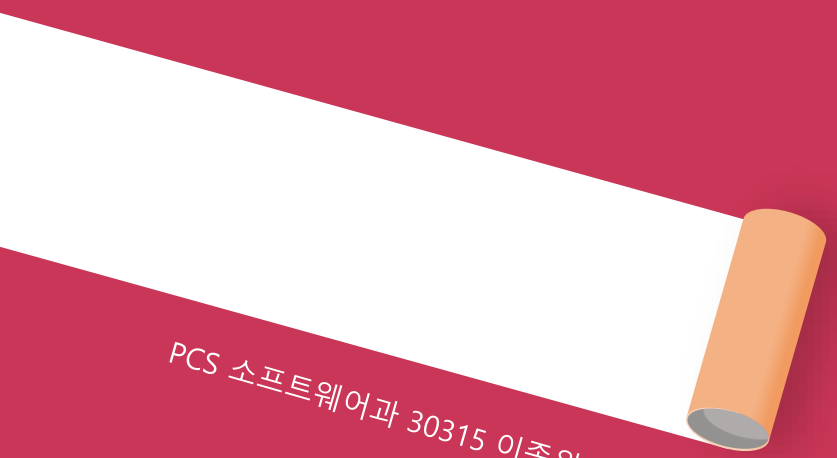
C, Operator

Visual Studio 2019 / C언어 - 연산자



PCS 소프트웨어과 30315 이종원

C , Operator

C언어 - 연산자

Operation은 종류가 다양 하다

- **Operation**
  - **산술 연산자** - +, -, *, /, %
  - **관계 연산자** - ==, !=, <, >, <=, >=
  - **비트 연산자** - &, |, ^, ~, >>, <<
  - **논리 연산자** - &&, ||, !
  - **대입 연산자** - =, +=, -=, *=, /=, %=

C, Arithmetic Operation

C언어 – 대입 연산자

대표적인 대입 연산자

```
#include <stdio.h>

int main() {
  int a=5+10;
}
```

출력결과 : 15

```
#include <stdio.h>

int main() {
  int a=5-10;
}
```

출력결과 : -5

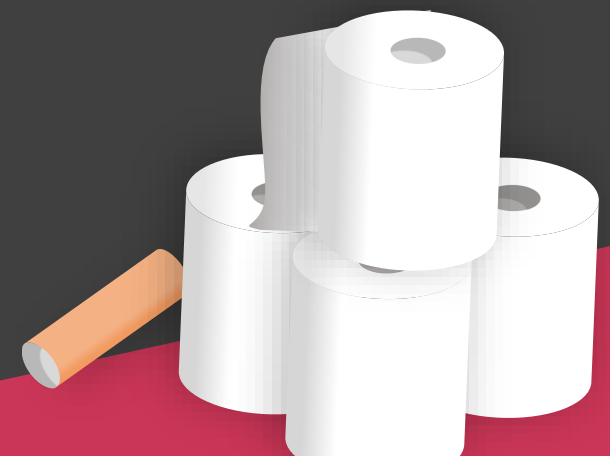
```
#include <stdio.h>

int main() {
  int a=5*10;
}
```

출력결과 : 50

```
#include <stdio.h>

int main() {
  int a=5/10;
}
```

출력결과 : 0

ex) sum = x + y;



C, Relational Operation

C언어 – 관계 연산자

관계 연산자

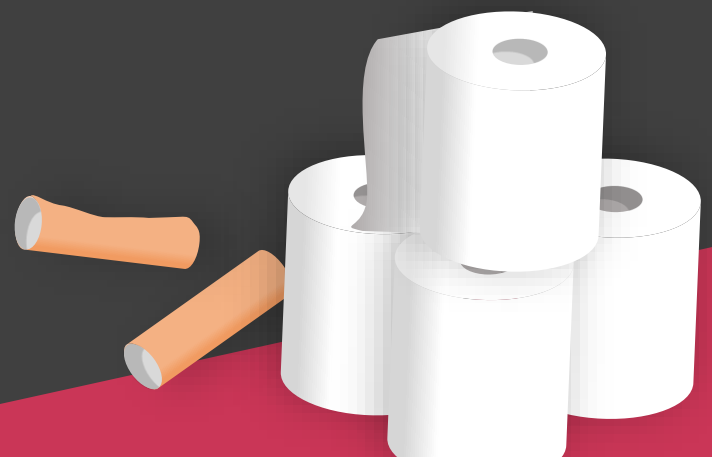
```
#include <stdio.h>

int main() {
  int a=1;
  int b=5;

  a==b;
}
```

출력결과 : 0

```
#include <stdio.h>

int main() {
  int a=1;
  int b=5;

  a!=b;
}
```

출력결과 : 1

1 : Ture, 0 : False



C, Relational Operation

C언어 – 관계 연산자

관계 연산자

```
#include <stdio.h>

int main() {
  int a=1;
  int b=5;

  printf("%d", a>b);
}
```

출력결과 : 0(false)

```
#include <stdio.h>

int main() {
  int a=1;
  int b=5;

  printf("%d", a<b);
}
```

출력결과 : 1(ture)

```
#include <stdio.h>

int main() {
  int a=1;
  int b=5;

  printf("%d", a>=b);
}
```

출력결과 : 0(false)

```
#include <stdio.h>

int main() {
  int a=1;
  int b=5;

  printf("%d", a<=b);
}
```

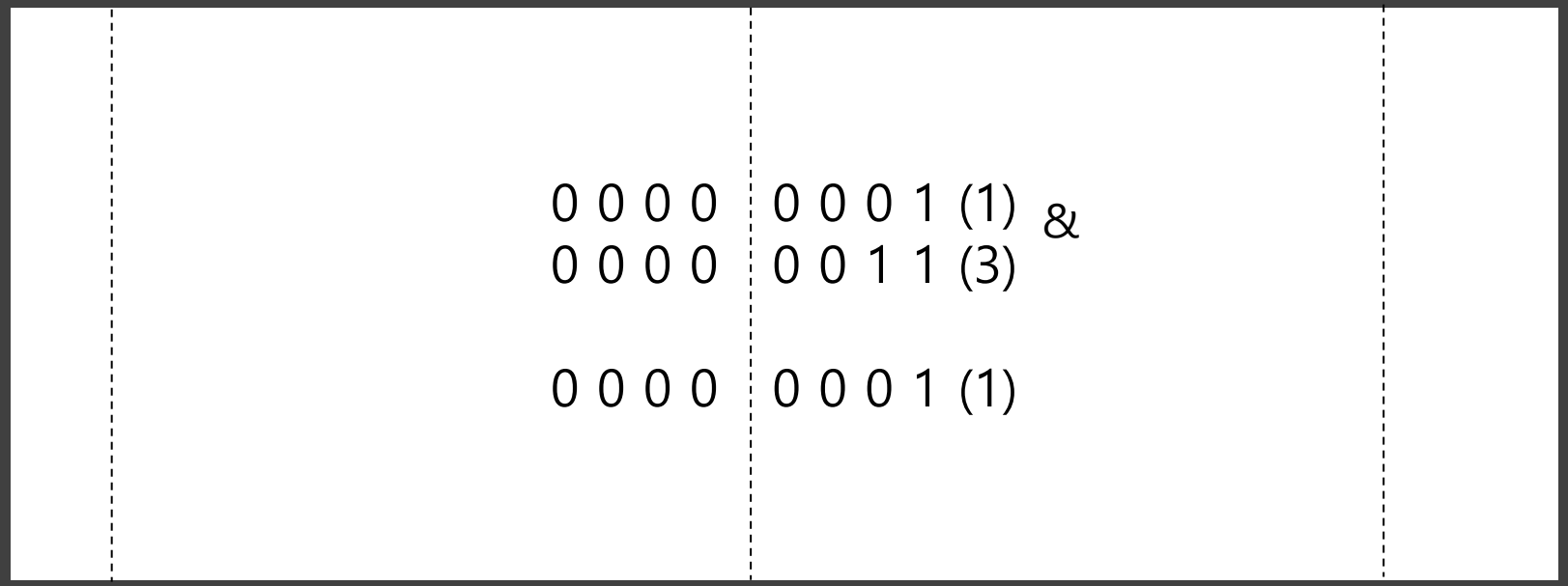
출력결과 : 1(ture)

< : less then, > : greater than

C , Bit-Wize Operator

C언어 – 비트 연산자

&(AND) 비트 단위 연산

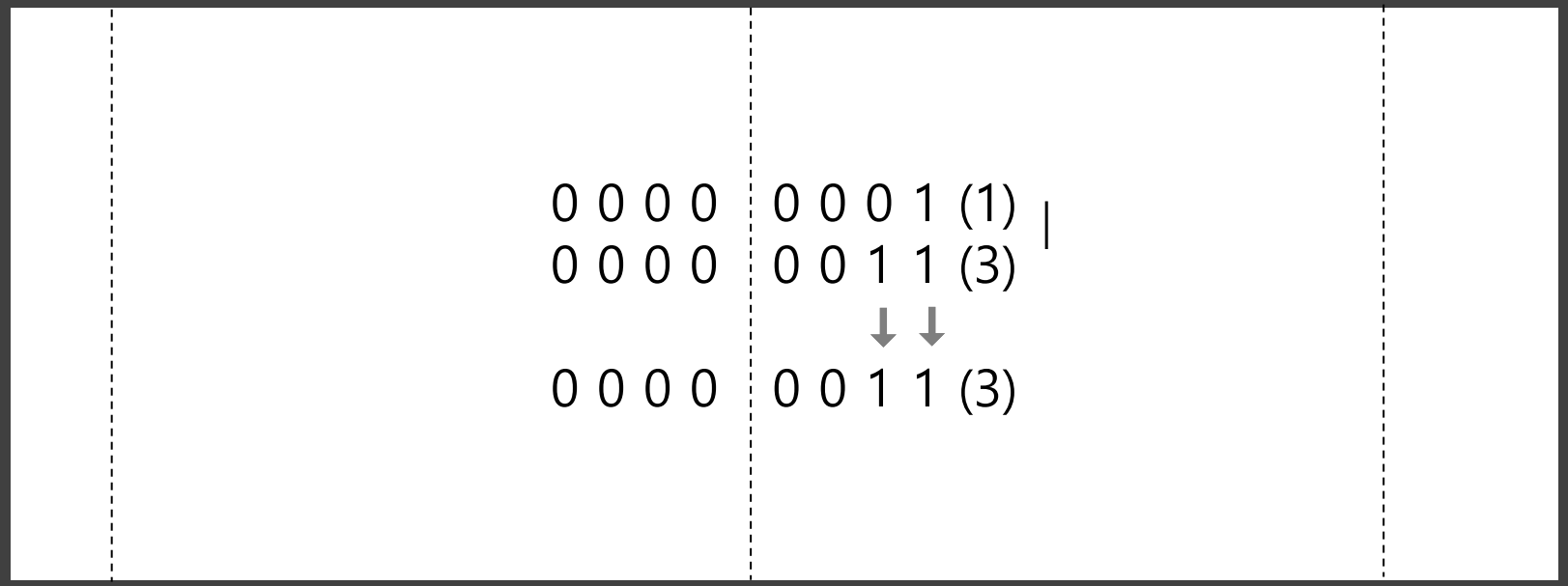


& : AND, | : OR, ^ : XOR, ~ : NOT, << : left shift , >> : right shift

C , Bit-Wize Operator

C언어 – 비트 연산자

|(OR) 비트 단위 연산

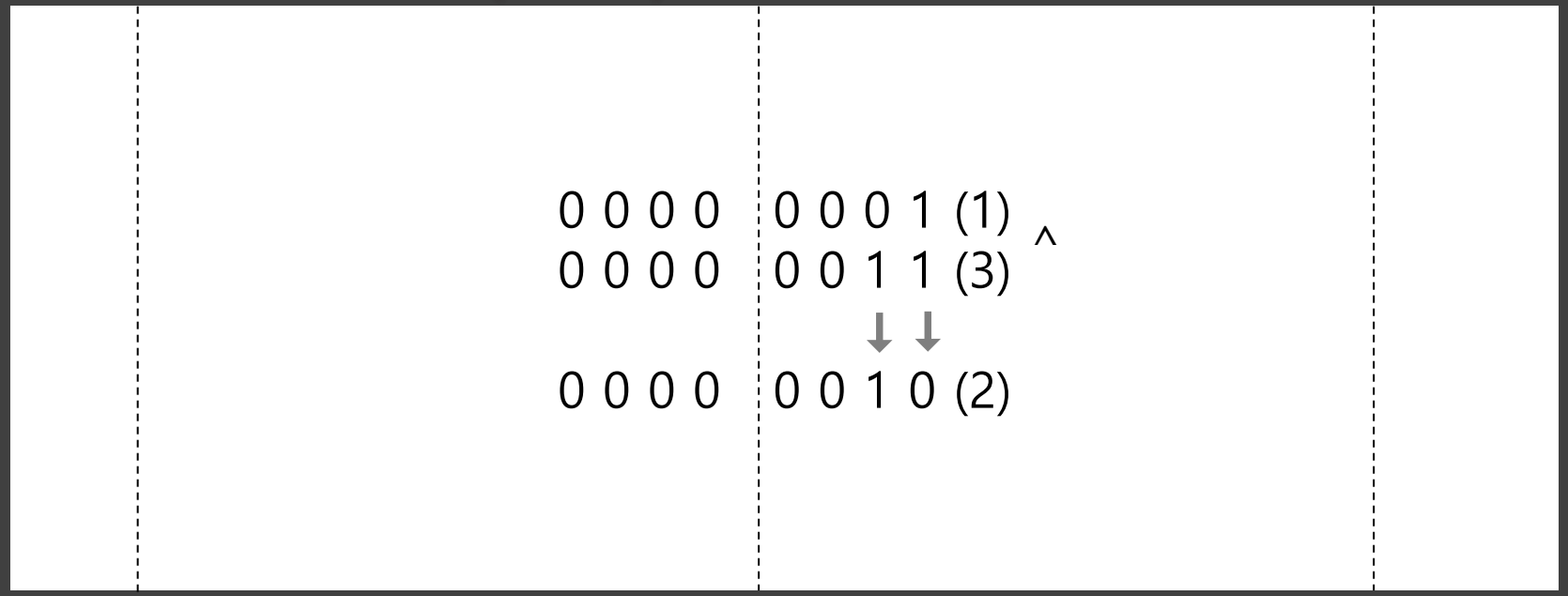


& : AND, | : OR, ^ : XOR, ~ : NOT, << : left shift , >> : right shift

C , Bit-Wize Operator

C언어 – 비트 연산자

^(XOR) 비트 단위 연산

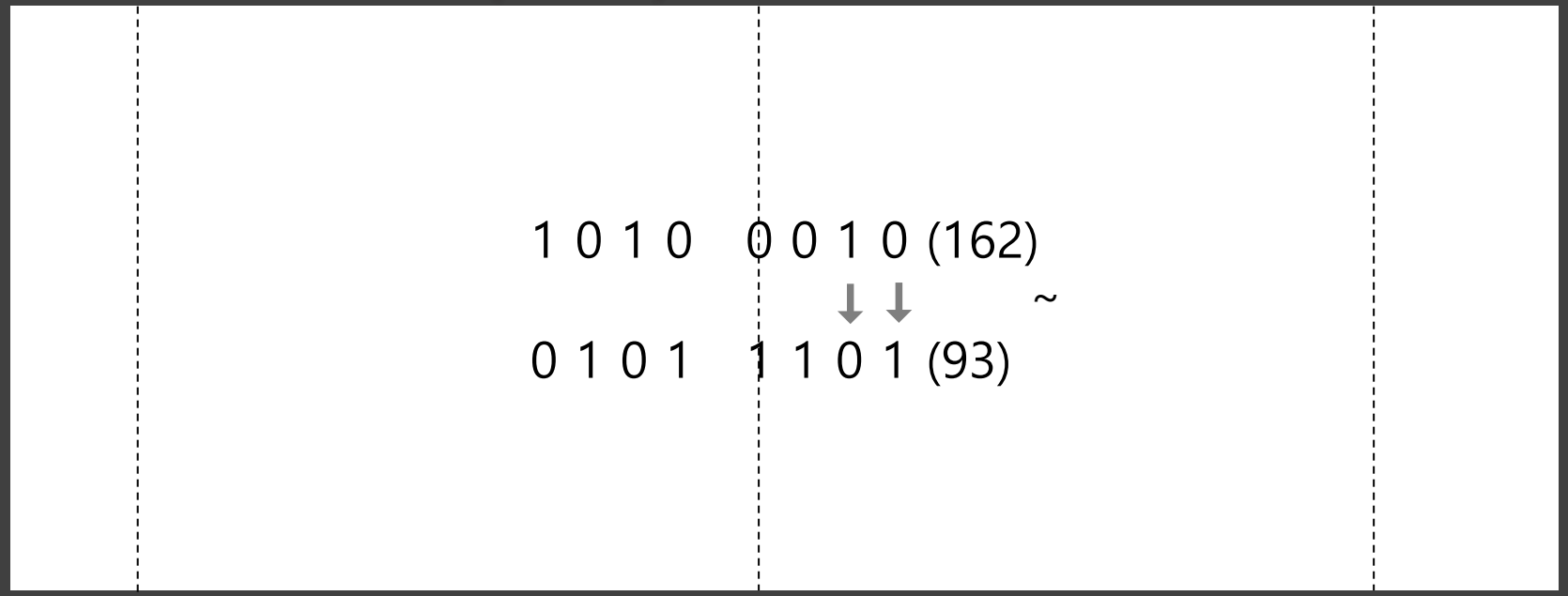


& : AND, | : OR, ^ : XOR, ~ : NOT, << : left shift , >> : right shift

C , Bit-Wize Operator

C언어 – 비트 연산자

~(NOT) 비트 단위 연산

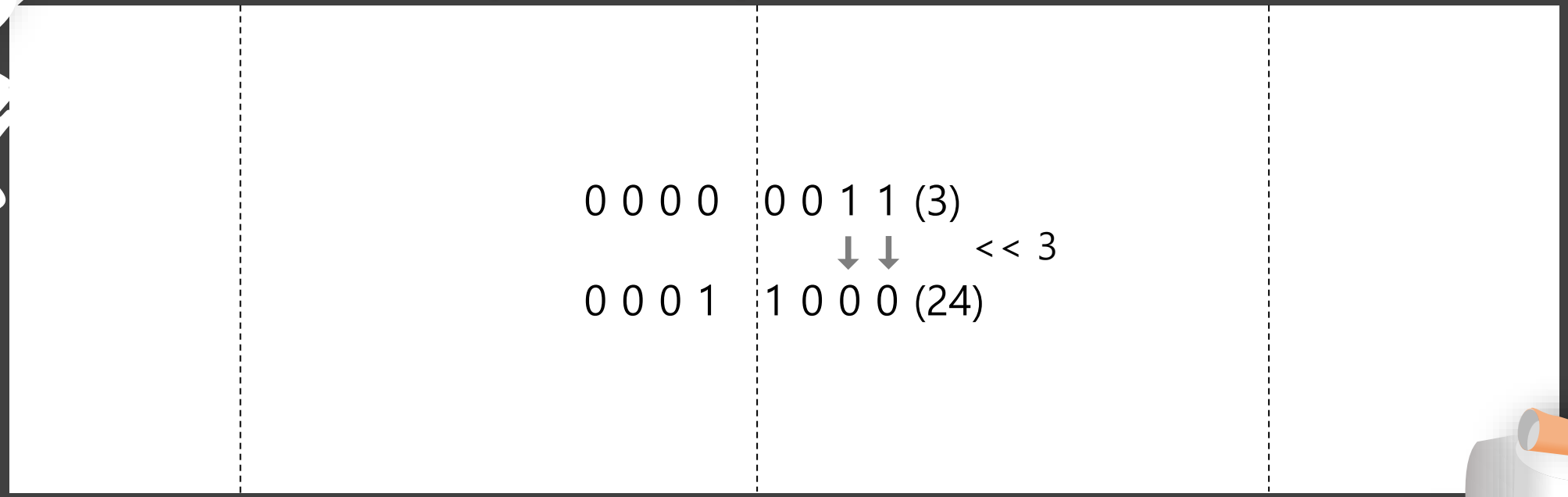


& : AND, | : OR, ^ : XOR, ~ : NOT, << : left shift , >> : right shift

C , Bit-Wize Operator

C언어 – 비트 연산자

<<(left shift) 비트 단위 연산자

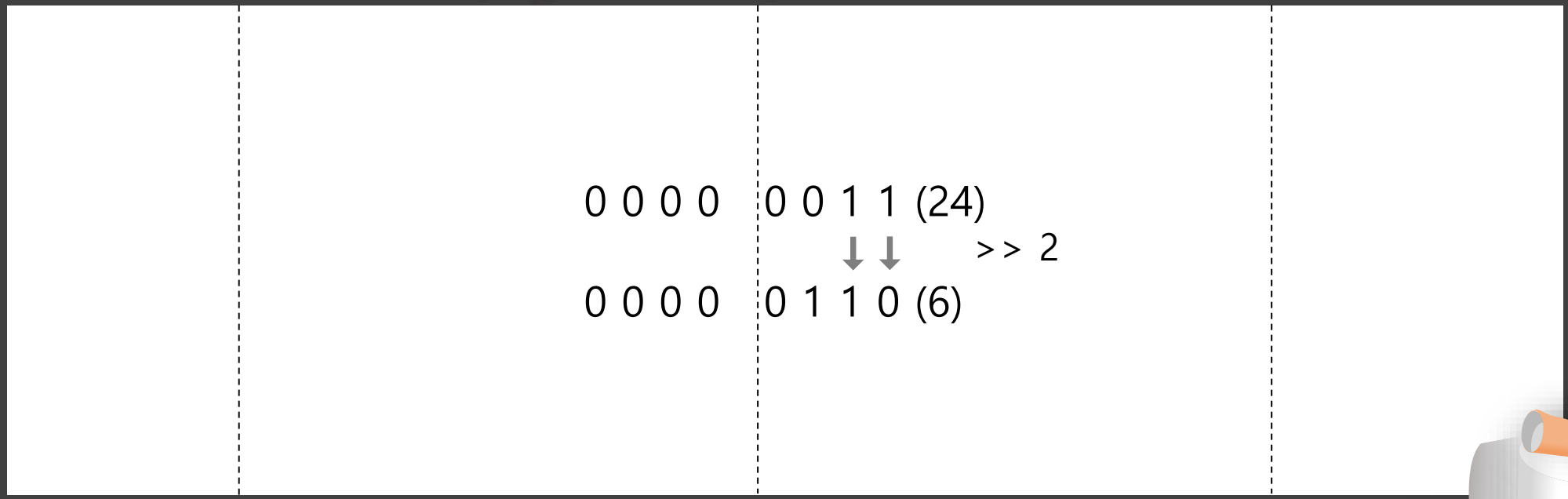


& : AND, | : OR, ^ : XOR, ~ : NOT, << : left shift , >> : right shift

C , Bit-Wize Operator

C언어 – 비트 연산자

>>(right shift) 비트 단위 연산자



& : AND, | : OR, ^ : XOR, ~ : NOT, << : left shift , >> : right shift

C , Bit-Wize Operator

C언어 – 비트 연산자

비트 연산자 연산 결과

&		
0	0	0
0	1	0
1	0	0
1	1	1

¦		
0	0	0
0	1	1
1	0	1
1	1	1

^		
0	0	0
0	1	1
1	0	1
1	1	0

~	
0	1
1	0

& : AND, | : OR, ^ : XOR, ~ : NOT, << : left shift , >> : right shift

C , Logical Operator

C언어 – 논리 연산자

논리 연산자

```
#include <stdio.h>

int main() {
  int man = 1;
  int age = 1;
  int korea = 1;

  printf("%d", man && age && korea);
}
```

출력결과 : 1

```
#include <stdio.h>

int main() {
  int man = 0;
  int age = 1;
  int korea = 0;

  printf("%d", man || age || korea);
}
```

출력결과 : 1

&& : AND, || : OR, ! : NOT

C , Logical Operator

C언어 – 논리 연산자

논리 연산자

```
#include <stdio.h>

int main() {
  int man = 0;
  int age = 0;

  printf("%d", !(man || age));
}
```

출력결과 : 1

```
#include <stdio.h>

int main() {
  int man = 1;
  int age = 0;

  printf("%d₩n", !(!(man || age)));
  printf("%d₩n", !(man || age));
}
```

출력결과 : 1
0

₩n : Enter 역할(강제 줄바꿈)

C , Assignment Operator

C언어 – 대입 연산자

대입 연산자

```
#include <stdio.h>

int main() {
  int a=0;
  int b=5;

  a = b; // b를 a에 대입
}
```

출력결과 : 5

```
#include <stdio.h>

int main() {
  int a=0;
  int b=5;

 a += b; // a = a + b와 같은 의미
}
```

출력결과 : 5

```
#include <stdio.h>

int main() {
  int a=0;
  int b=5;

  a -= b; // a = a - b와 같은 의미
}
```

출력결과 : -5

C , Assignment Operator

C언어 – 대입 연산자

대입 연산자

```
#include <stdio.h>

int main() {
  int a=0;
  int b=5;

  a *= b; // a = a * b와 같은 의미
}
```

출력결과 : 0

```
#include <stdio.h>

int main() {
  int a=0;
  int b=5;

  a /= b; // a = a / b와 같은 의미
}
```

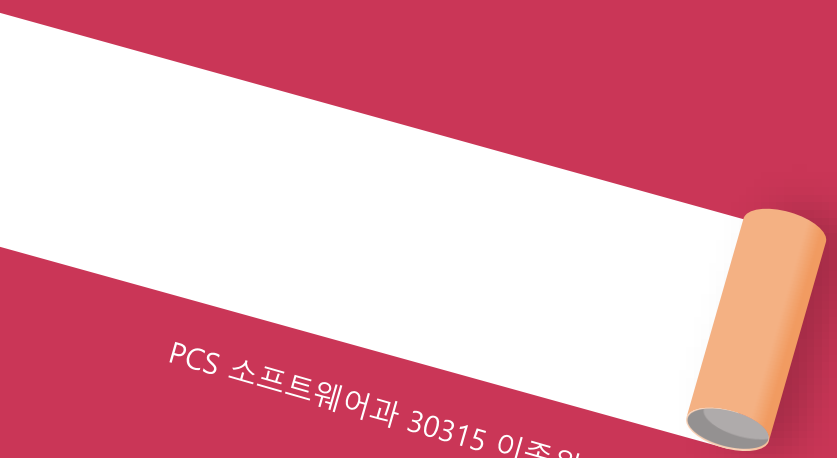
출력결과 : 0

```
#include <stdio.h>

int main() {
  int a=5;
  int b=2;

  a %= b; // a = a % b와 같은 의미
}
```

출력결과 : 1(나머지)

C, Operator

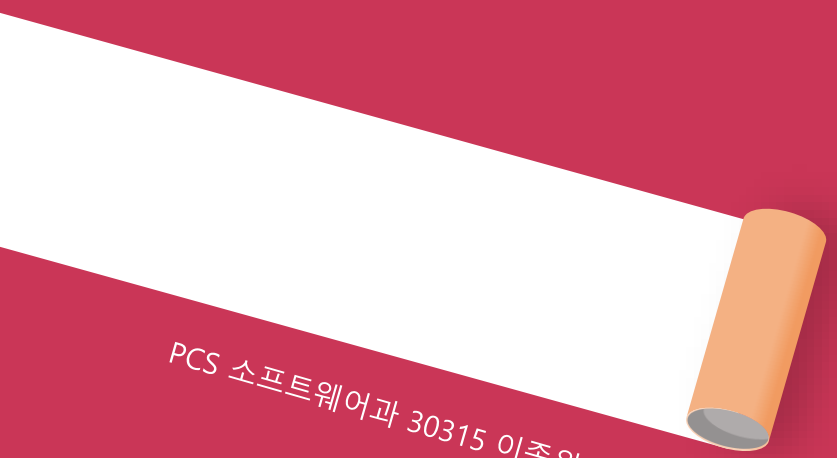
Visual Studio 2019 / C언어 - 연산자



PCS 소프트웨어과 30315 이종원

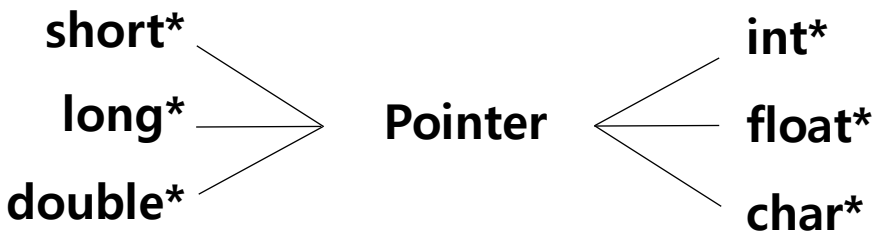
C, Pointer

Visual Studio 2019 / C언어 - 포인터



PCS 소프트웨어과 30315 이종원

포인터의 종류



* = Pointer(포인터)

대표적인 자료형

```
#include <stdio.h>

int main() {

  printf("%d₩n", sizeof(int)); // sizeof : 자료형의 크기
  printf("%d₩n", sizeof(float));
  printf("%d₩n", sizeof(char));
}
```

출력결과 : 4
4
1

- ex) sizeof(int) : int의 크기

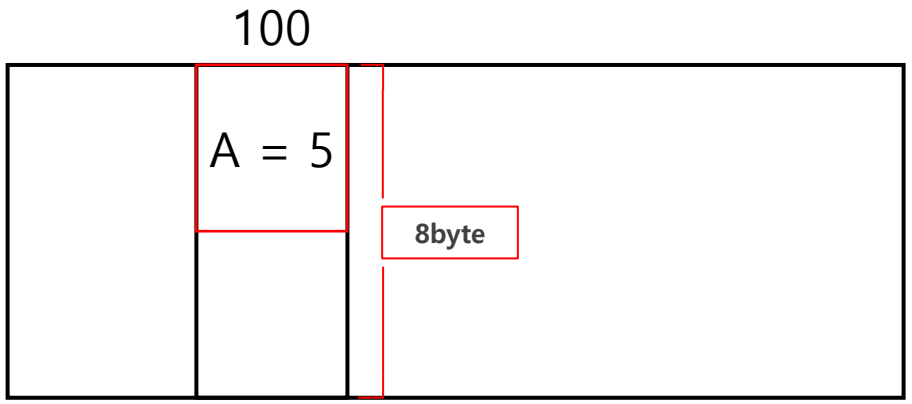
선언된 변수 예시

int = 4byte, int* = 8byte

ptr*

예시

선언 방법 : int* ptr;



A

정수형 변수

5

100

예시 주소

64bit의 운영체제의 한 메모리 주소는 8byte

ptr = &a;
(a의 주소를 표시)

변수의 앞에 사용

*(pointer) = asterisk

주소출력

```
#include <stdio.h>

int main() {

  int a = 5;
  int* ptr;

  ptr =&a;

  printf("%p", &a); // 포인터 주소 출력은 %p
  printf("₩n%p", *ptr);
}
```

출력결과 : 100
100

서식 지정자

정수형	실수형	문자	기타
%d : 10진 정수형으로 출력	%f : 실수형으로 출력	%c : 문자 출력	%e : 지수형 출력
%o : 8진 정수형 출력		%s : 문자열 출력	%u : 부호 없는 10진 정수형 출력
%x : 16진 정수형 출력			%g : e와 f 중에서 출력할 자리를 덜 차지하는 형태로 출력 (자동)
			%p : 포인터의 주소값 출력

변환문자

서식문자	%d	%o	%x	%f	%e	%u	%g
출력형태	char, shot, int	unsigned int	unsigned int	float, double	float, double	unsigned int	float, double

값 출력

```
#include <stdio.h>

int main() {

  int a = 5;
  int* ptr;

  ptr =&a;

  printf("%d", *ptr); // *ptr : ptr의 주소의 값 = 5
}
```

출력결과 : 5

포인터로 연산가능

```
#include <stdio.h>

int main() {

  int a = 5;
  int b = 10;

  int* ptr;
  int* ptr2;

  ptr = &a;
  ptr2 = &b;

  printf("%d", *ptr2 - *ptr); // *ptr2의 주소값(10) – ptr의 주소값(5) = 5, 5가 출력됨
}
```

출력결과 : 5

C , Pointer

C언어 - 포인터

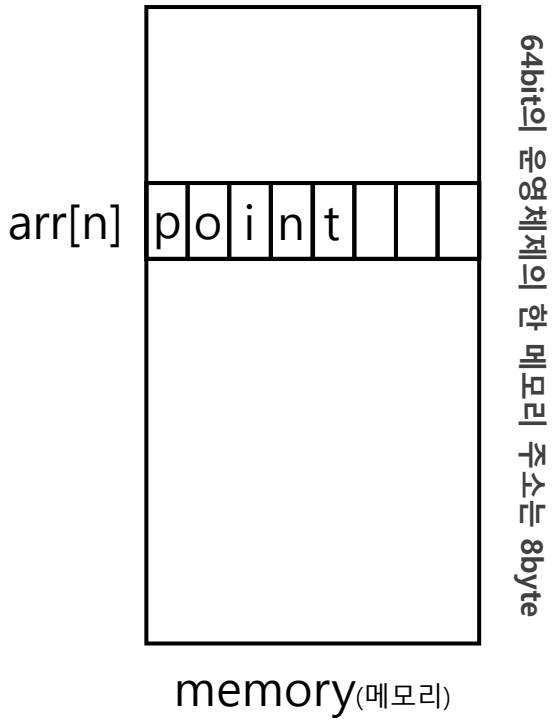
```
#include <stdio.h>

void swap(int* num1, int* num2){ // 주소 값을 받는 Pointer(포인터)
  int temp; // swap 연산자는 임시 저장공간 필요
  temp = *num1; // 임시 저장공간에는 num1이 가지고 값을 보관
  *num1 = *num2; // num1에 있는 값을 num2에 있는 값으로 변경
  *num2 = temp; // num2에 있는 값을 임시 저장공간에 있는 값으로 변경해라
}
int main() {
  int a = 5;
  Int b = 10;

  printf("%d%d", a, b); // 변경 전 5, 10
  swap(&a, &b); // a의 주소와 b의 주소를 넣음
  printf(("%d%d", a, b); // 스왑 후 10, 5로 변경
}
```

출력결과 : 510
105

포인터 배열

```
#include <stdio.h>

int main() {

  char arr[6] = {"point"};
  char* ptr;
  ptr = &arr[0];

  printf("%s", *ptr+1);
  return 0;
}
```

출력결과 : q // q가 출력된 이유는 연산자의 순서때문



연산자의 우선순위

```
#include <stdio.h>

int main() {

  char arr[6] = {"point"};
  char* ptr;
  ptr = &arr[0];

  printf("%s", *ptr+1);
  return 0;
}
```

출력결과 : q // q가 출력된 이유는 연산자의 순서때문

우선순위

(1) (2)

*ptr+1

3등급 4등급

C언어 연산자 우선순위

3	*	/	%
4	+	-	

우선순위가 *(asterisk)가 우선이기 때문에

*ptr로 정한 p가 +1이 되어 q가 출력

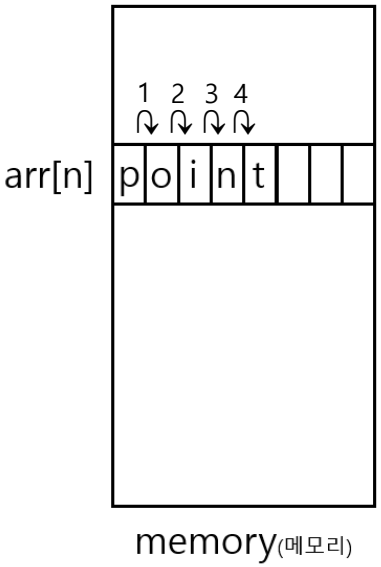
p의 다음 알파벳인 q가 출력됨

( )을 사용하여 우선순위를 변경하고

*(ptr + 1)로 실행하면 point에서
ptr로 정한 p의 다음 알파벳인
o가 출력됨

→

*(ptr + 4)로 실행하면 point에서
ptr로 정한 p의 + 4 알파벳인
t이 출력됨



더블, 트리플 포인터

더블 포인터 (이중 포인터)

포인터를 가르키는 더블 포인터

A ← ptr* ← ptr ← ptr*****

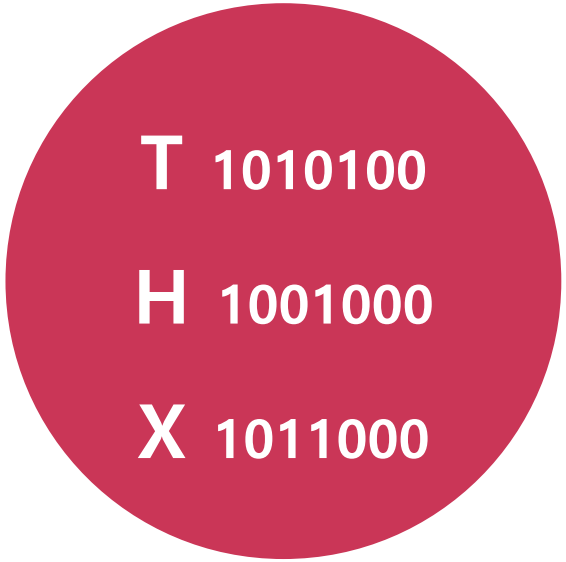
변수의 주소를 가르키는 포인터

더블 포인터를 가르키는 트리플 포인터

트리플 포인터 (삼중 포인터)

THX를 이진법으로 나타내면

THX = Thanks



감사합니다