Declaration of Storage Class

- Variables in C can have not only *data type* but also *storage class*
- It provides information about characteristics of variable

- Storage place of variable i.e CPU registers or memory
- Initial value of the variable ,if it is not specified in the program
- **Scope or visibility** of the variable i.e the portion of the program within which the variables are recognized.
- **Lifetime** of variable i.e how long the variable retains a particular value

- The storage class is another qualifier (like **long** or **unsigned)** that can be added to a variable declaration as shown below:

```
auto int count;
register char ch;
static int x;
extern long total;
```

Auto Storage Class

• Storage	Memory
• Default Initial Value	Garbage value i.e unpredictable value
• Scope or visibility	Local or visible the function to block in which the variable is declared
• Place of Declaration	function
• Life	Till control remains within the block in which variable is defined

- Automatic storage is a means of conserving memory, because automatic variables exist only when they are needed. They are created when the function in which they are defined is entered and they are destroyed when the function is exited.

```
 void main()
{
    auto int x=5;
    {
        auto int x=10;
        {
            auto int x=15;
            {
                printf(“\n%d”,x);
            }
    printf(“%d”,x);
}
printf(“%d”,x);
}
}
```

- Output

15 10 5

Static Storage Class

• Storage	• Memory
• Default Initial Value	• Zero
• Scope or Visibility	• Local to the block in which variable is declared.
• Place of Declaration	• function
• Life	• Value of variable persists between function calls

- But if the place of declaration of variable is outside the function then it is visible to all functions following its declaration
- The C compiler creates permanent storage for static variables (much like a global variable)
- Retain their value within their function or file.
- Known only to the code block in which they are declared.

Comparison

```
 void main()
{
    increment();
    increment();
    increment();
}
increment()
{
auto int i=1;
printf(“%d\n”,i);
i=i+1;
}
```

```
void main()
{
    increment();
    increment();
    increment();
}
increment()
{
static int i=1;
printf(“%d\n”,i);
i=i+1;
}
```

```
 int A(int);
void main()
{
     int i.r;
     for(i=1;i<=5;i++)
     {
          r=A(i);
          printf(“\t%d”,r);
     }
return;
}
A(int x)
{
static int s=0;
 s=s+x;
return(s);
}
```

Output

1,3,6,10,15

Extern Storage Class

- Storage
- Default Initial Value
- Scope or Visibility
- Place of Declaration
- Life

- Memory
- Zero
- Global
- Outside function
- Retains value throughout the program. As long as the program’s execution does not come to an end.

```
int i ;
void main()
{
  printf(“\n  i=%d”,i);
     increment();
     decrement();
}
increment()
{
     i++;
     printf(“\n i=%d”,i);
}
 decrement()
{
     i--;
     printf(“\n i=%d”,i);
}
```

Register Storage Class

• Storage	• CPU registers
• Default Initial Value	• Garbage value
• Scope or Visibility	• Local to the block in which it is declared
• Place of Declaration	• Inside function block
• Life	• Retains value as long as the control remains in that block in which variable is declared

```
void main()
{
        register int i;
        for(i=1;i<=100;i++)
        {
                printf(“%d”,i);
        }
}
```

Scope

- 3 types of Scope
- Local
- Function
- File/Global

Local

- Variables declared inside a block can be referenced inside that block only
- When control reaches the opening brace ,these variables come into existence and get destroyed as soon as control leaves the block

Global

- Variables, which can be accessed throughout the program.

```
int i;
Void main()
{
Statements;
}
```

i is global variable here.