

BASIC CONSTRUCTS OF C

NUMBER SYSTEMS IN C

- There are majorly four types of number systems used in a computer world:
 - Decimal (Base 10)
 - Binary (Base 2)
 - Octal (Base 8)
 - Hexadecimal (Base 16)

NUMBER SYSTEMS

- Binary number system:
 - In digital systems, instructions are given through electric signals
 - The easiest way to vary instructions through electric signals is two-state system – on and off.
 - The number system having just these two digits – 0 and 1 – is called **binary number system**.
 - Each binary digit is also called a **bit**
 - In any binary number, the rightmost digit is called **least significant bit (LSB)** and leftmost digit is called **most significant bit (MSB)**.

NUMBER SYSTEMS

- Decimal number system:
 - Decimal number system is a **base 10** number system having 10 digits from 0 to 9.
 - Any numerical quantity can be represented using these 10 digits
 - Most commonly used number system
- Octal number system
 - It has eight digits – 0, 1, 2, 3, 4, 5, 6 and 7.
 - Like decimal and binary number system, it's positional value is expressed in powers of 8.

NUMBER SYSTEMS

- Hexadecimal number system
 - It has 16 symbols – 0 to 9 and A to F
 - where A is equal to 10,
 - B is equal to 11 and so on till F.
 - Hexadecimal number system is also a positional value system with where each digit has its value expressed in powers of 16.

16^5	16^4	16^3	16^2	16^1	16^0
--------	--------	--------	--------	--------	--------

NUMBER SYSTEMS

HEXADECIMAL	DECIMAL	OCTAL	BINARY
0	0	0	0000
1	1	1	0001
2	2	2	0010
3	3	3	0011
4	4	4	0100
5	5	5	0101
6	6	6	0110
7	7	7	0111
8	8	10	1000
9	9	11	1001
A	10	12	1010
B	11	13	1011
C	12	14	1100
D	13	15	1101
E	14	16	1110
F	15	17	1111

QUICK EXERCISE

- Convert $(125)_{10}$ to binary, hexadecimal and octal
- Convert $(34)_8$ and $(34)_{16}$ into decimal
- Solutions:
 - $(125)_{10} = (1111101)_2 = (175)_8 = (7D)_{16}$
 - $(34)_8 = (28)_{10}$
 - $(34)_{16} = (52)_{10}$

COMPUTER MEMORY

- Computer memory is measured in terms of how many bits it can store.
- 1 byte (B) = 8 bits
- 1 Kilobytes (KB) = 1024 bytes
- 1 Megabyte (MB) = 1024 KB
- 1 Gigabyte (GB) = 1024 MB
- 1 Terabyte (TB) = 1024 GB
- 1 Exabyte (EB) = 1024 PB
- 1 Zettabyte = 1024 EB
- 1 Yottabyte (YB) = 1024 ZB

**LET US FIND OUT HOW DATATYPES
WORK...**

Number of bits	Values that can be stored (in binary)	Values that can be stored (in decimal)	Number of values	Range of values
1	0 1	0 1	2	0 ... 1
2	0 0 0 1 1 0 1 1	0 1 2 3	4	0 ... 3
3	000 001 010 011 100 101 110 111	0 1 2 3 4 5 6 7	8	0 ... 7
4	0000 0001 0010	0 1 2	16	0 ... 15
n			2^n	0.... 2^{n-1}

DATA TYPES IN C

- It is used to specify the type of data that we want to store.
- All data types will not occupy the same amount of memory space.
- Depending upon the memory allocated, the range of values will also differ
- Also data types can be further classified as **long**, **short**, **signed** and **unsigned**.
- The above keywords are called as **type qualifiers**.

DATATYPES AND THEIR RANGE

Type	Size	Range
char	1 byte	-128 to +127
unsigned char	1 byte	0 to 255
int	2 byte	-32768 to +32767
unsigned int	2 bytes	0 to 65535
long int	4 bytes	-2147483648 to +2147483647
unsigned long	4 bytes	0 to 4294967295
float	4 bytes	$\pm 3.4 * 10^{\pm 38}$
double	8 bytes	$\pm 1.7 * 10^{\pm 308}$
long double	10 bytes	$\pm 3.4 * 10^{\pm 4932}$

DATA TYPES

- long and short are used to modify the size.
- unsigned and signed are used to specify if a variable can store only positive values or both positive and negative values.
- In the absence of type qualifiers, the default is **short** and **signed**.
- Example:
long unsigned int x;
- The size of the data type varies from one processor to the other.

BITWISE OPERATORS

- Operates on the bits of the number
- It is a binary operator
- Converts both the operands into a binary system (only int and char allowed)
- Apply the bitwise operators which are:
 - & (Bitwise AND)
 - | (Bitwise OR)
 - ^ (Bitwise EX-OR)
 - >> (Bitwise Right shift)
 - << (Bitwise Left shift)

UNARY OPERATORS

- Can only operate on 1 operand
- The following operators are unary:
 - ! (Logical negation)
 - - (Unary negation)
 - ~ (Bitwise complement)
 - ++ (Unary increment)
 - -- (Unary decrement)
 - sizeof()
 - & (Address/Referencing operator)
 - * (Indirection/Dereferencing operator)
 - (type) (Typecasting operator)

QUICK EXERCISE

- Consider that short int is allocated 1 byte and long int is allocated 2 bytes. If we want to store the values -45, 546 and 35000 in variables x, y and z respectively. What should be the variable declaration?
 - `int x=-45;`
 - `long int y=546;`
 - `unsigned long int z= 35000;`

- What will be the final output?

```
int a=25, b=13,c,d;
```

```
c = a&b;
```

```
c=c<<3;
```

```
d=c++ %3;
```

```
d= (!d)? --c : c-- ;
```

Output:

72 72