

```

int temp;
temp = count;
func2();
printf("count is %
d", count); /* will print 100 */
}
void func2(void)
{
int count;
for(count=1; count<10; count++)
putchar(' ');
}

```

If a global variable and a local variable have the same name, all references to that variable name inside the code block in which the local variable is declared will refer to that local variable and have no effect on the global variable.

2.3 Storage Class Specifiers

C supports four storage class specifiers:

1. extern
2. static
3. register
4. auto

These specifiers tell the compiler how to store the subsequent variable. The general form of a variable declaration that uses one is shown here:

storage_specifier type var_name;

2.3.1 extern

The principal use of **extern** is to specify that an object is declared with external linkage elsewhere in the program. To understand why this is important, it is necessary to understand the difference between a declaration and a definition. A *declaration* declares the name and type of an object. A *definition* causes storage to be allocated for the object. The same object may have many declarations, but there can be *only one* definition. By preceding a variable name with the **extern** specifier, you can declare a variable without defining it.

Here is an example that uses **extern**. Notice that the global variables **first** and **last** are declared *after* **main()**.

```

#include <stdio.h>
int main(void)
{
    extern int first, last; /* use global vars */
    printf("%d %d", first, last);
    return 0;
}
/* global definition of first and last */
int first = 10, last = 20;

```

This program outputs **10 20** because the global variables **first** and **last** used by the **printf()** statement are initialized to these values. Because the **extern** declaration tells the compiler that **first** and **last** are declared elsewhere (in this case, later in the same file), the program can be compiled without error even though **first** and **last** are used prior to their definition. As mentioned, **extern** allows you to declare a variable without defining it. However, if you give that variable an initialization, the **extern** declaration becomes a definition.

One more example: