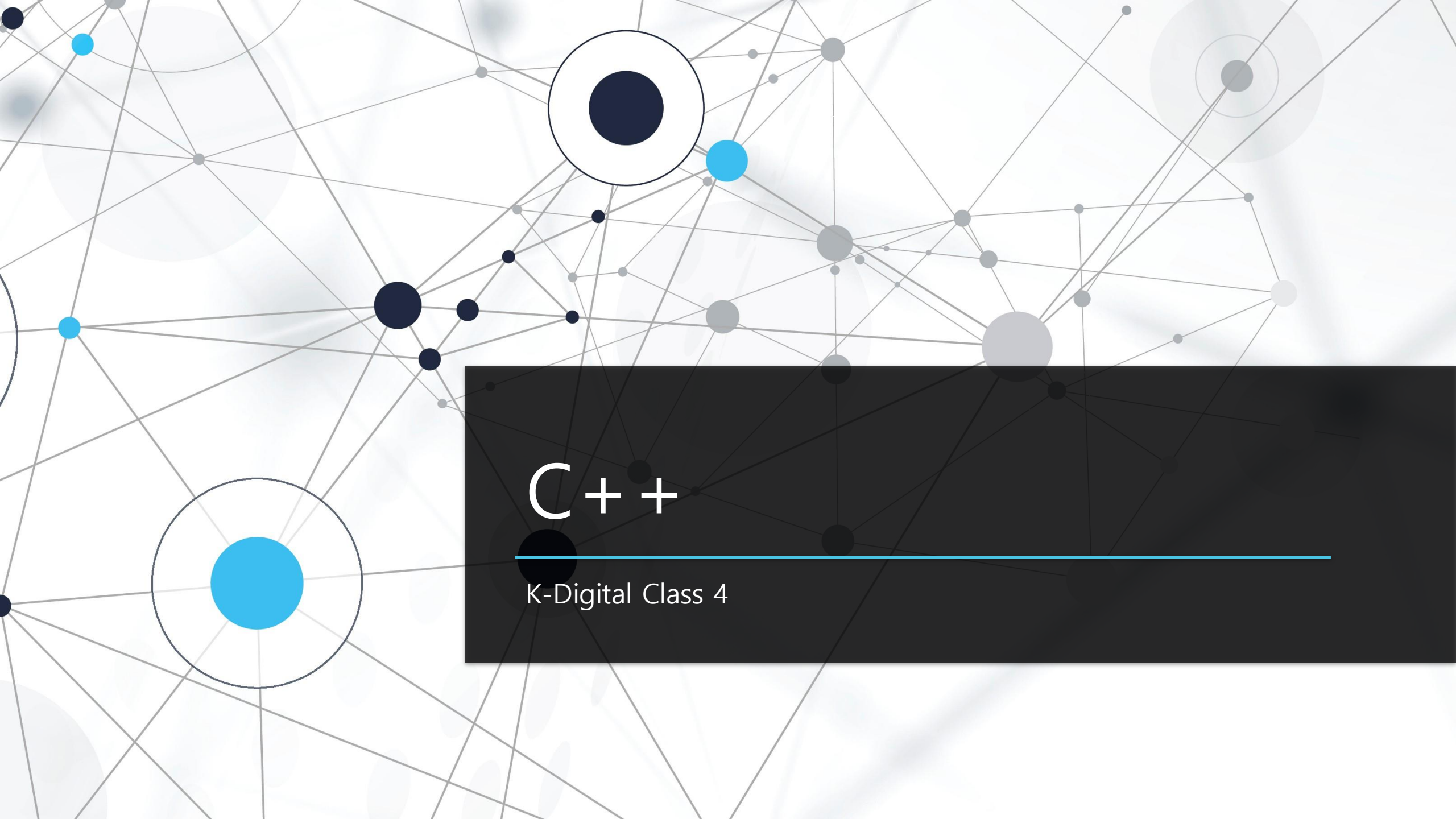


C++

K-Digital Class 4

C++ 변수(VARIABLE)

C++ Constants(상수)

- 다른 사람(또는 자신)이 기존 변수 값을 재정의하지 않도록 하려면 const 키워드를 사용한다.
- 이는 변수를 변경할 수 없고 읽기 전용을 의미하는 “Constant " 로 선언한다.

Example

```
const int myNum = 15;  // myNum will always be 15
myNum = 10;  // error: assignment of read-only variable 'myNum'
```

C++ 변수(VARIABLE)

C++ Constants(상수)

- 변경될 가능성이 없는 값이 있는 경우 항상 변수를 상수로 선언 하는 것이 합리적이다.

Example

```
const int minutesPerHour = 60;
const float PI = 3.14;
```

C++ DATA TYPE

- 변수(Variables)에서 설명한 대로 C++의 변수는 지정된 데이터 유형이어야 한다.

Example

```
int myNum = 5;               // Integer (whole number)
float myFloatNum = 5.99;     // Floating point number
double myDoubleNum = 9.98;   // Floating point number
char myLetter = 'D';         // Character
bool myBoolean = true;       // Boolean
string myText = "Hello";     // String
```

C++ DATA TYPE

Basic Data Types(기본 데이터 유형)

- 데이터 유형은 변수가 저장할 정보의 크기와 유형을 지정한다.

Data Type	Size	Description
int	4 bytes	Stores whole numbers, without decimals
float	4 bytes	Stores fractional numbers, containing one or more decimals. Sufficient for storing 7 decimal digits
double	8 bytes	Stores fractional numbers, containing one or more decimals. Sufficient for storing 15 decimal digits
boolean	1 byte	Stores true or false values
char	1 byte	Stores a single character/letter/number, or ASCII values

C++ DATA TYPE

C++ Numeric Data Types

- 35 또는 1000과 같이 소수 없이 정수를 저장해야 하는 경우 int를 사용하고 9.99 또는 3.14515와 같이 부동 소수점 숫자(소수점 포함)가 필요한 경우 float 또는 double을 사용한다.

int

```
int myNum = 1000;
cout << myNum;
```

float

```
float myNum = 5.75;
cout << myNum;
```

double

```
double myNum = 19.99;
cout << myNum;
```

C++ DATA TYPE

C++ Numeric Data Types

- float vs. double
  - 부동 소수점 값의 정밀도는 소수점 이하 값이 가질 수 있는 자릿수를 나타낸다.
  - Float의 정밀도는 소수점 이하 6자리 또는 7자리에 불과하지만 double 변수는 정밀도가 약 15자리이다.
  - 따라서 대부분의 계산에는 double을 사용하는 것이 더 안전하다.

Scientific Numbers

- 부동 소수점 숫자는 10의 거듭제곱을 나타내는 "e"가 있는 과학적 숫자일 수도 있다.

Example

```
float f1 = 35e3;
double d1 = 12E4;
cout << f1;
cout << d1;
```

C++ DATA TYPE

C++ Boolean Data Types

- Bool data type은 bool keyword로 선언되며 true 또는 false 값만 사용할 수 있다.
- 값이 반환되면 true = 1이고 false = 0이다.

Example

```
bool isCodingFun = true;
bool isFishTasty = false;
cout << isCodingFun;  // Outputs 1 (true)
cout << isFishTasty;  // Outputs 0 (false)
```

C++ DATA TYPE

C++ Character Data Types

- char 데이터 유형은 단일 문자를 저장하는 데 사용된다.
- 문자는 'A' 또는 'c'와 같이 작은따옴표(‘’)로 묶어야 한다.

Example

```
char myGrade = 'B';
cout << myGrade;
```

C++ DATA TYPE

C++ Character Data Types

- ASCII 값을 사용하여 특정 문자를 표시할 수 있다.

Example

```
char a = 65, b = 66, c = 67;
cout << a;
cout << b;
cout << c;
```

C++ DATA TYPE

C++ String Data Types

- String Type은 일련의 문자(텍스트) 열를 저장하는 데 사용된다.
- 이것은 기본 제공 유형이 아니지만 가장 기본적인 사용 유형처럼 작동 한다.
- 문자열 값은 큰따옴표(“”)로 묶어야 한다.

Example

```
string greeting = "Hello";
cout << greeting;
```

C++ DATA TYPE

C++ Modifier Types

- C++ 은 char, int, double data type 앞에 modifier 를 가질 수 있도록 허용한다.
- Modifier 는 base type의 의미를 수정하는데 사용되며, 그래서 다양한 상황에 대한 요구를 더 정확하게 만족시킨다.
- 그 data type modifier 의 리스트는 아래와 같다
  - signed
  - unsigned
  - long
  - short

C++ DATA TYPE

C++ Modifier Types

- signed, unsigned, long, short 는 integer base type 들에 적용될 수 있다. 부가적으로 signed 와 unsigned 는 char 에 적용될 수 있으며, long 은 double 에 적용될 수 있다.
- signed 와 unsigned 는 long 이나 short 앞에 사용될 수도 있다. 예를 들면 unsigned long int 이다.
- C++ 은 unsigned, short, long 의 integer 를 선언하기 위한 단축 표기를 허용한다. 당신은 int 없이 단순히 unsigned, short, long 만을 사용할 수 있다. int 는 내포되어 있다. 예를 들어 다음의 두 문장은 둘 다 unsigned integer 변수를 선언한다.

```
unsigned x;
unsigned int y;
```

C++ DATA TYPE

C++ Modifier Types

```cpp
#include <iostream>

using namespace std;

/* This program shows the difference between
   * signed and unsigned integers.
*/
int main() {
   short int i;           // a signed short integer
   short unsigned int j;  // an unsigned short integer

   j = 50000;

   i = j;
   cout << i << " " << j; cout << endl;

   return 0;
}
```

```
-15536 50000
계속하려면 아무 키나 누르십시오 . . .
```

위의 결과는 50,000을 short unsigned integer로 나타내는 비트 패턴을 short로 -15,536으로 해석하기 때문이다.

C++ DATA TYPE

Type Qualifier in C++(Type 한정)

Qualifier	Meaning
const	const type 의 개체는 프로그램 실행 동안에 변경될 수 없다.
volatile	volatile 은 컴파일러에게 변수의 값이 프로그램에 의해서 명시적으로 지정되지 않은 방식으로 변경될 수 있음을 통보한다.
restrict	restrict 에 의한 pointer qualifier 는 초기에 단지 그것이 가리키는 개체에 의해서만 접근 가능하다는 것을 의미한다. C99 에서만 restrict 라 불리는 새로운 유형의 qualifiler 가 추가되었다

C++ OPERATORS

- Arithmetic operators(산술 연산자)
- Assignment operators(대입 연산자)
- Comparison operators(비교 연산자)
- Logical operators(논리 연산자)
- Bitwise operators(비트 연산자)
- Misc Operators(기타 연산자)

C++ OPERATORS

Arithmetic Operators

Operator	Name	Description	Example
+	Addition	Adds together two values	x + y
-	Subtraction	Subtracts one value from another	x - y
*	Multiplication	Multiplies two values	x * y
/	Division	Divides one value by another	x / y
%	Modulus	Returns the division remainder	x % y
++	Increment	Increases the value of a variable by 1	++x
--	Decrement	Decreases the value of a variable by 1	--x

C++ OPERATORS

Assignment Operators

- 대입 연산자는 변수에 값을 할당하는 데 사용된다.

Example

```
int x = 10;
x += 5;
```

C++ OPERATORS

Assignment Operators

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

C++ OPERATORS

C++ Comparison Operators

- 비교 연산자는 두 값을 비교하는 데 사용된다.
- 비교의 반환 값은 true(1) 또는 false(0)이다.

Example

```
int x = 5;
int y = 3;
cout << (x > y); // returns 1 (true) because 5 is greater than 3
```

C++ OPERATORS

C++ Comparison Operators

Operator	Name	Example
==	Equal to	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

C++ OPERATORS

Logical Operators

Operator	Name	Description	Example
&&	Logical and	Returns true if both statements are true	x < 5 && x < 10
||	Logical or	Returns true if one of the statements is true	x < 5 || x < 4
!	Logical not	Reverse the result, returns false if the result is true	!(x < 5 && x < 10)

C++ OPERATORS

Bitwise operator

- 비트 연산자는 논리 연산자와 비슷하지만, 비트(bit) 단위로 논리 연산을 할 때 사용하는 연산자이다.
- 또한, 비트 단위로 왼쪽이나 오른쪽으로 전체 비트를 이동하거나, 1의 보수를 만들 때도 사용된다.

operator	asm equivalent	description
&	AND	Bitwise AND
|	OR	Bitwise inclusive OR
^	XOR	Bitwise exclusive OR
~	NOT	Unary complement (bit inversion)
<<	SHL	Shift bits left
>>	SHR	Shift bits right

C++ OPERATORS

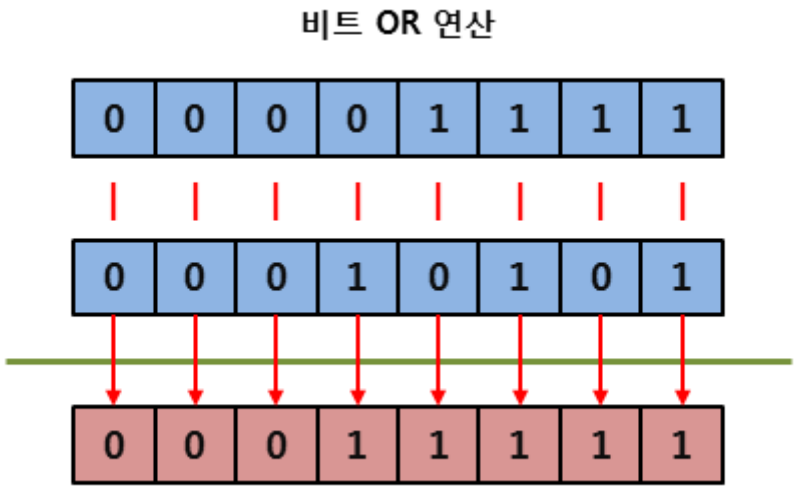
Bitwise operator

- 그림은 비트 AND 연산자(&)의 동작을 나타낸다.
- 이처럼 비트 AND 연산자는 대응되는 두 비트가 모두 1일 때만 1을 반환하며, 다른 경우는 모두 0을 반환한다.

C++ OPERATORS

Bitwise operator

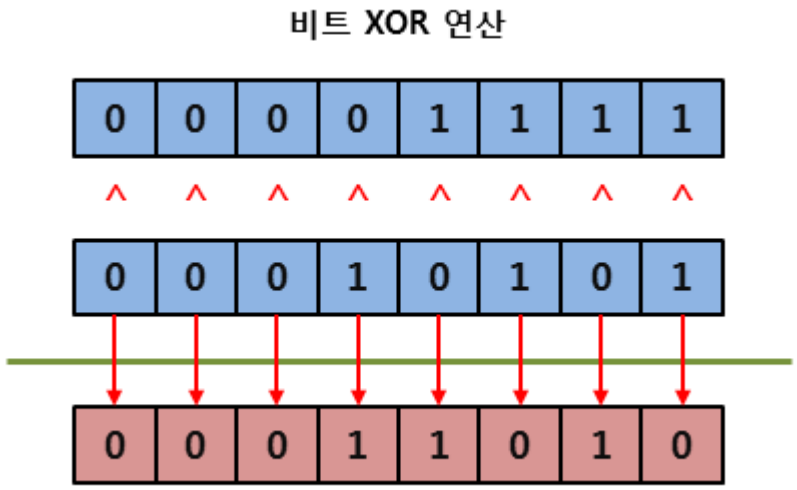
- 그림은 비트 OR 연산자(|)의 동작을 나타낸다.
- 이처럼 비트 OR 연산자는 대응되는 두 비트 중 하나라도 1이면 1을 반환하며, 두 비트가 모두 0일 때만 0을 반환한다.



C++ OPERATORS

Bitwise operator

- 그림은 비트 XOR 연산자(^)의 동작을 나타낸다.
- 이처럼 비트 XOR 연산자는 대응되는 두 비트가 서로 다르면 1을 반환하고, 서로 같으면 0을 반환한다.



C++ OPERATORS

Bitwise operator

- 그림은 비트 NOT 연산자(~)의 동작을 나타낸다.
- 이처럼 비트 NOT 연산자는 해당 비트가 1이면 0을 반환하고, 0이면 1을 반환한다.