CPNM Lecture 15 - Storage Class and Scope

Mridul Sankar Barik

Jadavpur University

2022-23

Introduction

- **Storage Class**: Defines the scope (visibility), lifetime, area for storage and initial values of variables

auto Storage Class I

- **Storage**: Memory, **Scope**: Local or Block, **Life**: Exists as long as Control remains in the block, , **Initial Value**: garbage
- Default storage class for all local variables defined inside functions
- Destroyed automatically when the function exits
- Assigned default initial garbage value
- Example:

```
int Count;
auto int Month;
```

static Storage Class I

- **Storage**: Memory, **Scope**: Local, **Life**: Retain value across function calls, **Initial Value**: zero
- Default storage class for global variables
- A static variable tells the compiler to persist the variable until the end of program
- initialized only once and remains into existence till the end of program
- Scope of internal static variable remains inside the function in which it is defined
- External static variables remain restricted to scope of file in which they are declared
- Assigned 0 (zero) as default value by the compiler

static Storage Class II

- Example:

```
void test();    //Function declaration

main(){
    test();
    test();
    test();
}
void test(){
    static int a = 0;          //Static variable
    a = a+1;
    printf("%d\t",a);
}

output :
1 2 3
```

register Storage Class I

- **Storage**: Register, **Scope**: Local or Block, **Life**: Exists as long as Control remains in the block, **Initial Value**: garbage
- Defines local variables that should be stored in a register instead of memory
- Register variable has faster access than normal variable
- Frequently used variables are kept in register
- Only few variables can be placed inside register
- We can never get the address of such variables
- Example:

```
register int  Miles;
```

extern Storage Class I

- **Global variables**
  - declared outside all functions
  - available to the entire program
  - values can be changed in any function
- **Storage**: Memory, **Scope**: Global, **Life**: Exists as long as program runs, **Initial Value**: zero
- `extern` keyword is used to define a global variable that is visible to all object modules
- Example:

`source1.c`

```
--------

extern int count;

write(){
    printf("count is %d\n", count);
}
```

extern Storage Class II

```
source2.c
--------

int count=5;
main(){
    write();
}

Compilation command will look like

$gcc source1.c source2.c -o program
```

Storage Class for Functions

- All functions in C are external by default and are accessible to all source files
- If a function is declared to be of static, then it is accessible only to functions in the file in which they are defined, not to functions in other files