Lecture - 08

Programming in C

Operators (Part 02)

Unary operators

- inc → ++ ← increase by 1
- dec → -- ← decrease by 1,
- Post decrement / Increment
- Pre-decrement / Increment

x++ ← Increase at last

++x ← Increase first

x = 10
x++
x => 11

x = 10
x--
x => 9

int a = 10;

printf('%d', a++ + ++a) ← increment

a++ → last increase; ++a → sum

10 + 11

=> 22

a = ~~11~~ 12

(B) Binary operator : 2 operands

x + y , 2-operands

a) Assignment operator :

Assign the value to the variable

```
int x = 10;
```

- `=` ← Assignment
- `10` ← literal / value / constant
- `x` ← Variable

* Assign the value 10 to the variable x.

* It Solve at last

Eg `int x = 10 + 20 - 2`

L ← R

Solve then Assign

`int x = y;` ← Assign the value of y to the variable x.

`int x = 10, y = 20, z = 30;` ← Allowed

- `int` ← Datatype
- `x = 10, y = 20` ← Variables
- `z = 30` ← Dynamic typing

Augmented Assignment operater

↳ Perform arithmetic operation first the assign the value.

Ex
```
int x = 100;
x += 200;
printf("%d", x);
```

$x += 200 \longrightarrow x = x + 200$

$x = x + 200$

$x = 100 + 200$

$x = 300$

$x = \cancel{100} \rightarrow 300$

300

Ex
```
int m = 10;
m -= 5;
printf("%d", m);
```

$m -= 5 \longrightarrow m = m - 5$

$m = 10 - 5$

$m = 5$

$m = \cancel{10} \rightarrow 5$

5

Old value

`+=` ← add then assign

`-=` ← subtract then assign

`*=` ← product then assign

`/=` ← divide then assign

`%=` ← remainder the assign

```
int a = 10
a *= 2;
printf("%d", a);
```

$a = a * 2$

$a = 10 * 2$

$a = 20$

20

```
int a = 10;
a %= 2 ; ← valid
printf("%d", a)
```

a = a % 2
a = 0

10 % 2 → 0

```
int b = 10;
b %= 2.00; → invalid
printf("%d", b);
```

b %= 2.0

⇒ b = b % 2.0 ← float

error

↳ Arithmetic

int % float ⇒ error

b) Arithmetic Operator :

↳ An operator that perform an arithmetic Operation [+ - * / %]

a) Arithmetic + operator : Add two numbers.

↳ int, float, double, short, char

```
int a = 10;
int b = 20;
printf("%d", a+b);   # 30
```

```
printf("%d", 3+1.5);
```

int + float = float

4.5

- 4
- Garbage Value
- 0 ✓

int + int = int

float + int = float

float + float = float

double + int = double

double + float = double

char + int = int ✓

short + int = int

short + short = int

float / double → Print('%d') → 0

Welcome | arithemetic.c

arithemetic.c > main()

```c
#include<stdio.h>

int main(){
    float a = 1.5;
    int b = 3;
    int c = a+b; // we are storing the value of a+b in c
    // and c is an integer type container.
    printf("%d\n", c); // printing the stored integral part of the operation a+b
    printf("%d\n", a+b); // it is finding the integer value, and got a float value. that is false. false = 0;

    char x = 'A'; // ASCII A = 65
    int y = 100;
    printf("%d", x+y);

    return 0;
}
```

```
4
0
165
PS>
```

b) Arithmetic - Operator : Subtract two numbers

↳ Same Rules applied for '-' Operator.

c) Arithmetic * Operator

↳ Multiply the values ← Asterisk

int * int = int

int * double = double

int * float = float

float * float = float

Char * int = int

Char * float = float

double * float = double

Unary - operator

Arithmetic - ,,

Unary - Op:

↳ int x = 10;

-x

printf("%d", x); → -10

Arithmetic - op:

↳ int x = 10

int y = 20

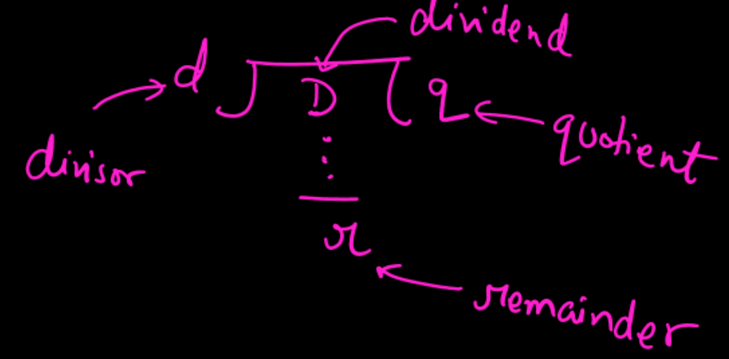
printf("%d", x-y);

⇓

-10

d) Division operator:

↳ Return the quotient after dividing two numbers.



⇒ int / int = int

⇒ float / int ⇒ float

⇒ Char / int = int

⇒ double / float ⇒ double

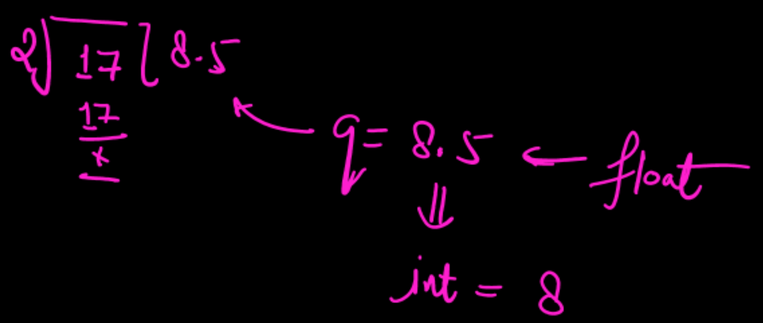
⇒ float / float ⇒ float

Eg

```
int x = 17;
int y = 2;
printf("%d", x/y);
```

⇓

8



e) Modulus operator : (%)

↳ It divide two integers and return the remainder.

↳ only works with integer

↳ The result must be an integer

result = a % b

a/b → remainder

```
int x = 11
int y = 2
printf("%d", x%y)
```

↓

1

2) 11 (5
10
1

1) int % int → int
2) Char % Char ⇒ int
3) Short % Short ⇒ Short int

```
printf("%f", 10.5 % 3.0); ← Error
```