PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ

Operações de Bitwise e shift de bits.

CURITIBA

2025

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ

SUMÁRIO

INTRODUÇÃO

O que são operadores de bitwise?

Operadores bitwise são utilizados quando precisamos realizar operações em nível de bits com números inteiros, ou seja, quando queremos trabalhar diretamente com sua representação binária. Essas operações bit a bit (ou bitwise) e os chamados shifts de bits são consideradas operações de baixo nível, pois atuam diretamente sobre os bits individuais que compõem um número.

Essas operações são extremamente úteis em contextos como programação de sistemas, criptografia, compressão de dados, e manipulação de flags, onde a performance e o controle fino sobre os dados são essenciais.

Como operadores temos o Operador & (Bitwise AND),Operador | (Bitwise OR), Operador ^ (Bitwise XOR), Operador ~ (Bitwise NOT) e também temos o Bit Shift(<<, >>), que embora ambos estejam relacionados com à lógica e manipulação de valores. Temos uma grande diferença no nível de atuação e no tipo de operação que realizam, por isso são citados separadamente.

DESENVOLVIMENTO

Operadores de Bitwise - Detalhada

1.Operador & (Bitwise AND)

O operador &, conhecido como Bitwise AND, realiza uma comparação entre os bits correspondentes de dois valores inteiros. Ele analisa cada posição de bit nos dois números e, para cada par de bits, retorna 1 somente se ambos forem iguais a 1. Se pelo menos um dos bits for 0, o resultado naquele ponto será 0. O resultado final é um novo valor inteiro, construído a partir dessas comparações bit a bit.

a = 10 # 1010
b = 7   # 0111

resultado = a & b # 0010 = 2

a = 14 # 1110
b = 4 # 0100

resultado = a & b # 0100 = 4

a = 9 # 1001
b = 5 # 0101

resultado = a & b # 0001 = 1

2.Opedor | (Bitwise OR)

O operador | (Bitwise OR) compara dois valores utilizando suas representações binárias, gerando um novo valor a partir dessa comparação. Cada bit dos dois valores é analisado e, se pelo menos um dos bits for 1, o resultado naquela posição será 1. Se ambos forem 0, o resultado será 0. Esse operador é utilizado para forçar bits a ficarem ativados, como no caso de configuração de flags, onde é necessário garantir que uma ou mais opções estejam habilitadas.

a = 10 # 1010
b = 7   # 0111

resultado = a | b # 1111 = 15

a = 14 # 1110
b = 4 # 0100

resultado = a | b # 1110 = 14

a = 9 # 1001
b = 5 # 0101

resultado = a | b # 1101 = 13

3.Operador ^ (Bitwise XOR)

O operador ^ (Bitwise XOR) compara dois valores com base em suas representações binárias, gerando um novo valor a partir dessa comparação. Cada bit dos dois valores é analisado e, se os bits forem diferentes, o resultado será 1 (verdadeiro); caso contrário, será 0 (falso). Esse operador é utilizado para obter 1 quando os bits são distintos. É amplamente usado em criptografia e também para trocar valores entre variáveis sem a necessidade de uma variável auxiliar.

a = 10 # 1010
b = 7   # 0111

resultado = a | b # 1101 = 13

a = 14 # 1110
b = 4 # 0100

resultado = a | b # 1011 = 11

a = 9 # 1001
b = 5 # 0101

resultado = a | b # 1100 = 13

4.Operador ~ (Bitwise NOT)

O operador ~ (Bitwise NOT), ao contrário dos operadores anteriores, age sobre um único operando. Ele inverte todos os bits do número, trocando os 0s por 1s e os 1s por 0s. Além disso, ele muda o sinal do número, transformando números positivos em negativos e vice-versa. Ou seja, ele realiza uma inversão completa da representação binária de um número.

a = 10 # 00000000000000000000000000 01010

resultado = ~a # 11111111111111111111111111 10101= -11

a = 14 # 00000000000000000000000000 01110

resultado = ~a # 11111111111111111111111111 10001 = -15

a = 9 # 00000000000000000000000000 01001

resultado = ~a # 11111111111111111111111111 10110 = -10

5.Operador << (Bitwise left shift)

O operador << (shift à esquerda) desloca os bits de um número para a esquerda, preenchendo com zeros à direita. Esse deslocamento é equivalente a multiplicar o valor original por potências de 2 (a << n → equivalente a a * (2 ** n)).

a = 10 # 01010
resultado = a << 1 # 10100 = 20

a = 14 # 01110
resultado = a << 1 # 11100 = 28

a = 9 # 01001

resultado = a << 1 # 10010 = 18

6.Operador >> (Bitwise right shift)

O operador >> (deslocamento de bits para a direita) pode parecer confuso quando analisado em números decimais, mas sua lógica fica clara ao observarmos os valores em formato binário. Esse operador move os bits do número para a direita, e o número à direita do operador indica quantas posições os bits devem ser deslocados. Cada deslocamento equivale a uma divisão inteira por 2.

a = 10 # 01010
resultado = a >> 1 # 00101 = 5

a = 14 # 01110
resultado = a >> 1 # 00111 = 7

a = 6 # 00110
resultado = a >> 1 # 00011 = 3

Conclusão

Em resumo, os operadores bitwise são ferramentas poderosas que atuam diretamente na representação binária dos números, permitindo manipulações eficientes e de baixo nível. Eles são especialmente úteis em áreas como programação de sistemas, criptografia, compressão de dados e desenvolvimento embarcado, onde o desempenho e o controle preciso sobre os dados são essenciais. Entender como funcionam operações como &, |, ^, ~, << e >> é fundamental para aproveitar todo o potencial desses recursos e escrever códigos mais otimizados e eficazes.

Referências

WHITNEY, Tyler; SHARKEY, Kent; SCHONNING, Nick; B, Mike; JONES, Mike; HOGENSON, Gordon; CAI, Saisang. Operadores bit a bit C. [S. l.], 2 abr. 2023. Disponível em: https://learn.microsoft.com/pt-br/cpp/c-language/c-bitwise-operators?view=msvc-170. Acesso em: 24 abr. 2025.

EXPLICANDO operações Bitwise. [*S. l.*], 13 mar. 2021. Disponível em: https://danilopinotti.com.br/blog/explicando_operadores_bitwise. Acesso em: 24 abr. 2025.

CONHEÇA os operadores Bitwise (Bit-a-Bit). [S. l.], 6 mar. 2015. Disponível em: https://imasters.com.br/desenvolvimento/conheca-os-operadores-bitwise-bit-bit. Acesso em: 24 abr. 2025.