Paper 102: Programming & Problem solving through C

Lecture-12:Unit-II
Storage Classes

Storage classes

- Every variable in a program has specific attributes:
- **storage class**: partially determines duration, scope and linkage
- *storage duration*: how long a variable exists (automatic or static)
- *scope:* Area of reference in the program (local or global)
- *linkage*: important only in multiple-source-file programs

- **auto**
  - default type for variables declared within blocks (local) and argument lists
  - created when block (or function) is entered, destroyed when control exits from the block
  - scope: local; duration: automatic
- **register**
  - same as **auto**, except variable held in hardware register for speed, if possible
  - scope: local; duration: automatic
- **static**
  - local variables that retain value between function calls
  - visible to block in which declared
  - scope: local; duration: till end of program
  - if initialized, initial value is assigned only in the first time the control enters the block
- **extern**
  - default class for global variables (those defined outside a function) and function names
  - scope: global; duration: till end of program

Storage classes: Examples

Automatic storage class

```
#include<stdio.h>
void main()
{
        auto int a=2;
        {
                auto int a=5;
                printf("\n %d",a);
                a=a+10;
        }
        printf("\n %d",a);
}
```

Register storage class

```
#include<stdio.h>
void main()
{
        register a=3;
        for(a=1;a<=5;a++)
                printf("\n  %d",a);
}
```

Static storage class

```
#include<stdio.h>
void static_var(void);
void main()
{
        int a=2;
        printf("\n %d",a);
        static_var();
        static_var();
        printf("\n %d",a);
}
void static_var()
{
        static int a=3;
        a=a+10;
        printf("\n %d",a);
}
```

Extern storage class

```
#include<stdio.h>
int a=2;
void add(void);
void sub(void);
void main()
{
        int a=3;
        printf("\n %d",a);
        add();
        printf("\n %d",a);
        sub();
        printf("\n %d",a);
}
void add()
{
  a=a+10;
  printf("\n %d",a);
}
void sub()
{
        a=a-10;
        printf("\n %d",a);
}
```

If external var and local var names are same, then it is the local var that is recognized in that module

External storage class

- External variables has to be defined, and declared in order to access it.
  - *int a=2;* *definition*
- it can be access in any function by declaring with the keyword *extern*
  - *extern int a;* *declaration*
- If a function requires to access an external variable that has been declared earlier in the program, then the function may access it without declaration within the function.
- If the function definition precedes the external variable definition, then the function must include a declaration for that external variable.
  - The name and data type should agree with the external definition
  - It cannot initialize the variables

```
#include<stdio.h>
void add(void);
void sub(void);
void main()
{
        extern int a;
        printf("\n %d",a);
        add();
        printf("\n %d",a);
        sub();
        printf("\n %d",a);
}
void add()
{       extern int a;
        a=a+10;
         printf("\n %d",a);
}
int a=2;
void sub()
{
        a=a-10;
        printf("\n %d",a);
}
```

Storage class and functions

- C programs can consists of more than one file
- In Multifile program, a function definition may be either *external* or *static*
- An external function will be recognized throughout the entire program
- A static function will be recognized only within the file in which it is defined
- General syntax of function

  Storage-class return-data-type function-name(list of arguments)

  - Storage class is extern by default

    extern int add(int a, int b);