C Programming

Class- BCA IInd Semester

Dr. Dharm Raj Singh

Assistant Professor, (HOD)

Department of Computer Application

Jagatpur P. G. College, Varanasi

Affiliated to Mahatma Gandhi Kashi Vidhyapith Varanasi

Email- dharmrajsingh67@yahoo.com

Outline

unit 5：Preprocessor

- Bitwise Operators
- Bitwise Shift Operators
- Masking
- XOR masking
- Bit Fields

Bitwise Operators

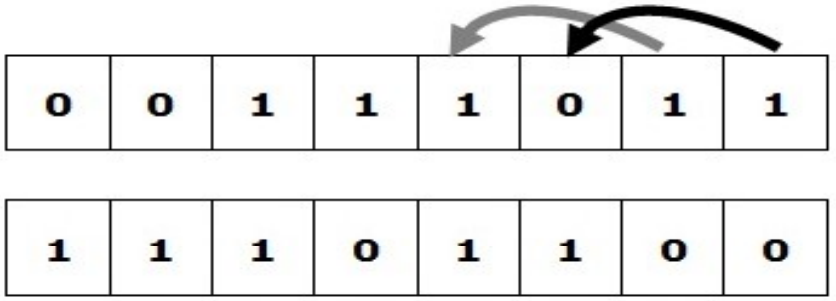
Bitwise operator works on bits and performs bit-by-bit operation.

Operator	Meaning	Remark
&	Bitwise AND	101 & 010 = 000
|	Bitwise OR	101 | 010 = 111
^	Bitwise EX-OR	111 ^ 010 = 101
<<	Shift left	<<100 = 001
>>	Shift right	>>100 = 010
~	Ones complement	~101 = 010

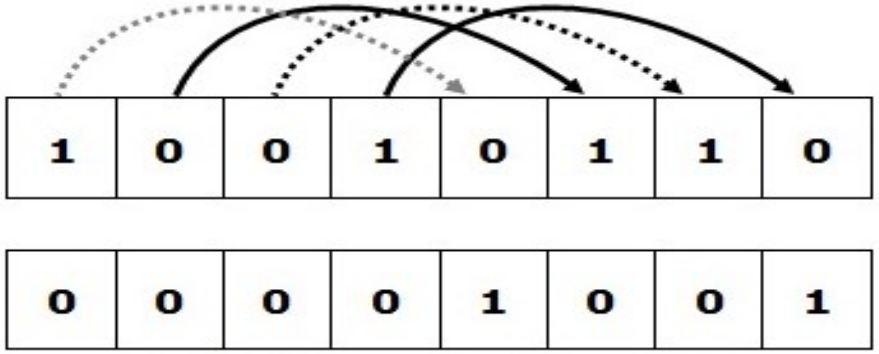
Bitwise Shift Operators (<<, >>)

- There are two shift operators – **Left shift** and **Right shift**. These operators shift the bits by the corresponding value, in other word's move the bits. The sign **<<** for left shift and **>>** for right shift.

For example C = A << 2; // left shift A by 2



For example D = B >> 4; // right shift B by 4



```
#include <stdio.h>   /*BITWISE*/
#include<conio.h>
void main()
{
  unsigned int a = 60;                    /* 60 = 0011 1100 */
  unsigned int b = 13;                    /* 13 = 0000 1101 */
  int c = 0;
  c = a & b;                             /* 12 = 0000 1100 */
  printf("Line 1 - Value of c is %d\n", c );
  c = a | b;                              /* 61 = 0011 1101 */
  printf("Line 2 - Value of c is %d\n", c );
  c = a ^ b;                              /* 49 = 0011 0001 */
  printf("Line 3 - Value of c is %d\n", c );
  c = ~a;                                 /*-61 = 1100 0011 */
  printf("Line 4 - Value of c is %d\n", c );
  c = a << 2;                            /* 240 = 1111 0000 */
  printf("Line 5 - Value of c is %d\n", c );
  c = a >> 2;                             /* 15 = 0000 1111 */
  printf("Line 6 - Value of c is %d\n", c );
  getch();
}
```

Masking

Masking is an operation in which we can selectively mask or filter the bits of a variable, such that some bits are to keep/change/remove a desired part of information.

Masking using Bitwise AND: More often in practice bits are "masked off" (or masked to 0) than "masked on" (or masked to 1). When a bit is ANDed with a 0, the result is always 0, i.e. Y AND 0 = 0. To leave the other bits as they were originally, they can be ANDed with 1, since Y AND 1 = Y.

```
  10100101 10100101
  00001111 00001111 (AND mask bits)
= 00000101 00000101
```

ch	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
mask	0	0	0	0	1	0	0	0
result = ch & mask	0	0	0	0	b_3	0	0	0

Masking using Bitwise OR: the principle of OR masking is that Y OR 1 = 1 and Y OR 0 = Y. Therefore, to make sure a bit is on, OR can be used with a 1. To leave a bit unchanged, OR is used with a 0.

```
  10010101 10100101
  11110000 11110000  (OR masking bits)
= 11110101 11110101
```

	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
ch	1	0	0	0	0	0	1	1
mask	0	0	0	0	1	0	0	0
ch | mask	1	0	0	0	1	0	1	1

XOR masking

- This can be achieved using the XOR (exclusive or) operation. XOR returns 1 if and only if an odd number of bits are 1. Therefore, if two corresponding bits are 1, the result will be a 0, but if only one of them is 1, the result will be 1. Therefore inversion of the values of bits is done by XORing them with a 1. If the original bit was 1, it returns 1 XOR 1 = 0. If the original bit was 0 it returns 0 XOR 1 = 1. Also note that XOR masking is bit-safe, meaning that it will not affect unmasked bits because Y XOR 0 = Y.

```
  10011101 10010101
  00001111  11111111  (XOR masking bits)
= 10010010 01101010
```

Bit Fields

- In C, we can specify size (in bits) of structure and union members. The idea is to use memory efficiently when we know that the value of a field or group of fields will never exceed a limit or is withing a small range.
- For example, consider the following declaration of date without the use of bit fields.

```
struct date {
   unsigned int d;
   unsigned int m;
   unsigned int y;
};

int main()
{
   printf("Size of date is %u bytes\n",  sizeof(struct date));
   struct date dt = { 31, 12, 2014 };
   printf("Date is %d/%d/%d", dt.d, dt.m, dt.y);
}
```

Output: Size of date is 12 bytes
Date is 31/12/2014

➢The above representation of ‘date’ takes 12 bytes on a compiler where an unsigned int takes 4 bytes. Since we know that the value of d is always from 1 to 31, the value of m is from 1 to 12, we can optimize the space using bit fields.

```
struct date
 {
    unsigned int d : 5;      // d has value between 1 and 31, so 5 bits  are
sufficient
    unsigned int m : 4;      // m has value between 1 and 12, so 4 bits are
sufficient
    unsigned int y;
};
 int main()
{
   printf("Size of date is %u bytes\n", sizeof(struct date));
   struct date dt = { 31, 12, 2014 };
   printf("Date is %d/%d/%d", dt.d, dt.m, dt.y);
   return 0;
}
```

Output: Size of date is 8 bytes
Date is 31/12/2014

```c
#include <stdio.h>

main()
{
        int x = 50, k=10;
        clrscr();
        printf("\n Value of x is %d ", x);
        printf("\n\n Value of k is %d ", k);

        printf("\n\n k = %d is Masking the value of x = %d \n", k, x);
        x = x ^ k;
        printf("\n After XOR Masking the Value of x is %d \n" , x);

        printf("\n k = %d is Masking the Changed Value of x = %d again", k,
        x = x ^ k;
        printf("\n\n Now the Value of x is changed again to %d " , x);
}
```

```
Output
Value of x is 50                //      Bit-Pattern : 00110010
Value of k is 10                //      Bit-Pattern : 00001010
k = 10 is Masking the value of x = 50
After XOR Masking the Value of x is 56          // Resultant    : 00111000
k = 10 is Masking the Changed Value of x = 56 again
Now the Value of x is changed again to 50       // Resultant    : 00110010
```

Exercise

1. Write a program to show right and left shifting using bitwise operators.
2. Write a program to illustrate the bitwise operator.
3. The information about colors is to be stored in bits of a **char** variable called **color. The bit number 0 to 6, each represent 7** colors of a rainbow, i.e. bit 0 represents violet, 1 represents indigo, and so on. Write a program that asks the user to enter a number and based on this number it reports which colors in the rainbow does the number represents.
4. In an inter-college competition, various sports and games are played between different colleges like cricket, basketball, football, hockey, lawn tennis, table tennis, carom and chess. The information regarding the games won by a particular college is stored in bit numbers 0, 1, 2, 3, 4, 5, 6, 7 and 8 respectively of an integer variable called **game. The college** that wins in 5 or more than 5 games is awarded the Champion of Champions trophy. If a number is entered through the keyboard, then write a program to find out whether the college won the Champion of the Champions trophy or not, along with the names of the games won by the college.

References

- Kanetkar, Yashavant P. "Let UsC Fifth Edition." (2017).
- Kernighan, Brian W., and Dennis M. Ritchie. *The C programming language*. 2006.
- Ritchie, Dennis M., Brian W. Kernighan, and Michael E. Lesk. The C programming language. Englewood Cliffs: Prentice Hall, 1988.
- McGraw-Hill, Herbert Schildt Tata. "The Complete Reference C fourth Edition”." (2005).
- Griffiths, David, and Dawn Griffiths. Head First C: A Brain-Friendly Guide. " O'Reilly Media, Inc.", 2012.
- Programming in C-Balguruswamy
- Structured programming approach using C-Forouzah & Ceilberg Thomson learning Publication

Declaration

“The content is exclusively meant for academic purpose and for enhancing teaching and learning. Any other use for economic/commercial purpose is strictly prohibited. The users of the content shall not distribute, disseminate or share it with anyone else and its use is restricted to advancement of individual knowledge. The information provided in this e-content is authentic and best as per knowledge”.

Dr. Dharm Raj Singh
Assistant Professor, (HOD)
Department of Computer Application
JagatpurP. G. College, Varanasi

Thanks