Bitwise Operators

Bitwise Operators and Expressions
Bitwise Complement
Bitwise Binary Logical Operators
Left and Right Shift Operators
Masks
Packing and Unpacking

Bitwise Operations and Expressions

- ภาษาซีสามารถดำเนินการกับข้อมูลในระดับของบิต (Bit manipulation)ได้
- ตัวดำเนินการระดับบิตใช้กับตัวถูกดำเนินการ(operand) ประเภท char, short, int, long int, unsigned เท่านั้น
- The bitwise operators act on integral expressions represented as string of binary digits.
- These operators are explicitly system-dependent.

Bitwise operations

- (Unary) Bitwise Complement : ~
- Logical operators
  - bitwise and : &
  - bitwise exclusive or : ^
  - bitwise inclusive or : |
- Shift Operators
  - left shift : <<
  - right shift : >>
- Operation Priorities

Bitwise Complement

- The operator ~ is unary.
- It inverts the bit string representation of its argument;
  - the 0s become 1s
  - the 1s become 0s
- using format : *~ integral operand*
- Example :

```
                                    int a = 9;
the binary representation of a is   0 0 0 0 0 0 0 0   0 0 0 0 1 0 0 1
the binary representation of ~a is  1 1 1 1 1 1 1 1   1 1 1 1 0 1 1 0
the integer value of expression ~a is -10
```

Bitwise Binary Logical Operators

- These operators are binary.
- The two operand are operated on bit-position by bit-position.

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


```
int   a = 15, b= -255;
```

Expression	Binary Representation
a	0000 0000 0000 1111
b	1111 1111 0000 0001
a & b	0000 0000 0000 0001
a ^ b	1111 1111 0000 1110
a | b	1111 1111 0000 1111

Left and Right Shift Operators

- These operators are binary.
- The two operands of shift operator must be integral expressions.
  - expr1 >> expr2
  - expr1 << expr2
- The bit representation of expr1 to be shifted by the number of places specified by expr2.
- For the left shift operator, on the low-order end, 0s are shifted in.
- For the right shift operator, on the high-order end, 0s are shifted in.

Declaration and initializations

unsigned a = 13, b= -255, c = 5;

Expression	Binary Representation	Action
a	0000 0000 0000 1101	unshifted
a << 1	0000 0000 0001 1010	left-shifted 1
a << c	0000 0001 1010 0000	left-shifted 5
b	1111 1111 0000 0001	unshifted
b >> 1	0111 1111 1000 0000	right-shifted 1
b >> c	0000 0111 1111 1000	right-shifted 5

Declaration and initializations

```
unsigned    a = 1, b= 2;
```

Expression	Binary Representation	Value
a << b >> 1	0000 0000 0000 0010	
a << 1 + 2 << 3		
a+b<<12*a>>b		
~b >> a		
~a + b << 3		

ตัวอย่าง

```
/* BITWISE.C : Demonstrate bitwise operator */
#include <stdio.h>
void main()
{
        printf("255 & 15 = %d\n", 255 & 15);
        printf("255 | 15 = %d\n", 255 | 15);
        printf("255 ^ 15 = %d\n", 255 ^ 15);
        printf("2 << 2 = %d\n", 2<<2);
        printf("16 >> 2 = %d\n", 16 >> 2);
        printf("~2 = %d\n", ~2);
}
```

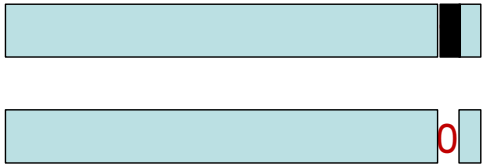
```
255 & 15 = 15
255 | 15 = 255
255 ^ 15 = 240
2 << 2 = 8
16 >> 2 = 4
~2 = -3
```


Masks

A mask is a constant or variable that used to extract desired bits from another variable or expression.

- int i, mask = 1;

```
 for (i=0; i<10; i++)
     printf("%d", i & mask);
```



- int v;

```
    :
if (v & (1<<2))
/*This statement is executed when the 3rd bit of v is 1.*/
  else
    :
```



```
/* Bit print of an integral expression*/
#include <stdio.h>
#include <limits.h>
int main() {
  int i, c = -15;
  int n = sizeof(int)*CHAR_BIT;
  int mask = 1 << (n-1);
  for (i=1; i<=n; ++i) {
      putchar(((c & mask) == 0)? '0':'1');
      c <<= 1;
      if (i% CHAR_BIT == 0 && i < n)
         putchar(' ');
  }
}
```

```
11111111 11110001
```

จงเขียนฟังก์ชัน bit_print() ซึ่งมีพารามิเตอร์เป็นข้อมูลชนิดจำนวนเต็ม เพื่อทำการแสดงค่าแต่ละบิตของพารามิเตอร์นั้น

```
#include <stdio.h>
#include <limits.h>
void bit_print(int );
void main() {
   int c;
   printf("Integer to be displayed its bit representation:");
   scanf("%d", &c);    BitPrint(c);
}
/* Bit print of an integral expression*/
```

```
}
```

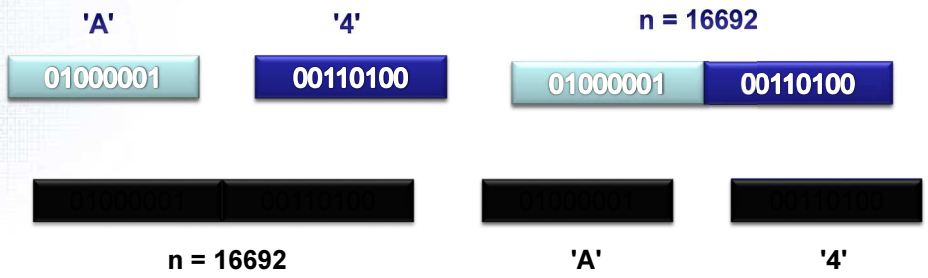

ตัวอย่างการรับเข้าข้อมูล (binary -> decimal)

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 32

int main ()
{
    char curr;
    unsigned int sum = 0;
    int i  = 0;

    while (((curr = getchar()) != '\n') && (i < MAX)) {
        sum = sum << 1;
        if (curr == '1') {
            sum = sum | 1;
        }
        i++;
    }
    printf("Decimal = %d\n", sum );

    return 0;
}
```

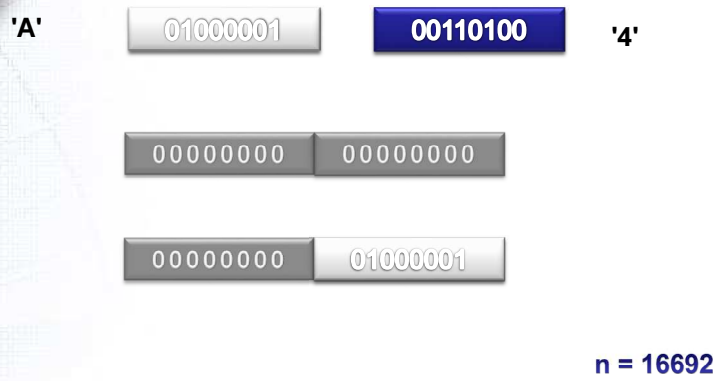
```
101
Decimal = 5
```


Packing and Unpacking

- The use of bitwise expressions allows for data compression/decompression across byte boundaries.



Packing



Packing

The use of bitwise expressions allows for data compression across byte boundaries.

Function Definition

```
/*Pack 2 characters into an int.*/
#include <limits.h>
int pack (char a, char b) {
    int p = a; /* p will be packed with a and b*/
    p = (p << CHAR_BIT) | b;
    return (p);
}
```

Function Invocation

```
:
printf("Packing of \'x\' and \'y\' is %X\n", pack( 'x', 'y'));
printf("Bit representation of packed \'x\', \'y\' is ", BitPrint(pack('x', 'y')));
```



```
/*Unpack 2 characters from an int.*/
#include <limits.h>
char unpack (int p, int k) {
    int n = k * CHAR_BIT;    /* k=0, 1 so n = 0, 8 */
    unsigned   mask = 255;

    mask <<= n;
    return ((p & mask) >> n);
}

int       p;
p = pack('a', 'b');
putchar(unpack(p, 1)); /* a is printed.*/
putchar('\n');
putchar(unpack(p, 0)); /* b is printed.*/
```

low byte

high byte

'a'	'b'

บิตที่ 15 0


```
/*   tbitwise.c : to illustrate bitwise operation
*/
#include <stdio.h>
#include <limits.h>
int  pack(char, char);
void BitPrint(int);
char  unpack(int, int);
void main (void) {
  int i;
  i = pack('A', '4');
  printf("Packing of \'A\' and \'4\', hexadecimal is %X\n", i);
  printf("Bit representation of %d is: ", i);
  BitPrint(i); /* BitPrint(pack('A', '4'))*/
  printf("Unpack low byte of %d is: %c\n", i, unpack(i, 0));
  printf("      High byte of %d is: %c\n", i, unpack(i, 1));
  //putchar(unpack(i, 1));
}
```

```
int  pack(char a, char b) {
  int p=a;
  p = (p << CHAR_BIT) | b;
  return (p);
}
char  unpack (int p, int k) {
     int n = k * CHAR_BIT;    /* k=0,
   1 so n = 0, 8 */
     unsigned   mask = 255;
     mask <<= n;
    return ((p & mask) >> n);
}
void  BitPrint(int c) {
  int n = sizeof(int)*CHAR_BIT;
  int i, mask= 1<< n-1;
  for (i=1; i<=n; ++i) {
    putchar((c & mask)==0 ? '0':
   '1');
    c <<= 1;
    if (i%CHAR_BIT ==0 && i<n)
      putchar(' ');
    }
  putchar('\n');
}
```


บทส่งท้าย ๗ จอ

จิตใจ จดจ่อ จึงจะเจอะเจอ