DSA Study Notes Day 7:

Chapter 7: Bitwise Operators & Scope

Bitwise Operators

Bitwise operators are used to perform bit-level operations on data. These operators directly manipulate bits of integers.

Bitwise AND (`&`)

- Compares each bit of the two numbers. If both bits are `1`, the result is `1`; otherwise, it is `0`.

Example:

```
int a = 4;  // Binary: 0100
int b = 8;  // Binary: 1000
int result = a & b;  // Binary: 0000, result = 0
```

Bitwise OR (`|`)

- Compares each bit of two numbers. If either bit is `1`, the result is `1`.

Example:

```
cpp
Copy code
int a = 4;  // Binary: 0100
int b = 8;  // Binary: 1000
int result = a | b;  // Binary: 1100, result = 12
```

Bitwise XOR (`^`)

- Compares each bit of two numbers. If the bits are different, the result is `1`; if the bits are the same, the result is `0`.

Example:

```
int a = 4;  // Binary: 0100
int b = 8;  // Binary: 1000
int result = a ^ b;  // Binary: 1100, result = 12
```

Bitwise Left Shift (<<)

- Shifts the bits of a number to the left by a specified number of positions, filling with `0` on the right.
- Effectively multiplies the number by `2` for each shift.

Example:

```
int num = 10;  // Binary: 1010
int result = num << 2;  // Binary: 101000, result = 40
```

Bitwise Right Shift (>>)

- Shifts the bits of a number to the right by a specified number of positions, filling with `0` on the left.
- Effectively divides the number by `2` for each shift.

Example:

```
int num = 8;  // Binary: 1000
int result = num >> 2;  // Binary: 0010, result = 2
```

Examples

1. **Bitwise AND (`&`)**
    - `6 & 10`
        - `6` (Binary: `0110`)
        - `10` (Binary: `1010`)
        - Result: `0010` (Decimal: `2`)
2. **Bitwise OR (`|`)**
    - `6 | 10`
        - `6` (Binary: `0110`)
        - `10` (Binary: `1010`)
        - Result: `1110` (Decimal: `14`)
3. **Bitwise XOR (`^`)**
    - `6 ^ 10`
        - `6` (Binary: `0110`)
        - `10` (Binary: `1010`)
        - Result: `1100` (Decimal: `12`)
4. **Left Shift**
    - `10 << 2`
        - `10` (Binary: `1010`)
        - Result: `101000` (Decimal: `40`)
5. **Right Shift**
    - `10 >> 1`

- `10` (Binary: `1010`)
- Result: `0101` (Decimal: `5`)

Operator Precedence

Operator precedence determines the order in which operations are performed in an expression. Operators with higher precedence are executed before those with lower precedence.

Operators	Precedence	Associativity
`!`, `+`, `-` (unary operators)	First	Right to Left
`*`, `/`, `%`	Second	Left to Right
`+`, `-`	Third	Left to Right
`<<`, `>>` (bitwise shift)	Fourth	Left to Right
`<`, `<=`, `>`, `>=` (comparison)	Fifth	Left to Right
`==`, `!=` (equality)	Sixth	Left to Right
`&` (bitwise AND)	Seventh	Left to Right
`^` (bitwise XOR)	Eighth	Left to Right
`	` (bitwise OR)	Ninth
`&&` (logical AND)	Tenth	Left to Right
`		` (logical OR)
`=, +=, -=, *=, /=, %=, <<=, >>=, &=, ^=, =` (assignment operators)	Last	

Scope

- **Local Scope**: Variables declared within a function or a block are accessible only within that function or block.
- **Global Scope**: Variables declared outside all functions are accessible from any part of the program.

Data Type Modifiers

These modify the size or range of data types:

1. **long**: Increases the range of an integer.
2. **short**: Decreases the range of an integer.
3. **long long**: Provides even larger integer ranges.
4. **signed**: Allows both positive and negative values.
5. **unsigned**: Allows only positive values.

Homework Solutions

1. **How to Check if a Number is a Power of 2 (without a loop):**

   A number is a power of `2` if it has only one `1` in its binary representation. This can be checked using the bitwise operation:

   ```
   bool isPowerOfTwo(int n) {
       return (n > 0) && ((n & (n - 1)) == 0);
   }
   ```

Function to Reverse an Integer

Question: Write a function to reverse an integer `n`.

Solution:

```
int reverseNumber(int n) {
    int reversedNum = 0;
    while(n > 0) {
        int lastDigit = n % 10;  // Get the last digit
        reversedNum = reversedNum * 10 + lastDigit;  // Append the last digit
to the reversed number
        n /= 10;  // Remove the last digit from the original number
    }
    return reversedNum;  // Return the reversed number
}
```

Day 7 Notes

Prepared by Munawar Johar