Storage Class

Every variable are classified into different class based on their scope, accessibility, initial value and storage device etc. This is called storage class. It specifies following about a variable.

Default initial value : The value that is assign to a variable, if not initialized.

Storage location **:** The storage device, where the variable is created.

Scope : Define the visibility, where the variable can be referenced.

Lifetime/Extent **:** It is the duration for which for which memory is allocated to a variable

Types of storage class: →

Automatic
Register
Static
External

Automatic storage class

This is default storage class of a local variable. It creates the variable in its block and destroys it when control leaves the block permanently.

Key word : auto

Initial value : Garbage for local and 0 for glob

Storage : RAM

Scope : Local

Life : Till the control remains within the block in which the variable is defined.

Register storage class

Declaring a variable as register makes the computation faster. As the variable is created in CPU registers, the variable whose size is more than the size of CPU register cannot be created in register. So it is automatically converted to **automatic.**

Key word : register

Initial value : Garbage

Storage : CPU register

Scope : Local

Life : Till the control remains within the block in which the variable is defined.

Static storage class

It maintains the value of a local variable in till the program execution ends. This is default storage class for global variable.

Key word	:	static
Initial value	:	0
Storage	:	RAM
Scope	:	Local
Life	:	Value of the variable persists between different function calls.

External storage class

This storage class is used to give a reference of already defined global variable in another file.

Key word	:	extern
Initial value	:	0
Storage	:	RAM
Scope	:	Global
Life	:	value of variable is available till end of the program's execution.

first.c

```
#include<stdio.h>
#include"second.c"
int as;
int main()
        {
         as=9;
        printf("in storage %d",as);
        show();
        }
```

second.c

```
#include<stdio.h>
void show()
        {
        extern int as;
        printf("\n In second.c: %d",as);
        }
```

Important Notes:

Go through the examples/programs discussed in class.