Day 01

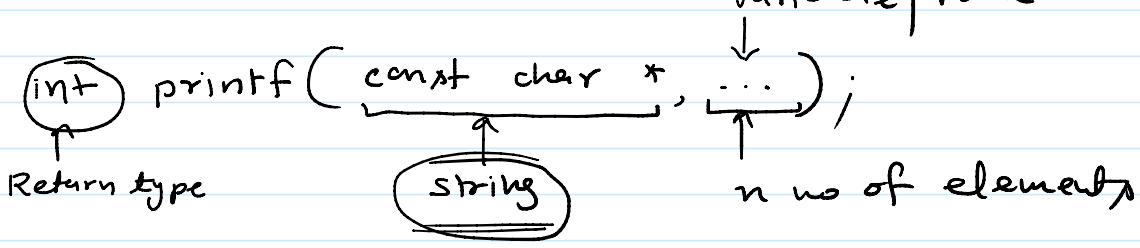
04 April 2024 12:33 PM

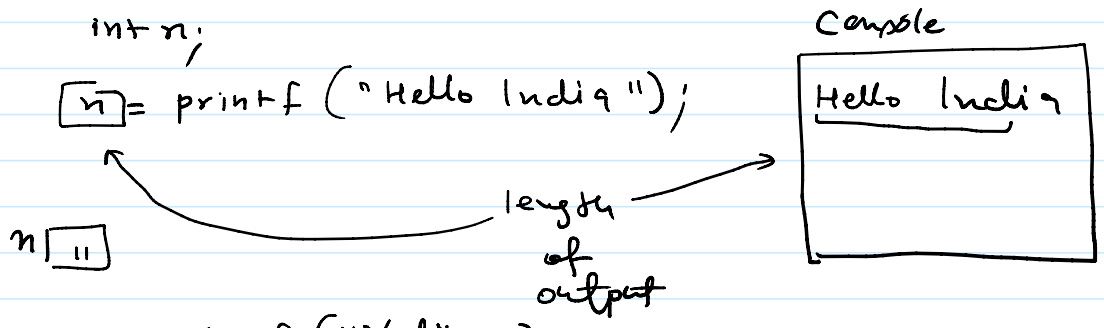
Syntax :→ input | output

scanf printf

stdio.h

printf ( ) :→ (pre defined function)





n = printf ("%d", 25); → 25

2

int → %d

float → %f

char → %c

string → %s

printf ("%s", "Hello India"); → Hello India

int a=25;

printf ("%d", a); → 25

→ printf ("%4d", a); __25

space

→ printf("%04d", 9); → 0025

→ printf("%-4d", 9); 25__ (space)

printf("%*d", 5, 25); ___25

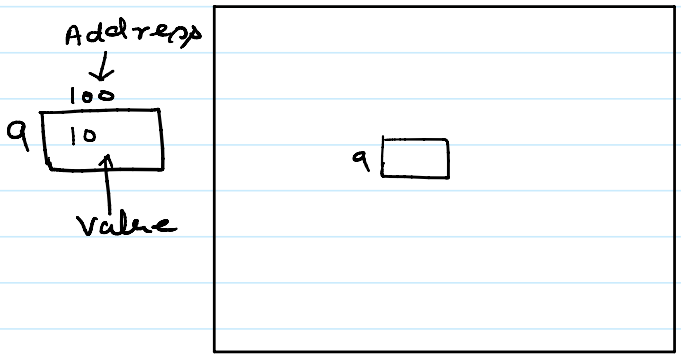
printf("%02d", 7583); → 7583

scanf() →

scanf(" — ", ...)

Address

int a;



&a

↑ Address of variable 'a'

int = scanf("%d", &a); (10)

String

Sum of 2 Nos :-

sum of 2 No/s without using any operator -

n = printf("%2c", ' '); (space) (space)

n = 2

n = printf("%2c%3c", ' ', ' ');

n = 5

n = printf("%*c%*c", 2, ' ', 1, ' ');

n = 5 a b

\n → new line

\r → Carriage Return

Data types :-

int a;

a = 10;

Integer :→

int short long long long

short a;
or
Short int a;

a [5] ← 2 bytes → 16 bits

1 bit

a | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |

Sign Bit
0 → +ve
1 → -ve

Data bits

$2^{16-1} \rightarrow 2^{15} \rightarrow 32768$

Range −32768 to 32767

long a;

a [ ] ← 4 bytes → 32 bits

long a;

← 4 bytes → 32 bits

long int a;

$2^{31} \rightarrow 2147483648$

Range −2147483648 to 2147483647

int → Compiler dependent

	16 bits	32 bits	64 bits
int →	2 byte	4 bytes	8 bytes

Turbo → 2 byts (32 bits)
64 bits → 4 byts

long long → 8 bytes

sizeof →

sizeof (data type | variable-name | value)

sizeof (short) → 2

long a;

sizeof (a) → 4

sizeof (5LL) → 8

Float :→

float , double , long double

float a; → 4 bytes

double a; → 8 bytes

long double → Compiler dependent

8 | 10 | 12

0.000123 → 1.23 e-4

$\frac{1.23}{10000} \rightarrow \frac{1.23}{10^4} \rightarrow 1.23 * 10^{-4}$

%e → 1.23 | e-4

(1.23: Mantissa, e-4: Exponent)

float a = 0.000123;

✓ printf("%f", a); → 0.000123

✗ printf("%e", a); → 1.23 e-4

1.230000 e-004

float b = 0.000000123;

✗ printf("%f", b); → 0.000000

✓ printf("%e", b); → 1.23 e-7

%g

printf("%g", a); → 0.000123

printf("%g", b); → 1.23 e-7

Character :- [A-Z, a-z], 0-9 (digits), symbols +, -, @, #

'a', '2', '+'

ASCII-8

unsigned char → 1 byte → 8 bits → 2^7 → 128 char

Range → -128 to 127 → 256

String :- "abcd"

'\0' → null char / End of String

printf("%d", sizeof("abcd")); → 5

""

Operators :-

operator

s + c

operands

5 + 2

Unary	Binary	Ternary
Single operand	2 Operands	3 Operands

① Arithmetic :- Associativity → Left to Right

+, -, *, /, %

low, high

precedency

a = 5 + c * 2

a = 5 + 12

a = 17

a = 5 * 2 / 3 * 7

(1) (2) (3)

10 / 3 * 7

3 * 7

a = 21

3 * +

a = 21

a = -5 → +5

2. Unary Operators

++ , -- , - , sizeof , typecasting

++ : inc by 1 ; -- : dec by 1 ; - : sign change

++

```
int a=5;
a++;
printf("%d", a); → 6
```

--

```
int a=5;
a--;
printf("%d", a); 4
```

Pre inc (++a)

```
int a=5;
++a;
printf("%d", a); → 6
```

a = 6

```
printf("%d", ++a); 6
```

(① ② ③)

Post inc (a++)

```
int a=5;
a++;
printf("%d", a); → 6
```

a = 6

```
printf("%d", a++); 5
```

(① ② ③)

```
int a=5, b;
b = ++a;
```

a 6, b 6

```
int a=5, b;
b = a++;
```

a 6, b 5

```
int a=5;
```

a 7 → 6 → 5

a = -a;

a = -a;

typecasting →

```
int a = 5;
printf("%f ", a);
```
→ 0.000000

(float: %f, int: a)

```
printf("%f ", (float) a);
```
→ 5.000000

(5.0; (float) a: Type casting)

5/2 → 2

(float)5/2 → 2.5

int a = 5;

~~a%0~~

Relational operators → Boolean op

>, <, >=, <=

output: ~~True~~ (1), ~~false~~ (0)

if(~~⊗~~)

```
printf("%d", 5>4);
```
→ 1

input

True → Non zero

false → zero

5>=10 → false

5>=5 → True

if(-5) ← True

if(0) ← false

Equality operators → Boolean

== !=

~~5>2>1~~

U

== !=

5 == 5 → True
5 == 4 → false
5 != 4 → True
5 != 5 → false

~~5 > 2 > 1~~

5 == 5 == 5 → 0 false

Assignment operator → Associativity → R to L

= ↑ Assign

+=, -=, *=, /=, %= ← short hand

Precedence
High [ ]
( )
;]
=
low → ,

a = 5;
variable ↑ variable / value / exp

a = 5+4; ✓

a = b;

5+4 = a X
a = 5 X

a 5

a = 5;
a = a+2; → a += 2

a = a+b; → a += b

a = a+10+20 → a += 10+20;

a 10
b 10
c 10
d 10

int a, b, c, d;

a = b = c = d = 10;
④ ③ ② ①

logical operators :-

② && (and)
③ || (or)
① ! (not) ← unary

Cond	! cond
T	F
F	T

printf("%d", !5); → 0

!(5>4) → false

Cond a	cond b	a && b	a || b
T	F	F	T
T	T	T	T
F	F	F	F

short Circuit :→

T || X

F && X

T || X && X || X && X || X

Conditional operator → (Ternary Operator)

Syntax → Condition ? True stmt : false stmt;

Bitwise operators → (int)

& and, | or, ^ XOR, << left shift, >> Right shift, ~ not

a	b	a&b	a|b	a^b
1	0	0	1	1
1	1	1	1	0]
0	0	0	0	0

printf("%d", 5&4); → 4

```
  1 0 1
4 1 0 0
-------
  1 0 0   Ans
```

printf("%d", 5|4); → 5

```
  1 0 1
| 1 0 0
-------
  1 0 1 →
```

printf("%d", 5^4); → 1

```
  1 0 1
^ 1 0 0
-------
  0 0 1
```

Double

<< left shift fill with zero

```
int a;
a = 5<<2;
printf("%d", a); → 10
a = 5 <<4;
printf("%d", a); → 80
```

5 = 0 0 0 0 0 1 0 1

a = 0 0 0 0 1 0 1 0

80 << 4 = 40 << 2 = 20 << 1 = 10 << 1 = 5

Half

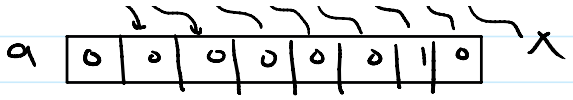
>> Right shift and fill with sign bit

```
int a;
a = 5>>1;
printf("%d", a); (2)
```

5 = 0 0 0 0 0 1 0 1

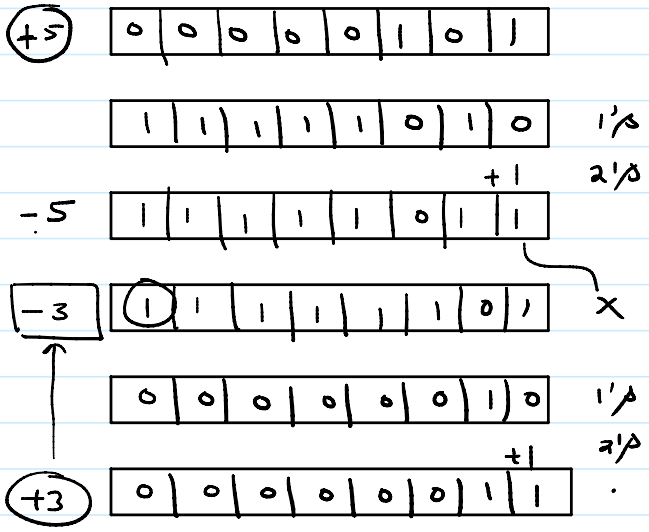
a = 0 0 0 0 0 0 1 0

printf("%d", a); (2)



a = (-5) >> 1;

printf("%d", a); -3



5/2 → 2.5

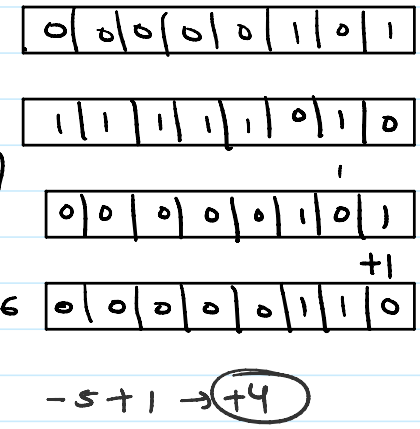
↑ floor divide → 2

-5/2 → -3

~ → Not (1's Complement) → (Add one then sign change)

int a = 5;

printf("%d", ~a); -6



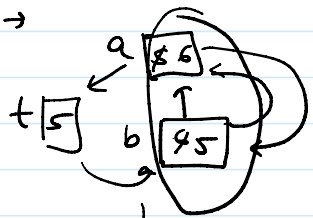
2 → !0

-5+1 → +4

a = -5;

printf("%d", ~a); → 4

Swap 2 Nos :→

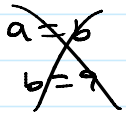


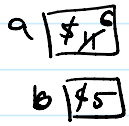
→ t = a
→ a = b
→ b = t

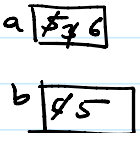
a = a+b
b = a-b
a = a-b

a = a*b
b = a/b

a = a^b ✓
b = a^b







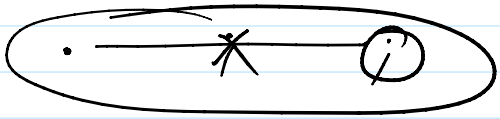
```
a = a*b
b = a/b
a = a/b
```

```
a = a^b ✓
b = a^b
a = a^b
```

b 85

```
  0 1 1        0 1 1        1 0 1
^ 1 0 1      ^ 1 1 0      ^ 1 1 0
-------      -------      -------
6 ← 1 1 0      1 0 1        0 1 1
```

H.W → swap 2 No within single line



ok