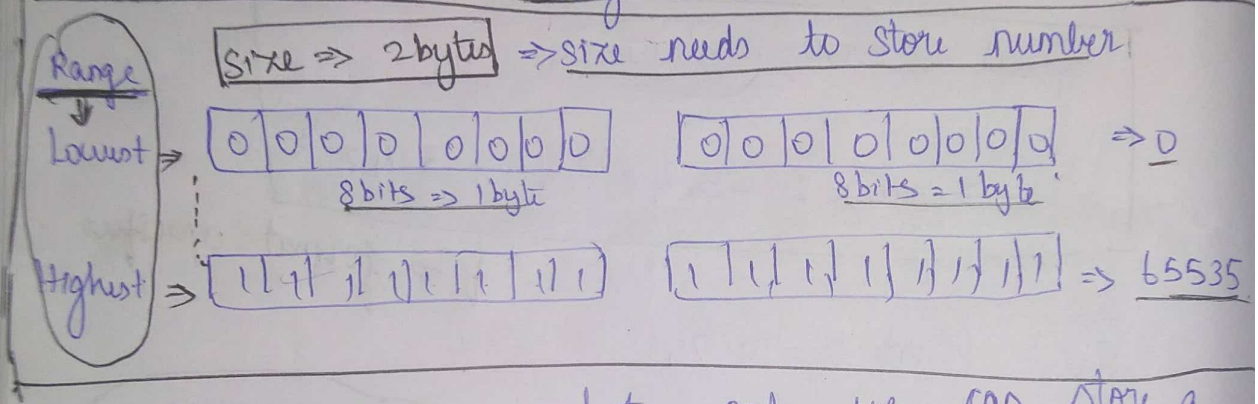
C_11 => Data Types in C - Part 2

* If we have 16 bit Compiler machine, then our integer size will be 2 bytes.

16 bit Machine :- (2 bytes)

Integer Size → 2 bytes.

Integer Range → Signed int => -32768 to 32767
Unsigned int => 0 to 65535



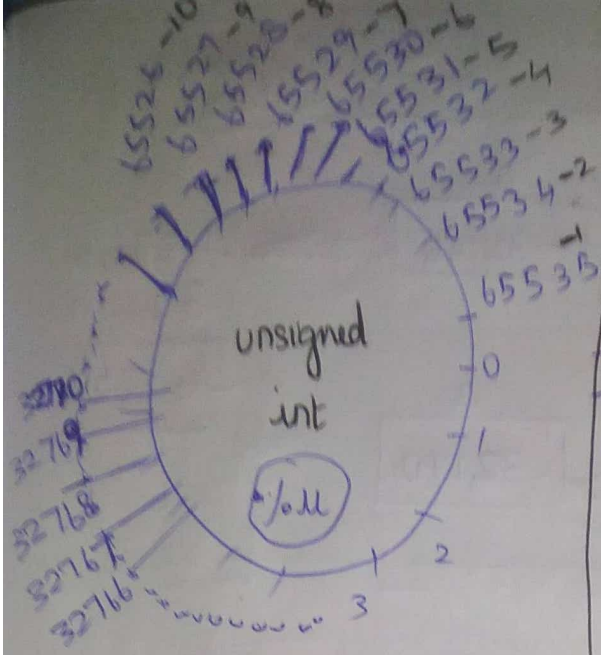
* Using this 2 bytes only we can store a number ranges from 0-65535 for unsigned int.

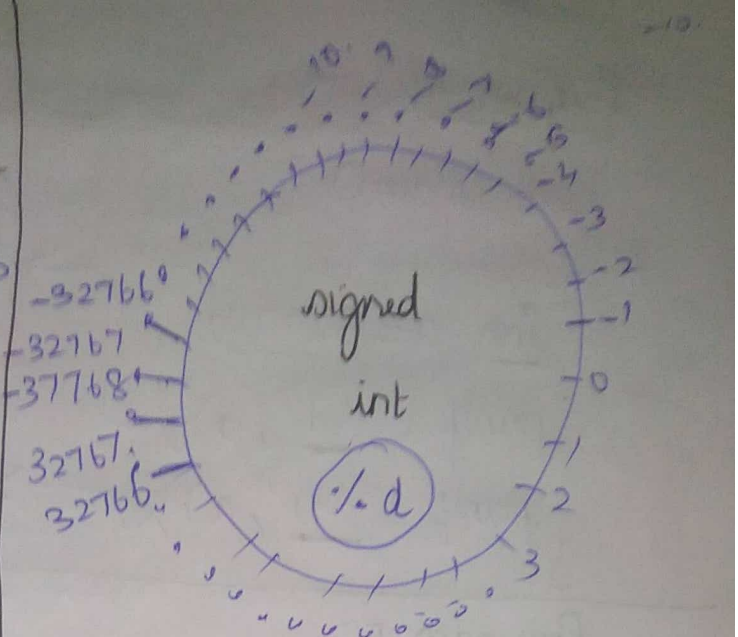
* For Signed int, $\frac{65535}{2} \Rightarrow 32768$.

-32768 to 32767

-32768 ..... , -2 , -1 , 0 , 1 , 2 , ... 32767
(0 → +ve)

* For unsigned int; format specifier is represented as %u.





Program 1:

```
#include <stdio.h>
#include <conio.h>
void main()
{
  int a = 32767;
  clrscr();
  printf("%d", a);
  getch();
}
```

a 1000
32767

Program 2:

```
void main()
{
  int a = 32768;
  clrscr();
  printf("%d", a);
  getch();
}
```

a
-32768

-65536
32767
= -32768

Note:

* 'int' by default both +ve & -ve will store.
* clrscr() gives after initialization of variable & getch() is written last.
* Execution starts from main()

Program 3:

```
void main()
{
  int a = 32768;
  clrscr();
  printf("%u", a);
  getch();
}
```

a
32768

Program 4:

```
Void main()
{
  int a = 32770;
  printf("%d", a);
  printf("%u", a);
}
```

a → -32766 ⟹ -65536 + 32770 = -32766

a → 32770

Program 5:

```
Void main()
{
  long int a = 32770;
  printf("%d", a);
  printf("%u", a);
}
```

Same value printed

long int so value beyond 32767.

a → 32770

a → 32770

We will have this value 32770; since long int have 4 bytes.

32770 lies within range of 4 bytes.

Program 6:-

```
Void main()
{
  int a = -10;
  printf("%u", a);
}
```

a → 65526

65536 − 10 = 65526

Total No. of bits ⇒ 65536

0 – 65535

Program 7:-

```
void main()
{
  long int a = 327777777;
  printf("%d", a);
  printf("%u", a);
}
```

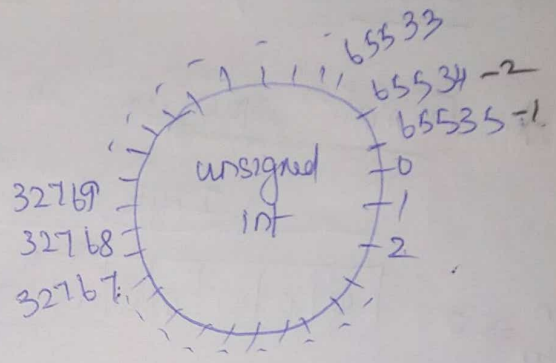
Same value printed because long int is the value lies inside the range of long int

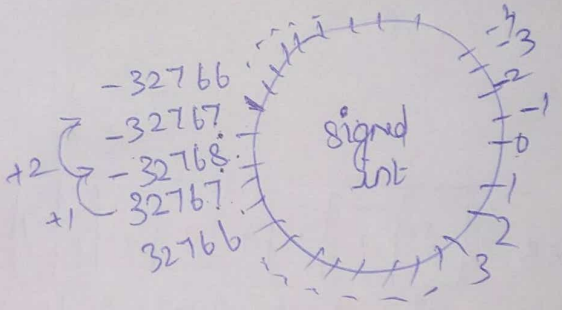
a → 327777777

a → 327777777

Program 8:-

```
void main()
{
  unsigned int a = -32767;
  printf("%d", a);
  printf("%u", a);
}
```





a → %u 32769

a → %d -32769 ⇒ Same value printed because %d takes signed value also.

But '%u' is unsigned; but -32767 is not there in unsigned int; so check with signed int and print it by MOD operation.

+(32766)
+(32767)

Signed Range

-(32768) ⇒ (32768)

-(32767) ⇒ (32769)

1) int a = 10;
%d → 10.

2) int a = -10;
%d → -10

3) int a = -10;
%u → 65526

65536
-10
65526

4) int a = -1;
%u → 65535

5) int a = 32767;
%d → 32767

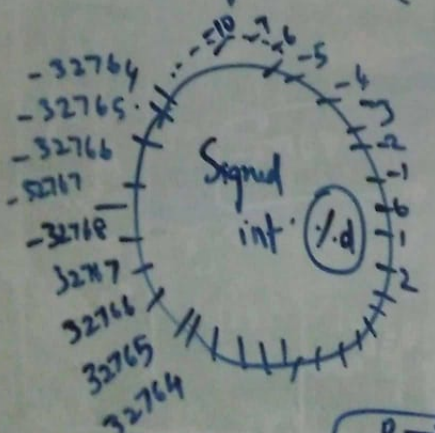
6) int a = 32768;
%d → -32768
%u → 32768

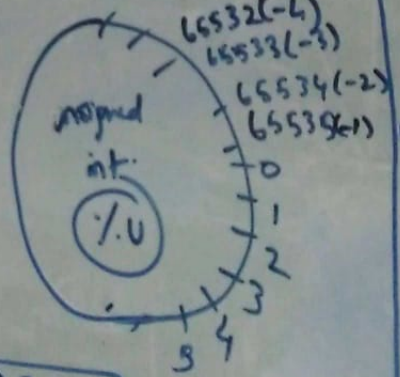
-65536
32768
-32768

7) unsigned int a = -32767;
%d → -32767.
%u → 32769.

65536
-32767
32769

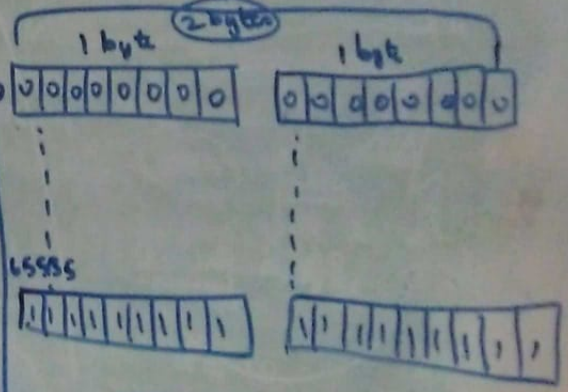
unsigned int → (0-65535)
Signed int → (-32768 to 32767)





0 → 0
-1 → 65535
-2 → 65534

2 bytes ⇒ 16 bits [int]



2^{16} = 65536 Values

Starts from 0;

Range → 0 - 65535.