# 11. Bit-wise Operators

**11.1 Introduction**

Bit-Wise operatorsను low level operators అని అ0టారు. వీటితో, మనము, bit level లో ప్రోగ్రాములు వ్రాయవచ్చు at. C లా0గ్వేజిలో వున్న Bit-Wise operators, complement, AND, OR, XOR, left shift(<<) and right shift(>>) అనే వాటి గురి0చి ఇక్కడ నేర్చుకొ0దాము. ఈ operators అన్నీ కూడా integer టైప్ operands మీదనే పని చేస్తాయి.

**Bit-Wise complement operator (~)**

ఇది unary operator, అనగా ఒక్క operand మాత్రమే వు0టు0ది. ఈ operator అప్లై చేస్తే, operands యొక్క bits complement అవుతాయి; అనగా, సున్నాలు ఒకట్లు, ఒకట్లు సున్నాలు అవుతాయి. ఉదాహరణకు:

**unsigned char a=15; 00001111**

**unsigned char b=~a; 11110000**

అనగా, variable **b** value 240 అవుతు0ది.

Bit-Wise left shift operator (<<)

ఈ operator ద్వారా, దీని మొదటి operand యొక్క bits ను రె0డో operand ఎ0త అయితే అన్ని సార్లు leftకు జరుగుతాయి. రె0డో operand d అయితే, మొదటి operand 2d చేతmultiply అవుతు0ది. ఉదాహరణకు, క్రి0ద exampleలో, variable b వాల్యూ 240 అవుతు0ది, ఎలాగా అనగా **a** ను **4** సార్లు **left shift** చేస్తున్నాము కనుక.

**unsigned char a=15; 00001111**

**unsigned char b=a<<4; 11110000**

## Example 1: ఈ ప్రోగ్రాములో ఓ integer variableకు 1 ఇచ్చి దాని వాల్యూను ప్రతీసారీ ఓ bit ప్రక్కకు జరుపుతూ ప్రి0ట్ చేస్తున్నాము. అప్పుడు మనకు, 1,2,4,8,... వ0టి న0బర్లు ప్రి0ట్ అవుతాయి.

**#include<stdio.h>**

**int main() {**

**unsigned char a=1;**

**printf("\n\n\n");**

**while(a){**

**printf("%d\n", a);**

**a <<=1;**

**}**

**return 0;**

**}**

**Bit-Wise right shift operator (>>)**

ఈ operator ద్వారా, దీని మొదటి operand యొక్క bits ను రె0డో operand ఎ0త అయితే అన్ని సార్లు right కు జరుగుతాయి. రె0డో operand d అయితే, మొదటి operand 2d చేత divide అవుతు0ది. ఉదాహరణకు, క్రి0ద ,exampleలో, variable b వాల్యూ 15 అవుతు0ది **a** ను **4** సార్లు **right shift** చేస్తున్నాము కనుక.

**unsigned char a=240; 11110000**

**unsigned char b=a>>4; 00001111**

**క్రి**0**ద ఇచ్చిన Table bitwise AND(&), OR(|), exclusive-OR(^) యొక్క** result ను చూపిస్తు0ది.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| X | Y | X&Y | X|Y | X^Y |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 |

**Bit-Wise AND Operator (&)**

ఈ operator అప్లై చేస్తే, రె0డు operands యొక్క respective bits 1’s అయితే result 1 అవుతు0ది, లేకపోతే result 0 అవుతు0ది. ఈ క్రి0ది ఉదాహరణను చూడ0డి.

**unsigned char a=15; 00001111**

**unsigned char b=197; 11000101**

**unsigned char z; \_\_\_\_\_\_\_\_\_**

**z=a&b; 00000101**

## Example 2: ఈ ప్రోగ్రాము ఓ integer ను రీడ్ చేసి అది even లేక odd అని చెపుతు0ది. అన్ని even న0బర్లకు last bit 0 వు0టు0ది, odd వాటికి 1 వు0టు0ది.

**#include<stdio.h>**

**int main(){**

**int Num;**

**unsigned mask=1;**

**printf(“Enter a number”);**

**scanf(“%d”, &Num);**

**if(Num&mask)**

**printf(“Odd Number\n”);**

**else**

**printf(“Even Number\n”);**

**return 0;**

**}**

**ఉదాహరణలు:**

**Num(17) 00010001**

**mask(1) 00000001**

**AND result 00000001. So, న**0**బరు odd.**

**Num(18) 00010010**

**mask(1) 00000001**

**AND result 00000000 So, న**0**బరు even**

**Bit-Wise OR Operator (|)**

ఈ operator అప్లై చేస్తే, ఏ ఒక్క operand యొక్క respective bits 1’s అయినా result 1 అవుతు0ది, లేకపోతే result 0 అవుతు0ది. ఈ క్ర్0ది ఉదాహరణను చూడ0డి.

**unsigned char a=15; 00001111**

**unsigned char b=197; 11000101**

**unsigned char z; \_\_\_\_\_\_\_\_\_**

**Z=a|b; 11000101**

**Bit-Wise Exclusive Operator (^)**

ఈ operator అప్లై చేస్తే, రె0డు operands యొక్క respective bits opposite అయితే result 1 అవుతు0ది, లేకపోతే result 0 అవుతు0ది. ఈ క్రి0ది ఉదాహరణను చూడ0డి.

**unsigned char a=15; 00001111**

**unsigned char b=197; 11000101**

**unsigned char z; \_\_\_\_\_\_\_\_\_**

**Z=a^b; 11001000**

**Example 3:** ఈ ప్రోగ్రాము రె0డు integers తీసుకొని వాటి వాల్యూస్ ను exchange చేస్తు0ది.

**#include<stdio.h>**

**int main(){**

**int p,q;**

**scanf(“%d%d”,&p,&q);**

**p=p^q;**

**q=p^q;**

**p=p^q;**

**printf("%d %d\n", p, q);**

**return 0 ;**

**}**

**ఉదాహరణ: p=15 and q=197.**

**p(15) 00001111**

**q(197) 11000101**

**p=p^q 11001010**

**q=p^p 00001111**

**p=p^q 11000101**

### 11.2 Bit-Fields

C లా0గ్వేజిలో, special టైప్ strucre, bit-field వు0ది. ఇక్కడ, ప్రతీ మె0బరుకు ఎన్ని bits కావాలో ఈ క్రి0ది విధముగా చెపుతాము.

**unsigned identifier: bit-length**;

**unsigned : bit-length**;

వీటి వలన మనము కొ0త memory సేవ్ చెయ్యవచ్చు. ఎ0త సేవ్ చేస్తాము అనేది, మన computer యొక్క architecture మీద అధారపడుతు0ది. ఈ టైప్ strucre ను అన్ని మె0బరులు positive వాల్యూస్ తీసుకొనేవి అయితేనే వాడగలము. Bit-fieldsను వేరే strucreలలో కూడా వాడవచ్చు. కానీ, వీటిలో వు0డే మె0బర్లకు మనము address operator ( & )ను అప్లై చెయ్యలేము.

**Example 4: ఈ ప్రోగ్రాము ఓ** integer ను రీడ్ చేసి దాని 2’s complement ను ప్రి0ట్ చేస్తు0ది.

**#include<stdio.h>**

**struct part{**

**unsigned val :15;**

**unsigned sign:1;**

**};**

**int main(){**

**static union compl{**

**int val;**

**struct part i\_val;**

**} input;**

**scanf("%d", &input.val);**

**input.i\_val.sign=(input.i\_val.sign>=0);**

**printf("%d %d %d \n", input.i\_val.sign, input.i\_val.val, input.val);**

**return 0;**

**}**

**11.3 Conclusions**

ఈ చాప్టరులో Bit-Wise operators గురు0చి నేర్చుకొన్నాము. కొన్ని ప్రాక్టికల్ ఉదాహరణలు కూడా ఇచ్చాము.