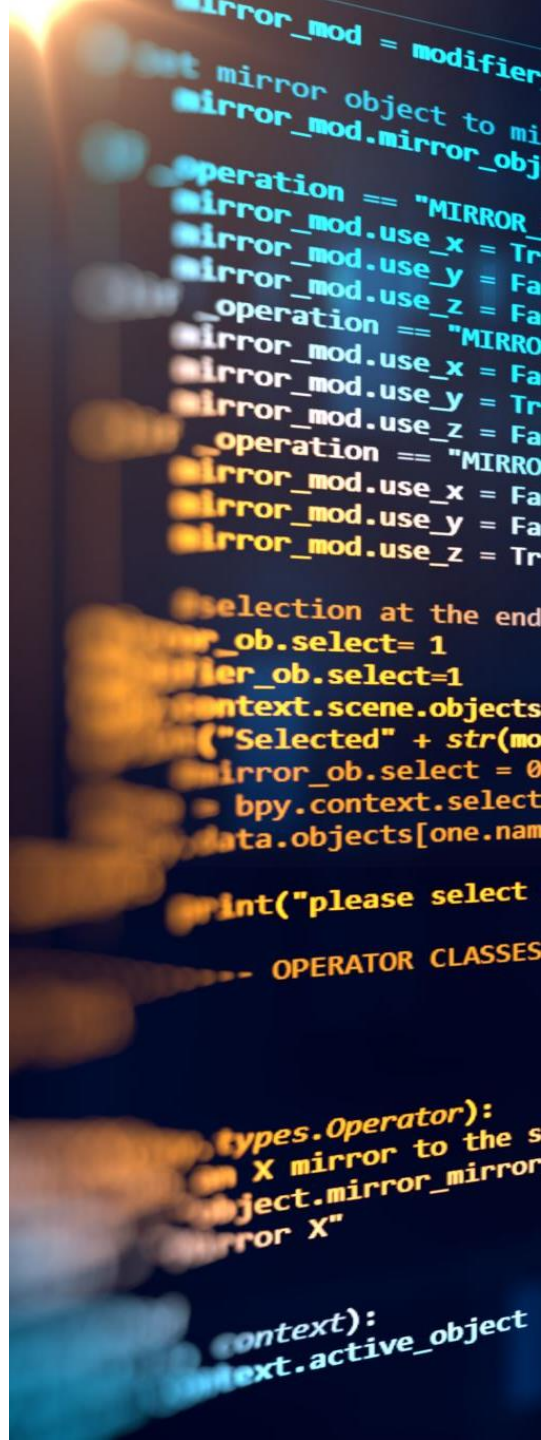
CS 1037
Fundamentals of Computer Science II

Bitwise Operators in C

Ahmed Ibrahim



Bitwise Operators in C

- Bitwise operators manipulate one or more bits from integral operands like char, int, short, and long.
- There are six types of bit operators:
  - AND (&) - Performs a bitwise AND operation between two operands.
  - OR (|) - Performs a bitwise OR operation.
  - XOR (^) - Performs a bitwise exclusive OR operation.
  - Complement (~) - Flips all the bits in an operand (bitwise NOT).
  - Left Shift (<<) - Shifts the operand bits to the left.
  - Right Shift (>>) - Shifts the bits of the operand to the right.

Bitwise AND (&)

Truth Table

xi	yi	xi & yi
0	0	0
0	1	0
1	0	0
1	1	1

Example

Variable	b3	b2	b1	b0
x	1	1	0	0
y	1	0	1	0
z = x & y	1	0	0	0

Bitwise OR (|)

Truth Table

xi	yi	xi | yi
0	0	0
0	1	1
1	0	1
1	1	1

Example

Variable	b3	b2	b1	b0
x	1	1	0	0
y	1	0	1	0
z = x | y	1	1	1	0

Bitwise XOR (^)

Truth Table

xi	yi	xi ^ yi
0	0	0
0	1	1
1	0	1
1	1	0

Example

Variable	b3	b2	b1	b0
x	1	1	0	0
y	1	0	1	0
z = x ^ y	0	1	1	0

Example of AND

```
#include <stdio.h>
int main()
{
    int c1 = 4, c2 = 6, c3;
    c3 = c1 & c2;
    printf("\nBitwise AND i.e. c1 &
    c2 = %d\n", c3);
    return 0;
}
```

Output: `// Perform Bitwise AND`
Bitwise AND i.e. c1 & c2 = 4

00000100 & 00000110

00000100

Example of OR and XOR

```c
#include <stdio.h>

int main()
{
    int c1 = 4, c2 = 6, c3_or, c3_xor;

    // Perform Bitwise OR
    c3_or = c1 | c2;
    printf("\nBitwise OR i.e. c1 | c2 = %d\n", c3_or);

    // Perform Bitwise XOR
    c3_xor = c1 ^ c2;
    printf("Bitwise XOR i.e. c1 ^ c2 = %d\n", c3_xor);

    return 0;
}
```

Output: // Perform Bitwise OR
Bitwise OR i.e. c1 | c2 = 6

00000100 | 00000110

00000110

Output: // Perform Bitwise XOR
Bitwise XOR i.e. c1 ^ c2 = 2

00000100 ^ 00000110

00000010

Left Shifting

Bit positions in an 8-bit binary number

	128	64	32	16	8	4	2	1	
	0	0	0	0	0	1	1	1	= 7
<< 1	0	0	0	0	1	1	1	0	= 14
<< 1	0	0	0	1	1	1	0	0	= 28
<< 1	0	0	1	1	1	0	0	0	= 56
<< 1	0	1	1	1	0	0	0	0	= 112
<< 1	1	1	1	0	0	0	0	0	= 224
<< 1	1	1	0	0	0	0	0	0	= 192 !

Decimal equivalent of each binary value after the left shift.

Example of Shift Operators

This code first performs a left shift by 2 bits on c1 and then a right shift by 2 bits, printing the result of both operations.

```
#include <stdio.h>

int main() {
    char c1 = 1, c2 = 2, c3 = 3;

    // Left shift operation
    c3 = c1 << 2;
    printf("\nLeft shift by 2 bits: c1 << 2 = %d\n", c3);

    // Right shift operation
    c3 = c1 >> 2;
    printf("Right shift by 2 bits: c1 >> 2 = %d\n", c3);

    return 0;
}
```