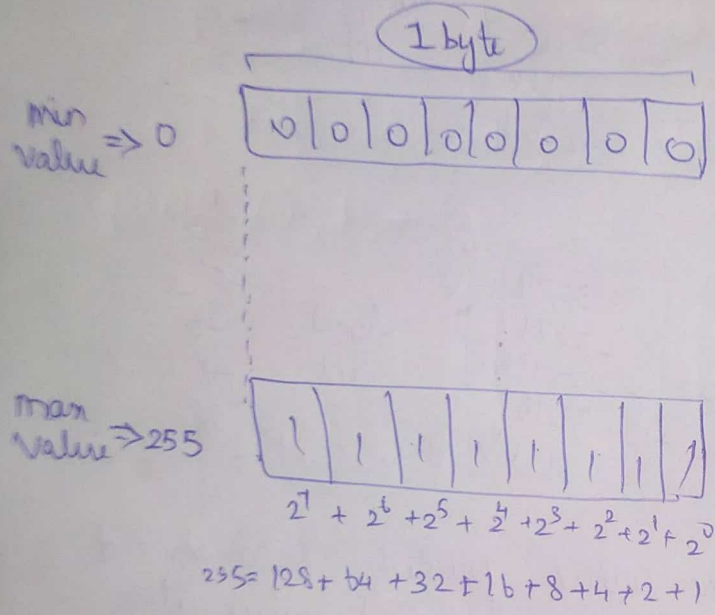
C_12 ⇒ Data Types in C - Part 3

* Character is defind using keyword 'char'
* Sign Modifiers:
  - Signed range: -128 to 127
  - Unsigned range: 0 to 255

1 byte = 8
$2^8 = 256$
0-255

* char requires memory value of 1 byte.
* Format specifier to print character is %c



UNSIGNED

$2^8 = 256$

0-255

SIGNED

$\frac{256}{2} \Rightarrow 128$

0-127

but signed

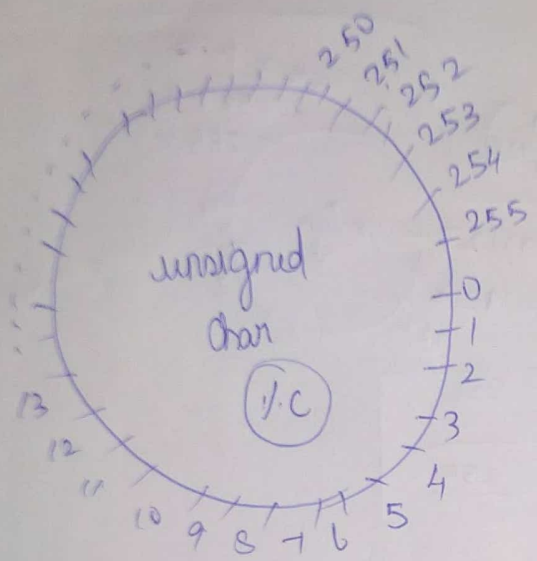
-128 to 127

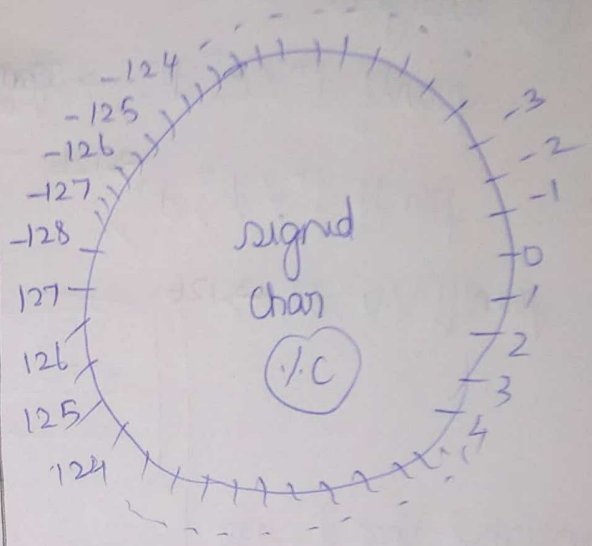
Character System:

* To represent symbols, letters, number in memory we have many character system.
* Most popular character system is ASCII.
* For every small case, upper case letter, number, special character we have ASCII value

A = 65	a = 97	0 =	Symbols
B = 66	b = 98	1 =	
⋮	⋮	2 =	
Z = ...	x = ...	⋮	
		9 =	

* If we combine all this, it will have less than 256 values.

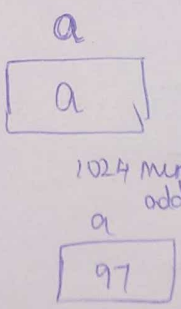




① Char a = 'a';

printf ("%c", a); → a

printf ("%d", a); → 97

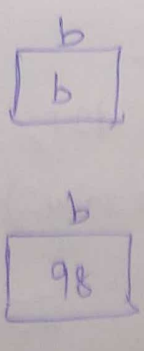


* In memory the character 'a' gets stored with binary value of 97; but for our understanding we put 'a'.

② Char b = 98;

printf ("%c", b); → b

printf ("%d", b); → 98



* If, %c → prints 'a'

%d → prints 97

③ char c = 'z';

printf("%c", c); → z

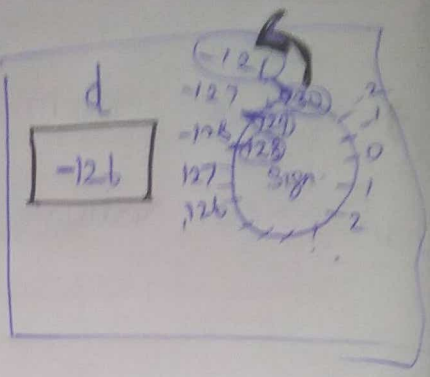
printf("%d", c); → 122

④ char d = 130;

256
130
−126

printf("%c", d); → garge value or symbol

printf("%d", d); → −126

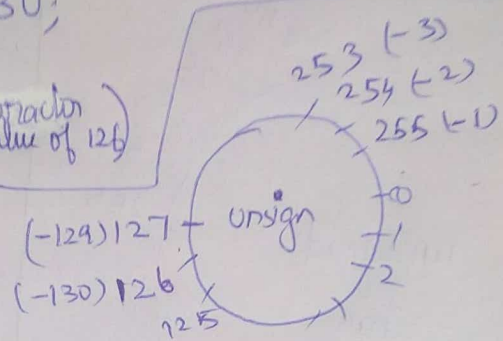


⑤ unsigned char d = −130;

printf("%c", d); → (character value of 126)

printf("%d", d); → 126

printf("%u", d); → 126



256
−130
126

256
−1
255

unsigned char d = −130

("%u", d) → 256
−130
+126

("%d", d) → 256
−130
+126

⑥ unsigned char d = −129;

("%c", d) → (Character value of 127)

("%d", d) → 127

("%u", d) → 127

256
−129

⑦ char a = '#'

printf("%c", a);

printf("%d", a);