

MA 511: Computer Programming

Lecture 16: Macro & Storage Classes

http://www.iitg.ernet.in/psm/indexing_ma511/y10/index.html

Partha Sarathi Mandal

psm@iitg.ernet.ac.in

Dept. of Mathematics, IIT Guwahati

Semester 1, 2010-2011

Macro Definition

- We have already seen that `#define` statement can be used to define symbolic constants within a C program.

Ex: `#define SIZE 100`
`int Array[SIZE];`

- It can be used to define **macros**
 - Single identifiers that are equivalent to expressions, complete statements or groups of statements.
 - It looks like functions in that sense.
 - These are treated differently during the compilation process.

Example: Macro

```
#include <stdio.h>
```

```
#define area length*width
```

```
main(){  
    int length, width;  
    printf("Length = ");  
    scanf("%d", &length);  
    printf("width = ");  
    scanf("%d", &width);  
    printf("area = %d", area);  
}
```

Example: Macro

```
#include <stdio.h>
main(){
    int c, i, n;
    printf("number of lines : ");
    scanf("%d", &n);
    printf("\n");
    for(i=1; i <= n; i++){
        for(c=1; c <= n-i; c++)
            putchar(' ');
        for(c=1; c <= 2*i-1; c++)
            putchar('* ');
        printf("\n");
    }
}
```

```
#include <stdio.h>

#define loop(n) for(i=1; i<= n; i++){ \
                for(c=1; c<=n-i; c++) \
                    putchar(' '); \
                for(c=1; c<=2*i-1; c++) \
                    putchar('* '); \
                printf("\n"); \
            }

main(){
    int c, i, n;
    printf("number of lines : ");
    scanf("%d", &n);
    printf("\n");

    loop(n)
}
```

```
n = 6
      *
     ***
    *****
   *********
  ***********
 *****
*****
```

C Storage Classes

- C has a concept of '*Storage classes*' which are used to define the scope (visibility) and life time of variables and/or functions.
 - **auto** is the default storage class for local variables.
 - **static** is the default storage class for **global variables**
 - **extern** defines a global variable that is visible to ALL object modules.
 - When you use 'extern' the variable cannot be initialized as all it does is point the variable name at a storage location that has been previously defined.

extern Storage Class

f1.c

```
void write_extern(void);  
extern int count;  
void write_extern(void) {  
    printf("f1 count is %i\n", count++);  
}
```

f2.c

```
int count=5;  
main() { write_extern();  
    printf("f2 count is %i\n", count++);  
}
```

```
cc f1.c f2.c -o file  
./file
```

```
f1 count is 5  
f2 count is 6
```

global variable static storage class

```
#include<stdio.h>
void func(void);
static count1=10; /*Global variable -static is the default*/
main() {
    while (count1-->0)
        func();
}

void func(void) {
    /* 'count2' is local to 'func' - it is only initialized at run time. */
    /* Its value is NOT reset on every invocation of 'func' */
    static count2=5;
    count2++;
    printf(" count2 is %d count1 is %d\n", count2, count1);
}
```

Output:

```
count2 is 6 count1 is 9
count2 is 7 count1 is 8
count2 is 8 count1 is 7
count2 is 9 count1 is 6
count2 is 10 count1 is 5
count2 is 11 count1 is 4
count2 is 12 count1 is 3
count2 is 13 count1 is 2
count2 is 14 count1 is 1
count2 is 15 count1 is 0
```

Summary of **extern** and **static**

Objective	How Achieved
To access variable x external to all functions and defined in file i from file j	Declare x as extern in file j
To make a variable x external to all functions and defined in file i not accessible to any other file	Declare x as static in file i
To use a function f(x) defined in file i in file j	No spl declaration needed
To make a function float f(float x) defined in file i inaccessible to all other files	Declare f(x) as: static float f(float x);