7장 비트단위 연산자와 열거

박 종 혁 교수

UCS Lab

Tel: 970-6702

Email: jhpark1@seoultech.ac.kr

비트단위 연산자

- 이진숫자의 문자열로 표현된 정수적 수식에 사용
- 시스템 종속적
- 여기서는 8 비트 바이트, 4 바이트 워드, 2의 보수로 표현되는 정수, 그리고 ASCII 문자 코드를 갖는 컴퓨터를 가정함

비트단위 연산자

논리 연산자	(단항) 비트단위 보수 비트단위 논리곱 비트단위 배타적 논리합 비트단위 논리합	~ & ^ |
이동 연산자	왼쪽 이동 오른쪽 이동	<< >>

비트단위 보수 : ~ 연산자

- **1의 보수 연산자 또는 비트단위 보수 연산자**
- **주어진 인자의 비트열 표현을 반대로 함; 0은 1로, 1은 0으로**
- **예**

```
int   a = 70707;
```

  - a의 이진수 표현

```
00000000 00000001 00010100 00110011
```

  - ~a의 이진수 표현

```
11111111 11111110 11101011 11001100(-70708)
```

2의 보수

- **n이 음이 아닌 정수일 때**
  - n의 2의 보수는 n을 이진수로 표현한 비트열
  - -n의 2의 보수표현은 n의 비트열에서 비트단위 보수를 구하고 거기에 1을 더한 비트열

n의 값	이진수	비트단위 보수	-n의 2의 보수	-n의 값
7	00000000 00000111	11111111 11111000	11111111 11111001	-7
8	00000000 00001000	11111111 11110111	11111111 11111000	-8
9	00000000 00001001	11111111 11110110	11111111 11110111	-9
-7	11111111 11111001	00000000 00000110	00000000 00000111	7

(참고) 0의 2의 보수 : 모든 비트가 0임

-1의 2의 보수 : 모든 비트가 1임

비트단위 이진 논리 연산자

- **이진 논리 연산자**
  - &(논리곱), ^(배타적 논리합), |(논리합)
  - 정수적 수식을 피연산자로 가짐
  - 적절히 형 변환된 두 피연산자는 대응되는 비트끼리 연산됨

a	b	a & b	a ^ b	a | b
0	0	0	0	0
1	0	0	1	1
0	1	0	1	1
1	1	1	0	1

비트단위 이진 논리 연산자

선언 및 초기화		
int a = 33333, b = -77777;		
수식	표현	값
a	00000000 00000000 10000010 00110101	33333
b	11111111 11111110 11010000 00101111	-77777
a & b	00000000 00000000 10000000 00100101	32805
a b	11111111 11111110 01010010 00011010	-110054
a | b	11111111 11111110 11010010 00111111	-77249
~(a | b)	00000000 00000001 00101101 11000000	77248
(~a & ~b)	00000000 00000001 00101101 11000000	77248

왼쪽과 오른쪽 이동 연산자

- 이동 연산자의 두 피연산자는 정수적 수식이어야 함
- 각 피연산자에 정수적 승격이 일어남
- 전체 수식의 형은 승격된 왼쪽 피연산자의 형이 됨

왼쪽 이동 연산자

- **expr1 << expr2**
  - expr1의 비트 표현을 expr2가 지정하는 수만큼 왼쪽으로 이동
  - 하위 비트로는 0이 들어옴
  - 수식에서 c는 int 형으로 승격됨
  - 따라서 결과는 4 바이트에 저장됨

왼쪽 이동 연산자

선언 및 초기화		
char c = 'Z';		
수식	표현	동작
c	00000000 00000000 00000000 01011010	이동되지 않은 상태
c << 1	00000000 00000000 00000000 10110100	왼쪽으로 1 이동
c << 4	00000000 00000000 00000101 10100000	왼쪽으로 4 이동
c << 31	00000000 00000000 00000000 00000000	왼쪽으로 31 이동

오른쪽 이동 연산자

- **expr1 >> expr2**
  - 왼쪽 이동 연산자와 대칭적이지 않음
  - 부호가 없는 정수적 수식에서는 상위 비트로 0이 들어옴
  - 부호가 있는 형일 때에는 시스템에 따라 상위 비트로 0이 들어오는 것도 있고, 1이 들어오는 것도 있음

오른쪽 이동 연산자

선언 및 초기화		
int a = 1 << 31; /* shift 1 to the high bit */ unsigned b = 1 << 31;		
수식	표현	동작
a	10000000 00000000 00000000 00000000	이동되지 않은 상태
a >> 3	11110000 00000000 00000000 00000000	오른쪽으로 3 이동
b	10000000 00000000 00000000 00000000	이동되지 않은 상태
b >> 3	00010000 00000000 00000000 00000000	오른쪽으로 3 이동

이동 연산자

선언 및 초기화			
unsigned a = 1, b = 2;			
수식	동등한 수식	표현	값
a << b >> 1	(a << b) >> 1	00000000 00000010	2
a << 1 + 2 << 3	(a << (1 + 2)) << 3	00000000 01000000	64
a + b << 12 * a >> b	((a + b) << (12 * a)) >> b	00001100 00000000	3072

마스크

- **마스크 : 다른 변수나 수식으로부터 원하는 비트를 추출하는 데 사용되는 상수나 변수**
- **int 형 상수 1의 비트 표현:**

```
00000000 00000000 00000000 00000001
```

  - 이것을 사용하여 int 형 수식의 최하위 비트를 알아낼 수 있음

패킹

- **4 개의 문자를 하나의 int 형에 패킹하는 함수**

```
#include <limits.h>
int pack(char a, char b, char c, char d){
    int   p = a;
    p = (p << CHAR_BIT) | b;
    p = (p << CHAR_BIT) | c;
    p = (p << CHAR_BIT) | d;
    return p;
}
```

언패킹

- **32 비트 int 안에 있는 문자를 검색하는 함수 (마스크 사용)**

```
#include <limits.h>
char unpack(int p, int k){      /* k = 0, 1, 2, or 3 */
   int  n = k * CHAR_BIT;    /* n = 0,8,16, or 24 */
   unsigned  mask = 255;     /* low-order byte */
   mask <<= n;
   return ((p & mask) >> n);
}
```

열거

- 키워드 enum은 열거형을 선언하는데 사용됨
- 이것은 유한집합을 명명하고, 그 집합의 원소로서 식별자를 선언하는 수단을 제공함

열거

- **예제**

```
enum day {sun, mon, tue, wed, thu, fri, sat};
```

  - 사용자 정의형 enum day 생성
  - day : 태그 이름
  - 열거자는 식별자 sun, mon, ..., sat이고, 이들은 int 형 상수임
  - 디폴트로 첫 번째 원소는 0이고, 각 원소는 이전 원소의 값보다 하나 큰 값을 가짐
  - 이것은 형 정의임

열거

- **enum day 형 변수의 선언**

```
enum day  d1, d2;
```

- **d1, d2 변수의 사용**

```
d1 = fri;
if (d1 == d2)
    .....      /* do something */
```

열거

- **선언 예제 1**

```
enum suit {clubs = 1, diamonds, hearts, spades} a, b, c;
```

- **선언 예제 2**

```
enum fruit {apple = 7, pear, orange = 3, lemon} frt;
```

- **선언 예제 3**

```
enum veg {beet = 17, carrot = 17, corn = 17} vege1, vegw2;
```

- **선언 예제 4**

```
enum {fir, pine}   tree;
```

- **선언 예제 5**

```
enum veg {beet, carrot, corn}  veg;
```

질의 및 응답