Storage Classes in C++ Programming

Storage class of a variable defines the **lifetime and visibility** of a variable. Lifetime means the duration till which the variable remains active and visibility defines in which module of the program the variable is accessible. They are:

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

Storage Class	Keyword	Lifetime	Visibility	Initial Value
Automatic	auto	Function Block	Local	Garbage
External	extern	Whole Program	Global	Zero
Static	static	Whole Program	Local	Zero
Register	register	Function Block	Local	Garbage

1. Automatic Storage Class

This variable is **visible** only within the function it is declared and its **lifetime** is same as the lifetime of the function as well. This is the default storage class we have been using so far. It applies to local variables only and the variable is visible only inside the function in which it is declared and it dies as soon as the function execution is over. If not initialized, variables of class auto contains garbage value.

- The value is lost after the execution of function.

Syntax: auto datatype var_name1 [= value];

Example:

```
int var; // by default, storage class is auto
auto int var;    // auto int j=10;
```

Example Program:

```
#include<stdio.h>

void display();
void main()
{
        auto int a=10;        //OR int a=10;
        printf("\n A1 : %d",a);
        display();
        printf("\n A3 : %d",a);
}
void display()
{
        int a=20;                //OR auto int a=20;
        printf("\n A2 : %d",a);
}
```

Output :

```
A1 : 10
A2 : 20
A3 : 10
```

2. External Storage Class

External storage class reference to a **global variable** declared outside the given program. *extern* keyword is used to declare external variables. They are **visible** throughout the program and its **lifetime** is same as the lifetime of the program where it is declared. This visible to all the functions present in the program.

```
Syntax : extern datatype var_name1;
 Example: extern float a;
```

The extern keyword is optional, there is no need to write it.

- The scope of external variable is the entire program.
- If not initialized external variable is assigned a zero value.
- The value is not lost after the execution of function.

Example Program:

```
#include<stdio.h>

              void display();
              extern int a=10;                    //global variable
              void main()
              {
                     printf("\nA : %d",a);
                     increment();
                     display();
                     printf("\nA : %d",a);

              }
              void increment()
              {
                     a = 20;
              }
              void display()
              {
                     printf("\nA : %d",a);
              }
```

Output :

```
A : 10
A : 20
A : 20
```

3. Static Storage Class

Static storage class ensures a variable has the **visibility** mode of a local variable but **lifetime** of an external variable. It can be used only within the function where it is declared but destroyed only after the program execution has finished. The default initial value of static variable is **0**. The value of a static variable persists between function calls

.

Syntax: static datatype var_name1 [= value];
Example: static int x = 101;
static float sum;

During multiple calling static variables retains their previous value.

- We must declare variable as static.
- Static variables can't be accessed outside the function.
- If not initialized static variables have zero as initial value.

Example of static storage class

```
#include<stdio.h>
voiddisplay();
void main()
{
display();
display();
display();
                        }
void display()
                    {
static int a=1;
 printf("\nA : %d",a)
a++;
                    }
```


Output:

A : 1
A : 2
A : 3

In the above example, we does not use static keyword then the output will be :

```
Output :   A : 1
           A : 1
           A : 1
```

4. Register Storage Class

Variables of class 'register' are stored in CPU registers instead of memory which allows faster access. It has its lifetime and visibility same as automatic variable. The scope of the variable is local to the function in which it is defined and it dies as soon as the function execution is over. It contains some garbage value if not initialized. The purpose of creating register variable is to increase access speed and makes program run faster. As register spaceis very limited, so only those variables which requires fast access should be made register Itis declared as:

Syntax: register datatype var_name1 [= value];
Example: register int count
register char a;

C program to create automatic, global(extern) variables.

```
     #includ
      e<stdi
      o.h>
      void
      main()
       {
register int a=10;
printf("\nA : %d",a);
     }
```

```
Output :

 A : 10
```