PROGRAM STRUCTURE

STORAGE CLASS

Earlier we mentioned that variables are characterized by their data type like integer, floating point type, character type etc. Another characteristic of variables or arrays is done by storage class. It refers to the permanence and scope of variables or arrays within a program. There are 4 different storage class specification in C – automatic, external, static, and register. They are identified by the key words auto, external, static, and register respectively.

AUTOMATIC VARIABLES

They are declared in a function. It is local and its scope is restricted to that function. They are called so because such variables are created inside a function and destroyed automatically when the function is exited. Any variable declared in a function is interpreted as an automatic variable unless specified otherwise. So the keyword auto is not required at the beginning of each declaration.

EXTERNAL VARIABLE (GLOBAL VARIABLE)

The variables which are alive and active through out the entire program are called external variables. It is not centered to a single function alone, but its scope extends to any function having its reference. The value of a global variable can be accessed in any program which uses it. For moving values forth and back between the functions, the variables and arrays are declared globally i.e., before the main program. The keyword *external* is not necessary for such declaration, but they should be mentioned before the main program.

STATIC VARIABLES

It is, like automatic variable, local to functions is which it is defined. Unlike automatic variables static variable retains values throughout the life of the program, i.e. if a function is exited and then re-entered at a later time the static variables defined within the function will retain their former values. Thus this feature of static variables allows functions to retain information permanently through out the execution of the program. Static variable is declared by using the keyword static.

Example : static float a ;
Static int x ;

Consider the function program:

```
# include<stdio.h>
long int Fibonacci (int count )

main()
{
int i, m=20;

for (i =1 ; i < m ; ++i)
printf( "%ld\t",fibonacci(i));
}

long int Fibonacci (int count )
{
static long int f1=1, f2=1 ;
long int f ;
f = (count < 3 ) ? 1 : f1 + f2 ;
f2 = f1
f1= f ;
return (f ) ;}
```

In this program during the first entry to the function f1 and f2 are assigned 1, later they are replaced by successive values of f1 and f. as f1 and f2 are declared static storage class. When the function is exited the latest values stored in f1 and f2 will be retained and used when the function is re-entered.

. .