Top Down Design

- Large problems can be divided into smaller sub - problems
  - each sub - problem can be solved separately in order to reach to the solution of the original problem
- Large problems that are to be solved by computer
  - divided into smaller sub - problems
  - each sub - problem is called a task that need to be solved
  - **a subprogram** is written to solve each task
  - **typical** FORTRAN program consists of
    - one **main** program
    - several **subprograms**

Main Program and Subprograms

- **One main** program
- **Zero or more** subprograms
- The subprograms may appear **before** or **after** the main program
- **Execution** of the program **starts** from the **main** program
- A subprogram may be **called** from the **main** program or from another **subprogram**
- The calling (main program or subprogram) **passes information** to the called subprogram through arguments
- The called subprogram starts executing . When completes, it will **return** value(s) to the calling (main program or subprogram)
- **Two** types of **subprograms**
  - FUNCTION SUBPROGRAMS
  - SUBROUTINE SUBPROGRAMS

FUNCTION SUBPROGRAMS

Function Header

type FUNCTION fname (a list of arguments)

Where

- type is the type for the function name (REAL, INTEGER - - -) ;
- fname is the name of the function; and
- a list of arguments is the optional list of dummy arguments .

Function Body

The function body is similar to a FORTRAN program

Function Structure

```
TYPE FUNCTION FNAME (A LIST OF DUMMY ARGUMENTS)

DECLARATION OF DUMMY ARGUMENTS AND VARIABLES TO BE USED IN THE FUNCTION

EXECUTABLE STATEMENTS
- - -
- - -
FNAME = EXPRESSION
- - -
- - -
RETURN
END
```

Examples on Function Subprograms:

Example 1: Write a real function VOLUME that computes the volume of a sphere ($4/3\pi r^3$) given its radius.

Solution:

```
C      FUNCTION  SUBPROGRAM
       REAL  FUNCTION  VOLUME (RADIUS)
       REAL  RADIUS, PI
       PI = 3.14159
       VOLUME = 4.0 / 3.0 * PI * RADIUS ** 3
       RETURN
       END

C      MAIN  PROGRAM
       REAL  RADIUS, VOLUME
       PRINT*, 'ENTER A RADIUS'
       READ*, RADIUS
       PRINT*, ' THE  VOLUME  OF THE SPHERE = ' , VOLUME ( RADIUS )
       END
```

Examples on Function Subprograms:

Example 2: Write a logical function ORDER that checks whether three different integer numbers are ordered in increasing or decreasing order.

Solution:

```
C      FUNCTION  SUBPROGRAM
       LOGICAL  FUNCTION  ORDER(X, Y, Z)
       INTEGER  X, Y, Z
       LOGICAL  INC, DEC
       DEC = X .GT. Y .AND. Y .GT. Z
       INC  = X .LT. Y .AND. Y .LT. Z
       ORDER = INC .OR. DEC
       RETURN
       END

C      MAIN  PROGRAM
       LOGICAL  ORDER
       INTEGER  X, Y, Z
       PRINT*, 'ENTER THREE DIFFERENT INTEGER NUMBERS'
       READ*, X, Y,  Z
       IF ( ORDER( X, Y, Z ) ) THEN
          PRINT*, 'THE NUMBERS ARE ORDERED'
       ELSE
          PRINT*, 'THE NUMBERS ARE  NOT ORDERED'
       ENDIF
       END
```

Examples on Function Subprograms:

Example 3: Write a function subprogram to evaluate the function f(x) defined below.

$f(x) = 2x^2 + 4x + 2$ if $x < 5$

$f(x) = 0$ if $x = 5$

$f(x) = 3x + 1$ if $x > 5$

Solution:

```
C     FUNCTION  SUBPROGRAM
      REAL  FUNCTION  F(X)
      REAL   X
      IF (X .LT. 5) THEN
         F = 2 * X ** 2 + 4 * X + 2
      ELSEIF (X .EQ. 5) THEN
         F = 0
      ELSE
         F = 3 * X + 1
      ENDIF
      RETURN
      END

C     MAIN  PROGRAM
      REAL  X , F
      READ*, X
      PRINT*, 'F(X) = ', F( X )
      END
```

Function Rules

The following rules must be observed in writing programs with function subprograms:

- **Actual** and **dummy** arguments must **match** in **type**, **order** and **number**. The names of these arguments may or may not be the same.
- **Actual** arguments may be **expressions**, **constants** or **variable** names. **Dummy** arguments must be **variable** names and should never be expressions or constants.
- The type of the function name must be the same in both the calling program and the function description.
- The **result** from the function subprogram, to be returned to the calling program, should be stored in the **function name.**
- A return statement transfers control back to the calling program. Every function should have at **least one** return statement.
- The function may be placed either before or after the main program.
- A **function** is **called** or invoked as part of an **expression**.
- A FORTRAN function cannot call itself.

Special Cases of Functions

- Intrinsic (built-in) Functions

Function	Function Value	Comment
SQRT(X)	Square Root of X	X is a real argument
ABS(X)	Absolute Value of X	
SIN(X)	Sine of angle X	Angle is in radians
COS(X)	Cosine of angle X	Angle is in radians
TAN(X)	Tangent of angle X	Angle is in radians
EXP(X)	e raised to the power X	
LOG(X)	Natural Logarithm of X	X is real
LOG10(X)	Logarithm of X to base 10	X is real
INT(X)	Integer value of X	Converts a real to an integer
REAL(K)	Real value of K	Converts an integer to real
MOD(M, N)	Remainder of M/N	Modulo function

Statement Functions

fname (a list of arguments) = expression

Where

- fname is the name of the statement function;
- a list of arguments is the optional list of dummy arguments ; and
- expression computes the function value.

Examples of statement functions:

Example 1: Write a statement function to compute the area of a triangle, given its two sides and an angle.

```
REAL  SIDE1, SIDE2, ANGLE, AREA
AREA( SIDE1, SIDE2, ANGLE ) = 0.5 * SIDE1 * SIDE2 * SIN (ANGLE)
READ*, SIDE1, SIDE2, ANGLE
PRINT*, ' THE AREA  OF THE TRIANGLE = ' , AREA( SIDE1, SIDE2, ANGLE )
END
```

Example 2: Write a statement function to compute the total number of seconds, given the time in hours, minutes and seconds.

```
REAL  HOUR, MINUTE, SECOND, TOTSEC
TOTSEC ( HOUR, MINUTE, SECOND ) = 3600 * HOUR +60 * MINUTE + SECOND
READ*,   HOUR, MINUTE, SECOND
PRINT*, ' THE TOTAL NUMBER OF SECONDS = ' , TOTSEC ( HOUR, MINUTE, SECOND )
END
```

Complete Example on Function Subprograms

Example : The sum of three integer numbers: Write an integer function ISUM to sum three integer numbers. Also write a main program to test the function ISUM.

Solution:

```
C      MAIN   PROGRAM
       INTEGER  X, Y, Z, ISUM
       READ*, X, Y, Z
       PRINT*, ' SUM OF THE NUMBERS = ' , ISUM (X, Y, Z)
       END

C      FUNCTION   SUBPROGRAM
       INTEGER  FUNCTION  ISUM(A, B, C)
       INTEGER A, B, C
       ISUM  = A + B + C
       RETURN
       END
```

Exercises

What is the output of the following program?

```
INTEGER A, B, X, Y, Z, F
A = 2
B = 3
X = F(4, A)
Y = B * 3
Z = F(Y, X)
PRINT*, X, Y, B, Z
END
INTEGER FUNCTION F(X,Y)
INTEGER X, Y, Z
Z = 2*Y
F = X+Z
RETURN
END
```

The output of the above program is

```
8      9      3      25
```

What is the output of the following program?

```
INTEGER  FUNCTION  FUN(J, K, M)
REAL  SUM
SUM = J + K + M
FUN = SUM /3.0
RETURN
END
INTEGER  FUN, FUS, J, K
FUS(J, K) = J * K / 2
PRINT*, FUS(FUN(2, 3, 4), FUN(5, 6, 7))
PRINT*, FUN(FUS(2, 3), FUS(4, 5), FUS(6, 7))
END
```

The output of the above program is

```
9
11
```

Structure & Rules of the Subroutines

Subroutine is a subprogram that has the following Header :

SUBROUTINE SNAME (a list of dummy arguments)

where

- SNAME is the name of the subroutine; and
- a list of dummy arguments is optional.

a subroutine is called or invoked by an executable statement,
the **CALL** statement. The general form of the statement is as follows:

CALL SNAME (a list of actual arguments)

- The subroutine actual and dummy arguments must match in type, number and order.
- At the invocation of a subroutine , values of the actual arguments are copied in the dummy arguments.
- At the return of a subroutine , values of the dummy arguments are copied back in the actual arguments.
- At least one RETURN statement must be present to ensure transfer of control from the subroutine to the calling program (or subprogram)
- The subroutine does not return a value in its name.

Examples on Subroutine Subprograms:

Example 1: Write a subroutine that exchanges the value of its two real arguments.

Solution:

```
C       SUBROUTINE SUBPROGRAM
        SUBROUTINE  EXCHNG (NUM1, NUM2)
        REAL NUM1, NUM2, TEMP
        TEMP = NUM1
        NUM1 = NUM2
        NUM2 = TEMP
        RETURN
        END

C       MAIN PROGRAM
        REAL  NUM1, NUM2
        PRINT*, 'ENTER TWO REAL NUMBERS'
        READ*, NUM1, NUM2
        PRINT*, 'INPUT: ' , NUM1, NUM2
        CALL  EXCHNG (NUM1, NUM2)
        PRINT*, 'NUMBER1 = ' , NUM1
        PRINT*, 'NUMBER2 = ', NUM2
        END
```

Examples on Subroutine Subprograms:

Example 2: Write a subroutine that takes three different integer arguments X, Y and Z and returns the maximum and the minimum.

Solution:

```
C       SUBROUTINE SUBPROGRAM
        SUBROUTINE  MINMAX (X, Y, Z, MAX, MIN)
        INTEGER   X, Y, Z, MAX, MIN
        MIN   = X
        MAX = X
        IF (Y .GT. MAX)    MAX = Y
        IF (Y .LT. MIN)    MIN = Y
        IF (Z .GT. MAX)    MAX = Z
        IF (Z .LT. MIN)    MIN = Z
        RETURN
        END

C       MAIN PROGRAM
        INTEGER   X, Y, Z, MAX, MIN
        PRINT*, 'ENTER THREE DIFFERENT INTEGER NUMBERS'
        READ*, X , Y , Z
        CALL   MINMAX (X, Y, Z, MAX, MIN)
        PRINT*, 'THE MAXIMUM NUMBER = ' , MAX
        PRINT*, 'THE MINIMUM NUMBER  = ' , MIN
        END
```

Examples on Subroutine Subprograms:

Example 3: Sum and Average: Write a subroutine to sum three integers and compute their average. The subroutine should return the sum and average of the three numbers. Write a main program to test the subroutine.

Solution:

```
C     MAIN  PROGRAM
      INTEGER  X, Y, Z, TOTAL
      REAL  AVG
      PRINT*, 'ENTER THREE INTEGER NUMBERS'
      READ*, X, Y, Z
      CALL  SUBSUM  (X, Y, Z, TOTAL, AVG)
      PRINT*, 'TOTAL IS ', TOTAL
      PRINT*, 'AVERAGE IS ' , AVG
      END

C     SUBROUTINE  SUBPROGRAM
      SUBROUTINE  SUBSUM (A, B, C, TOTAL, AVG)
      INTEGER  A, B, C, TOTAL
      REAL  AVG
      TOTAL = A + B + C
      AVG = TOTAL / 3.0
      RETURN
      END
```

Exercises

What is the output of the following program?

```
INTEGER  A, B
LOGICAL  FLAG
READ*, A, B
FLAG =  A .GT. B
CALL  SUB (A, B)
PRINT*, A, B, FLAG
END
SUBROUTINE  SUB (A, B)
INTEGER  A, B, T
LOGICAL  FLAG
T = A
A = B
B = T
FLAG = A .GT. B
RETURN
END
```

Assume the input is

```
 6        3
```

The output of the above program is

```
3        6        T
```

What is the output of the following program?

```
SUBROUTINE  CHANGE (W, X, Y, Z)
INTEGER  W, X, Y, Z
W = X
X = Y
Y = Z
Z = W
RETURN
END
INTEGER  A , B
READ*, A, B
CALL  CHANGE (A * 2, B * 3,  A,  B)
PRINT*, A * 2, B * 3
END
```

The output of the above program is

```
8          36
```

Assume the input is

```
3          4
```

What is the output of the following program?

```
REAL  X ,Y
X = 3.0
Y = 1.0
CALL  F (X ,Y)
PRINT*, X ,Y
END
SUBROUTINE  F (A ,B)
REAL  A , B
CALL  G (B ,A)
B = A + B
A = A - B
RETURN
END
SUBROUTINE  G (C ,D)
REAL  C , D
C = C + D
D = C - D
RETURN
END
```

The output of the above program is

```
- 4.0          5.0
```