601.220 Intermediate Programming

Bitwise operations

Outline

- Bitwise operators

Bitwise operators

A bitwise operator performs a function across all bits in its operands

Bitwise AND - &

Bitwise AND (&) performs logical AND (&&) across all bits:

```
12 = 00001100 (In Binary)
25 = 00011001 (In Binary)

Bit Operation of 12 and 25
  00001100
& 00011001

  --------
  00001000  = 8 (In decimal)
```

Bitwise AND - &

```c
// bitwise_and.c:
#include <stdio.h>
int main() {
    int a = 12;
    int b = 25;
    printf("%d & %d = %d", a, b, a & b);
    return 0;
}
```

```
$ gcc -std=c99 -pedantic -Wall -Wextra bitwise_and.c
$ ./a.out
12 & 25 = 8
```

Bitwise OR - |

Bitwise OR (|) performs logical OR (||) across all bits:

```
12 = 00001100 (In Binary)
25 = 00011001 (In Binary)

Bitwise OR Operation of 12 and 25
  00001100
| 00011001
  --------
  00011101  = 29 (In decimal)
```

Bitwise OR - |

```c
// bitwise_or.c:
#include <stdio.h>
int main() {
    int a = 12;
    int b = 25;
    printf("%d | %d = %d", a, b, a | b);
    return 0;
}
```

```
$ gcc -std=c99 -pedantic -Wall -Wextra bitwise_or.c
$ ./a.out
12 | 25 = 29
```

Bit Shifting Left - <<

`x << n` shifts bits of `x` to the left `N` positions

`N` 0s are "shifted in" at right-hand side

`N` bits "fall off" left-hand side

```
25 = 00011001 (In Binary)

Bitwise left-shift of 25 by 5 positions (25 << 5)
0000011001
      << 5

----------
1100100000 = 800 (In decimal)
```

Bit Shifting Left - <<

```c
// bitwise_lshift.c:
#include <stdio.h>
int main() {
    int a = 25;
    int b = 5;
    printf("%d << %d = %d", a, b, a << b);
    return 0;
}
```

```
$ gcc -std=c99 -pedantic -Wall -Wextra bitwise_lshift.c
$ ./a.out
25 << 5 = 800
```

Bit Shifting Right - >>

Similar for bitwise right shift (>>)

```
25 = 00011001 (In Binary)

Bitwise right-shift of 25 by 4 positions (25 >> 4)
0000011001
      >> 4

----------
0000000001  = 1
```

Bit Shifting Right - >>

```c
// bitwise_rshift.c:
#include <stdio.h>
int main() {
    int a = 25;
    int b = 4;
    printf("%d >> %d = %d", a, b, a >> b);
    return 0;
}
```

```
$ gcc -std=c99 -pedantic -Wall -Wextra bitwise_rshift.c
$ ./a.out
25 >> 4 = 1
```

Converting an int to its Binary Representation (as a string)

```c
// bitwise_convert.c:
#include <stdio.h>
int main() {
    int num = 53;
    char bin_str[33] = {'\0'};
    int tmp = num;
    for(int i = 0; i < 32; i++) {
        if((tmp & 1) != 0) {     // least significant bit set?
            bin_str[31-i] = '1'; // prepend 1
        } else {
            bin_str[31-i] = '0'; // prepend 0
        }
        tmp >>= 1;               // shift right by 1
    }
    printf("%d in binary: %s\n", num, bin_str);
    return 0;
}
```

```
$ gcc bitwise_convert.c -std=c99 -pedantic -Wall -Wextra
$ ./a.out
53 in binary: 00000000000000000000000000110101
```

Bitwise operators

Operator		Description
&	bitwise AND	The bits in the result are set to 1 if the corresponding bits in the two operands are both 1.
|	bitwise inclusive OR	The bits in the result are set to 1 if at least one of the corresponding bits in the two operands is 1.
^	bitwise exclusive OR	The bits in the result are set to 1 if exactly one of the corresponding bits in the two operands is 1.
<<	left shift	Shifts the bits of the first operand left by the number of bits specified by the second operand; fill from the right with 0 bits.
>>	right shift	Shifts the bits of the first operand right by the number of bits specified by the second operand; the method of filling from the left is machine dependent.
~	one's complement	All 0 bits are set to 1 and all 1 bits are set to 0.

Fig. 10.6 | Bitwise operators.

From Deitel & Deitel: C++ How to Program, 8th ed

More: `en.wikipedia.org/wiki/Bitwise_operation`

Zoom poll!

What is the result of `(15 >> 2) || 7`?

A. 7

B. 15

C. 0

D. 1

E. 8