Paper 102: Programming & Problem solving through C

Lecture-08:Unit-I
C Fundamentals

Qualifiers

- Qualifiers modify the behavior of the variable type to which they are applied.
- Declaration such as

int a;

Specifies that a is an integer which takes both positive and negative integer. i.e a is a signed integer by default.

can also be written as signed int a;

qualifier

If the variable a is used for storing only positive integers then it can be declared as

unsigned int a;

- Qualifiers can be classified into two types:
  - Size qualifiers
  - Sign qualifiers

Size qualifier

- Alters the size of the basic data type (int, char, float, double).
- There are two size qualifiers that can be applied to integers
  - short (min size is 16 bits)   e.g short int a;
  - Long (must be greater than or equal to an int, min size is 32 bits) e.g. long int a;
- The type qualifier long can also be applied to double, where the precision is more for long double.

  e.g. long double a;

  Precision of long double>= precision of double >= precision of float

Sign qualifier

- There are two types
  - Signed
  - Unsigned
- They can be applied to data types int and char.

  e.g. unsigned int b;

  ```
  signed int b;
  signed char;
  unsigned char;
  ```

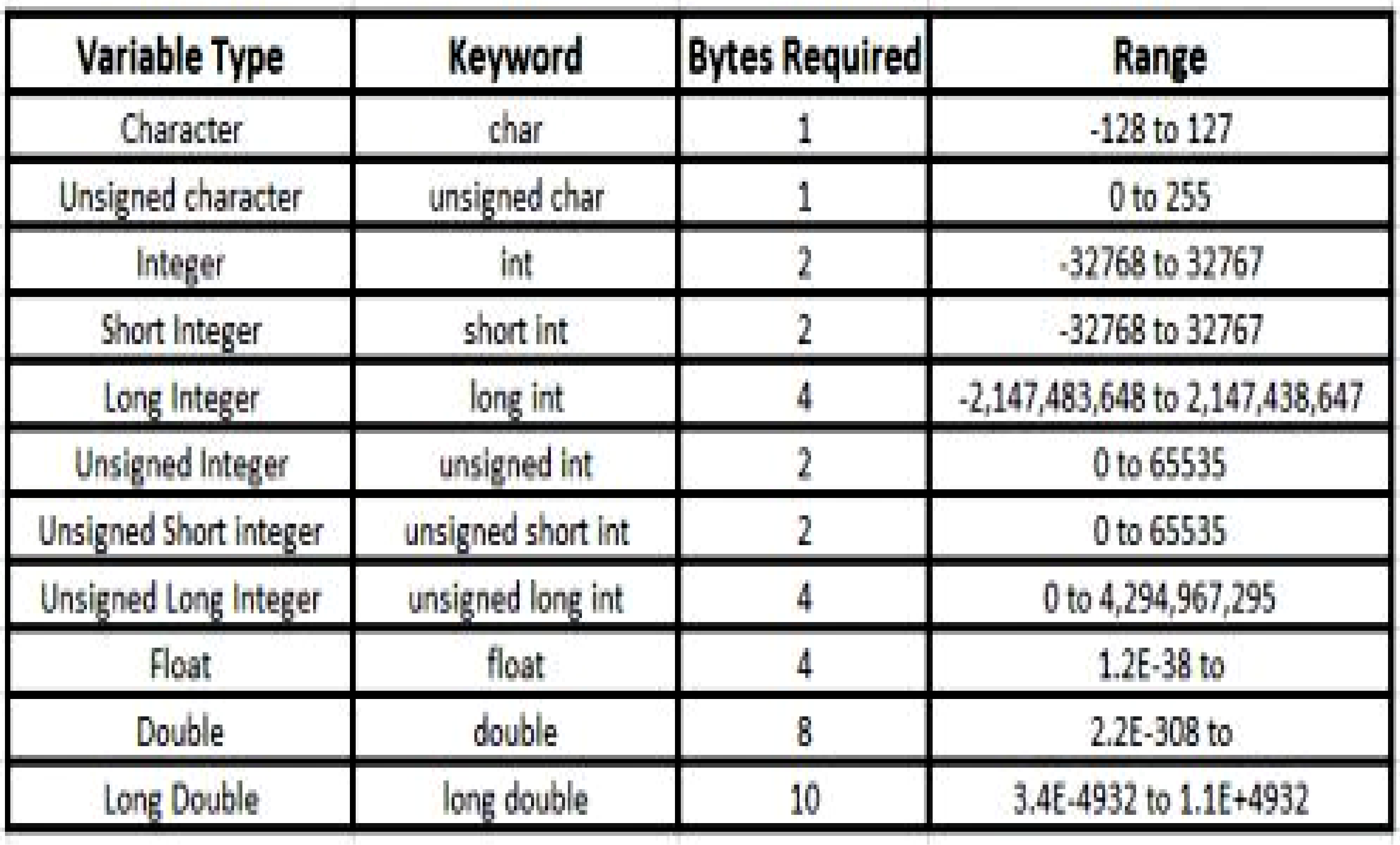
sizeof operator

- The sizeof operator is treated as a unary operator
- It gives the size of any data type in bytes

Example

```
/* program to illustrate the sizeof operator*/
#include<stdio.h>
void main()
{
        printf(“size of short int is %d”, sizeof(short int));
        printf(“\n size of an int is %d”, sizeof(int));
        printf(“\n size of long int is %d”, sizeof(long int));
}
```

C Data types and Ranges



Variable Type	Keyword	Bytes Required	Range
Character	char	1	-128 to 127
Unsigned character	unsigned char	1	0 to 255
Integer	int	2	-32768 to 32767
Short Integer	short int	2	-32768 to 32767
Long Integer	long int	4	-2,147,483,648 to 2,147,438,647
Unsigned Integer	unsigned int	2	0 to 65535
Unsigned Short Integer	unsigned short int	2	0 to 65535
Unsigned Long Integer	unsigned long int	4	0 to 4,294,967,295
Float	float	4	1.2E-38 to
Double	double	8	2.2E-308 to
Long Double	long double	10	3.4E-4932 to 1.1E+4932

Const qualifier

- values that do not change throughout the execution of the program can be declared using the const qualifier

  const float pi=3.14;

- This not only prevent the value from being changed, but also can be modified only in the declaration part and not throughout the rest of the program wherever it is used.

Bitwise operators

- A bitwise operator operates on each bit of data.
- These operators are used for testing, complementing or shifting bits to the right or left.
- They are meant for integers only.

A list of bitwise operators are given below:

Operator	Meaning
&	bitwise AND
|	bitwise OR
^	bitwise XOR
<<	shift left
>>	shift right
~	bitwise complement

Bitwise AND and OR

- The statement

  int a=13,b=7,c;

  c=a & b;

Uses the bitwise AND operator.

Each bit in c will be 1 only if the corresponding bits in both a and b are 1.

(assuming int is 16 bits)

Bitwise AND operator: a & b

```
 a     0000 0000 0000 1101        - decimal 13
 b     0000 0000 0000 0111        - decimal  7
a&b    0000 0000 0000 0101        - decimal  5
```

Bitwise OR operator: a | b

```
 a     0000 0000 0000 1101        - decimal 13
 b     0000 0000 0000 0111        - decimal  7
a|b    0000 0000 0000 1111        - decimal  15
```

Bitwise XOR

- The statement

int a=13,b=7,c;

c=a ^ b;

Bitwise XOR operator: a ^ b

```
 a      0000 0000 0000 1101          - decimal 13
 b      0000 0000 0000 0111          - decimal  7
a^b     0000 0000 0000 1010          - decimal  10
```

Left shift operator (<<)

c=a<<3;

The value of a is shifted to the left by three bit position

```
Drop off ←        0000 0000 0000 1101 ←inserts 0's       -decimal 13
After left shift by 3 places i.e. a<<3
                  0000 0000 0110 1000                     -decimal 104
```

The effect is similar to multiplying a by 2*2*2, i.e. 2^3

Right shift operator (>>)

c=a>>2;

- The value of a is shifted to the right by three bit position

inserts 0's $\longrightarrow$ 0000 0000 0000 1101 $\longrightarrow$ Drop off -decimal 13

After right shift by 2 places i.e. a>>2

0000 0000 0000 0011 -decimal 3

The effect is similar to dividing a by 2*2.

Shifting negative numbers: (only for right shift)

- If the number is negative, then 1 is inserted in the left for every bit shifted to the right.
  e.g. a = -3;
- the binary representation of a would be the two's complement of the magnitude.
- The binary representation of 3 is 0000 0000 0000 0011
- Complement it bitwise, 1111 1111 1111 1100
- Add 1 to this sequence to get 2's complement
  1111 1111 1111 1101
  This is the binary representation of -3

  c= a >> 2;
- This shift the value of a by 2 bit positions to the right and insert two 1's.
- Hence the value of c is 1111 1111 1111 1111
- To see the magnitude of this negative number, take 2's complement again resulting to 0000 0000 0000 0001 which is -1 in decimal

- Bitwise complement operator ~
- <<= and >>= bitwise assignment operator