LECTURE - 04

Programming in C

Variable :

↳ Memory address which stores a value

→ address 5010 = 25;

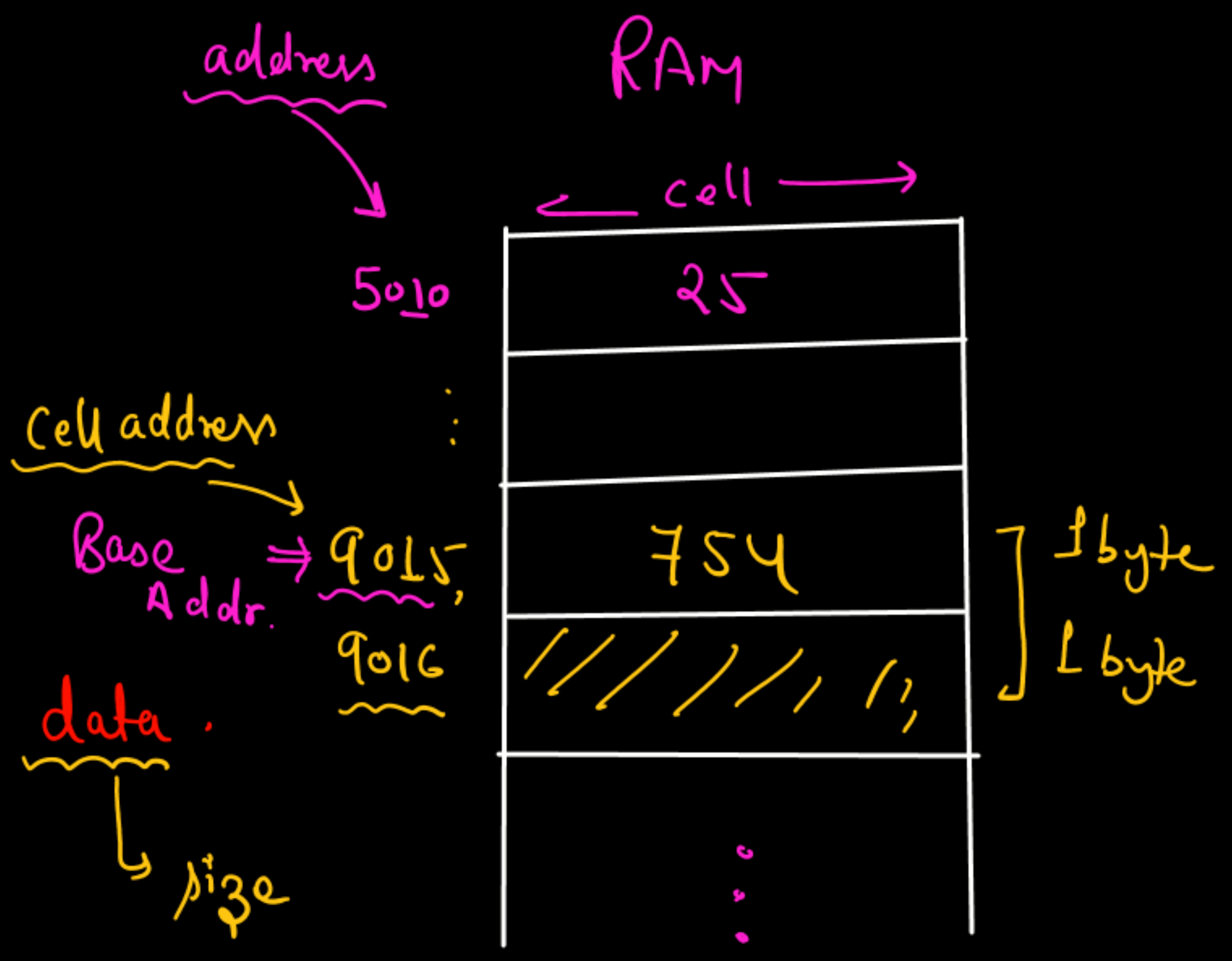
↳ Variable

⇒ Variables are the Containers which hold some data.

↳ inside main Memory

↳ size

* We are assigning an Identifier to the Base address of a value.



1 cell - 1 byte

data = 2 bytes

declaring a variable:

Syntax : format of writing a code.

declare ⇓ Empty Container

define ⇓ filled Container

datatype variableName ; } declare

└ Identifier

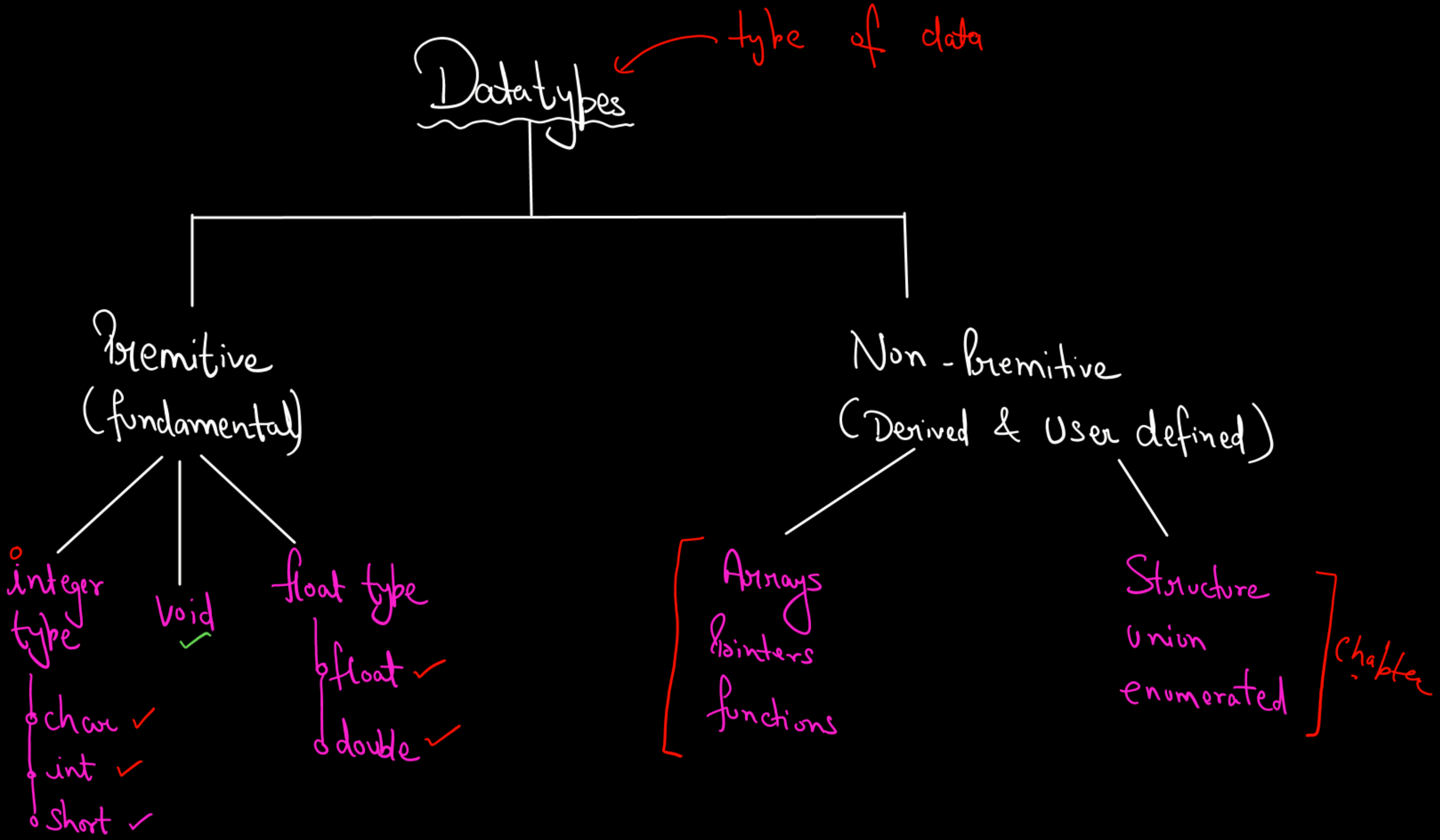
eg int y ;

⇒ इसने Memory Cells को reserve कर दिया लेकिन कोई Value pass नहीं की क्योंकि हम future में उस cell को Value assign करेंगे ← declaring a variable

datatype VariableName = value;

int x = 75;

↖ var ← literal


type of data
Datatypes
Premitive
(fundamental)
Non-Premitive
(Derived & User defined)
integer type
char ✓
int ✓
short ✓
void ✓
float type
float ✓
double ✓
Arrays
Pointers
functions
Structure
union
enumerated
Chapter

(A) Integer Type :

alphabets, numbers, symbols

1) Character : Alphanumeric letter

↳ single

Ex "ABC@123" ← Word (String)

Char Char Char

* Represented within Single Quotes ex 'A', 'B', '$', '@'

Size = 1 byte (8 bits)

Range = -2^{8-1} to $2^{8-1}-1$

= -128 to 127

Signed

-ve +ve

Aditi

'A'

Character

i ⇒ Character

8 bits → $2^8 = 256$

range = 0 to 255

2's Compliment form

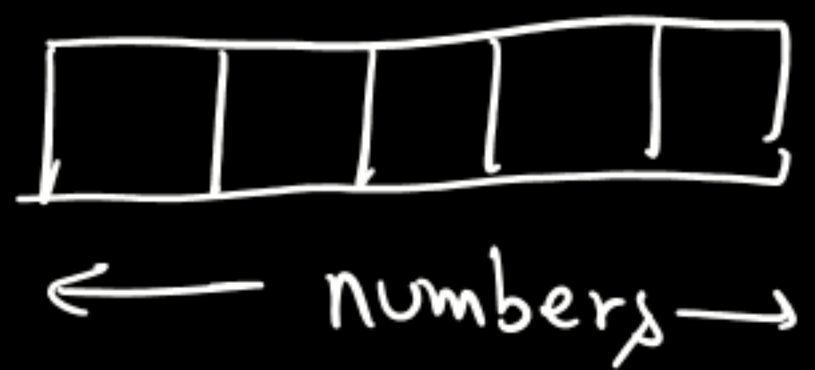
↳ Range = (-2^{n-1} to $2^{n-1}-1$)

n = no. of bits

Characters

- ↳ Signed → +, - Symbols are their (2's Complement)
- ↳ Unsigned → No negative number
  - ↳ No Sign bit

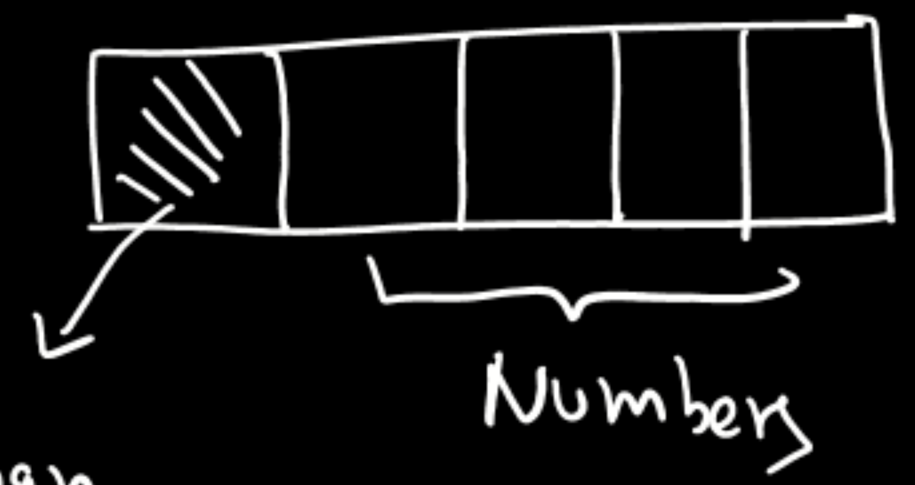
5 bits



$2^5 = 32$

↳ 0 to 31

5 bits



Sign bit

↳ $2^4 = 16$

0 → +ve		+16
1 → -ve		-16

-16 to +16

Unsigned Char ⇒ 8 bits

↳ $[0 \text{ to } 2^n - 1]$

0 to 256-1

[0 to 255]

Range → length of numbers

ii) integer

↳ A Complete number (No decimal Point).

⇒ +ve & -ve

Eg int m= 48; ← m is an integer type variable which holds 48.

Size of int ⇒ Old Compiler (Turbo C++)

↳ 2 bytes

1byte = 8bits

⇒ New Compiler (GCC / G++) ⇒ 4 bytes (4x8 = 32 bits)

Range of int :

↳ Unsigned int ⇒ $[0 \text{ to } 2^n - 1]$ ⇒ 0 to $2^{32} - 1$ ⇒ 0 to $\underbrace{4294967296}_{10 \text{ numbers}}$

↳ Signed int ⇒ $(-2^{n-1} \text{ to } 2^{n-1} - 1)$ ⇒ -2^{31} to $2^{31} - 1$

⇒ -2147483648 to 2147483647

└ 31 bits = data

↳ 1 bit = Sign

long int → 8 bytes (64 bits)

↳ Range ⇒ -2^{64-1} to $2^{64-1}-1$ ⇒ -2^{63} to $2^{63}-1$

Unsigned long int →

↳ 0 to $2^{64}-1$

Suppose,

$x = 10.0$ ← not integer

$y = 10$ ← int

$10.0 = 10$ ← Normal maths

$10.0 \neq 10$

⇓ float ⇓ int

float → Floating Point Represent ⇒ $\left[(-1)^s \; 1.M \times 2^{e-bias}\right]$ $\left[(-1)^s \; 0.M \times 2^{e-bias}\right]$

⇓ matissa, exponent, sign

int ↳ direct Representation

↳ 2's Complement

↳ direct

iii) Void

↳ Represent Nothing

⇒ Void type has no value

⇒ Used in functions

⇒ It has No Size

Void x ; ← x खाली है ।

x = Void ;

↖ खाली

Short

↳ integer → But takes half of the size of integer

↳ int = 4 bytes

short = 2 bytes

⇒ Everything is same

float : (half Precision) →

↳ A numeric datatype that Can hold the numbers with decimal point

Eg float x = 2.718 ;

datatype   Variabe   literal / Constant

* 2's Complement = $[-2^{n-1}$ to $2^{n-1}-1]$

* Size : old Compilers = 2 bytes

new Compilers = 4 bytes (32 bits)

Eg float m = 2.15809f ;

float

* Precision : 6 to 7 decimal digits

Range (depends on Precision)

↳ Range = $\sim -1.2 \times 10^{38}$ to -3.4×10^{38}

* Computer recognise a number having decimal point as 'double'

$-\underbrace{3.1}_{\text{matissa}} \times 2^{23}$

Sign

Exp

Ex

$\overset{\text{int}}{2} + \underbrace{1.85791358}_{\text{double}}$

$\overset{\text{int}}{2} + \underbrace{1.85791358}_{\text{float}} f$ ← Specifier

double

↳ Store number with point

float ⇒ 4 bytes

double ⇒ 8 bytes

long double ⇒ 16 bytes

old Computers = 12 bytes

* float and double are same thing but they have different Precision

↳ double Precision

1 Sign bit

11 bits Exponent

52 bits mantissa