STORAGE CLASSES:

All variables will have not only data type, but also a storage class.

In order to define a variable totally, we need to mention both its data type and storage class.

Storage Class of a Variable defines:

i) Where the variable is stored
ii) Default initial value of variable.
iii) Scope of variable i.e, in which functions the variable is available.
iv) What is a life of a variable i.e, how long will the variable exist.

There are 4 types of storage classes in C:

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

Inside **CPU**, 2 types of storage spaces are present
i) memory unit. ii) registers. (limited in number)

a) Automatic storage classes:

Features of a variable having an automatic storage class will be:

Storage	→ memory
Default initial value	→ garbage value
Scope	→ Local to the block in which it is defined.
Life	→ Till the control remains within the block, the variable is defined.
Keyword	→ auto

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

Output: 1011 (unpredictable/unexpected value if differs from compiler to compiler)

Program to demonstrate scope and life of automatic variable:

```
main()
{
 auto int i;
{
        {
                {
                        printf("%d",i);
                }
                printf("%d",i);
        }
}
```

Output: 1 1 1

Reason: in the above program, i is a automatic variable, whose scope is local, to the block, in which it is defined.

There fore when the control comes out of the block, in which the variable is defined. The variable and its value are lost.

```
 main()
{
auto int i=1;
        {
                auto int i=2;
                {
                        auto int i=3;
                        printf("%d",i);
        }
                printf("%d",i);
        }
        printf("%d",i);
}
```

Output:

3 2 1

Reason: Compiler treats 3 i's as totally different variables, since they are defined in different blocks.Once control comes out of the inner most block, the variable i with value 3 is lost. Therefore, i in 2nd printf() statement. Refers to i with value 2. When control comes out of the next

inner most blocks i=2 is lost and 3rd pritnf() statement. Refers to i with value i.

b) Register Storage class:

Features of a variable defined under register storage class are:

Storage	→ CPU register
Default initial value	→ garbage value
Scope	→ Local to the block in which variable is defined.
Life	→ Till the control remains within the block, the variable is defined.
Keyword	→ register

A value stored in CPU register, can be accessed faster than a value stored in memeory. If a variable is to be used at many places/ many times in the program, then it would be better to declare its storage class as register.

Disadvantage: CPU registers will be limited in number. If all the register of CPU are busy with some other task, then variable storage class is taken as auto.

```
#include<stdio.h>
mian()
{
register int i;
for (i=1;i<=10;i++)
printf("%d \t ",i);
}
```

Output: 1 2 3 4 5 6 7 8 9 10

c) Static Storage Class:

Features of a variable defined with static storage class are:

Storage	→ Memory
Default initial value	→ Zero
Scope	→ Local to the block in which variable is defined.
Life	→ Till the control remains within the block, the variable is defined.
Keyword	→ static

Difference between static and automatic variables is that they don't disappear, when the function is not active. Their value exists and when control comes back to same function again, the static variables have the same values, they had last time.

Ex:

```
#include<stdio.h>
void increment();
main()
{
increment();
increment();
increment();
}
void inclremnt()
{
static int i=1;
printf("%d",i);
i=i+1;
}
```

Output: 1 2 3

Reason: In the above program, if we declare variable i as integer and static variable the output will be 1 2 3.

Ex:

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

Output: 1 1 1

Reason: In the above program, when I is of automatic storage class, each time increment() is called, it will be re-initialized to 1. When function terminates value i=2 will be lost.

Instead of 'auto' if we use 'static', i will be initialized to 1 only once. During first call of increment 'i' will be 2 and i is of static type, the value will not be lost. Similarly during second function call of increment(), i will be 3.

d) External Storage Class:

Features of a variable defined with External storage class are:

Storage	→ Memory
Default initial value	→ Zero
Scope	→ Global
Life	→ As long as execution of program doesn't come to an end.
Keyword	→ Extern

External variables are declared outside the function, So they are available to all the functions in program.

Ex:

```
#include<stdio.h>
int i=20;
main()
{
extern int j;
printf("%d",i);
printf("%d",j);
}
int j=40;
```

Out put: 20 40

Keyword Extern indicates that variable j is defined some where after or outside the main().

Here i and j are global variable. Therefore they are defined outside function, both enjoy external storage class.

Difference between them is:

extern int j; → Declaration; initially memory is not registered for it.

Int j=40; → Defination

S.No	Storage Classes	Keyword	Storage	Default Value	Scope	Life Time
1	Automatic	Auto	CPU	Garbage	Local	→ Till the control remains within the block, the variable is defined.
2	Register	Register	Register	Garbage	Local	→ Till the control remains within the block, the variable is defined.
3	Static	Static	CPU	Zero	Local	Persist between diff function calls
4	External	Extern	CPU	Zero	Global	The variable exits through out the execution of the program.