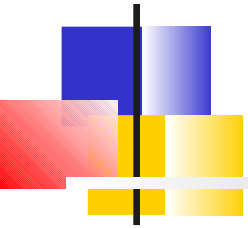
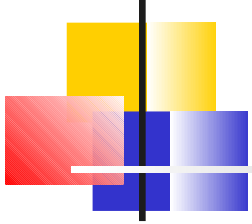


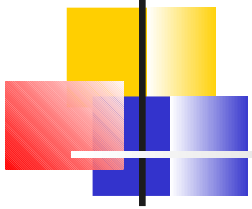
Program Organization

Chapter 10



Objectives

- You will be able to
  - Say what the **scope** of a variable is based on how it is declared.
  - Say what the **lifetime** of a variable is based on how it is declared.



Properties of Variables

- Scope
  - Where it is visible
- Duration (or Lifetime)
  - When it comes into existence
  - When it ceases to exist

Local Variables

Local variables are variables defined inside a function definition

- By default, local variables have
  - ***Automatic* storage duration**
    - Memory is allocated automatically when the function is called.
    - Memory is deallocated when the function returns.
  - **Block scope**
    - Variable is visible in the block in which it is declared.

Block Scope

- In C89 local variables must be declared at the beginning of a block
  - Before any executable statements.
  - Visible from declaration to end of block.

- C99 permits variables to be declared anywhere within a block
  - Visible from declaration to end of block
  - In Unix, gcc permits this even with C89.
  - **You are allowed to use this part of C99**

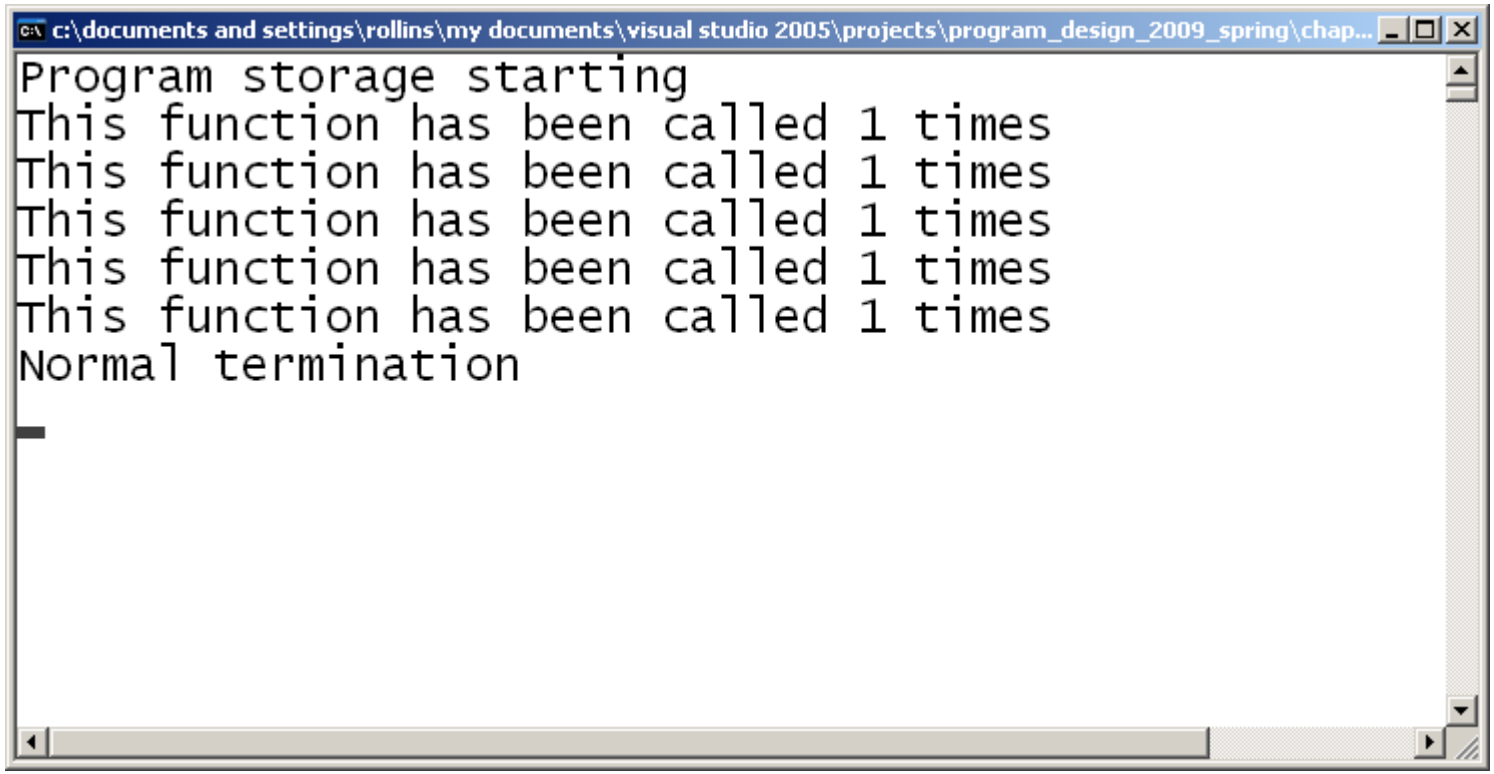
Static Local Variables

- The keyword ***static*** at the start of a local variable declaration makes its duration permanent.
  - Occupies the same memory location throughout the execution of the program.
    - Allocated in the *heap* rather than the runtime stack.
    - *Not* a separate copy for each invocation.
- Visible only within the block where it is defined.
- Variable initialization only done the first time the function is called

storage.c

```
#include <stdio.h>
void fn()
{
    int counter = 0;
    counter++;
    printf ("This function has been called %d times\n", counter);
}

int main()
{
    int i = 0;
    printf ("Program static_storage starting\n");
    for (i = 0; i < 5; i++)
    {
        fn();
    }
    printf ("Normal termination\n");
    getchar();
    getchar();
}
```

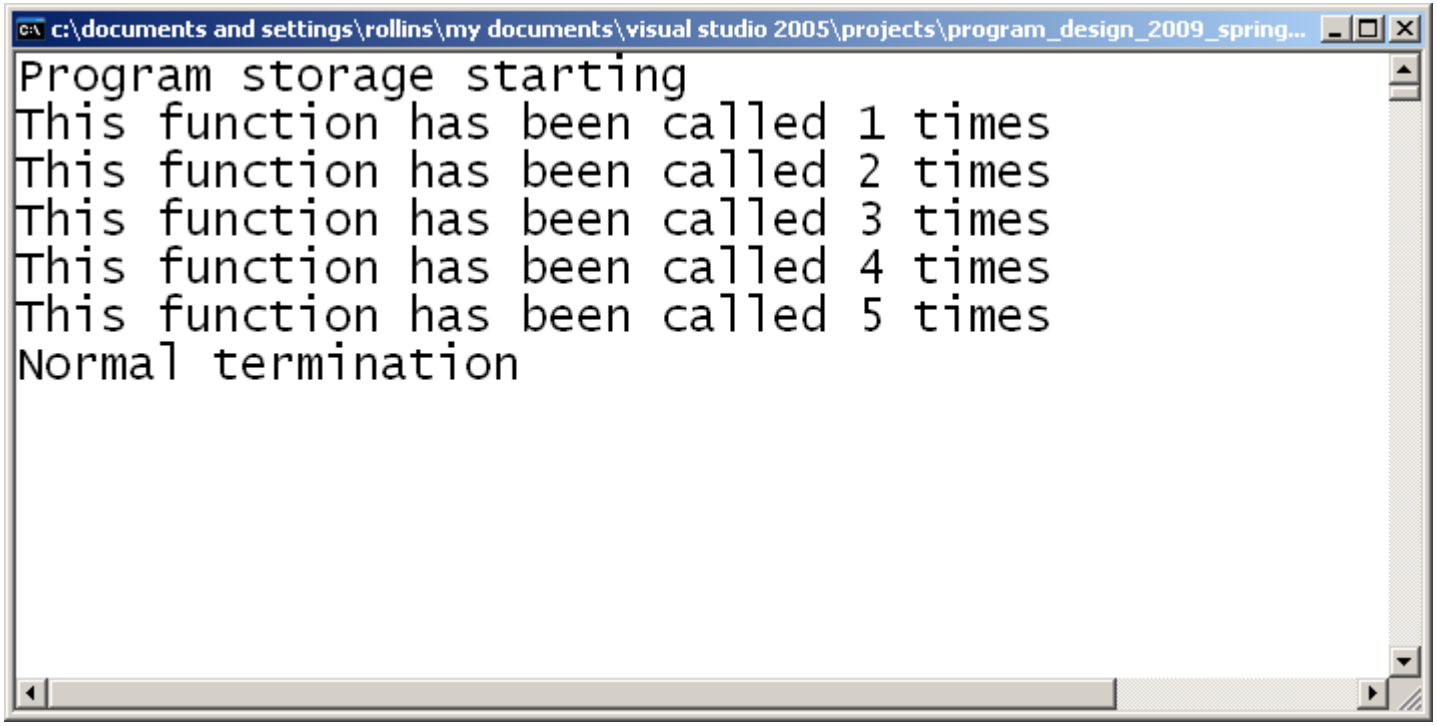
Program Running



A Static Local Variable

```
#include <stdio.h>
void fn()
{
    static int counter = 0;
    counter++;
    printf ("This function has been called %d times\n", counter);
}
int main()
{
    int i = 0;
    printf ("Program storage starting\n");
    for (i = 0; i < 5; i++)
    {
        fn();
    }
    printf ("Normal termination\n");
    getchar();
    getchar();
}
```

Program Running

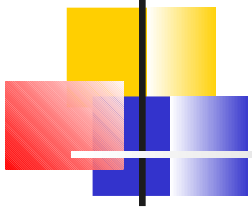


variable

```
#include <stdio.h>

void fn()
{
    static int counter = 0;
    counter++;
    printf ("This function has been called %d times\n", counter);
}

int main()
{
    int i = 0;
    printf ("Program storage starting\n");
    for (i = 0; i < 5; i++)
    {
        fn();
    }

    printf ("The function was called %d times\n", counter);
    printf ("Normal termination\n");
    getchar();
    getchar();
}
```

Incorrect!

This gets a compile error.



External Variables

- A variable declared outside a function declaration is an ***external*** or ***global*** variable.
- Duration is the entire execution of the program.
  - Like static local variables.
- Scope is from declaration to end of file.
  - Also visible to functions in other files.
  - Potential input and output for ***every function***.

A Global Variable

```
#include <stdio.h>

int counter = 0;

void fn()
{
    counter++;
    printf ("This function has been called %d times\n", counter);
}

int main()
{
    int i = 0;
    printf ("Program storage starting\n");
    for (i = 0; i < 5; i++)
    {
        fn();
    }
    printf ("The function was called %d times\n", counter);
    printf ("Normal termination\n");
    getchar();
    getchar();
}
```

Program Running