Course Name: O Level (2nd Sem B2 and B3 Batch) **Subject:** C Language
Topic: Storage Class in C Contined.. **Date:** 20-April-2020

Static Storage Class:

As the name suggest, the value of static exist until the end of the program. A static variable may be either an internal type or an external type depending on the piece of declaration. Internal static variables are those which are declared inside a function. A static variable is initialized only once, when the program is compiled. It is never initialized again. An external static variable is declared outside that program. The difference between a static external variable and simple external variable is that the static external variable is available only within the file where it is defined while the simple external variables can be accessed by other files.

Keyword	:	**static**
Storage Location	:	**Main memory**
Initial Value	:	**Zero and can be initialize once only**
Life	:	**depends on function calls and the whole application or program**
Scope	:	**Local to the block**

Syntax : static [data_type] [variable_name];
Example : static a;

Compare the two programs and their output given to understand the difference between the **automatic** and **static** storage classes.

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

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

Output of the above programs would be	
1 1 1	1 2 3

In the above program **increment( )** function gets called from **main( )** thrice. Each time it increments the value of **i** and prints it. The only difference in the two programs is that one uses an **auto** storage class for variable **i**, whereas the other uses **static** storage class. Like **auto** variables, **static** variables are also local to the block in which they are declared. The difference between them is that **static** variables don't disappear when the function is no longer active. Their values persist. If the control comes back to the same function again the **static** variables have the same values they had last time around. In the above example, when variable **i** is **auto**, each time **increment( )** is called it is re-initialized to one. When the function terminates **i**

vanishes and its new value of 2 is lost. The result: no matter how many times we call **increment( ), i** is initialized to 1 every time. On the other hand, if **i** is **static**, it is initialized to 1 only once. It is never initialized again. During the first call to **increment( )**, **i** is incremented to 2. Because **i** is static, this value persists.

What is the difference between static and auto variables?

Static variable	Auto variable
We have to specify the storage class to make a variable static.	It is the default storage class.
If it is not assigned any value then it will give 0 as out put.	If it is not assigned any value then it will give garbage value as output.
It is visible to the block in which it is declared and also in the function where it will passed.	It is visible to the block in which the variable is declared.
It retains its value between different function calls. It holds its last value.	It retains its value till the control remains in the block in which the variable is declared.

External Storage Class:
Variables that are both alive and active throughout the entire program are known as external variables. They are also known as global variables. Unlike function in the program external variables are declared outside a function.

Keyword	:	**extern**
Storage Location	:	**Main memory**
Initial Value	:	**Zero**
Life	:	**Until the program ends.**
Scope	:	**Global to the program.**

Syntax **:** extern [data_type] [variable_name];
Example **:** extern int a;

Example:

```
#include<stdio.h>
#include<conio.h>
void increment (void);
void decrement (void);
int i ;
void main( )
{
printf ( "\ni = %d", i ) ;
increment( ) ;
increment( ) ;
decrement( ) ;
decrement( ) ;
getch();
}
increment( )
{
i = i + 1 ;
printf ( "\non incrementing i = %d", i ) ;
}
decrement( )
{
i = i - 1 ;
printf ( "\non decrementing i = %d", i ) ;
```

```
}
```

Output:

```
i = 0
on incrementing i = 1
on incrementing i = 2
on decrementing i = 1
on decrementing i = 0
```

Storage classes in C, at galance:

- A storage class is used to represent additional information about a variable.
- Storage class represents the scope and lifespan of a variable.
- It also tells who can access a variable and from where?
- Auto, extern, register, static are the four storage classes in 'C'.
- auto is used for a local variable defined within a block or function
- register is used to store the variable in CPU registers rather memory location for quick access.
- Static is used for both global and local variables. Each one has its use case within a C program.
- Extern is used for data sharing between C project files.

Try Yourself:

1. Difference between static and local storage class.
2. Write a short note on storage class in c.
3. Difference between static external variable and simple external variable.