
Purpose of using storage class:

It tell us

1. Where the variable would be stored.
2. What will be the initial value if not explicitly assigned?
3. Scope i.e. portion of a program where the variable is visible.
4. Longevity i.e. the life of a variable.

In C we have **four** storage classes.

1. Automatic storage class
2. Extern/External storage class
3. Static storage class
4. Register storage class

1. **Automatic Storage Class**:

   a. Variables having automatic storage class are called automatic variables. Automatic variables are declared inside a function/block in which they are to be utilized.

   b. They are created when the function is called or control enters into the block and destroyed automatically when the function is exited or control leaves the block.

   c. The features of these variables are as follows:

1.	Storage:	Main Memory.
2.	Default initial Value:	An unpredictable value, which is often called a garbage value.
3.	Scope:	Inside the block/function in which it is declared.
4.	Longevity:	Till the control remains within the block/function.

Examples:

```
void main()
{
        auto int m = 30;
        void fun1();
         fun1();
         printf("\n m = %d",m);
}
void fun1()
{
       int  m = 10;
       printf("\n m = %d", m);
}

Output:
            m = 10
            m = 30
```

```
void main()
{
        auto int m = 30;
        {
                auto int m = 20;
                printf("\n m = %d",m);
         }
         printf("\n m = %d",m);
}

Output:
            m = 20
            m = 30
```

REMEMBER: **Global Variables cannot be declared as AUTO**.

NOTE: If Storage class is not specified during declaration of a local variable, the storage class will be **auto** by default.

2. External Storage class

a. Variables having external storage class/declared at the beginning of the program outside any function are called external variables.

b. The features of such variables are as follows:

1. Storage: Main Memory.
2. Default initial Value: Zero.
3. Scope: Global from point of its declaration.
4. Longevity: As long the program execution does not cease.

Examples:

#include <stdio.h>
int i=7; //Declaration & definition
void main()
{
printf("\n i = %d ", i);
}
 	
Output:
i = 7
	#include <stdio.h>
	void main()
	{
	extern int i; //Declaration
	printf("\n i=%d ", i);
	}
	int i=7; //Definition
	Output:
	i = 7

3. Static Storage class

a. The static variable is of two types –

- ✓ Internal static variable –
  When a local variable is declared as static, it is called Local (Internal) Static variable.

- ✓ External static variable –
  When an external variable declared as static, it is called External static variable.

b. Features of **Internal** Static variable –

1. Storage: Main Memory.
2. Default initial Value: Zero.
   (If explicitly initialized, **initialization takes place only once** when the function is called for the first time.)
3. Scope: Within the function in which it is declared.
4. Longevity: Till the end of the program execution.

Example:

```
#include <stdio.h>
void main()
{
        int i;
        void series();
        for(i = 1; i <= 5; i++)
                        series();
 }
void series()
{
        static int n = 10;
        printf("%d  ", n);
        n = n + 5;
}
```

Output will be:
10 15 20 25 30

In order to differentiate between internal static and automatic variables, the definition of the function **series()** is modified in the above program as follows:

```
void series()
{
        auto int n = 10;
        printf(" %d ", n);
        n = n + 5;
 }
```

Output will now be:
10 10 10 10 10

4. Register Storage class

We use this storage class to make a request to the compiler that the variable to be stored in one of the CPU registers, instead of memory because CPU

register access is faster than memory access. This storage class is used for variables that are frequently accessed.

If the CPU fails to keep the variable in its registers, then the compiler converts the storage class of the variable to AUTO and the variable stored in the memory.

Features of register variable –

1. Storage: CPU register.
2. Default initial Value: Garbage Value.
3. Scope: Within the block/function in which it is declared.
4. Longevity: Till the control remains within the block/function in which the variable is declared.

Example:

```
void main()
{
        register int i;
        for(i = 1;i <= 3; i++)
                printf("%d   ", i);
}
```

Output will be-

1 2 3

A register is identified by a name and it is not a memory location. So use of address operator is illegal.

Which to use when?

Static: We use this storage class if we want the value of a variable to persist between different function calls.

Register: If we have to access the variable frequently, e.g. loop counter, we use it.

Extern: We use it for those variables that are being used by almost all the functions in the program to avoid passing of these variables as arguments when making a function call.

Auto: We use it mostly in the program because often it so happens that once we have used the variables in a function, we do not mind to loose them.