In C language, each variable has a storage class which decides the following things:

* **scope** i.e where the value of the variable would be available inside a program.
* **default initial value** i.e if we do not explicitly initialize that variable, what will be its default initial value.
* **lifetime** of that variable i.e for how long will that variable exist.

The following storage classes are most oftenly used in C programming,

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

**1.Automatic variables : auto**

**Scope :** Variable defined with auto storage class are local to the function block inside which they are defined.

**Default Initial Value:** Any random value i.e garbage value or zero.

**Lifetime:** Till the end of the function/method block where the variable is defined.

A variable declared inside a function without any storage class specification, **is by default an automatic variable**. They are created when a function is called and are destroyed automatically when the function's execution is completed. **Automatic variables can also be called local variables because they are local to a function.** By default they are assigned garbage value by the compiler.

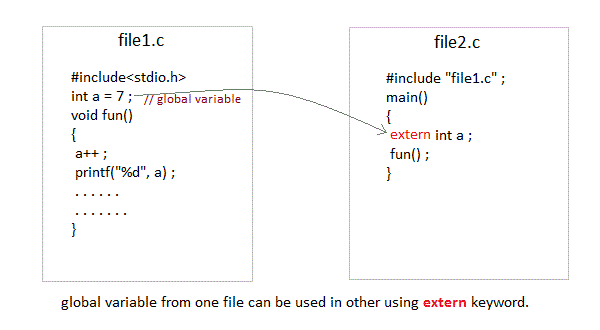
#include<stdio.h>  
void main()  
{  
 int detail;  
 // or   
 auto int details; //Both are same  
}

### **External or Global variable**

**Scope:** Global i.e everywhere in the program. These variables are not bound by any function, they are available everywhere.

**Default initial value:** 0(zero).

**Lifetime:** Till the program doesn't finish its execution, you can access global variables.

****

**Ex :**

**Hello.c**

#include <stdio.h>

**int a =7;**

void fun()

{

a++;

printf("%d\n",a);

}

**World.c** //compile Hello.c before executing World.c

#include <stdio.h>

**#include "Hello.c";**

void main(){

**extern int a;**

fun();

a++;

printf("%d\n",a);

}

//output:

8

9

**Note** : make sure both programs are in same folder.

**Static variables**

**Scope**: Local to the block in which the variable is defined, but the value will be persisted until the end of the program.

**Default initial value:** 0(Zero).

**Lifetime:** Till the whole program doesn't finish its execution.

A static variable tells the compiler to **persist/save** the variable until the end of program. Instead of creating and destroying a variable every time when it comes into and goes out of scope, static variable is initialized only once and remains into existence till the end of the program. A static variable can either be internal or external depending upon the place of declaration. Scope of internal static variable remains inside the function in which it is defined. External static variables remain restricted to scope of file in which they are declared.

They are assigned 0 (zero) as default value by the compiler.

Ex :

#include<stdio.h>

void main()  
{  
 test();  
 test();  
 test();  
}  
  
void test()  
{  
 static int a = 0; //a static variable  
 a = a + 1;  
 printf("%d\t",a);  
}

Output :

1 2 3

**Register Variable**

**Scope**: Local to the function in which it is declared.

**Default** initial value: Any random value i.e garbage value

**Lifetime**: Till the end of function/method block, in which the variable is defined.

Register variables inform the compiler to store the variable in **CPU register instead of memory.** Register variables have **faster accessibility** than a normal variable. Generally, the frequently used variables are kept in registers. **But only a few variables can be placed inside registers**. One application of register storage class can be in using loops, where the variable gets used a number of times in the program, in a very short span of time.

NOTE: **We can never get the address of such variables.**

#include <stdio.h>

void main()

{

**register** int b =5;

printf("%d",b);

}

**Which storage class should be used and when**

To improve the speed of execution of the program and to carefully use the memory space occupied by the variables, following points should be kept in mind while using storage classes:

* We should use static storage class only when we want the value of the variable to remain same every time we call it using different function calls.
* We should use register storage class only for those variables that are used in our program very oftenly. CPU registers are limited and thus should be used carefully.
* We should use external or global storage class only for those variables that are being used by almost all the functions in the program.
* If we do not have the purpose of any of the above mentioned storage classes, then we should use the automatic storage class.