Programming and Problem Solving through C Language
O Level / A Level

Chapter - 7 : Storage Classes

Storage Class Types

Storage Specifier	Storage	Initial Value	Scope	Life Time
auto	Stack	Garbage	Within Block	End of Block
extern	Data Segment	Zero	Global, Multiple Files	Till end of program
static	Data Segment	Zero	Within Block	Till end of Program
register	CPU Register	Garbage	Within Block	End of Block

Register Variable

- These variables are stored in one of the machine's register instead of RAM and are declared using **register** keyword.

  eg. register int count;
- This means that the variable has a maximum size equal to the register size (usually one word) and cannot have the unary '&' operator applied to it (as it does not have a memory location).
- As register memory access are much faster than a RAM memory access, keeping frequently accessed variables in the register lead to faster execution of program.
- Register should only be used for variables that require quick access - such as counters.
- It should also be noted that defining 'register' does not mean that the variable will be stored in a register.
- Since only few variables can be placed in the register, it is important to carefully select the variables for this purpose.
- However, C will automatically convert register variables into non-register variables once the limit is reached.
- Don't try to declare a global variable as register. Because the register will be occupied during the lifetime of the program.

Static Variables

- The value of static variables persists until the end of the program.
- It is declared using the keyword **static** like

  static int x;

  static float y;
- It may be of external or internal type depending on the place of the declaration.

- Static variables are initialized only once, when the program is compiled.
- Static is the default storage class for global variables.
- The two variables below (count and road) both have a static storage class.

```
static int count;
int road;
void main( )
{
        printf(“%d”, road);
}
```

- Static variables can be ‘seen’ within all functions in this source file.
- At link time, the static variables defined here will not be seen by the object modules that are brought in.
- Static can also be defined within a function.
- If this is done, the variable is initialized at run time, but is not reinitialized, when the functio n is called.
- Inside a function, static variable retains its value during various calls.

Example

```
static int count;
void func( );
static count=5;  // Global variable - static is the default

void main( )
{
       while ( count -- )
       {
         func( );
       }
}

void func( )
{
      static int i=5;
      i++;
      printf(" i is %d and count is %d\n", i , count);
}
```

Output

```
i is 6 and count is 4
i is 7 and count is 3
i is 8 and count is 2
i is 9 and count is 1
i is 10 and count is 0
```

Internal Static Variable

- Are those which are declared inside a function?
- Scope of Internal static variables extends up to the end of the program in which they are defined.
- Internal static variables are almost same as auto variable except they remain in existence (alive) throughout the remainder of the program.
- Internal static variables can be used to retain values between function calls.

```
void main( )
{
  int i;
  for(i=1 ; i<=3 ; i++)
    func( )
}

void func( )
{
  static int x=0;
  x=x+1;
  printf(" x= %d \n",x);
}

Output
        x=1
        x=2
        x=3
```

External Static Variables

- An external static variable is declared outside of all functions and is available to all the functions in the program.
- An external static variable seems similar simple external variable but their difference is that static external variable is available only within the file where it is defined while simple external variable can be accessed by other files.

Static Function

- Static declaration can also be used to control the scope of a function.
- If one wants a particular function to be accessible only to the functions in the file in which it is defined and not to any function in other files, declare the function to be static.

eg. **static int power(int x , int y)**

```
        {


        }
```