Introduction to Computing

MCS1101B

Lecture 5

By
Soumadip Biswas
Associate Professor, IEM

IEM
श्रद्धावान लभते ज्ञानम्

Recap

- Control Statements
  - Branching
  - Looping

- Branching
  - if
  - if else
  - if else if else if …
  - ? :
  - Nested if else
  - switch

- Looping
  - while
  - for
  - do while
  - break, continue

Nested Loops: Printing a 2-D Figure

- How would you print the following diagram?

```
* * * * * *
* * * * * *
* * * * * *
* * * * * *
```

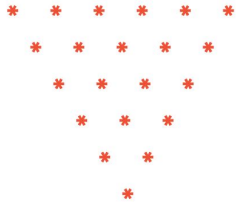
- Nested Loops
  - break and continue with nested loops

Half Pyramid

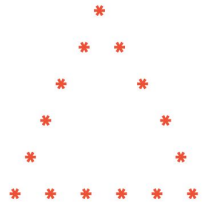
Inverted
Half Pyramid

Hollow Inverted
Half Pyramid

Full Pyramid



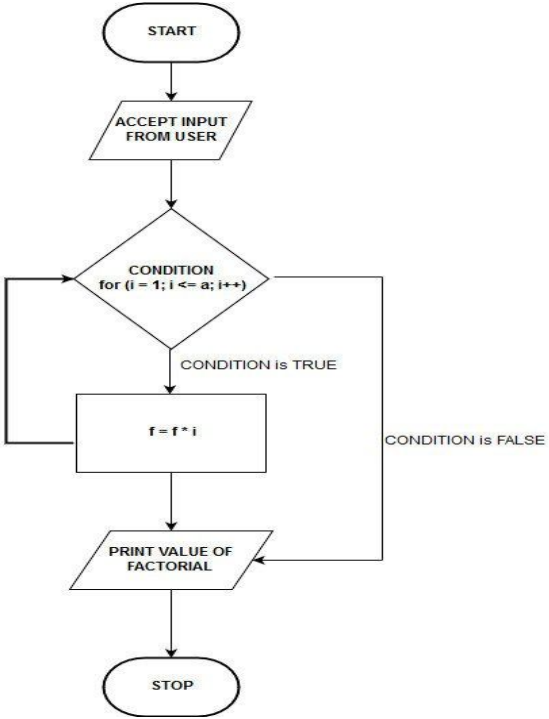
Inverted Full Pyramid



Hollow Full Pyramid

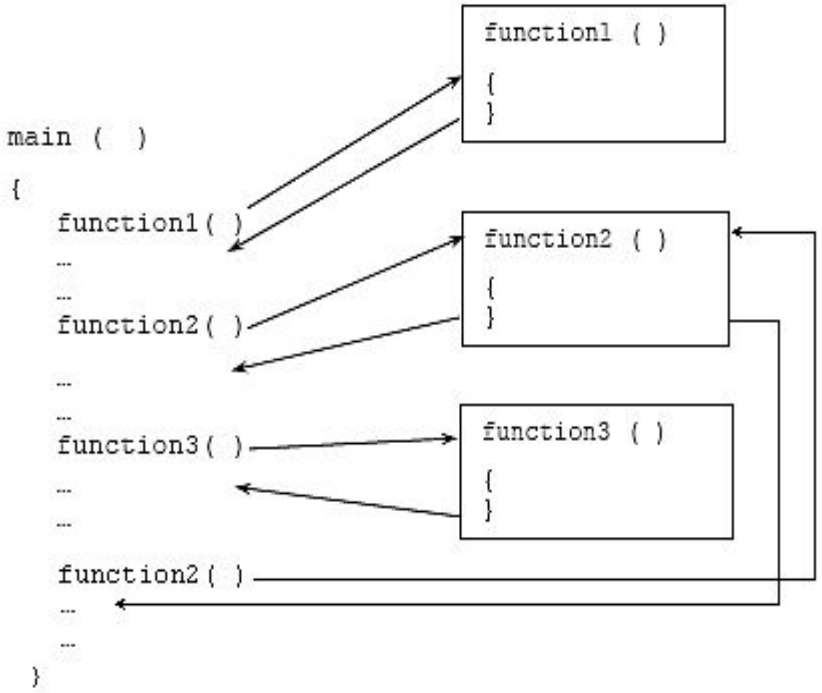
Sequence of Execution

- The flow of a program i.e. the steps and branches can be represented in graphically
- Represented using Flow chart
  - Example: a for loop ⇒
  - *Let's understand this on the board*



Functions

- A program segment that carries out some specific, well-defined task
- Example
  - A function to add two numbers
  - A function to find the largest of n numbers
- A function will carry out its intended task whenever it is **called** or **invoked**
  - Can be **called** multiple times



Functions (contd.)

- Examples
  - Print a banner
  - Factorial computation
  - Gcd computation

- A function definition has two parts:
  - The first line, called header
  - The body of the function
  - May or may not have a return value

```
return-value-type   function-name   ( parameter-list )
{
        declarations and statements
}
```

Function Prototypes

- Compiler needs to know some details of a function(see list below) before it is being used (called) in a program
  - Name of the function
  - Return type of the function
  - The sequence of the parameters-types (parameter names are optional) of that function
  - The definition/body of the function is optional
- The collection of these minimum requirements is known as *function prototype*

Example

- Function prototype
- Start of function body
- Local variables
- A while loop
- Start of the loop block
- Statements
- End of loop block
- Return statement
- End of function body

```
int gcd (int A, int B)
{
        int temp;
        while ((B % A) != 0)
        {
                temp = B % A;
                B = A;
                A = temp;
        }
return (A);
}
```

Functions (contd.)

Passing of variables

- Variables values are copied when then are passed (by calling) to a function
- The actual variables are not passed
- So a change made to a variable within a function will not reflect in the variable at the end of the caller

The return statement

- Return statement is optional
- Return type in the function prototype must be present
- Return statement causes the sequence of execution to return to the caller

Scope of Variables

- Part of the program from which the value of the variable can be used (seen)
- Scope of a variable - Within the block in which the variable is defined
  - Block = group of statements enclosed within { }
- Local variable – scope is usually the function in which it is defined
  - So two local variables of two functions can have the same name, but they are different variables
- Global variables – declared outside all functions (even main)
  - scope is entire program by default, but can be hidden in a block if local variable of same name defined

In The Next Class…

- You will learn about array and pointers
- You will learn more about functions