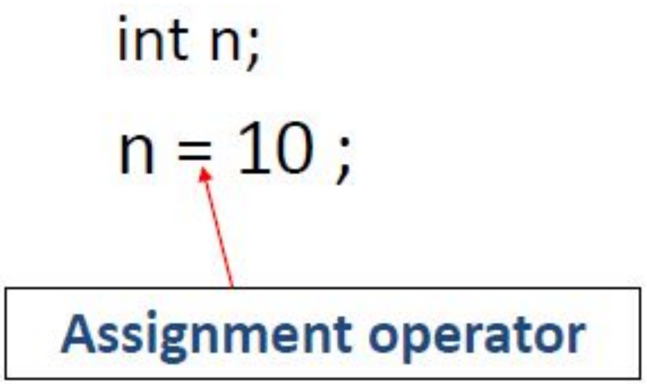
Programming Using C

SCS 1202

Lesson 02 - Operators

Assignment operator

- An assignment statement assigns as expression in particular , a variable to the right of the operator assign to a modifiable value on the left hand side of the assignment .



```
int n;
n = 10 ;
```

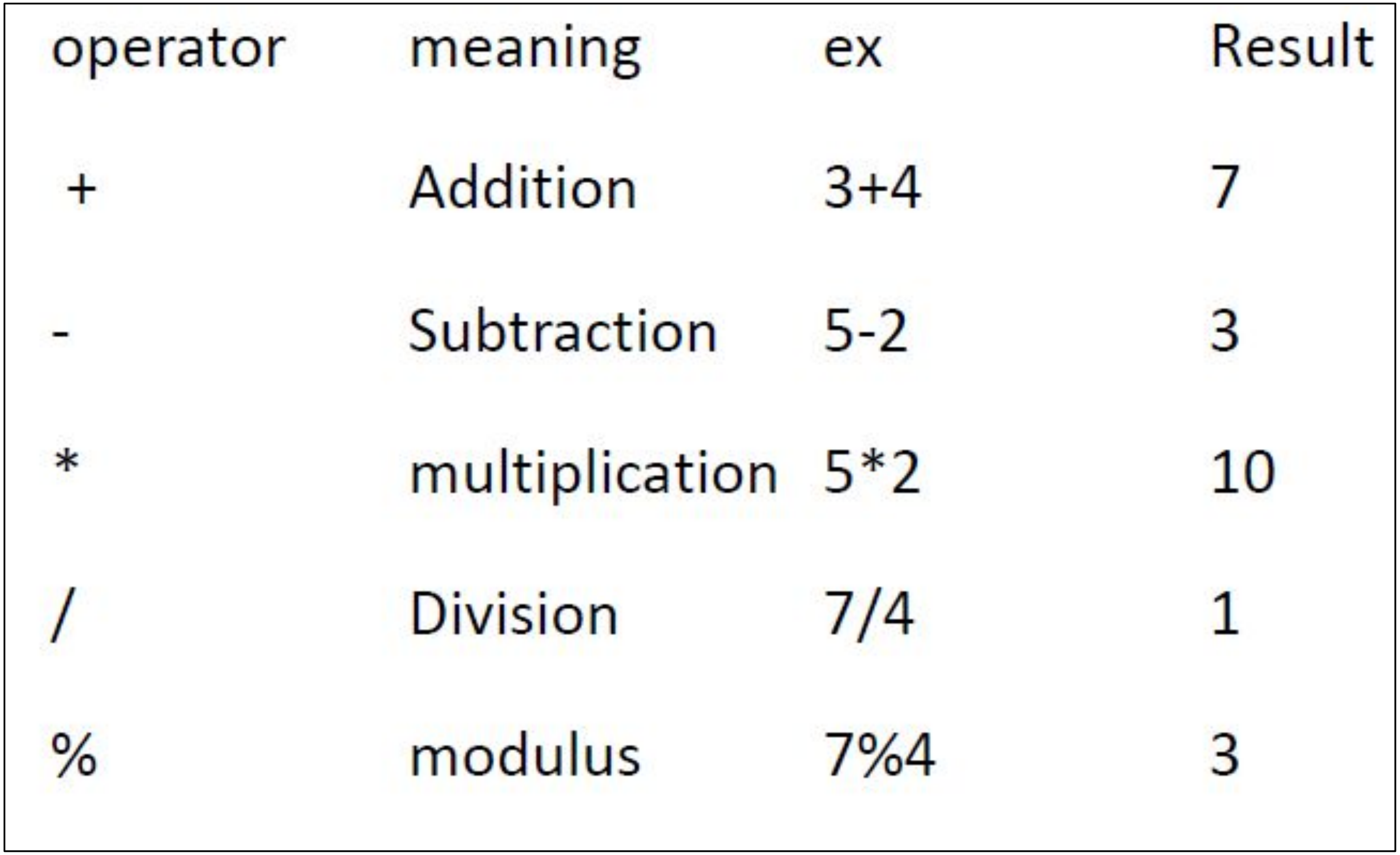
Assignment operator

- int a= 2;
  a + = 3;
  This means the a = a+3
  output – 5

- int b=5
  b - =2 ;
  This means the b = b -2
  output -3

```
void main()
{
    int a; int b;
    a=2; b=3;
    printf("value of a before =%d\n", a) ;
    a+=3;
    printf("value of a =%d\n", a) ;
    printf("value of b before =%d\n", b) ;
    b-=1;
    printf ("value of b =%d\n", b);
}
```

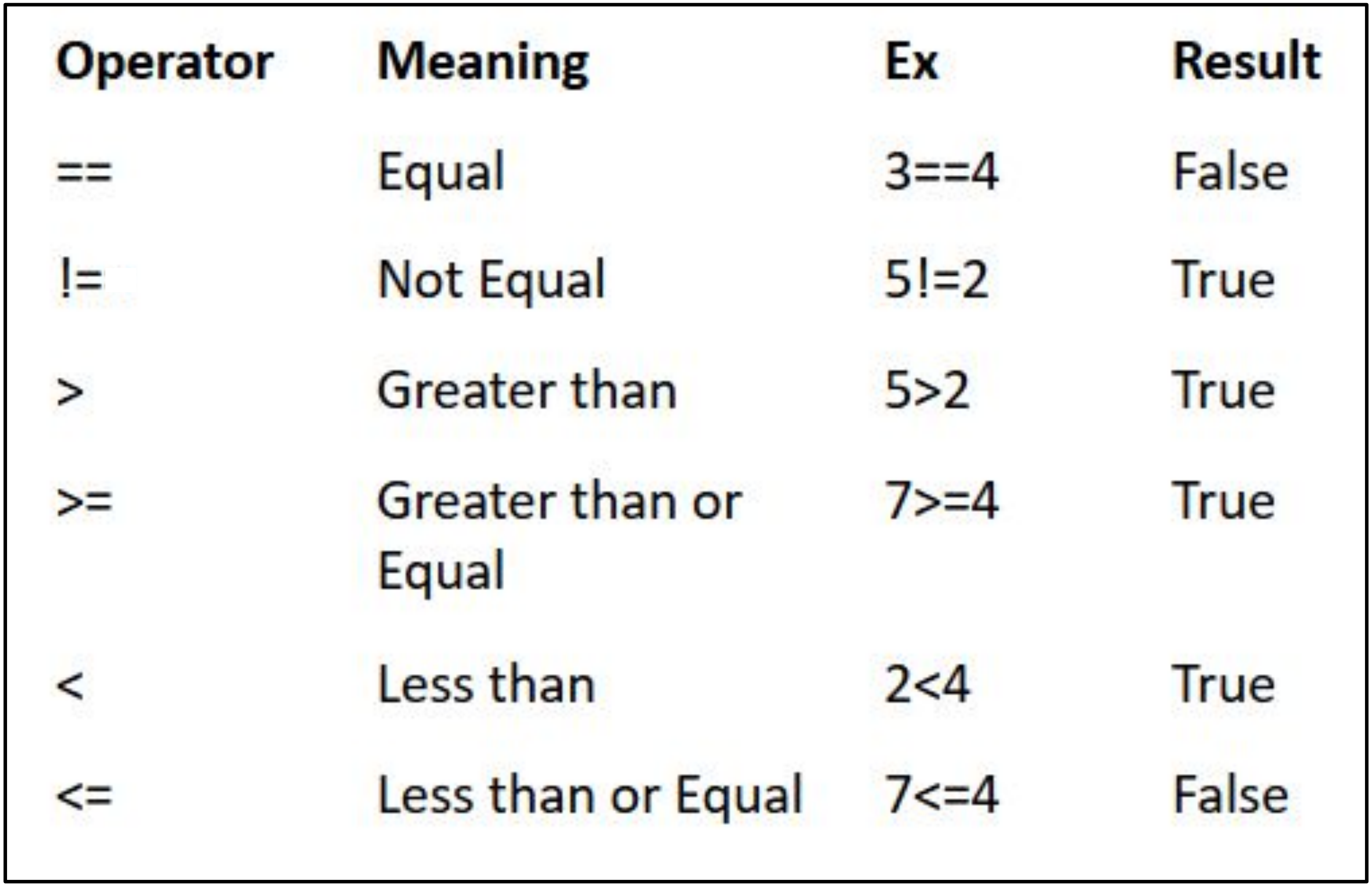
Arithmetic operators



operator	meaning	ex	Result
+	Addition	3+4	7
-	Subtraction	5-2	3
*	multiplication	5*2	10
/	Division	7/4	1
%	modulus	7%4	3

```c
void main()
{
    int a; int b;
    a=2; b=3;
    printf("value of a+b =%d\n", a+b) ;
    printf("value of a-b =%d\n", a-b) ;
    printf("value of a*b =%d\n", a*b) ;
    printf ("value of a/b =%d\n", a/b);
    printf ("value of a%b =%d\n", a%b);
}
```

Relational operators

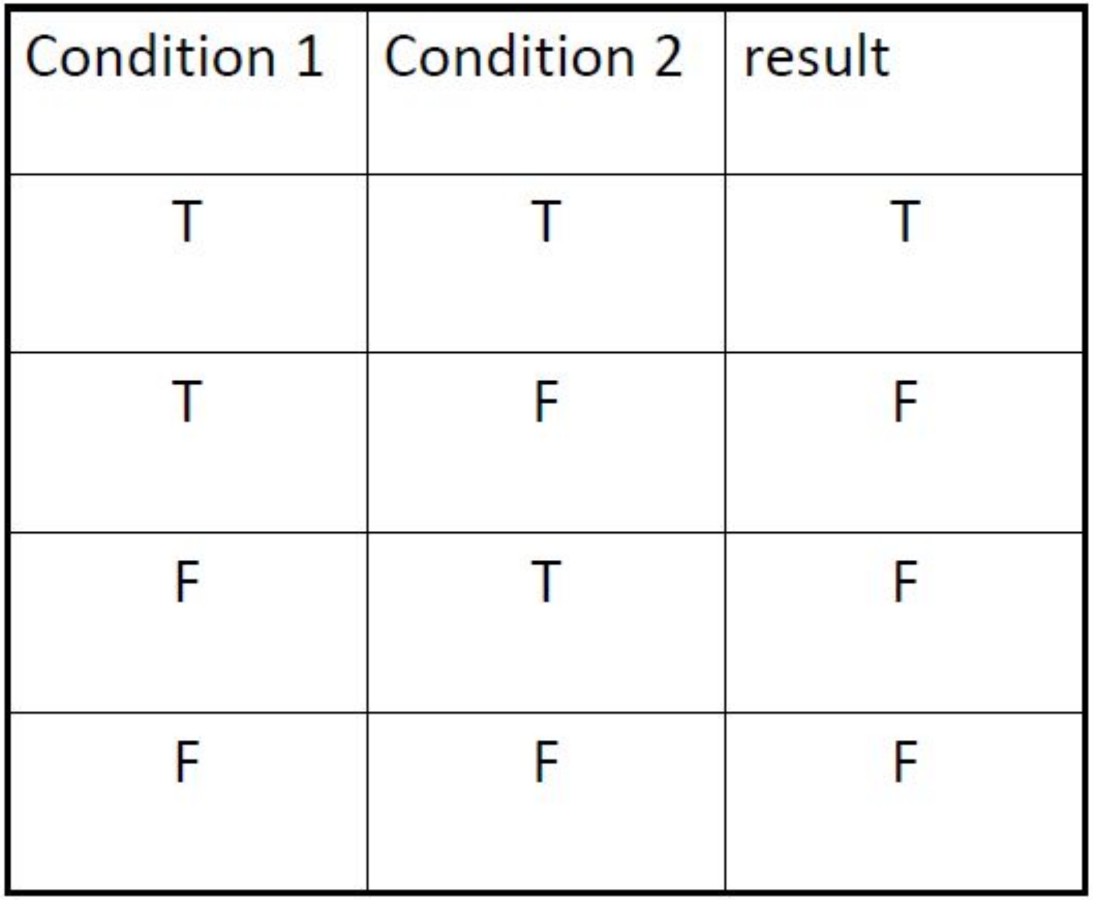


Operator	Meaning	Ex	Result
==	Equal	3==4	False
!=	Not Equal	5!=2	True
>	Greater than	5>2	True
>=	Greater than or Equal	7>=4	True
<	Less than	2<4	True
<=	Less than or Equal	7<=4	False

```
#include<stdio.h>
void main()
{
    int f, g, d;
    g=7; f=8 ;
    d=(int)(g<=f);
    printf(" %d", d);
}
```

Logical operators

- AND (&&)



Condition 1	Condition 2	result
T	T	T
T	F	F
F	T	F
F	F	F

```
#include<stdio.h>
void main()
{
    int f, g, d;
    g=1;f=0 ;
    d=(int)(g&&f);
    printf("%d", d);
}
```

Logical operators

- OR (||)

Condition 1	Condition 2	result
T	T	T
T	F	T
F	T	T
F	F	F

Logical operators

- NOT (!)

A	!A
T	F
F	T

Bitwise operators

- These may only be applied to integral operands, that is, char, short, int, and long, whether signed or unsigned

& bitwise AND

| bitwise inclusive OR

^ bitwise exclusive OR

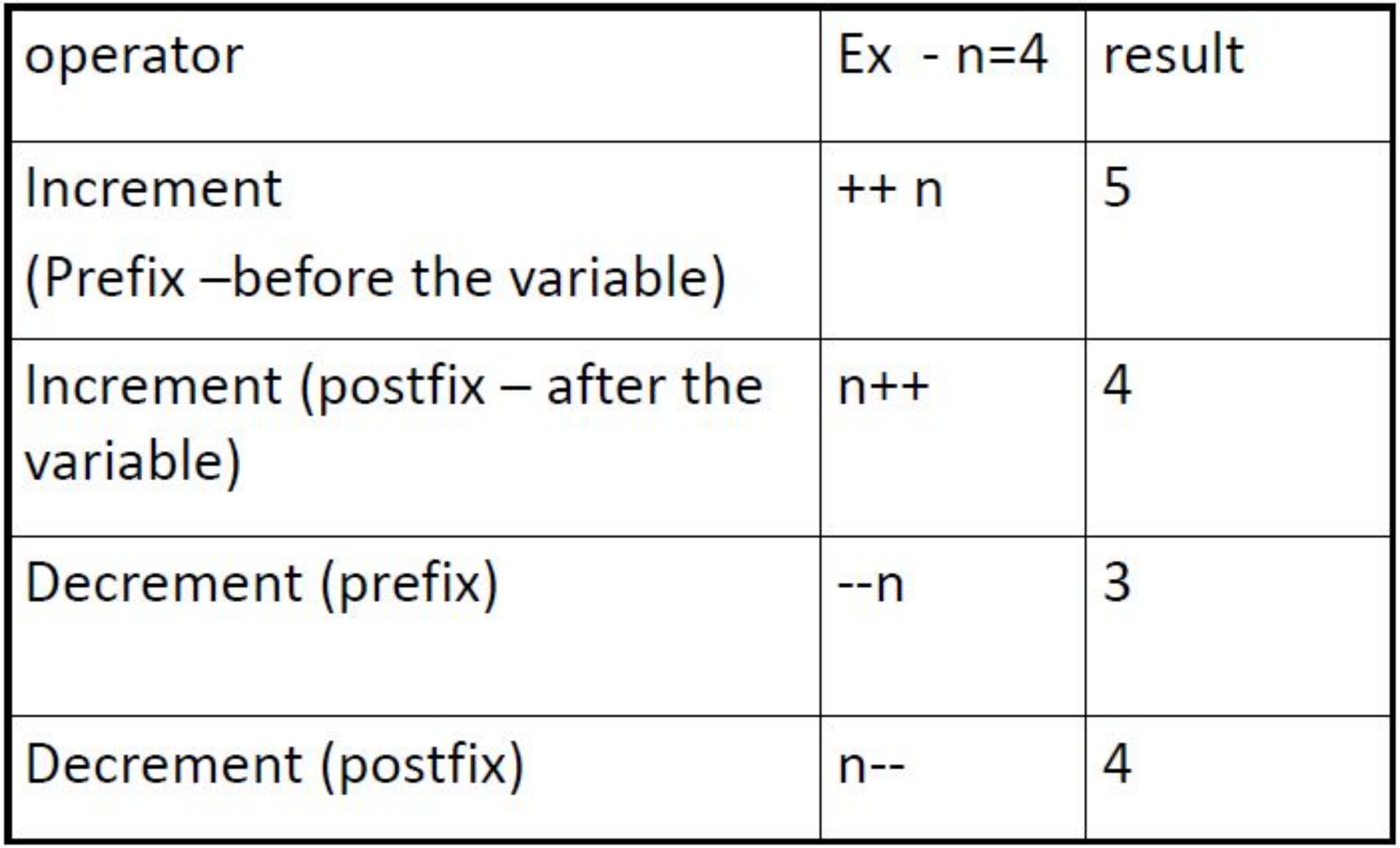
<< left shift

>> right shift

~ one's complement (unary)

```
#include<stdio.h>
void main()
{
    int a, b, d;
    a=2; b=3;
    d=(int)(a&b);
    printf("%d", d);
}
```

Increment and decrement

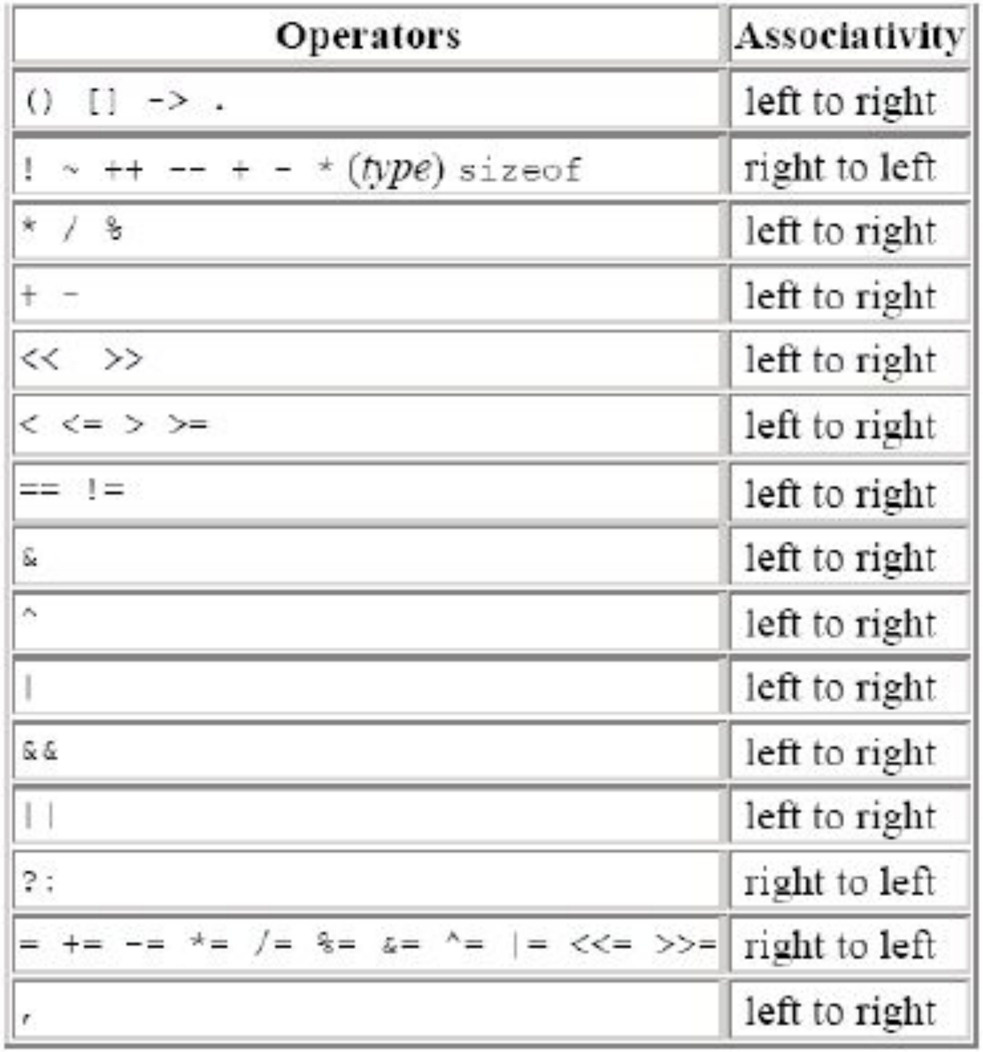


operator	Ex - n=4	result
Increment (Prefix –before the variable)	++ n	5
Increment (postfix – after the variable)	n++	4
Decrement (prefix)	--n	3
Decrement (postfix)	n--	4

```
void main()
{
    int a, b;
    a=3; b=3;
    printf(" a++ =%d\n", a++);
    printf(" ++a=%d\n", ++b);
}
```

```
void main()
{
    int c, d;
    c=3; d=3;
    printf("b-- =%d\n", c--);
    printf("--b =%d\n",--d);
}
```

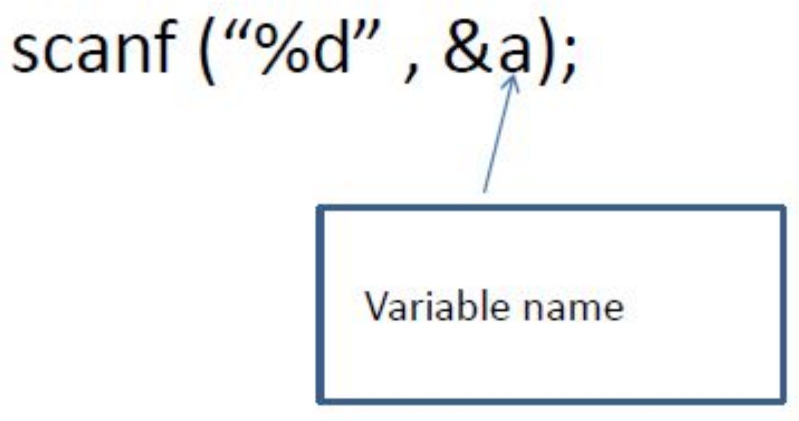
Precedence and Associativity of Operators



Operators	Associativity
() [] -> .	left to right
! ~ ++ -- + - * (*type*) sizeof	right to left
* / %	left to right
+ -	left to right
<< >>	left to right
< <= > >=	left to right
== !=	left to right
&	left to right
^	left to right
|	left to right
&&	left to right
||	left to right
?:	right to left
= += -= *= /= %= &= ^= |= <<= >>=	right to left
,	left to right

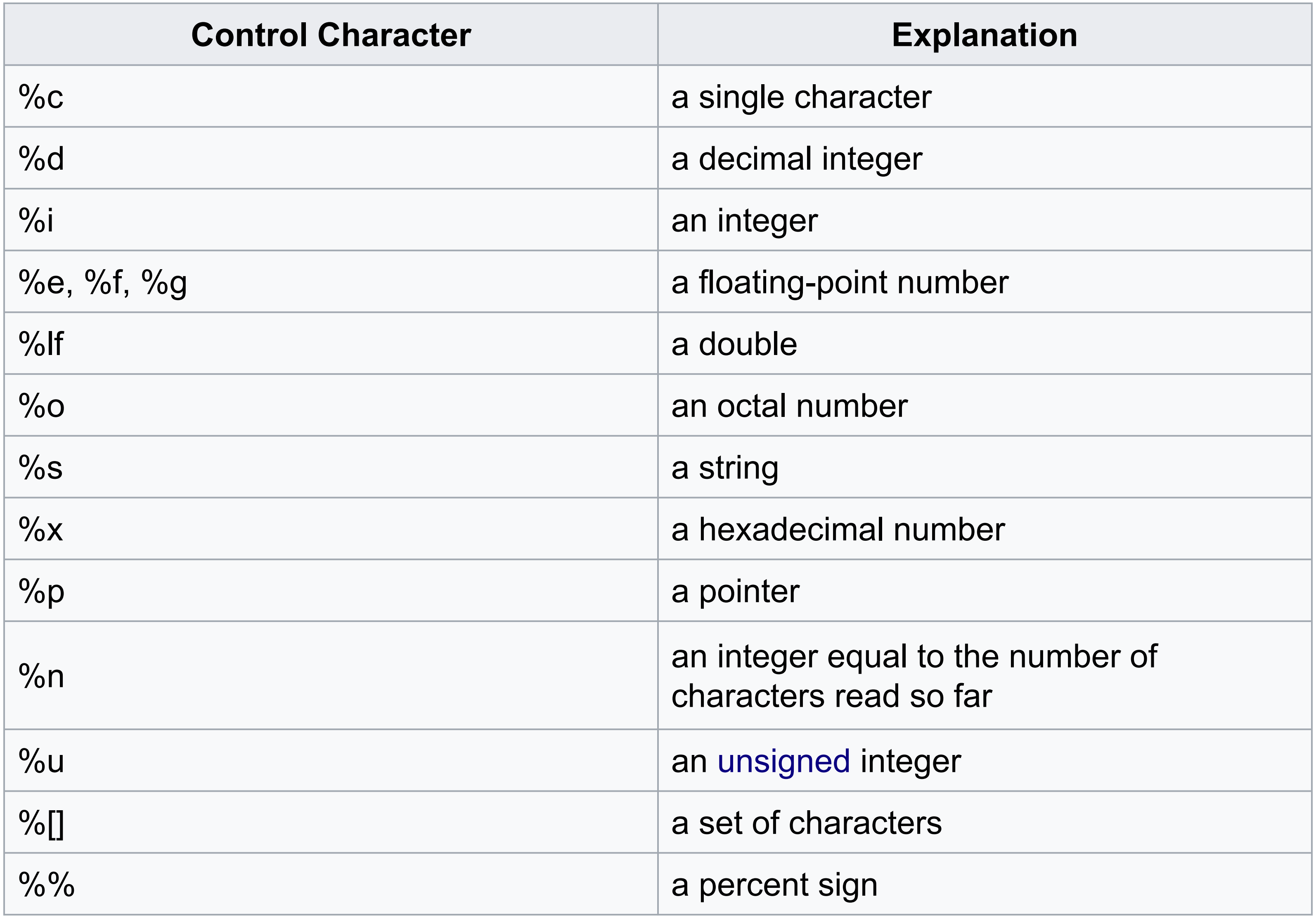
Getting input from keyboard

- int a;



```
scanf (“%d” , &a);
```

- char ch; scanf("%c", &ch);
- float no; scanf(“%f”, &no);



Control Character	Explanation
%c	a single character
%d	a decimal integer
%i	an integer
%e, %f, %g	a floating-point number
%lf	a double
%o	an octal number
%s	a string
%x	a hexadecimal number
%p	a pointer
%n	an integer equal to the number of characters read so far
%u	an unsigned integer
%[]	a set of characters
%%	a percent sign

Why we use “&” sign

- scanf() expects pointers (the address to the variable) as arguments.

- Some types - like arrays (e.g. int arr[6]), strings (e.g. char name[20]) and pointers (e.g. char* address) are already addresses and you can pass them to scanf() by simply using their names.

```
char name[20];      scanf("%s", name);
```

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int a, b;
    a=0,b=0 ;
    printf("input number1 ");
    scanf("%d", &a);
    printf("input number2 ");
    scanf("%d", &b);
    printf("\n addition is %d", a+b);
}
```