# Storage Class (C)

There are four storage classes —.

1. Automatic
2. Register
3. Static
4. External

(1) Automatic → Automatic variables are declared inside a particular function and they are created when the function is called and destroyed when the function exits. These are local to a function in which they are defined.

By default, all the variables declared without a storage specification is automatic.

The key to declare an automatic storage class is auto.

```
#include<stdio.h>
void main()
{
   auto int i=1;   {
   auto int i=2;
   {
      auto int i=3;  printf("%d", i);
   }
    printf("%d",i);
}
printf("%d", i);
}
```

(2) Register → A variable is usually stored in memory but it is also possible to store a variable in CPU register by defining it as register variable.

It is much faster than a memory access. The keyword to declare an register storage class is register.

```
register int i;
```

(3) Static → The static storage class is declared with the keyword static.

(a) The scape of variable is local. But the value of variable persists until the end of the program.

(b) When the program is compiled, the it is never initialized again.

```
#include <stdio.h>
```

```
void stat_func() {
    static int x;
    x++;
printf("x=%d", x);
}
```

```
void main () {

  stat-func();
  stat-func();
  stat-func();

}
```

(4) External → Variables which are common to all functions and accessible by all the functions of a program are external variables. It is declared outside of all functions.

(a) Its default value (initial) is 0.

(b) The scope of the external variables are global.

```
#include <stdio.h>
 int i=10;
 void function1 () {
      printf("%d", i); i++;
 }

 void function2() {
      printf ("%d", i); i--;
}
```

```
void main () {
    printf ("%d", i);
     function1 ();
     function2 ();
    printf ("%d", i);
}
```

(5) Extern → Global variables are usually declared in the beginning

01 December Thursday

of the program before all the function.

However, c provides a facility to declare any variable as global. This is using by the keyword storage class extern.

```
eg. #include <stdio.h>

     void function1 () {
         extern int i;
          i = 20;
         printf ("%d", i);
     }
         int i=10;
```

```
void main()
{   function1();
    printf("%d", i);

}
```

## Table (Storage Class)

① Auto :—

ⓐ Storage → Memory

ⓑ Default Value → Garbage Value

ⓒ Scope → Local to the block, in which the variable is declared.

ⓓ Lifetime → Till the control remains within the block in which it is defined.

(*) • A variable is an entity that has a value and is known to the program by a name.

• A variable definition associates a memory location with the variable name.

• A variable can have only one value assigned to it at any given time during the execution of the program. Its value may be change during the execution of the program.

(*) Precision → Number of significant digits after the decimal point.

eg. float = Single precision floating pt. no.

double = Double " " " .

August 13
Saturday

(*) Qualifiers → Additional data types.

(a) It alters the characteristics of data type such as size or sign.

eg. Short and Long. (Size qualifiers)

(b) eg. Signed and unsigned (Sign qualifiers)

P.T.O

15 August
Monday

## Enumeration Constants

(✱) Enumeration is a user defined type with values ranging over a finite set of idutifiers, called enumeration constants.

e.g. enum color {red, blue, green};

(values) 0 1 2

16 August
Tuesday

e.g. enum wheather {hot, warm=0, cold, we

0 0 1 2

interchangeldy used.

e.g. enum Week {sun, mon, tue, wed, thu, fri, sat};

| AUG | M | T | W | T | F | S | S | M | T | W | T | F | S | S | M | T | W | T | F | S | S | M | T | W | T | F | S | S | M | T | W |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2011 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |

## Difference between Character Constant and String Constant :-

'A' → Character Constant —

@ Occupying Single Byte.

"A" → String Constant —

@ Occupying Two Bytes

ⓑ All string terminates with a null character or '\0' with the value of 0.

## Logical Operators.

① Logical AND →

a>b && x==10

② Logical OR →

a<m || a<n

③ Logical Not →

$$!(x >= y)$$

## Bitwise Operators

① Bitwise AND →

$\boxed{a = 13 \\ b = 7}$

integer ↓ 16 bits (2 bytes)

$\boxed{C = a \& b;}$

20 August Saturday

a = 0000 0000 0000 1101

b = 0000 0000 0000 0111

a&b = 0000 0000 0000 0101

[ each bit in C will be 1 only if the corresponding bits in both a and b are 1. ]

21 SUNDAY

② Bitwise OR →

$$\boxed{c = a|b;}$$

a = 0000 0000 0000 1101

b = 0000 0000 0000 0111

a|b = 0000 0000 0000 1111

[each bit in c will be 1 only if the corresponding one of the bits of a or b is 1.]

③ Bitwise XOR →

$$\boxed{c = a \wedge b}$$

a = 0000 0000 0000 1101

b = 0000 0000 0000 0111

0000 0000 0000 1010

[c will be 1 whenever the corresponding bits. in a and b differ.]

24 August
Wednesday

④ Left - Shift →

$\boxed{c = a << 3}$

drop off ← 0000 0000 0000 1101 ← Insert 0's

↓ using left shift a<<3

0000 0000 0110 1000 ㊀*

⑤ Right - Shift →

25 August
Thursday

$\boxed{c = a >> 3}$

Insert 0's → 0000 0000 0000 1101 ⟶ drop off

↓ using right shift a>>3.

0000 0000 0000 0001 (**)

(*) Shifting a variable to the left by one bit position is to multiply it by 2.

eg. here a is shifted by 3 positions.

i.e/ a is multiplied by $2^3$.

i.e/ initial value a=13.

$\downarrow$ a<<3.

$a = 13 * 2^3 = 13 * 8 = 104.$

(**) Shifting a variable to the right by one bit position is to divide by 2.

eg. here also a is shifted by 3 positions.

i.e/ a is divided by $2^3$.

i.e/ initial value a=13

$\downarrow$ a>>3

13(10·7
8
50
5
1

$a = 13/8 = 10$

## Shifting Negative Numbers

$a = -3$.

i) Representation of 3 →

0000 0000 0000 0011

ii) 1's Complement →

1111 1111 1111 1100

iii) 2's Complement →

30 August Tuesday

$$\begin{array}{rr} & 1111\ 1111\ 1111\ 1100 \\ \text{(add)} & 1 \\ \hline & 1111\ 1111\ 1111\ 1101 \\ \hline \end{array}$$

↓

Representation of (−3).

→

$$\boxed{C = a >> 2}$$

i.e> shift 2 bit positions to the right and insert two 1's to the left.

insert 1's → 1111 1111 1111 1101 ⟶ drop off

↓

1111 1111 1111 1111

i.e> 2's Complement →.

0000 0000 0000 0000   1's compliment

1

0000 0000 0000 0001   2's compliment

(maguitude is (-1)).

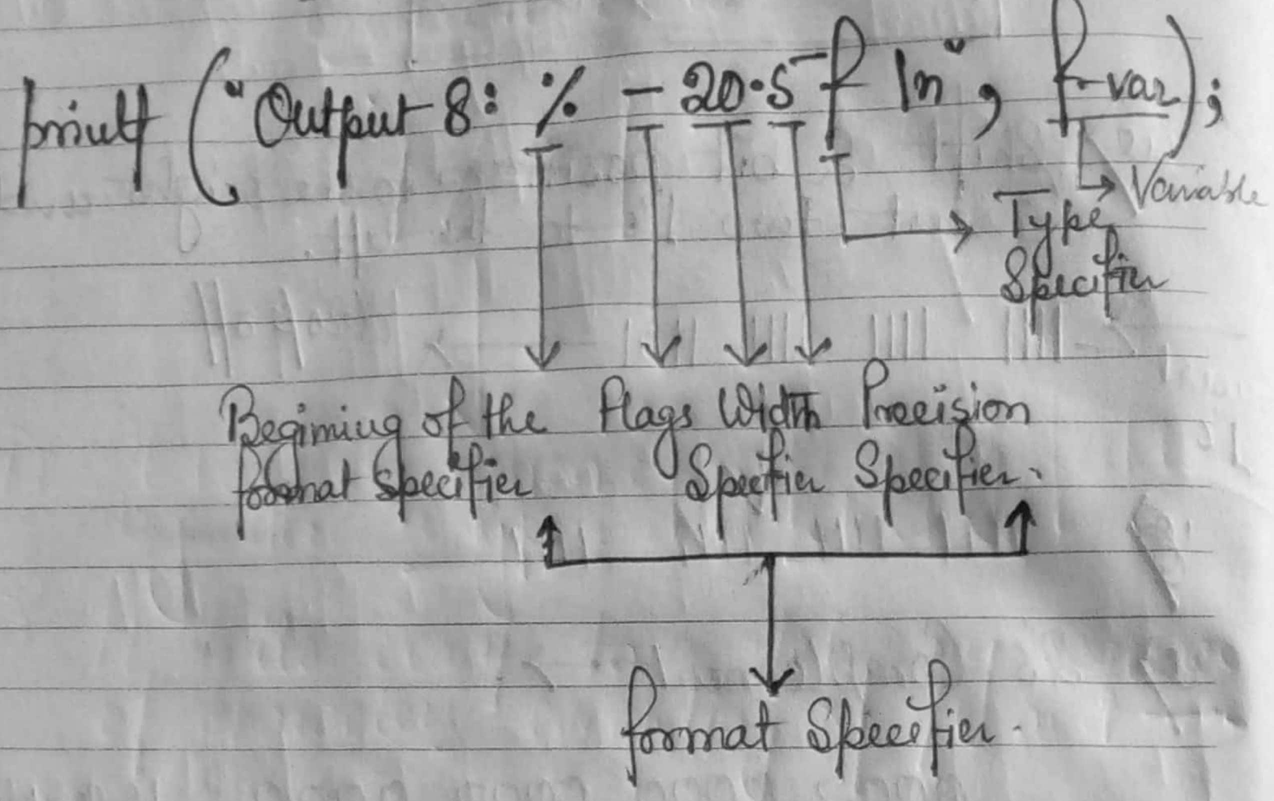
Note :— ① In case of unsigned integers, inserted left bit is always zero.

② Case of right shift —
left bit caries with the sign.

③ Case of left shift —.
right bit caries with the sign.

02 September Friday

# Format Specifier



```
printf ("Output 8: % -20.5f \n", f_var);
```

03 September Saturday

# Chart of Format Specifiers

## Integer :—

① %d } → Signed decimal integer
② %i }

③ %o → unsigned decimal integer.

04 SUNDAY

④ %x, %X → unsigned hexadicimal int.

⑤ %f, %e, %g → Signed value of floating point

⑥ %u → unsigned decimal integer.

## Qualified Data Types :—

| | | |
|---|---|---|
| ① | %ld, %li | long decimal int. |
| ② | %lu | unsigned long decimal int. |
| ③ | %hd, %hi | Short decimal int. |
| ④ | %hu | unsigned Short decimal int. |
| ⑤ | %le, %lf, %lg | Signed double. |
| ⑥ | %Le, %Lf, %Lg | Signed long double. |
| ⑦ | %lo | octal long integer |
| ⑧ | %lx | hexadecimal long int. |

## Character Datatype :—

| | | |
|---|---|---|
| ① | %c | Single character |
| ② | %s. | Sequence of charater |
| ③ | %% | prints the % charatter. |