout<<"\n int size : "<<sizeof(23);
22     cout<<"\n double size : "<<sizeof(3.5);
23     cout<<"\n float size : "<<sizeof(3.5f);
24     cout<<"\n char size : "<<sizeof('A');
25     cout<<"\n string size : "<<sizeof("Apple")<<"\n";
26 }
```



```
C:\Windows\system32\cmd.exe
<===== Data Type =====>
int size : 4
long size : 4
float size : 4
double size : 8
char size : 1
<===== Variable =====>
int size : 4
float size : 4
char size : 1
<=====Constant =====>
int size : 4
double size : 8
float size : 4
char size : 1
string size : 6
```

## 05 Other operators

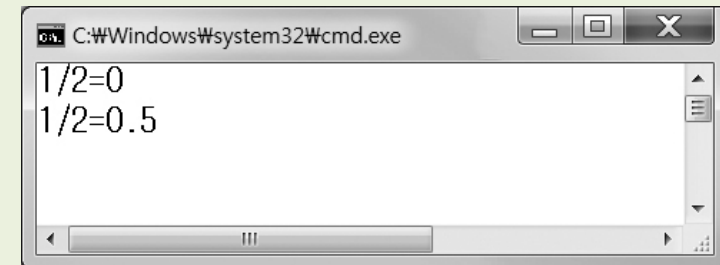
### ■ The cast operator cast

- Cast to change the data type is called (type conversion or type cast). If you convert the data types from a large to a small value types should note that the value may be lost

The cast operator default format	Use the cast operator
(Types) Expressions	<pre>int num01 = (int) 3.5; double num02 = (double) 5;</pre>

## Example 2-24. To coercion in order to obtain the desired result (02\_24.cpp)

```
01 #include <iostream>
02 using namespace std;
03 void main()
04 {
05     int c=1;
06     int d=2;
07     double res02;
08     res02= c / d;
09     cout<<c<<"/"<<d<<"="<<res02<<endl;
10     res02 = (double)c/(double)d;
11     cout<<c<<"/"<<d<<"="<<res02<<endl;
12 }
```



# Homework

---

- Chapter 2 Exercise: 2, 3, 4, 5, 6, 8, 9, 11, 14, 15, 17