Scope of a variables

Scope refers to the visibility of variables. In other words, which parts of your program can see or use it.
The range of code in a program over which a variable has a meaning is called as scope of the variable.

There are two types of scopes normally in C :
1. Local scope
2. File scope or Global scope

Local scope

A block of statement in C is surrounded by curly braces i.e. { and }. We can declare variables inside the block, such variables called as local, can be referenced only within the body of the block. When the control reaches the opening brace, these variables come into existence and get destroyed as soon as the control leaves the block through the closing brace. So the local variables are available to all the blocks enclosed by the block in which they have been declared.

```
e.g. while (flag)
{
   int x;
   if(i>x)
   {
      int j;
      ........
      ........
   }
}
```

The scope of x is the while block, and j is if block. The variable x can also be referenced to in the if block because this block is completely enclosed in the while block(parent if-block(child)). On the other hand, scope of j is only the if-block. Since a function is a block in itself, all the variables declared in it has the local scope.

Global Scope:

The global variables are declared outside all blocks and functions, they are available to all the blocks and functions. Thus the scope of a global variable is in the entire program file. This type of scope is known as file scope

e.g. global variables rate and qty of the following has file scope.

```
int qty, rate;
void main( )
{
   int gross, net;
   .......
   .......
   fun()
}
fun( )
```

```
{
   char c;
   int tot;
   tot=qty*rate;
  ..............
   ..............
 }
```

Scope rules

1. A local variable can be accessed only within the block in which it is declared. The local variable declared in a function exists only during the execution of the function. Therefore these variables are also known as automatic because they automatically created and destroyed.
2. The scope of formal parameters is local to the function in which it is defined.
3. Since the global variables' are not declared in a particular function, they are available to all the functions. Thus the scope of a global variable is in the entire program.

Storage Class

Def: A storage class defines the scope (visibility) and life-time of variables and/or functions within a C Program.

Each and every variable declared in C contains not only its data type but also has a storage class specified with it. If we do not specify storage class of a variable, the compiler will assume its storage class as default i.e. automatic.

Characteristics of storage classes

1. Storage place of the variable i.e. Memory or CPU registers.
2. The initial value of the variable has to be initialized, if it is not specified in the program.
3. The scope or the visibility of the variable.
4. The lifetime of the variable i.e. how long the variable exists.

Types of Storage class

1. Automatic storage class (with specifier auto)
2. Register storage class (with specifier register)
3. Static storage class (with specifier static)
4. External storage class (with specifier extern)

Automatic

S.No	Storage	Memory
1.	Default initial value	Garbage value i.e. an unpredictable value.
2.	Scope or visibility	Local or visible to the block in which variable is declared e.g. if the variable is declared in a function then it is only visible to that function
3.	Life time	It retains its value till the control remains in the block in which the variable is declared. e.g. if the variable is declared in a function, it will retain its value till the control is in that function.
4.	Place of store	memory

Demonstration of auto storage class with default value

```
void main( )
{
  auto int i, j = 5;
  int k, l= 10;
  printf ("\n value of i = %d \n value of j = %d", i, j);
  printf ("\n value of k = %d \n value of l = %d", k, l);
}
```

Output.

Value of i = 2009 // garbage value
Value of j = 5
Value of k = 1005
Value of l = 10

Register Class

Storage	CPU registers
Default initial value	Garbage value

Scope	Local to the block in which variable is declared (same as automatic)
Life time	It retains its value till the controsl remains in the block in which the variable is declared. (same as automatic)

The keyword register can be prefixed to an ordinary variable for declaring it as a register variable.
eg. register int i;

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

We use frequently loop counters as a register storage class, because speed of the program increases by using a variable as register storage class.

Static storage class

Storage	Memory
Default initial value	Zero
Scope or visibility	Local to the block in which the variable is declared (same as auto storage class)
Lifetime	It retains its value between different function calls i.e. when a function is called for the first time, static variable is created with initial value zero, and in subsequent calls it retains its present value.

If one desires that the local variable should retain its value even after the execution of the function in which it is declared,

then the variable must be declared as static. The value of the static variable is not destroyed when the function terminates. The value is retained and is available to the user even if the control returns back to the function again.

Demonstration of static storage class

```
#include <stdio.h>
#include <conio.h>
void main( )
{
   int i, r;
   clrscr( );
   for (i = 1; i <= 5; i ++)
   {
      r = tot (i);
      printf " (\ t %d", r);
   }
   getch( );
}
tot (int x)
{
    static int s = 0;
    s = s + x;
    return (s);
}
```

The function main ( ) calls the function tot ( ) five times, each time, sending the value of i varying from 1 to 5. In the function value is added to s and the accumulated result is returned. Once this program is executed, the following output is produced.

1 3 6 10 15

i.e. the contents of variables are retained in each function call.

```
#include <stdio.h>
#include <conio.h>
static int i = 100;
void abc( )
{
  static int i = 10;
  printf ("\n First = %d", i);
}
main( )
{
  static int i = 5;
  clrscr( );
  abc( );
  printf ("\t second = %d", i);
  getch( );
}
```

The output is

First = 10 second = 5

External Storage class

Storage	Memory
Default initial value	Zero
Scope or visibility	Global
Lifetime	It retains its value throughout the program. So the life time is as long as the program's execution does not come to an end.

External variables are declared outside all functions i.e. at the beginning of the program. When the size of the program becomes very large then it has to be divided into parts and

each part is stored in a separate function. Global variables should be available to all the functions with the help of extern specifier.

Eg:

```
extern int rate
void main()
{
 ...
 fun1()
 compute()
}
fun1 ()
{
 ....
 ....
}
compute ()
{
 int tot,net;
 net=tot*rate;
}
```

Difference between local and global variable

Local variable	Global variable
A variable which is defined inside a function is known as local variable.	A variable which is defined outside all the functions are called gobal variables.
Local variable are also called as internal variable.	Global variable are also called as external variables.
The scope of local variable is limited to the function in which they are declared.	The scope of Global variable is throughout in the program.
The memory is allocated to	Global variable remain in

local variables each time, the control enters the function and released when the control leaves the function.	**memory throughout the execution of the program.**
Local parameters can be accessed within the only function in which they are defined.	**Global parameters can be accessed in any of the functions in a program**
They can be used with auto-matic, static and register sto-rage class.	**They can be used with external storage class only.**