Operators and Expressions

Dr. Nachiket Tapas

Based on number of operands

- Unary operator (single operand)
- Binary operator (two operands)
- Ternary operator (three operands)

Types of operators

- Arithmetic operators
- Relational operators
- Logical operators
- Increment and decrement operators
- Assignment operators
- Conditional operators
- Bitwise operators
- Special operators

Arithmetic Operators

+	Addition or unary plus
-	Subtraction or unary minus
*	Multiplication
/	Division
%	Modulo division

Unary Operators

Operators that take only one argument

- -5
- +3
- -no1

The / Operator: for integers

When both operand of / are of type integer

- Result is integer part of the division
- Result is of type integer (floor value of the actual result)

9/2 gives output 4

1/2 gives output 0

The / Operator: for float

When either or both operand of / are of type float

- Result is same as real division
- Result is of type float

9.0/2 gives output 4.5

1.0/2 gives output 0.5

The % Operator

The remainder operator or % operator returns integer remainder of the division.

Both operands must be integer

4%2 gives output 0

31%3 gives output 1

Division / and Remainder %

Second operand cannot be 0

- else run time error

What will be the output of the following?

8/-3

8%-3

Relational Operators

<	is less than
<=	is less than or equal to
>	is greater than
>=	is greater than or equal to
==	is equal to
!=	is not equal to

Relational Operators (contd.)

The result of the expression is always TRUE (1 or non-zero) or FALSE (0).

```
#include<stdio.h>

void main() {

        printf("%d", 8 < 3);

        printf("%d", 8 <= 3);

        printf("%d", 8 > 3);

        printf("%d", 8 >= 3);

        printf("%d", 8 == 3);

        printf("%d", 8 != 3);

}
```

Logical Operators

&&	Logical AND
||	Logical OR
!	Logical NOT

The result of the expression is always TRUE or FALSE.

Truth Table

A	B	A && B
0	0	0
0	1	0
1	0	0
1	1	1

A	B	A || B
0	0	0
0	1	1
1	0	1
1	1	1

A	! A
0	1
1	0

Usage

```
#include<stdio.h>
void main() {
        printf("%d",  3 < 8 && 3 < 9);
        printf("%d",  3 < 8 || 3 > 9);
        printf("%d", !8 );
}
```

Operator Chain

A || B || C || D || …. || Z

Condition check till true is found.

A && B && C && D && …. && Z

Condition check till the end.

Increment and Decrement Operators

Pre-increment	++A
Post-increment	A++
Pre-decrement	--A
Post-decrement	A--

Program

```
#include<stdio.h>

void main() {

        int a = 2;

        printf(“%d”, a++);

        printf(“%d”, ++a);

        printf(“%d”, a--);

        printf(“%d”, --a);

}
```

What about the following program?

```
#include<stdio.h>

void main() {

        int a = 2;

        a++;

        printf(“%d”, a);

        ++a;

        printf(“%d”, a);

}
```

Assignment Operators

A = A + 1	A += 1
A = A - 1	A -= 1
A = A * 5	A *= 5
A = A / 5	A /= 5
A = A % 5	A %= 5

The advantage of assignment operators:

1. Reduced code
2. Evaluated only once

Usage

```
#include<stdio.h>
void main() {
            int I = 1, sum = 0;
            sum += I;
            printf("Sum of numbers till %d is %d\n", I, sum);
            I += 1;
            sum += I;
            printf("Sum of numbers till %d is %d\n", I, sum);
            I += 1;
            sum += I;
            printf("Sum of numbers till %d is %d\n", I, sum);
            I += 1;
}
```

Interesting Code

Code1:

```
A = 5;

A = A++ + 5;
```

O/P: 10

Conditional Operators

- Ternary operator

exp1 ? exp2 : exp3

If exp1 evaluates to true, exp2 is executed. Else exp3 is executed.

Usage

```
#include<stdio.h>
void main() {
        int number, output;
        printf("Enter a numbers:\n");
        scanf("%d", &number);
        output = number % 2 ? 0 : 1;
        printf("Is number even: %d", output);
}
```

Bitwise Operators

&	Bitwise AND
|	Bitwise OR
^	Bitwise XOR
<<	Shift Left
>>	Shift Right

Example

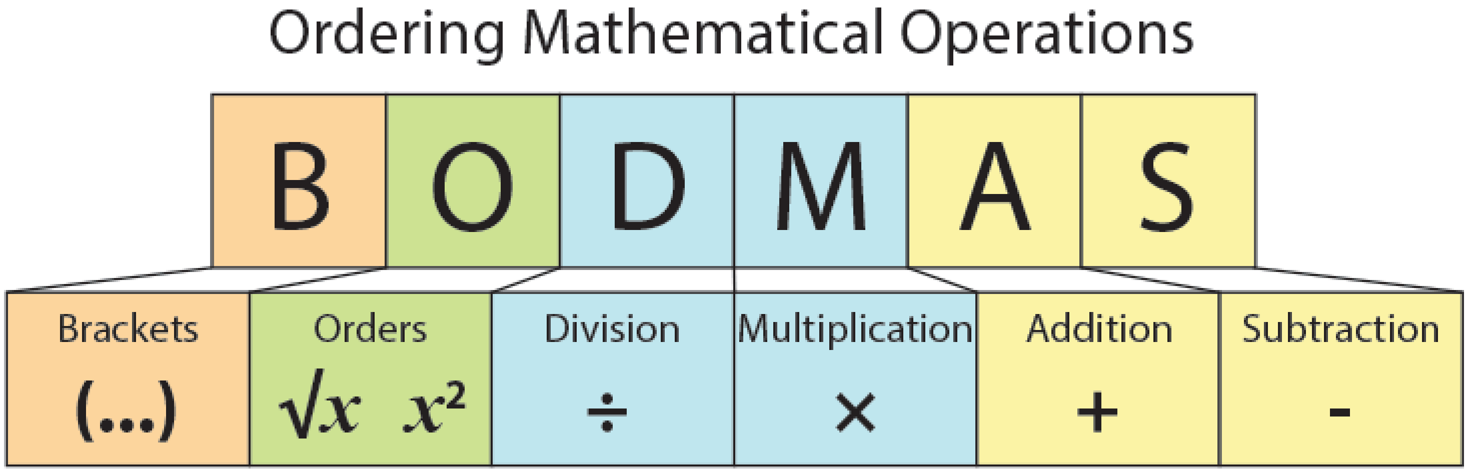
```
#include <stdio.h>
int main() {
        int a = 5, b = 9;
        printf("%d\n", a&b);
        printf("%d\n", a|b);
        printf("%d\n", a^b);
        printf("%d\n", a<<1);
        printf("%d\n", a>>1);
        return 0;
}
```

Special Operators

- comma operator
  - value = (x=5, y=7, x+y);
  - printf(“%d”, value);

- sizeof()
  - printf(“%d”, sizeof(int));

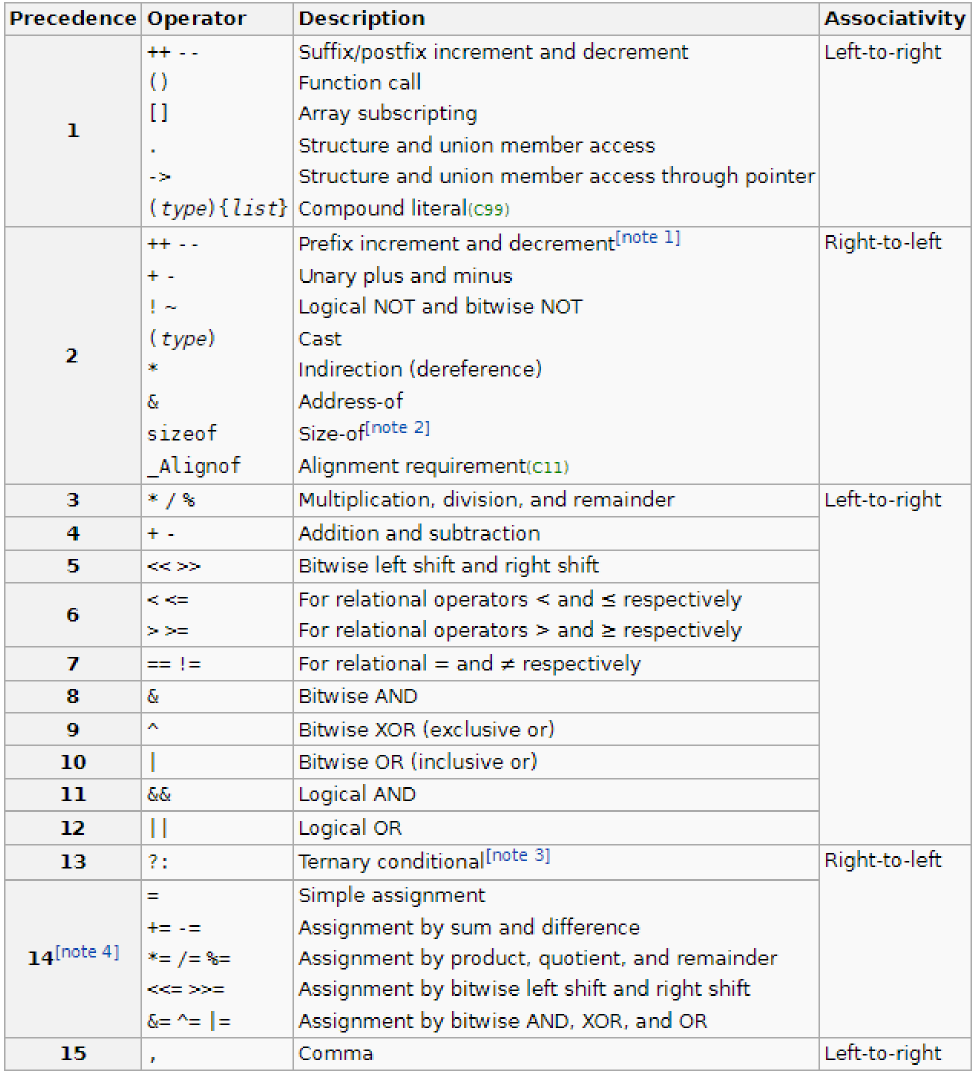
Operator Precedence



Ordering Mathematical Operations

B	O	D	M	A	S
Brackets	Orders	Division	Multiplication	Addition	Subtraction
(...)	$\sqrt{x}$ x^2	÷	×	+	-

Operator Precedence in C



Precedence	Operator	Description	Associativity
1	++ -- () [] . -> (*type*){*list*}	Suffix/postfix increment and decrement Function call Array subscripting Structure and union member access Structure and union member access through pointer Compound literal(C99)	Left-to-right
2	++ -- + - ! ~ (*type*) * & sizeof _Alignof	Prefix increment and decrement[note 1] Unary plus and minus Logical NOT and bitwise NOT Cast Indirection (dereference) Address-of Size-of[note 2] Alignment requirement(C11)	Right-to-left
3	* / %	Multiplication, division, and remainder	Left-to-right
4	+ -	Addition and subtraction	
5	<< >>	Bitwise left shift and right shift	
6	< <= > >=	For relational operators < and ≤ respectively For relational operators > and ≥ respectively	
7	== !=	For relational = and ≠ respectively	
8	&	Bitwise AND	
9	^	Bitwise XOR (exclusive or)	
10	|	Bitwise OR (inclusive or)	
11	&&	Logical AND	
12	||	Logical OR	
13	?:	Ternary conditional[note 3]	Right-to-left
14[note 4]	= += -= *= /= %= <<= >>= &= ^= |=	Simple assignment Assignment by sum and difference Assignment by product, quotient, and remainder Assignment by bitwise left shift and right shift Assignment by bitwise AND, XOR, and OR	
15	,	Comma	Left-to-right

Code

```
#include <stdio.h>
main() {
   int a = 20;
   int b = 10;
   int c = 15;
   int d = 5;
   int e;
   e = (a + b) * c / d; // ( 30 * 15 ) / 5
   printf("Value of (a + b) * c / d is : %d\n", e );
   e = ((a + b) * c) / d; // (30 * 15 ) / 5
   printf("Value of ((a + b) * c) / d is : %d\n" , e );
   e = (a + b) * (c / d); // (30) * (15/5)
   printf("Value of (a + b) * (c / d) is : %d\n", e );
   e = a + (b * c) / d; // 20 + (150/5)
   printf("Value of a + (b * c) / d is : %d\n" , e );
   return 0;
}
```

```
Value of (a + b) * c / d is : 90
Value of ((a + b) * c) / d is : 90
Value of (a + b) * (c / d) is : 90
Value of a + (b * c) / d is : 50
```

Thank You!!