Home Work No.6

10th October 2022

Content:

1) Review
2) Bitwise operators
3) Practice Codes
4) Home Tasks

Review:

C++ is/has:

1. Strictly typed
2. Static typed
3. Compiled language
4. Faster and resource and performance efficient code.
5. Designed to develop Operating system and embedded software
6. Curly brackets { … } used for blocks
7. Block have local scope
8. Collection of statements/Commands
9. Statement terminator ";"
10. Contains most of programming constructs
11. Best programming language for teaching and learning?
12. Should tell students about selecting appropriate data selection.

Remember:

Data is stored in binary form in system memory.

- Integral data [char (1byte), short (2 bytes), int (4 bytes) and long (8 bytes)] are by default signed and stored in 2's complement binary form. We can make Integral data unsigned explicitly by using reserved keyword unsigned.

• Floating-point data float (4-bytes), double (8-bytes) and long double (16-bytes) are represented in binary notation using IEEE-32 bit and IEEE-64 bit floating point binary representation.

Bitwise operators:

Six bitwise operators:

Operator	Symbol	Form	Operation
left shift	<<	x << y	all bits in x shifted left y bits
right shift	>>	x >> y	all bits in x shifted right y bits
bitwise NOT	~	~x	all bits in x flipped
bitwise AND	&	x & y	each bit in x AND each bit in y
bitwise OR	|	x | y	each bit in x OR each bit in y
bitwise XOR	^	x ^ y	each bit in x XOR each bit in y

1. Bitwise AND operator &

The output of bitwise AND is 1 if the corresponding bits of two operands is 1. If either bit of an operand is 0, the result of corresponding bit is evaluated to 0.

& Operator Truth Table		
Digit 1:	Digit 2:	Result:
0	0	0
0	1	0
1	0	0
1	1	1

Let us suppose the bitwise AND operation of two integers 12 and 25.

```
12 (In decimal) = 0000000000001100 (In Binary)

25 (In decimal) = 0000000000011001 (In Binary)



Bit Operation of 12 and 25

  0000000000001100

& 0000000000011001

  ________________

  0000000000001000 = 8 (In decimal)
```

2. Bitwise OR operator |

The output of bitwise OR is 1 if at least one corresponding bit of two operands is 1. In C++ Programming, bitwise OR operator is denoted by |.

Operator Truth Table		
Digit 1:	**Digit 2:**	**Result:**
0	0	0
0	1	1
1	0	1
1	1	1

```
12 (In decimal) = 0000000000001100 (In Binary)

25 (In decimal) = 0000000000011001 (In Binary)

Bitwise OR Operation of 12 and 25

  0000000000001100

| 0000000000011001

  ________________

  0000000000011101  = 29 (In decimal)
```

3. Bitwise XOR (exclusive OR) operator ^

The result of bitwise XOR operator is 1 if the corresponding bits of two operands are opposite/different. It is denoted by ^.

^ Operator Truth Table		
Digit 1:	**Digit 2:**	**Result:**
0	0	0
0	1	1
1	0	1
1	1	0

```
12 = 0000000000001100 (In Binary)

25 = 0000000000011001 (In Binary)

Bitwise XOR Operation of 12 and 25

  0000000000001100

^ 0000000000011001

  ________________

  0000000000010101  = 21 (In decimal)
```

4. Bitwise complement operator ~

Bitwise compliment operator is an unary operator (works on only one operand). It changes 1 to 0 and 0 to 1, i.e., it gives the 1's complement of a value. It is denoted by ~.

^ Operator Truth Table		
Digit 1:	**Digit 2:**	**Result:**
0	0	0
0	1	1
1	0	1
1	1	0

```
35 (In decimal) = 0000000000100011 (In Binary)



Bitwise complement Operation of 35

~ 0000000000100011

  ____________________

  1111111111011100  = -36 (In decimal)
```

//Execute the following codes and explain what is happened.

……code1.cpp……

```
int main()
{
        cout << (2.2 | 1.1);

}
```

……code1a.cpp……

```
int main()
{
        cout << ('a' | 'b');

}
```

……code2.cpp……

```
int main ()
{
        short  a=0;
        cout<<~a;
        return 0;
}
```

……code3.cpp……

```
int main ()
{
```

```
	short alpha=15;
	short beta = 245;
	cout<<endl;
	cout<< (alpha | beta)<<endl;
	cout<< (alpha & beta)<<endl;
	cout<< ~alpha<<endl;
	cout<< ~beta <<endl;
	cout<< (alpha ^ beta)<<endl;
	return 0;
}
```

……..code4.cpp……

```
int main ()
{
	short a=1;
	a=a<<1;
	cout<<a;
	a=a<<1;
	cout<<a;
	return 0;
}
```

……..code5.cpp……

```
int main ()
{
	short a=1;
```

```
	a=a<<15;

	cout<<a;

	return 0;

}
```

……..code5a.cpp…….

```
int main() {

	short a = 1;

	a = (a << 18);

	cout << a;

}
```

Tasks:

1) Compile all code segments (1 to 4). Write the appropriate reason of the output. (Write outputs on paper appropriately using binary values)

2) Rewrite all code segments (1 to 4) by using hexadecimal data now compare outputs with problem 1's outputs. (Write outputs on paper appropriately using binary values)

3) Rewrite all code segments (1 to 4) by using octal data now compare outputs with problem 1's and 2's outputs.

4) Value masking using Bitwise OR operator (Masking bits to 1/ **MASKED ON**):

To turn certain bits on, the bitwise OR operation can be used, following the principle 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.

Write a C++ program for OR masking using Bitwise OR operator.