Lecture – 6

Operator precedence and associativity:

1. ( ) , [] → Left to right
2. + + (postfix) , - - (postfix) → Right to left
3. ! (not) , ~ (1's complement) , + (unary) , - (unary) , + + (prefix) , - - (prefix) , & (address) , * (indirection) , sizeof → Right to left
4. * , / , % (modulus)→ Left to right
5. + (binary) , - (binary) → Left to right
6. << (shift left) , >> (shift right) → Left to right
7. < , < = , > , > = → Left to right
8. = = , ! = → Left to right
9. & (bitwise AND) → Left to right
10. ^ (bitwise XOR) → Left to right
11. | (bitwise OR) → Left to right
12. && (logical AND) → Left to right
13. | | (logical OR) → Left to right
14. ? : (a ? x : y) → Right to left
15. = , * = , / = , % = , + = , - = , & = , ^ = , | = , << = , >> = → Right to left
16. , (comma) → Left to right

Dealing with expressions:

Example 1:

```
void main(void)
{
int x=2,n=2;
x=n++;
printf("%d",x);
x=++n;
printf("%d",x);
}
```

24

Example 2:

```
void main(void)
{
int x=2, y=3;
x*=y;
printf("%d",x);
x=x*y;
printf("%d",x);
x*=y+1;
printf("%d",x);
}
```

61872

Example 3:

```
void main(void)
{
int x=2, y=2;
printf("%d",x++>y);
printf("%d", x&077);
}
```

03

Example 4:

```
void main(void)
{
int x=2;
printf("%d",!(x)&&x++);
printf("%d", x);
}
```

02

!(x) is false thus it returns 0. If one operand of AND operation is false then the result will be false. In this case complier will return 0 without executing the full expression. So x++ will not execute in this case. This is called "short circuit evaluation".

Example 5:

```
void main(void)
{
int a,b,c,p,q; a=b=c=0;
printf("%d%d%d",a,b,c);
a=(p=2)+(q=3);
p*=c=a+5;
printf("%d%d%d", a,c,p);
}
```

00051020

Example 6:

```
void main(void)
{
int x=5, y=2;
printf("%d",x+++y);
}
```

7

Mark operator: exp1 ? exp2 : exp 3

Example:

```
void main(void)
{
int x=5,y;
y= x>5 ? x-=2 : x+=2;
printf("%d%d",x,y);
}
```

77