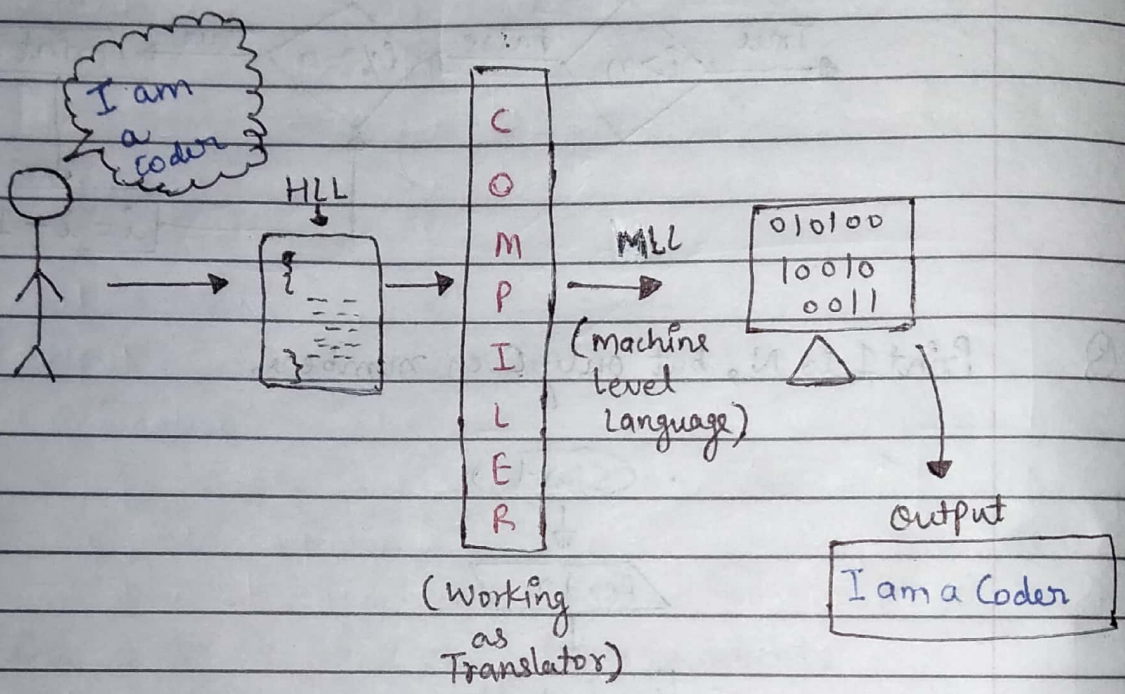
Page No.
Date 25 8 23

Need of Programming Languages?

→ Mode of communication b/w human and computer.
→ Follow standardized syntax while coding.
→ Translation tools like compiler/Interpreter are used to compile the code.

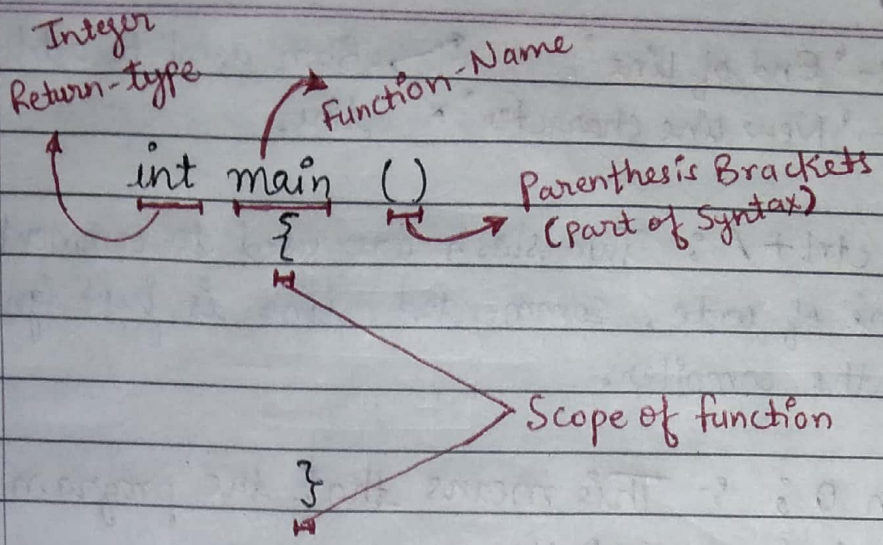
Compilation Process :-



Terms :-

Starting point of code → int main()

→ function :- A block of code which reads input & provide output.
↳ Inputs can be multiple.
↳ functions are re-usable.



→ First Code

```
#include <iostream>

using namespace std;

int main ()
{

cout << "Supreme 2.0";

}
```

→ #include :- Pre-processor directive, used to import necessary files in current programme.

→ <iostream> :- Write file name inside angular brackets to use it.

→ namespace std :- To use the specific identifiers mentioned in namespaces (Just a portion of code).

→ int main () :- Entry point, from where flow of execution starts.

→ cout :- It is an identifier used to print the statement written in double-quotes.

→ << :- Part of syntax, must to use with cout.

→ " " :- Used to write string in it.

→ ; :- Terminator, to end up the line.

→ { } :- Tells the scope of main function.

Page No.
Date

→ endl :- 'End of line' ↘
→ '\n' :- 'New line character' ↗ Both used to add a new line.

→ // or ctrl + / :- Two slash are used to comment the line of code. Commented line is just ignored by the compiler.

H.W→ return 0; :- This means that the program executed successfully.
→ return 1; :- Means the program does not executed successfully and there is some error.

→ cout << "Babbar"; cout << "Babbar";
output :- BabbarBabbar
Explanation :- If we use ';' and write some code just after the semi-colon, then it will gets executed by the compiler and compiler will treats that code as a next task to be executed after the semi-colon.

Data types and Variables :-

Assignment Operator ↑

```
int  marks = 99;
```

↓ Datatype (Type of data to be stored in memory)
↓ Name assigned to a variable
↘ Value assigned to the corresponding variable

Page No.
Date

What is Variable?

Assigning name to a memory location is known as variable.

What is Datatype?

The type of data stored in a variable is known as datatype.

Ex:- `int a = 8;`

- `int` → Data Type
- `a` → Variable Name
- `8` → Value

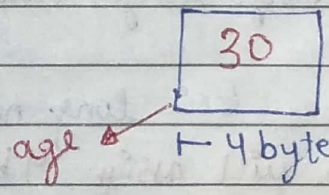
* It Tells
  1. Type of Data
  2. Size

Types of Datatype :-

1. → For storing numeric values like -5, 0, 4 we use :- int

Ex:- `int age = 30;`

* This will assign a 4 byte memory block to the variable whose name is 'age' and value is '30'



→ Declaration :-
Ex:- `int age;`

→ Initialisation :-
Ex:- `int age = 21;`

→ Size Of (variable_name) :-

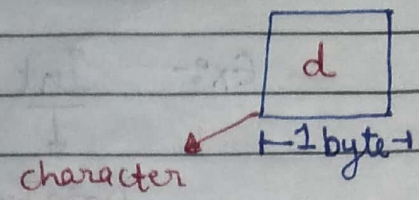
* This is used to get the size of any variable according to the system configrations.

Page No.
Date

2.→ For storing characters like a, d, K, we use :- char

Ex:- char character = 'd';

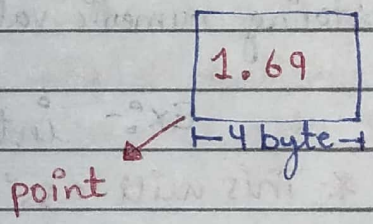
* This will assign 1 byte memory block to the variable named 'character' whose value is 'd'.



3.→ For storing floating point/decimal values like 1.24, we use :- float

Ex:- float point = 1.69;

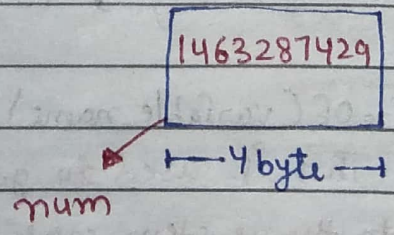
* This will assign 4 byte memory block to the variable named 'point' whose value is '1.69'.



4.→ For storing large integer value we use :- long

Ex:- long num = 1463287429;

* This will assign 4 byte memory block to the variable named 'num' whose value is '1463287429'.



⊙ If we try to print a variable who don't have any assigned value, then it will print any random grabage value in the terminal.

Ex:- int random;
cout << random;

output
43426387

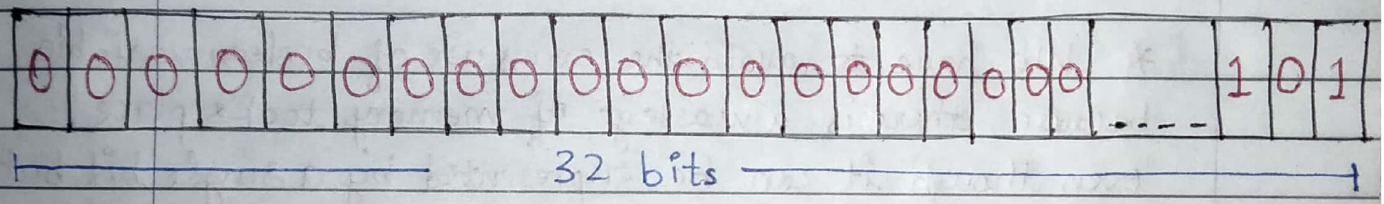
* The garbage value is always a unique one.

How data is stored?

1. for integers :-

* Int takes 4 byte = 32 bits

Ex:- int a = 5;

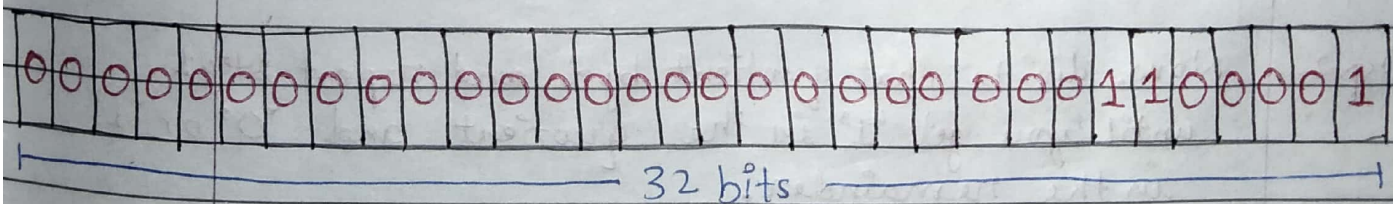


* The binary value of 5 = 101 so, it will be stored in the memory along with 29 zero's bcz we can't keep it empty so we put '0'.

2. for char :-

* char takes 1 byte = 8 bits

Ex:- char alphabet = 'a';



→ Ascii value of 'a' = 97.

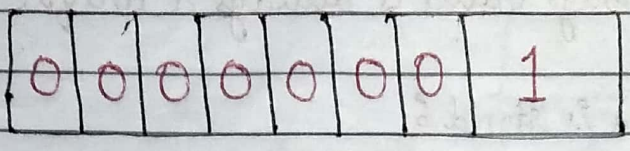
* The binary value of 97 is 1100001 so, it will be stored in the memory and empty bits are ~~store~~ filled with '0'.

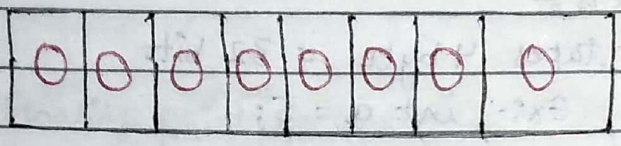
Page No.
Date

3. for boolean variable :-

* It will take 1 byte = 8 bits.
* Ex:- bool flag = true;

→ It is either true or false.
1 represents true & 0 represents false.





* We have to avoid the excess use of boolean variable because there is a wastage of memory tooks place. Even though it can be represented by a single bit but it took '7' extra bits and the reason behind this is → "Smallest addressable space in memory is 1 byte"

Finding Binary Equivalent :-

Step 1. Divide the number by 2 and stop when you get single digit quotient.

Step 2. Now, divide that quotient by 2 and repeat step 1 until you get '1' in the quotient and '0' or '1' in the remainder.

Step 3. Finally, note the last quotient in left side and write the previous remainders on ~~left~~ ~~right to~~ left direction to right direction.

Page No.
Date

→ Ex:- Finding binary of 15.

$2\overline{)15}\,(7$ → quotient ≠ 1

-14

remainder = 1 ← 1 $\quad 2\overline{)7}\,(3$ → quotient ≠ 1

-6

remainder = 1 ← 1 $\quad 2\overline{)3}\,(1$ → quotient = 1

-2

remainder = 1 ← 1 $\quad$ Stop!

Last quotient on the left side

Ans:- 1 1 1 1 → Previous remainders from left to right dir.

→ Ex:- Finding binary of 16.

$2\overline{)16}\,(8$ → quotient ≠ 1

-16

remainder = 0 ← 0 $\quad 2\overline{)8}\,(4$ → quotient ≠ 1

-8

remainder = 0 ← 0 $\quad 2\overline{)4}\,(2$ → quotient ≠ 1

-4

remainder = 0 ← 0 $\quad 2\overline{)2}\,(1$ → quotient = 1

-2

remainder = 0 ← 0 $\quad$ Stop!

Ans:- 1 0 0 0 0

Last quotient

Previous remainders

How -ve numbers are stored :-

→ 1's compliment :-

* Just flip 1 with 0 and 0 with 1.

Ex :-

101	1010	000011 → 111100
1's compliment → 010	0101	10101100 → 01010011
		11111100 → 00000011
		10111001 → 01000110
		11100111 → 00011000

→ 2's compliment :-

Step 1 → Find 1's compliment.

Step 2 → Add 1 to it.

Ex :- Q 101

find 1's Ⓐ 010

Add 1 B +1

011 → 2's compliment of 101

Q 1011

Ⓐ

0100

Ⓑ +1

0101 → 2's Comp.

→ 1000

① ① ①

1's comp. Ⓐ 0111

Add 1 B + 1

1000 → 2's compliment

1

+1

10

→ To store -ve numbers :-

* Step 1 :- Ignore - Sign.
* Step 2 :- find binary equivalent.
* Step 3 :- Take 2's compliment.

Page No. _____
Date

Ex:- int a = -5;

Step 1:- Ignore - sign, so number will be 5

Step 2:- Binary Equivalent of 5 - 00000000 00000000 00000000 00000101

Step 3:- Find 2's compliment →

Step 1:- → 1's compliment - 11111111 11111111 11111111 11111010

Step 2:- → Add 1 - +1

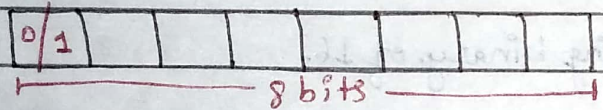
11111111 11111111 11111111 11111011

So, -5 will be stored in the memory like this

Range :- (Of a unsigned data)

1. char –

Takes 1 byte = 8 bits



Total combinations → $2^8 = 256$

If we store them from 0, then Range = $0 \rightarrow 255$

Generic formula = $0 \rightarrow (2^8 - 1)$

for 'n' bits = $0 \rightarrow (2^n - 1)$

2. int –

Takes 4 bytes = 32 bits 'n bits'

Total combinations = 2^{32} → $'2^n'$

Range = $0 \longrightarrow 2^{32} - 1$

Generic formula = $'0 \longrightarrow 2^n - 1'$

Page No.
Date

3. long -

Takes 8 bytes → 64 bits 'n bits'

Total combinations → 2^{64} '2^n'

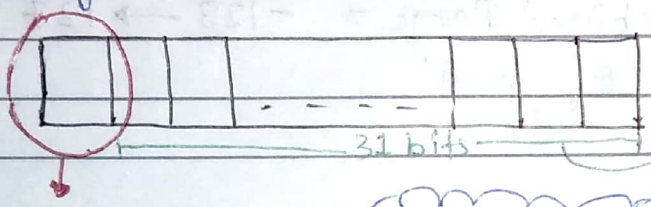
Range → 0 → $2^{64}-1$ '$0 \to 2^n-1$'

Signed Vs Unsigned data :-

→ Signed data → It can store positive, negative and zero.
* By default int is signed.

→ Unsigned data → It can only store positive and zero.

⇒ For storing signed data in 32 bits

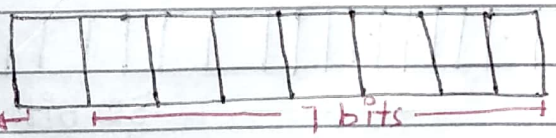


* Now find range acc to 31 bits.

Left significant Bit is called as 'Signed bit'

Rule -
If sign bit contains 0 → +ve no.
If sign bit contains 1 → -ve no.

→ Ex :- signed char -



'n bits'

char → 1 byte = 8 bits

Total comb = 2^{n-1}

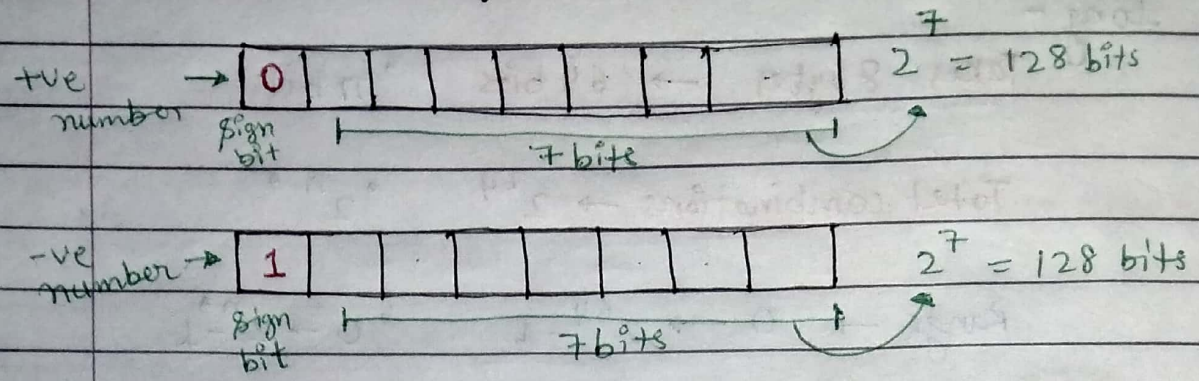
total combination = 2^7

Range = -ve +ve

$-2^7 \longrightarrow 2^7-1$

$(-2^{n-1} \longrightarrow 2^{n-1}-1)$

char → 1 byte → 8 bits



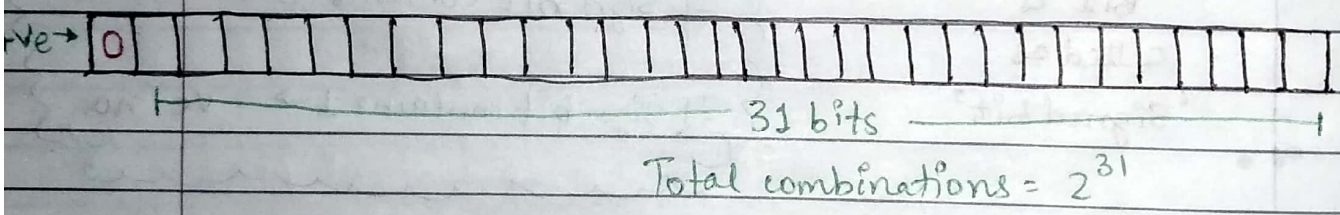
Range of +ve → 0 ⟶ 128-1 ⇒ 0 - 127

Range of -ve → -1 ⟶ -128

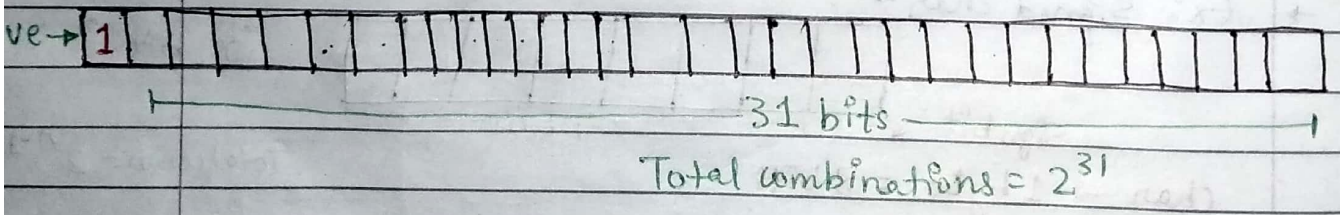
(0 is already considered in +ve)

Combined Range = -128 ⟶ 127
Of signed char

→ Ex:- int (signed)



Total combinations = 2^{31}



Total combinations = 2^{31}

Range of +ve → 0 ⟶ $(2^{31}-1)$ Range of -ve → -1 → -2^{31}

Combined Range = $(-2^{31} \longrightarrow 2^{31}-1)$
Of signed int