Storage Classes

A storage Class define the storage type of variable , Scope of the variable and lifetime of variable. These are 4 types of storage classes that are used in C Programming.

Syntax: <storageclass><data_type><name of variable>

Types:

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

1. Automatic Storage Class

- The automatic storage class is the default storage class of C language. It means if we don't declare any storage class the compiler considers it automatic storage class.
- It is represented by keyword auto.
- The auto variables are stored in memory and their default value is garbage.
- Every local variable is automatic in C by default.

Syntax: auto data_type variable_name

Example

```
#include<stdio.h>
#include<conio.h>
void main()
{
auto int i;
```

```
printf("%d",i);

getch();

}
```

2. Static Storage Class

- The static variables are different from all other variables. The static variables are local to the function or block in which they are declared.
- The keyword used to declare static variable is static.
- The default value of static variable is zero.
- Static local variables are visible only to the function or the block in which they are defined.

Syntax: static data_type variable_name

Example

```
#include<stdio.h>
#include<conio.h>
void main()
{
clrscr();
static int i;
static float f;
static char s[100];
printf(" %d %f %s",,i,f); // the initial default value of i, and f will be printed.
getch();
}
```

3. External storage class

- A variable that is declared outside any function is a Global variable. They are also called global variables.

- The variables that remain alive and active throughout the entire program are known external variables.
- The extern keyword is used to declare extern variable
- The scope of these variables is global i.e. visible to all function
- Default initial value of the static integral variable is 0 otherwise null.

Syntax: extern data_type variable_name

Example

```
#include<stdio.h>
#include<conio.h>
 int  x = 5;
int  fun();
void  main()
{
printf("  %d ", x);
printf("\n  %d ", fun();
getch();
}
int  fun()
{
x = x + 5;
return(  x );
}
```

4. Register Storage Class

- The keyword **register** is used to declare a register storage class.
- We can store pointers into the register, i.e., a register can store the address of a variable.
- The variables defined as the register is allocated the memory into the CPU registers depending upon the size of the memory remaining in the CPU.

Syntax: register data_type variable_name

Example

```
#include <stdio.h>
int main()
{
register int a; // variable a is allocated memory in the CPU register. The initial default value of a is 0.
printf("%d",a);
}
```