Operators and Expressions

Structured Programming Language (CSE-1271)

Course Instructor : Mohammed Mamun Hossain
Assistant Professor, Dept. of CSE, BAUST

Outline

1. Operators
2. Expressions
3. Some examples

Operators

- An operator is a symbol that tells the computer to perform certain mathematical and logical manipulations. Operators are used in programs to manipulate data and variables. Such as, +, --, <, > etc.

- C operators can be classified into a number of categories. They are as follows:

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

Arithmetic Operators

The arithmetic operators are +, -, *, /,%.

Integer arithmetic: Here operands are integer. For **a=14** and **b=4,**

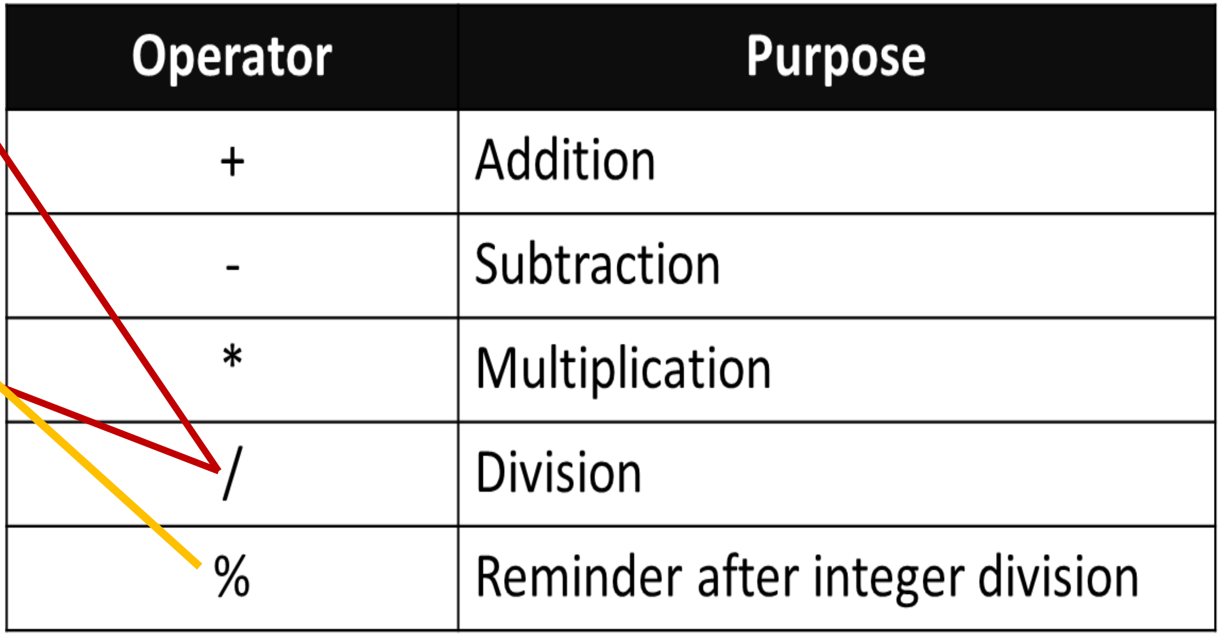
a+b=18 **a/b=3**(decimal part)

a%b=2(remainder of division)

- **Real arithmetic**: Here, operands are only real number. Such as, **if a=6.0** and **b=7.0**

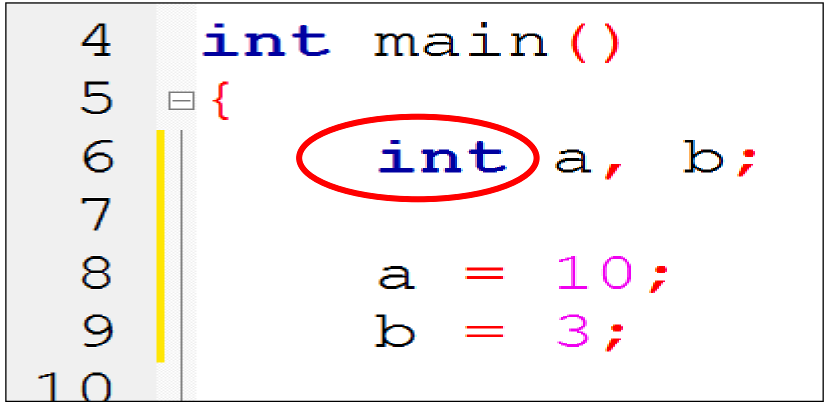
then **a/b=0.857143**

There are five arithmetic operators in C.



Operator	Purpose
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Reminder after integer division

Arithmetic Operators



```
int main()
{
    int a, b;

    a = 10;
    b = 3;
```

a + b → 13

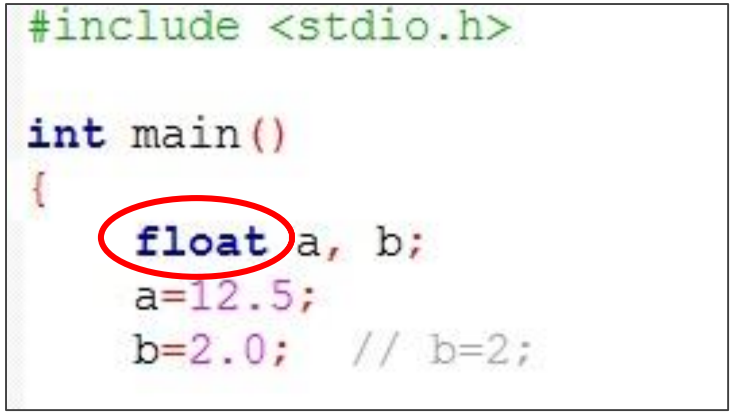
a - b → 7

a * b → 30

a / b → 3 Fractional part omitted.

a % b → 1 Remainder of division

Arithmetic Operators



```
#include <stdio.h>

int main()
{
    float a, b;
    a=12.5;
    b=2.0;   // b=2;
```

a + b → 14.5

a - b → 10.5

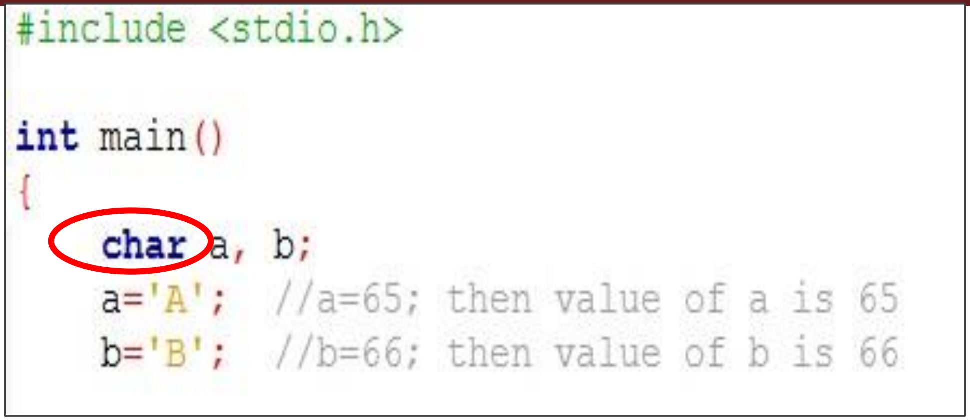
a * b → 25.0

a / b → 6.25

a % b →

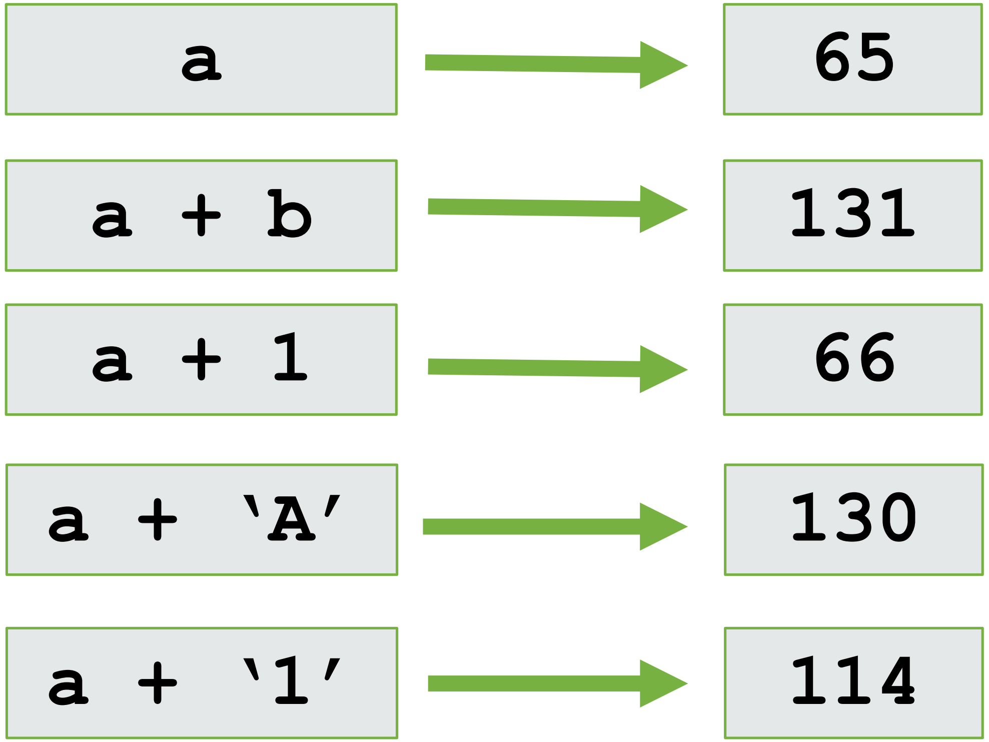
Not Possible!!!

Arithmetic Operators



```
#include <stdio.h>

int main()
{
    char a, b;
    a='A';  //a=65; then value of a is 65
    b='B';  //b=66; then value of b is 66
```



a → 65

a + b → 131

a + 1 → 66

a + ‘A’ → 130

a + ‘1’ → 114

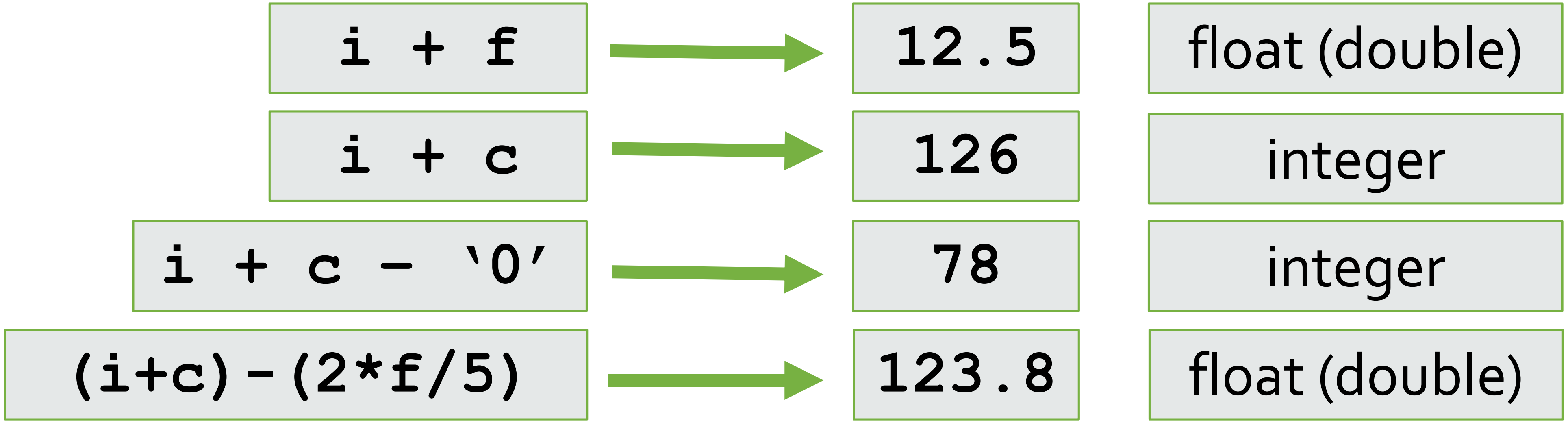
Type Conversion

- Operands that differ in type may undergo type conversion before the expression takes on its final value.

- In general, the final result will be expressed in the highest precision possible, consistent with the data types of the operands.

```
int i = 7;
float f = 5.5;
char c = ‘w’;
```

ASCII Value
w = 119
0 = 48



i + f	12.5	float (double)
i + c	126	integer
i + c – ‘0’	78	integer
(i+c)-(2*f/5)	123.8	float (double)

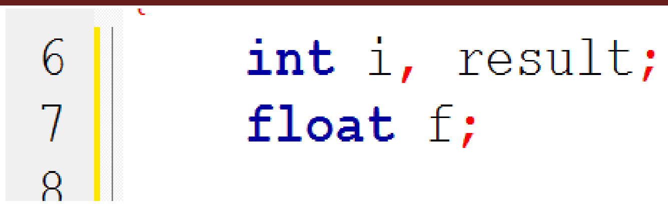
Type Cast

- To transform the type of a variable temporarily.
- To do so, the expression must be preceded by the name of the desired data type, enclosed in parentheses

(data type) expression

```
int number;
(float) number;
```

Check Type Cast



```
    int i, result;
    float f;
```

```
i = 7;
f = 8.5;
```

```
result = (i + f) % 4;
```

Invalid

Again

```
float num = 10.5;
num % 2;
```

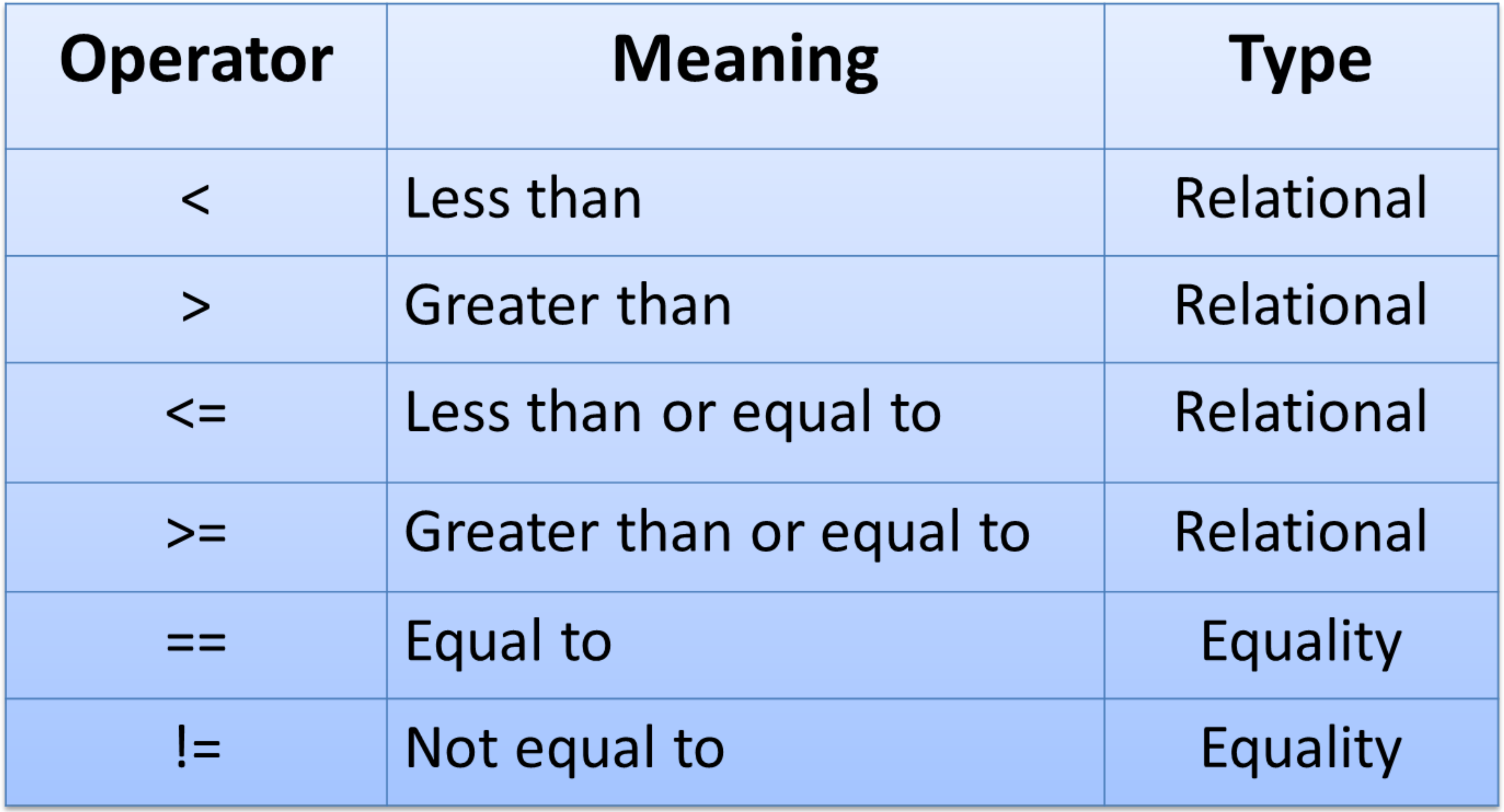
✗

```
float num = 10.5;
((int)num) % 2;
```

✓

Relational Operator

The operators which are used to compare two numbers and take decision depending on their relation are called relational operators.



Operator	Meaning	Type
<	Less than	Relational
>	Greater than	Relational
<=	Less than or equal to	Relational
>=	Greater than or equal to	Relational
==	Equal to	Equality
!=	Not equal to	Equality

Relational Operator

Given the following C declarations:

int **a =1, b = 2, c = 3, d = 1**;

✓ `a == d` is true

✓ `c > b` is true

✓ `c >= b` is true

✓ `a >= c` is false

✓ `a != d` is false

✓ `a <= d` is true

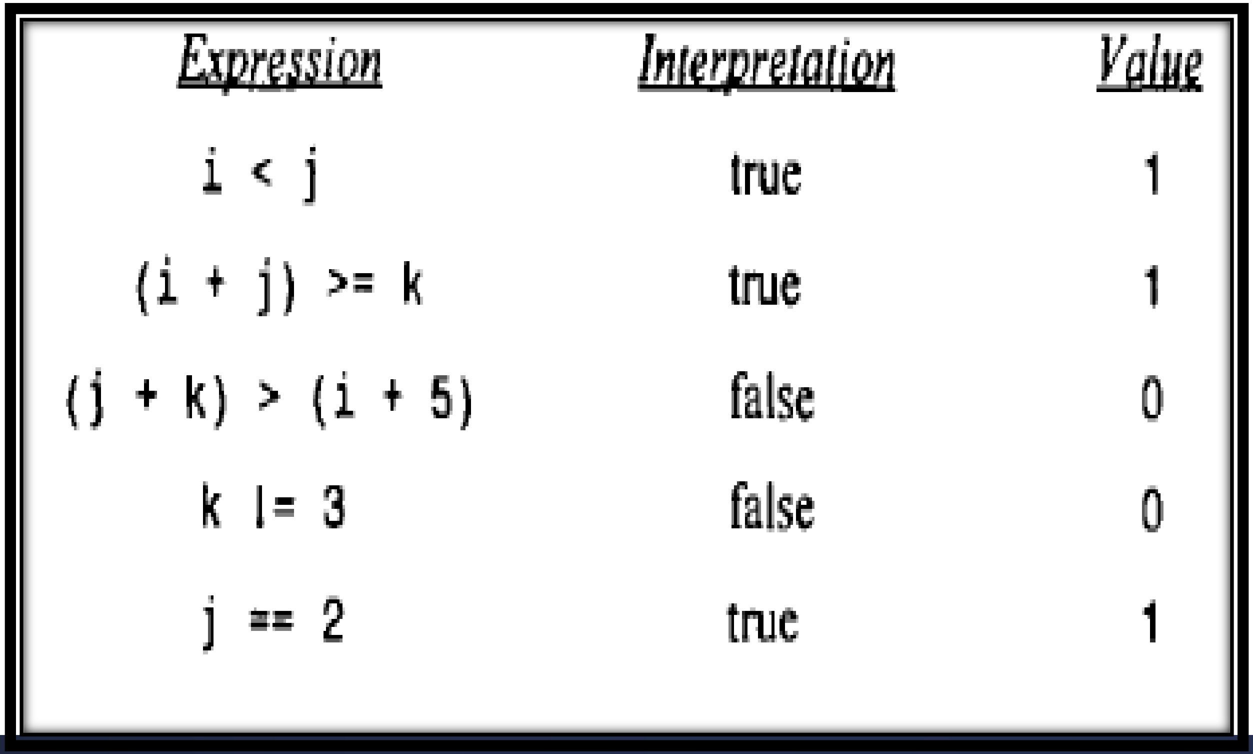
Relational Operator

Suppose that i, j and k are integer variables.

Where, `i=1;`

```
j=2;
k=3;
```

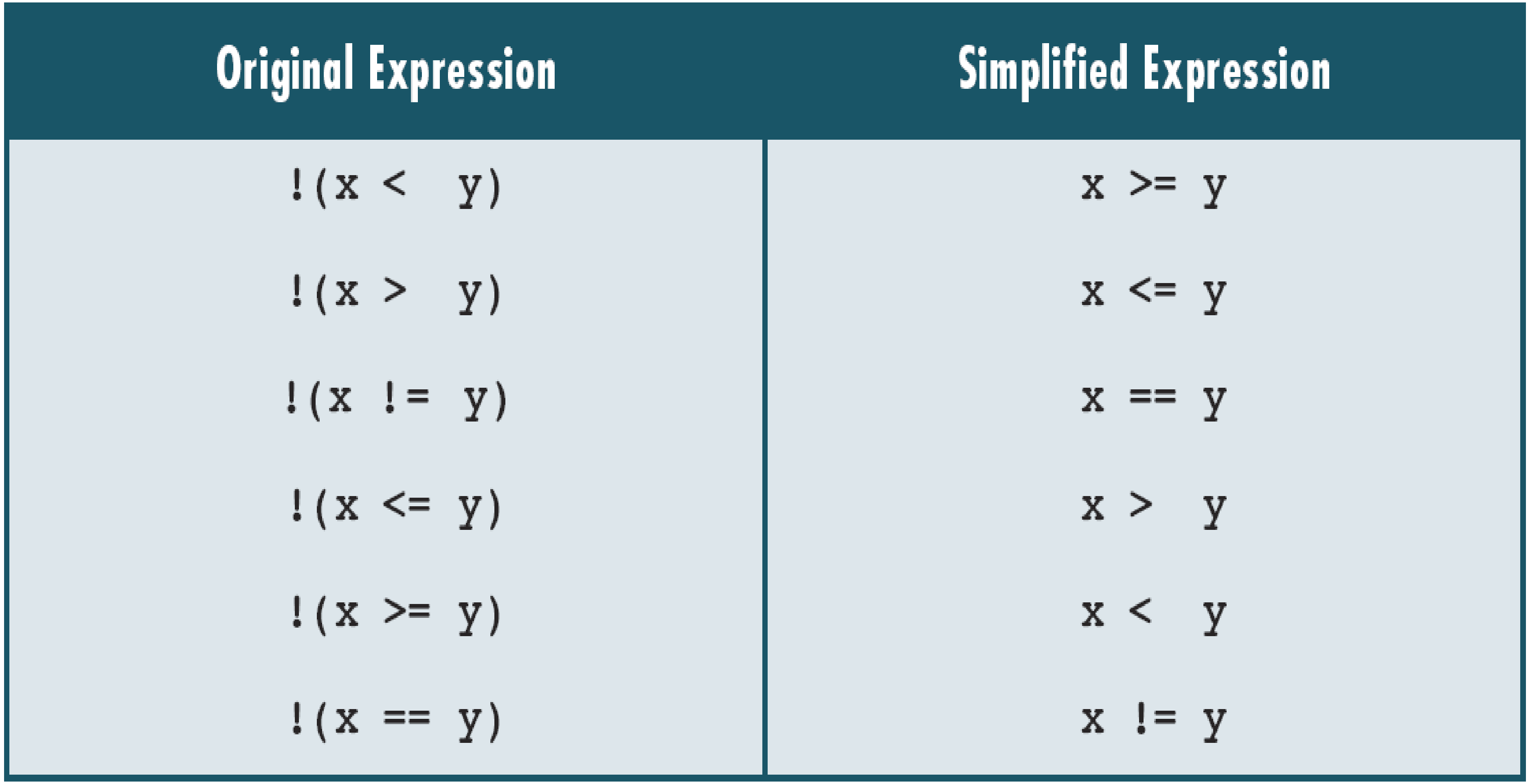
Several logical expressions involving these variables are shown below.



Expression	Interpretation	Value
i < j	true	1
(i + j) >= k	true	1
(j + k) > (i + 5)	false	0
k != 3	false	0
j == 2	true	1

Relational Operator

Simplified Expression

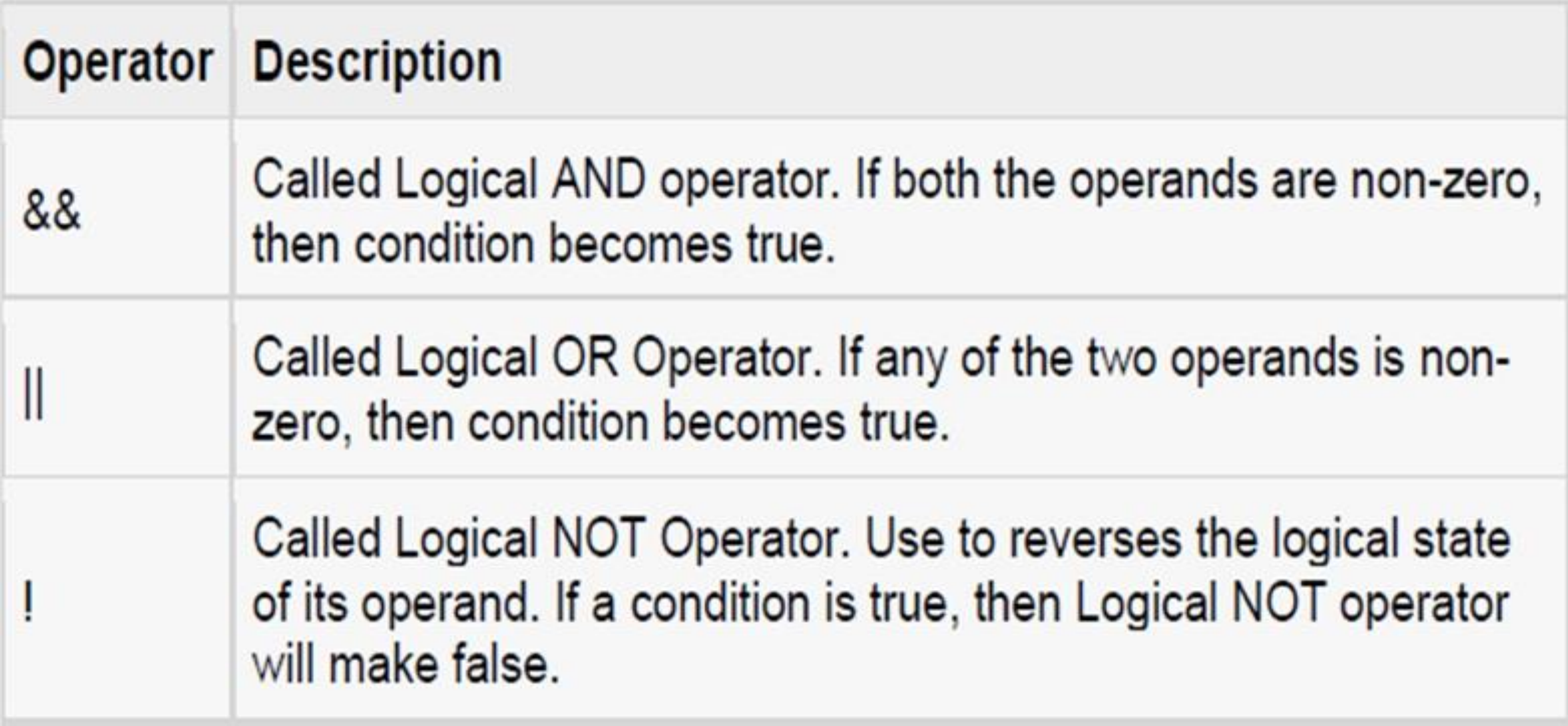


Original Expression	Simplified Expression
`!(x <  y)`	`x >= y`
`!(x >  y)`	`x <= y`
`!(x != y)`	`x == y`
`!(x <= y)`	`x >  y`
`!(x >= y)`	`x <  y`
`!(x == y)`	`x != y`

Logical Operator

Operators which are used to combine two or more relational expressions are known as logical operators.

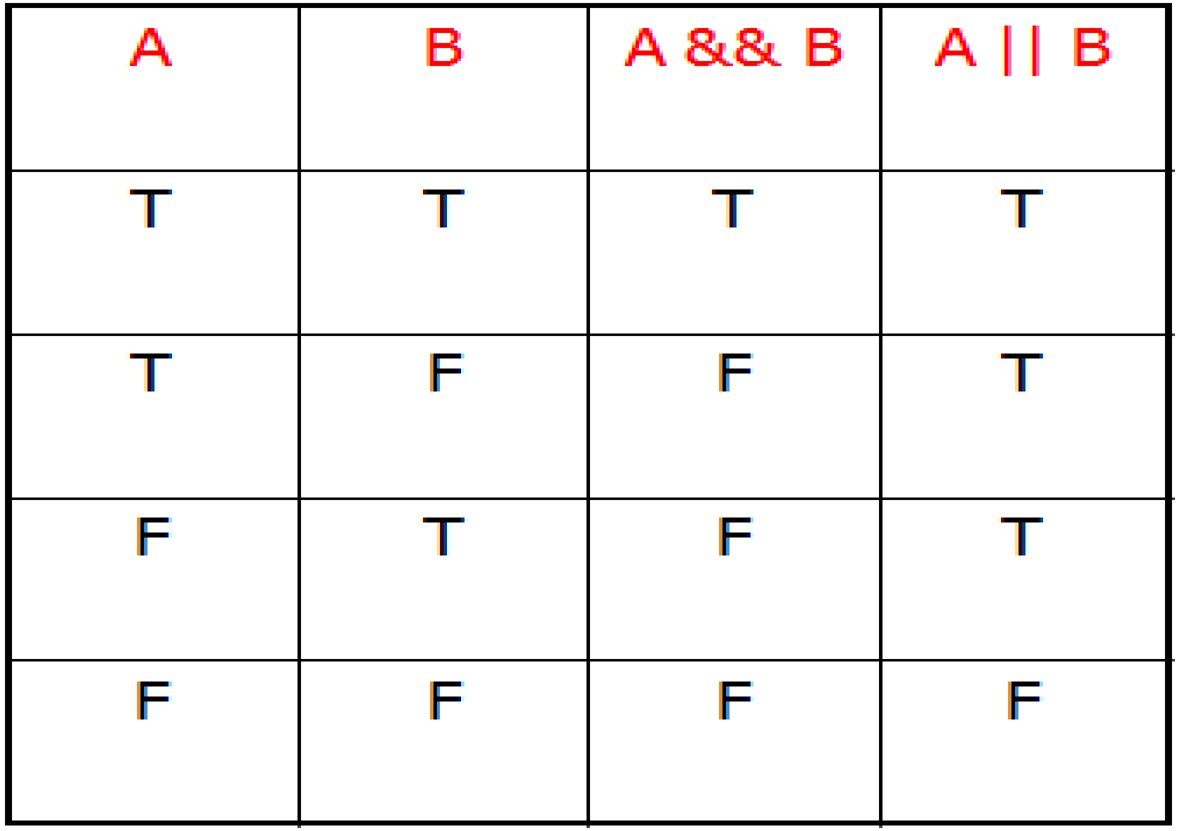
There are three logical operators.



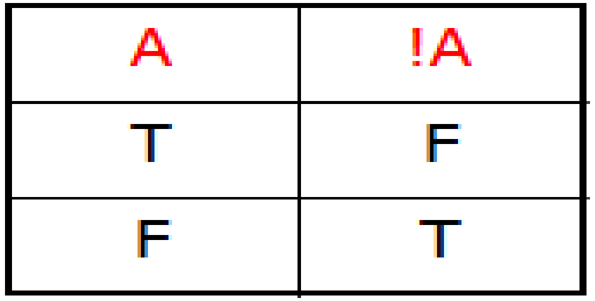
Operator	Description
&&	Called Logical AND operator. If both the operands are non-zero, then condition becomes true.
||	Called Logical OR Operator. If any of the two operands is non-zero, then condition becomes true.
!	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true, then Logical NOT operator will make false.

Logical Operator

Truth table of Logical Operator



A	B	A && B	A || B
T	T	T	T
T	F	F	T
F	T	F	T
F	F	F	F



A	!A
T	F
F	T

Assignment Operator

Operators which are used to assign the result of an expression to a variable are known as assignment operators.

Consider an example:

`x = 100;` meaning: 100 is assigned to x

```
x += (y+1);
```

The operator += means ‘**add y+1 to x**’ or

‘**increment x by y+1**’.

For **y=2;** the above statement results x= 103,

`x += 3;` that is (**x = x + 3**)

Increment Decrement Operator

C allows two very useful operators increment (++) and decrement (--) operators.

The operator **++** adds 1 to the operand, while **--** subtracts 1.

Rules for ++ and – operators:

i) Increment (++) and decrement (--) operators are unary operators

ii) When postfix ++ (or --) is used with a variable in an expression, the expression is evaluated first using the original value of the variable and then the variable is incremented (or decremented) by one.

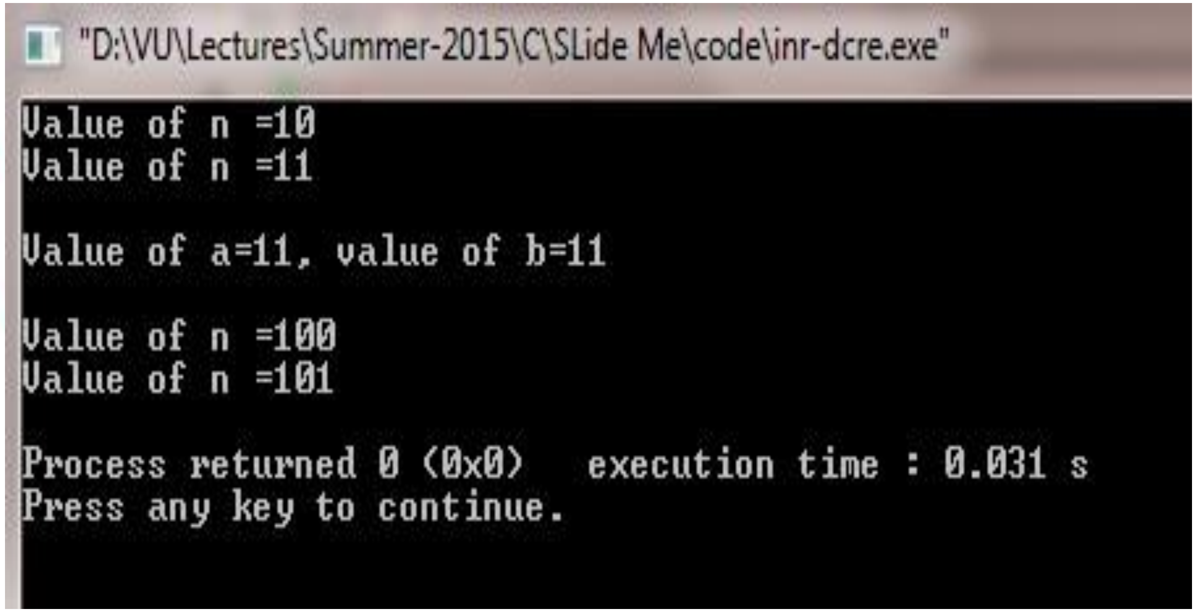
iii) When prefix ++ (or --) is used in an expression, the variable is incremented (or decremented) first and then the expression is evaluated using the new value of the variable.

Increment Decrement Operator

```
int a=10, b=10, n;
n=a++;
printf("Value of n =%d\n",n);
n=++b;
printf("Value of n =%d\n",n);

printf("\nValue of a=%d, value of b=%d\n\n",a,b);

n=100;
printf("Value of n =%d\n",n++);
n=100;
printf("Value of n =%d\n",++n);
```



"D:\VU\Lectures\Summer-2015\C\SLide Me\code\inr-dcre.exe"

```
Value of n =10
Value of n =11

Value of a=11, value of b=11

Value of n =100
Value of n =101

Process returned 0 (0x0)   execution time : 0.031 s
Press any key to continue.
```

Conditional Operator

A ternary operator pair “ **? :** ” is available in C

exp1?exp2:exp3.

- exp1 is evaluated first. If it is nonzero (true) then the expression exp2 is evaluated and becomes the value of the expression.
- if exp1 is false, then exp3 is evaluated and becomes the value of the expression.

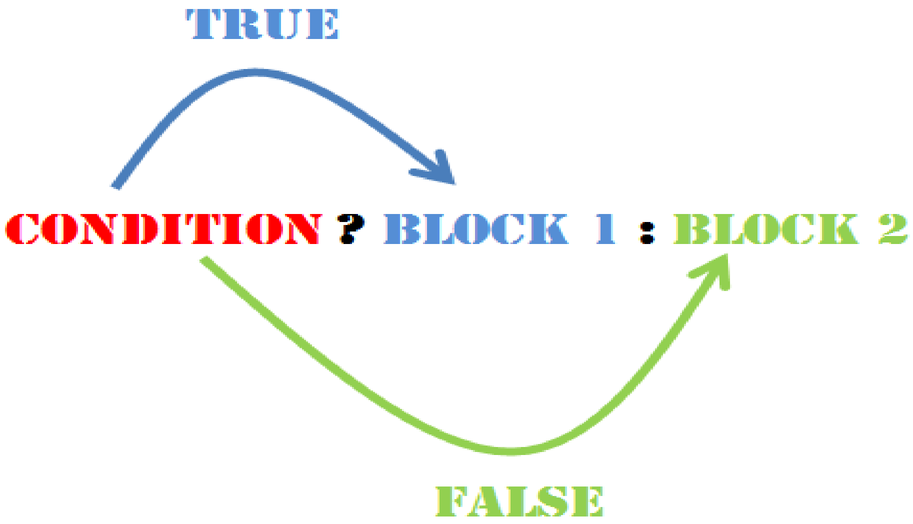
Suppose **a=10;**

b=15;

x=(a>b)?a:b;

In this example,

x will be assigned the value of b.



TRUE

CONDITION ? BLOCK 1 : BLOCK 2

FALSE

Bitwise Operator

Operators which are used to manipulate data at their bit level are known as bitwise operators. Bitwise operators and their meanings are given below,

Assume if A = 60; and B = 13;

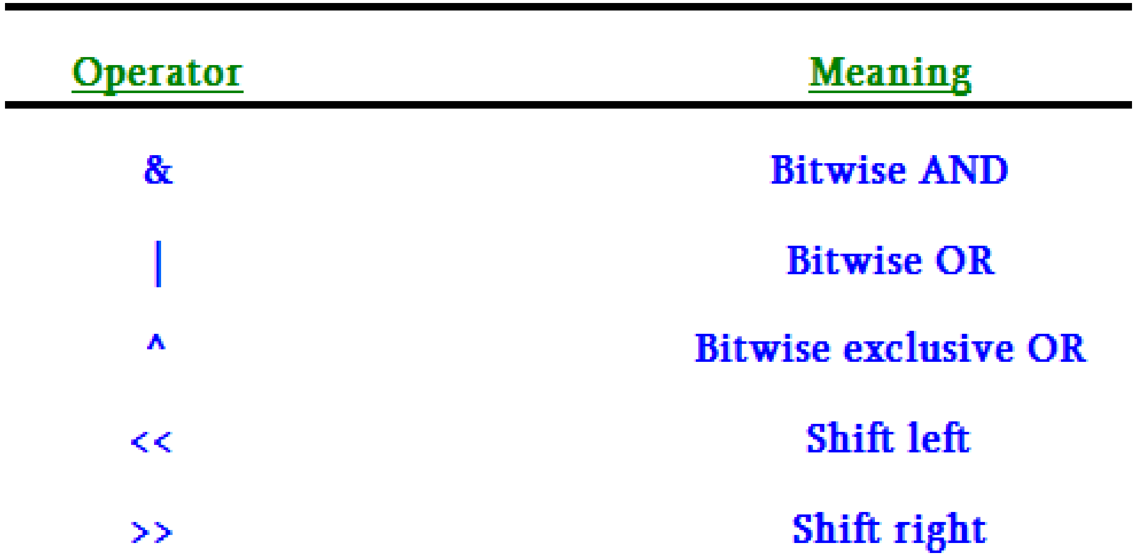
A = 0011 1100

B = 0000 1101

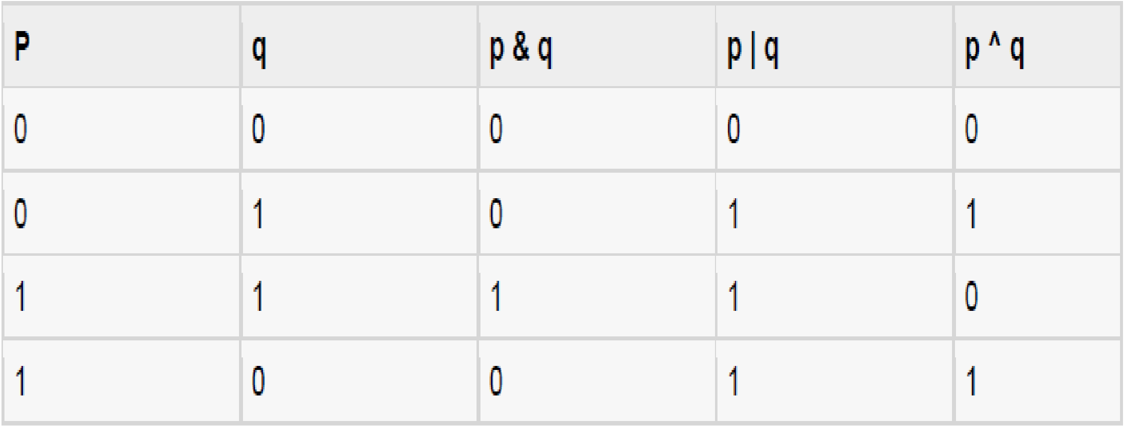
A&B = 0000 1100

A|B = 0011 1101

A^B = 0011 0001



Operator	Meaning
&	Bitwise AND
|	Bitwise OR
^	Bitwise exclusive OR
<<	Shift left
>>	Shift right



p	q	p & q	p | q	p ^ q
0	0	0	0	0
0	1	0	1	1
1	1	1	1	0
1	0	0	1	1

Special Operator

C supports some special operators of interest such as **comma** operator, **sizeof** operator, **pointer** operators (**&** and *****) and **member selection** operators (. and ->).

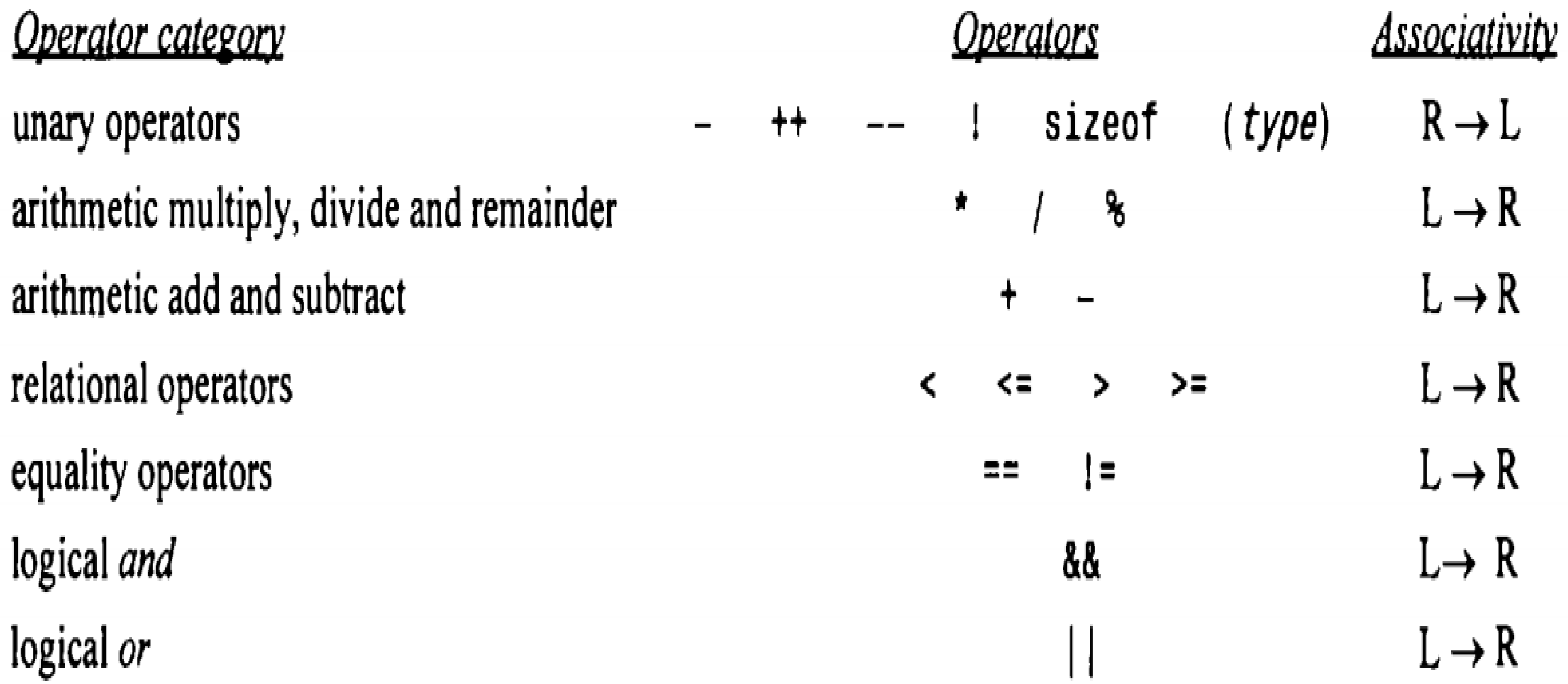
All these are special operators.

Example of sizeof operator:

```
int a;
float b;
double c;
char d;
printf("Size of int=%d bytes\n",sizeof(a));
printf("Size of float=%d bytes\n",sizeof(b));
printf("Size of double=%d bytes\n",sizeof(c));
printf("Size of char=%d byte\n",sizeof(d));
```

Operator Precedence

An operator’s **precedence** determines its order of evaluation.



Operator category	Operators	Associativity
unary operators	- ++ -- ! sizeof (*type*)	R → L
arithmetic multiply, divide and remainder	* / %	L → R
arithmetic add and subtract	+ -	L → R
relational operators	< <= > >=	L → R
equality operators	== !=	L → R
logical *and*	&&	L → R
logical *or*	||	L → R

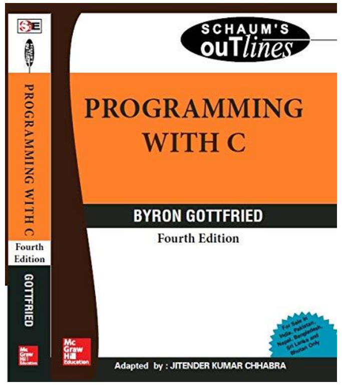
Thank You.

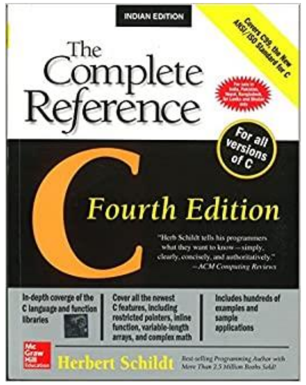
Questions and Answer

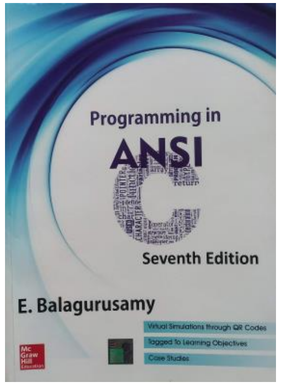
References

Books:

1. Programming With C. *By Byron Gottfried*
2. The Complete Reference C. *By Herbert Shield*
3. Programming in ANSI C *By E. Balagurusamy*
4. Teach yourself C. *By Herbert Shield*







Web:

1. www.wikbooks.org

and other slide, books and web search.