GE3 COMPUTER SCIENCE

C AND C ++ LECTURE SERIES *FOR*

B.SC 3RD SEMESTER *BY*

SUBHADIP MUKHERJEE

DEPARTMENT OF COMPUTER SCIENCE

KHARAGPUR COLLEGE

LECTURE 9

FUNCTIONS

PASSING ARGUMENTS TO A FUNCTION

- ***The value of the corresponding formal argument can be altered within the function, but the value of the actual argument within the calling routine will not change***

```
#include <stdio.h>

void modify(int a);       /* function prototype */

main()
{
    int a = 2;

    printf("\na = %d  (from main, before calling the function)", a);
    modify(a);
    printf("\n\na = %d  (from main, after calling the function)", a);
}

void modify(int a)
{
    a *= 3;
    printf("\n\na = %d  (from the function, after being modified)", a);
    return;
}
```

FUNCTIONS

PASSING ARGUMENTS TO A FUNCTION

- *passing **by** value is restricted to a one-way transfer **of** information.*

FUNCTIONS

RECURSION

- ***Recursion*** is a process by which a function calls itself repeatedly, until some specified condition has been satisfied.

```
#include <stdio.h>

long int factorial(int n);          /* function prototype */

main()

{
     int n;
     long int factorial(int n);

     /* read in the integer quantity */

     printf("n = ");
     scanf("%d", &n);

     /* calculate and display the factorial */

     printf("n! = %ld\n",  factorial(n));
}
```

```
long int factorial(int n)          /* calculate the factorial */

{
     if (n <= 1)
        return(1);
     else
        return(n * factorial(n - 1));
}
```

FUNCTIONS

RECURSION

$$n! = n \times (n-1)!$$

$$(n-1)! = (n-1) \times (n-2)!$$

$$(n-2)! = (n-2) \times (n-3)!$$

. . . . . .

$$2! = 2 \times 1!$$

$$1! = 1$$

$$2! = 2 \times 1! = 2 \times 1 = 2$$

$$3! = 3 \times 2! = 3 \times 2 = 6$$

$$4! = 4 \times 3! = 4 \times 6 = 24$$

. . . . . .

$$n! = n \times (n-1)! = \cdots$$

FUNCTIONS

STORAGE CLASSES

There are four different storage-class specifications in C: ***automatic, external, static*** and ***register***

```
auto int a, b, c;
extern float root1, root2;
static int count = 0;
extern char star;
```

FUNCTIONS

STORAGE CLASSES

- ***Automatic variables*** are always declared within a function and are local to the function in which they are declared;
- ***External variables,*** in contrast to automatic variables, are not confined to single functions. Their scope extends from the point of definition through the remainder of the program.
- Static variables are defined within a function in the same manner as automatic variables, except that the variable declaration must begin with the **static** storage-class designation.

FUNCTIONS

Fibonacci Series

```
#include <stdio.h>

long int fibonacci(int count);

main()

{
    int count, n;

    printf('How many Fibonacci numbers? ");
    scanf("%d", &n);
    printf('\n');

    for (count = 1; count <= n; ++count)
        printf('\ni = %2d    F = %ld", count, fibonacci(count));
}
```

```
long int fibonacci(int count)

/* calculate a Fibonacci number using the formulas

   F = 1 for i < 3, and F = F1 + F2 for i >= 3 */

{
    static long int f1 = 1, f2 = 1;
    long int f;

    f = (count < 3) ? 1 : f1 + f2;
    f2 = f1;
    f1 = f;
    return(f);
}
```

FUNCTIONS

Fibonacci Series

```
i =  1      F = 1
i =  2      F = 1
i =  3      F = 2
i =  4      F = 3
i =  5      F = 5
i =  6      F = 8
i =  7      F = 13
i =  8      F = 21
i =  9      F = 34
i = 10      F = 55
i = 11      F = 89
i = 12      F = 144
i = 13      F = 233
```

```
i = 14      F = 377
i = 15      F = 610
i = 16      F = 987
i = 17      F = 1597
i = 18      F = 2584
i = 19      F = 4181
i = 20      F = 6765
i = 21      F = 10946
i = 22      F = 17711
i = 23      F = 28657
i = 24      F = 46368
i = 25      F = 75025
i = 26      F = 121393
i = 27      F = 196418
i = 28      F = 317811
i = 29      F = 514229
i = 30      F = 832040
```

COMPILE AND RUN A C CODE

Thank You

End of Lecture 9

Subhadip Mukherjee

Department of Computer Science

Kharagpur College

Kharagpur, India