

이주현

비트연산1

(주) 한컴에듀케이션

비트 연산자

- 비트 연산(bit operation) 이란?
 - ✓ 한 개 혹은 두 개의 이진수에 대해 비트 단위로 적용되는 연산이다.

- 비트 연산자
 - ✓ C, C++에는 다음 6개의 연산자가 있다.
 - ✓ 단항 연산자 : ~
 - ✓ 이항 연산자 : & , | , ^ , << , >>

비트 연산자

~	bitwise NOT tilde , wave	단항연산자 예) ~0111	각 자리의 비트를 반전시킨다.
&	bitwise AND ampersand 앤드기호	이항연산자 예) 0101 $\begin{array}{r} \& 0011 \\ \hline 0001 \end{array}$	두 수의 같은 자리의 비트를 비교하여 둘 다 1이 있을 경우에만 1을 반환한다.
	bitwise OR vertical bar 세로줄	이항연산자 예) 0101 $\begin{array}{r} 0011 \\ \hline 0111 \end{array}$	두 수의 같은 자리의 비트를 비교하여 둘 중 하나라도 1이 있다면 1을 반환한다.
^	XOR caret 꺽쇠	이항연산자 예) 0101 exclusive or $\begin{array}{r} ^ 0011 \\ \hline 0110 \end{array}$	두 수의 같은 자리의 비트를 비교하여 서로 다를 경우에만 1을 반환한다.
<<	LEFT SHIFT 왼쪽 두 꺽쇠	이항연산자 예) 1 << 3 = 8 3 << 2 = 12	$a << b = a * (2^b)$
>>	RIGHT SHIFT 오른쪽 두 꺽쇠	이항연산자 예) 13 >> 2 = 13 / 4 = 3 3 >> 4 = 3 / 16 = 0	$a >> b = a / (2^b)$

문제 분석 & 해법 연구

위 6개 비트 연산자를 사용하여 아래 질문에 답하시오.

질문에 사용된 모든 변수는 음이 아닌 정수이다.

문항별 사용되는 비트연산자는 1개를 초과할 수 있다.

(1) a가 홀수인지 if문과 비트 연산자를 이용하여 판별하고자 한다.

?자리에 들어갈 비트 연산자는?

`if(a ? 1){} // c, c++인 경우`

문제 분석 & 해법 연구

위 6개 비트 연산자를 사용하여 아래 질문에 답하시오.

질문에 사용된 모든 변수는 음이 아닌 정수이다.

문항별 사용되는 비트연산자는 1개를 초과할 수 있다.

(1) a가 홀수인지 if문과 비트 연산자를 이용하여 판별하고자 한다.

?자리에 들어갈 비트 연산자는?

`if(a ? 1){} // c, c++인 경우`

- 정수는 컴퓨터에 2진 비트로 저장된다.
- 따라서 홀수라면 정수의 LSB(least significant bit)가 1이다.
- `if(a & 1)`

문제 분석 & 해법 연구

(2) a가 짝수인지 if문과 비트 연산자를 이용하여 판별하고자 한다.
?자리에 들어갈 비트연산자는?

if(?a ? 1) {} /// c, c++인 경우

if((?a ?1) == 1)) {} /// java의 경우

if((a & 2) == 0)

문제 분석 & 해법 연구

(2) a가 짝수인지 if문과 비트 연산자를 이용하여 판별하고자 한다.
?자리에 들어갈 비트연산자는?

if(?a ? 1) {} /// c, c++인 경우

if((?a ?1) == 1)) {} /// java의 경우

- 정수는 컴퓨터에 2진 비트로 저장된다.
- 따라서 짝수라면 정수의 LSB(least significant bit)가 0이다.
- LSB가 0인 것을 1로 만든 후 1과 & 연산을 하면 될 것이다.
- 0인 비트를 1로 만드는 것은 단항 연산자 ~를 이용할 수 있다.
- if(~a & 1)

문제 분석 & 해법 연구

(3) a와 2의 k제곱을 곱한 결과(결과는 int 범위를 넘지 않는다.)를 구하고자 한다.
?자리에 들어갈 비트연산자는?

result = a ? k;

문제 분석 & 해법 연구

(3) a와 2의 k제곱을 곱한 결과(결과는 int 범위를 넘지 않는다.)를 구하고자 한다.
?자리에 들어갈 비트연산자는?

`result = a ? k;`

➤ $a * 2^k$ 은 $a \ll k$ 로 나타낼 수 있다.

문제 분석 & 해법 연구

(4) a 를 2의 k 제곱으로 나눈 몫 p 와 나머지 r 을 구하고자 한다.

?자리에 들어갈 비트연산자들은?

$$p = a \ ?\ k, r = a \ ?\ ((1 \ ?\ k) - 1);$$

문제 분석 & 해법 연구

(4) a 를 2의 k 제곱으로 나눈 몫 p 와 나머지 r 을 구하고자 한다.
?자리에 들어갈 비트연산자들은?

$$p = a \ ?\ k, r = a \ ?\ ((1 \ ?\ k) - 1);$$

- 2의 k 제곱으로 나눈 몫은 $a / 2^k$ 이다.
 $a / 2^k$ 은 a **>>** k 로 얻을 수 있다.
- 2의 제곱수(2^k)로 나눈 나머지는
 - ✓ 2^k 이상의 값은 모두 0으로 만들고
 - ✓ 2^k 미만의 값은 그대로 남기는 것과 같다.
 - ✓ 0101 1011 을 16으로 나눈 나머지는 0000 **1011** 이다.
- 따라서 2의 k 제곱(2^k)으로 나눈 나머지는
 a **&** $((1 \ll k) - 1)$ 로 구할 수 있다.

문제 분석 & 해법 연구

(5) 다음은 a와 b의 값을 서로 바꾸는 코드이다.

?자리에 들어갈 비트연산자들은?

$a = a \ ? \ b, b = a \ ? \ b, a = a \ ? \ b;$

문제 분석 & 해법 연구

(5) 다음은 a와 b의 값을 서로 바꾸는 코드이다.

?자리에 들어갈 비트연산자들은?

$a = a ? b, b = a ? b, a = a ? b;$

➤ 두 수 a, b가 주어지고 $c = a \wedge b$ 라고 하면 세 수의 관계는

- ✓ $a == b \wedge c == c \wedge b$ (교환법칙이 성립하므로)
- ✓ $b == c \wedge a == a \wedge c$
- ✓ $c == a \wedge b == b \wedge a$

➤ 따라서 \wedge (xor)연산을 이용하여 해결할 수 있다.

$a = a \wedge b, b = a \wedge b, a = a \wedge b;$

문제 분석 & 해법 연구

(6) a의 k번째(2의 k제곱 자리) 비트가 0인지 1인지 알아보고자 한다.
?자리에 들어갈 비트연산자들은?

result = (a ? k) ? 1;

문제 분석 & 해법 연구

(6) a의 k번째(2의 k제곱 자리) 비트가 0인지 1인지 알아보고자 한다.
?자리에 들어갈 비트연산자들은?

```
result = (a ? k) ? 1;
```

- k번째 비트를 1의 자리로 이동시킨 후 그 값이 1인지 확인하는 방법을 사용할 수 있다.
- ```
result = (a >> k) & 1;
```

# 문제 분석 & 해법 연구

---

(7) a의 k번째(2의 k제곱 자리) 비트만 0으로 바꾸고자 한다.

?자리에 들어갈 비트연산자들은?

$a = a ? (?(1 ? k));$

# 문제 분석 & 해법 연구

---

(7) a의 k번째(2의 k제곱 자리) 비트만 0으로 바꾸고자 한다.  
?자리에 들어갈 비트연산자들은?

$a = a ? (?(1 ? k));$

- k번째 비트만 0인 수를 만들어 a와 &연산을 하는 방법을 생각할 수 있다.
  - 먼저 k번째 비트만 1인 수는  $(1 << k)$ 이다.
  - 다음 k번째 비트만 0인 수는  $\sim(1 << k)$ 가 된다.
- 따라서  $a = a \& (\sim(1 << k));$

# 문제 분석 & 해법 연구

---

(8) a의 k번째(2의 k제곱 자리) 비트만 0이면 1로, 1이면 0으로 바꾸고자 한다.  
?자리에 들어갈 비트연산자들은?

$a = a ? (1 ? k);$

# 문제 분석 & 해법 연구

---

(8) a의 k번째(2의 k제곱 자리) 비트만 0이면 1로, 1이면 0으로 바꾸고자 한다.  
?자리에 들어갈 비트연산자들은?

$a = a ? (1 ? k);$

- k번째 비트만 1인 수를 만들어 a와 ^연산을 하는 방법을 생각할 수 있다.
- $a = a ^ (1 << k);$

# 문제 분석 & 해법 연구

---

(9) a가 2의 제곱수인지 판별하고자 한다. ?자리에 들어갈 비트연산자들은?

```
result = a ? (a - 1);
```

```
if(a > 0 && result==0) printf("%d is the power of 2.\n", a);
```

# 문제 분석 & 해법 연구

---

(9) a가 2의 제곱수인지 판별하고자 한다. ?자리에 들어갈 비트연산자들은?

```
result = a ? (a - 1);
if(a > 0 && result==0) printf("%d is the power of 2.\n", a);
```

- 2의 제곱수들은 전체 비트 중에 1인 비트는 1개 뿐이다.
- a가 2의 제곱수라면 a와 a-1은 같은 자리의 비트를 비교할 때 두 자리 모두 1인 경우가 존재하지 않는다.
- 이것은 2의 제곱수를 판별하는 특징이 된다.  
그런데 0또한 이러한 특징이 있으므로 0은 예외처리 해야한다.
- result = a **&** (a - 1);

# 문제 분석 & 해법 연구

---

(10) a와 b의 같은 자리 비트를 비교한 결과  
서로 다른 비트가 1개 이하인지 알아보고자 한다.

?자리에 들어갈 비트연산자들은?

bit = a ? b;

result = bit ? (bit -1);

if(result==0) printf("%d and %d differ by below 1bit.\n", a, b);

# 문제 분석 & 해법 연구

---

(10) a와 b의 같은 자리 비트를 비교한 결과  
서로 다른 비트가 1개 이하인지 알아보고자 한다.  
?자리에 들어갈 비트연산자들은?

```
bit = a ? b;
result = bit ? (bit -1);
if(result==0) printf("%d and %d differ by below 1bit.\n", a, b);
```

- 두수의 ^ 결과가 2의 제곱수이거나 0이면 된다.
- bit = a **^** b;
- result = bit **&** (bit -1);

# 문제 분석 & 해법 연구

---

(11) a의 ls1b를 구하고자 한다.

(first set bit :  $2^0$ 부터 시작하여 처음 만나는 1인 비트의 위치로 정의한다.)

(least significant 1-bit :  $2^0$ 부터 시작하여 처음 만나는 1인 비트의 가중치로 정의한다.)

?자리에 들어갈 비트연산자는?

예를 들어 1, 5, 7 등의 ls1b 는 1이고 2, 6, 10 등의 ls1b 는 2이다.

`ls1b = a ? -a;`

```
printf("%d 's ls1b is %d.\n", a, ls1b);
```

- 음의 정수는 2의 보수 형태로 저장된다.
- `ls1b = a & -a;`
- 위 식은 Fenwick Tree(BIT-binary indexed tree)에서  
아주 중요하게 쓰인다.

감사합니다.