Class-2

14/2/22 Programming / what is program :

→ flowchart → pseudocode.

first program [Hello Dunia]

→ "placement tho lagni hi hai"

code Blocks / online
↓
ide one
repl it
gdb.

①
```
int main ()
{
}
```
↓
starting pt

②
```
int main()
{
cout<<"placement lagegi"<<endl;
}
```
↓
[cout ka def → computer ko Pata chalne we include iostream.]

③
```
#include<iostream>
using namespace std;

int main() {
 cout<< "placement lagegi"<<endl;
}
```
↓
directive

endl; → new line

H/w :- stackover flow.

① can i create custom header file ?

② can we create our own namespace ?

③ what all other namespaces are present ?

④ int ⇔ void [explore karna karna hai]

Data type and variable

int a=5 ; → [type of data & size of data]

char ch='z';

float

double

bool

(i) int me, 4 bytes.

↓

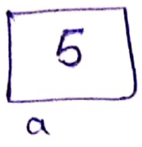
[-ve, 0, +ve canbe stored]

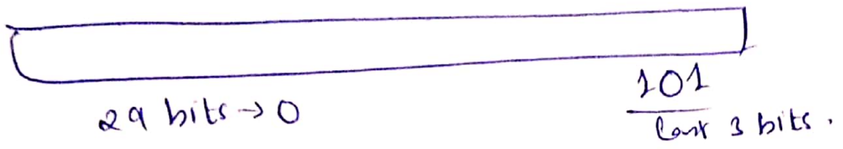
1 bytes → 8 bits

int a=5

↓

4 bytes





(ii) bool c = true;

b = false;

=> size of bool → 1 byte [smallest address].

(iii) float f = 1.2 .

size of float = 4 byte

(iv) double d= 1.23

size = 8 byte => precise.

=> variable

int a = 5

conventions :-

→ abc ✓

→ ABC ✓

→ A1, babbar1 ✓

→ a_b ✓

→ 1abc ✗

code2:-

```
int main(){
    int a =5;
    cout<<"value of a"<<a<<endl;
}return 0;
}
```

#12:- bool b=0;

```
cout<<"value of b is"<<b<<endl;

float f=1.23;

cout<<f<<endl;

cout<<"the size of float is'<< sizeof(f)<<endl;

cout<< sizeof(b)<<endl;
```

H/w → short

long

long long.

```
char ch='d';  → single

cout<< ch<< sizeof(ch') <<endl;
```

=> char ch='da'; → // Not possible.

=> int a=5;

1 byte = 8 bit

variable name = a

type = int

size = 4 byte = 32 bits

a=5 → in memory is stored as 00000000 00000000 00000000 00000101

a=4 → in Binary = 100.

0 0 0 0 0 ___ 0 0 0 100.

29 bit

=> the above, we follow for only the numbers.

→ How -ve no's are stored in memory.

int x = -5

step1:- ignore -ve sign

step2: convert into Binary sign. rep

step3: take 2's compliment

5 → [5] → 101.

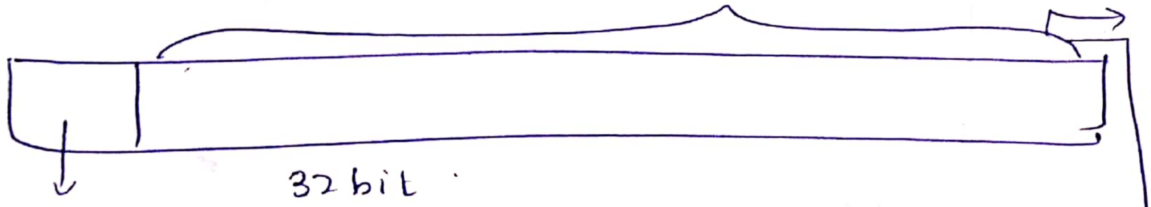
5 → $\underbrace{000 \text{——} 00}_{29 \text{ bit}}$ 101

[1's compliment ↓ bit filp]

[2's comp → take 1's comp + 1].

```
    00000000   00000000   00000000   00000         101
1's 111  _  ____  .  _  _  _  ___    _____       11010
                                                   + 1
    ____________________________________________________
    111  ___________________________________   111 011
```

-5 [memory me aisa store hota hai]



31 bits me actual num

+ve/-ve.

1 → -ve

0 → +ve

Iska 2's comp nikale tho → ans → 5 aata

Dry run → -4, -8, -11

range = -2^{31} se $2^{31}-1$.

→ How are we gng to differentiate B/w int or char memory.

data type can say

⇒ Range of float, double, long, short

Operators.

① Arithmetic [+, −, *, /, %]

code3:-

```
int main(){
    int a=5;
    int b=3;
    int ans = a+b;
    return 0;
}
```

code4:-

We know, $\frac{5}{3} = \frac{9}{6} = \textcircled{1}.666\ldots$ ↓

$\frac{int}{int} = int$

$\frac{float}{int} = float$

ans → $\boxed{1}$ in memory

coz, it's stored in int.

ex:- for code 4:-

```
int ans = 5.0/3 ;
cout << ans;
```

o/p:- 1..

ex:- cout<<(5.0/3);

o/p: 1.6.

$\frac{double}{int}$ → double.

Type casting → implicit
→ explicit → ex:- char ch = 'a'
int num = (int)ch;

code4:

```
float val1 = 5.0;
int val2 = 3;
int ans = val1/val2;
cout << ans << endl;
cout << (5.0/3) << endl;
```

#code 5:-

```
char cho = 'a';

int num = (int)ch;

cout<<num<<endl;
```

O/P:- 97 ← a

Modulo operator:- [%]

5 % 3 = remainder = 2 .

Relational operators:- =, >, <, >=, <=, !=

assignment operator: =

comparision : == [a==b]

ex:
```
int x=5;
int y=5;
bool b = (x==y);

o/p:- 1
```

ex:
```
int x=5;
int y=3;
bool a = (x>y);

o/p: 1
```

logical operators:- &&, ||, !

↓

→ along with cond's.

→ $\text{bool ans} = (\underset{T}{\underline{\qquad}})\&\&(\underset{T}{\underline{\qquad}}) = T$

→ $\text{bool ans} = (\underset{T}{\underline{\qquad}})||(\underset{F}{\qquad}) = T$

NOT operator:- (!)

a=5; | a=0
!a = 1

Bitwise Operator :-

↳ Bit level

```
int a=5;        // 101
int b = 6;      // 110        // use of bitwise?
                   ______
int ans = a&b;     100 = 4
```

O/p :- 4

| (or)

```
a|b →   a=5  → 101
        b=6    110
              _____
               111 = 7
```

$\boxed{ans = 7}$

~ (NOT) :- 0 → 1

1 → 0

✗✗.

XOR → Exclusive OR (^)

x	y	
0	0	→ 0
0	1	→ 1
1	0	→ 1
1	1	→ 0

H/w :- Arithmetic, logical, Relation, Bitwise

↓

code, Exp, play, experiment. → code on editor.

Left shift operator :- (num*2)

5 << 1 → shift 5, by 1 bit

↓

00 ---00101

00 - 1010

$\underbrace{00 - 1010}_{10}$

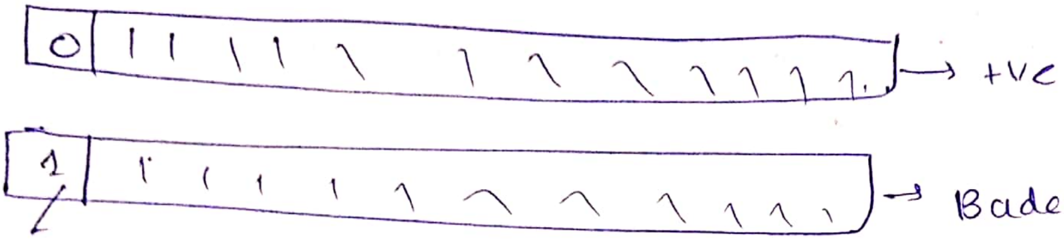
$\boxed{5 << 2}$ → shift 5 by 2 bits.

000 ---- – 00101

00 0 ——— 10100 → 20

→ padding with zero .

ex:-



→ Bade number ky left shift . ✗

Right shift operator [num/2]

$\boxed{45 >> 1}$ 000 - – 0101

00 ——— - - 10

→ padding with zero ,

5 >> 2 → 00 - - — - 01 → $\frac{5}{2\times 2}$.

H/w:- <<, >> → -ve numbers .

H/w:- Prime no ka flowchart → try to do it ✓ / Don't skip

Padding with zero → in case of +ve number = Yes .

-ve = compiler dependent

H/w:- Dry run ;- find some exceptions

The defining decade .

Pyscology of Money

Atomic Habits ,