International Islamic University Chittagong

Department of Computer Science & Engineering

Autumn - 2022

Course Code: CSE-2321
Course Title: Data Structures

Mohammed Shamsul Alam

Professor, Dept. of CSE, IIUC

Lecture – 2

Preliminaries

Mathematical Notation and Functions

Floor and Ceiling Functions

- ❑ ∟x┘, called the *floor* of x, denotes the greatest integer that does not exceed x.
- ❑ ⟦x˥, called the *ceiling* of x, denotes the least integer that is not less than x.

If x is itself an integer, then └ x ┘ = ┌ x ┐; otherwise, ∟ x ┘+ 1 = ⟦ x ˥.

∟3.14┘= 3; ⟦3.14˥ = 4; ∟-8.5┘= -9; ⟦-8.5˥ = -8; ∟7┘= 7; ⟦x˥ = 7.

Mathematical Notation and Functions

Remainder Function

- Let **k** be any integer and let **M** be a positive integer. Then ***k (mod M)*** will denote the integer remainder when **k** is divided by **M**.
- **k (mod M)** is the unique integer **r** such that **k = M q + r** where **0 ≤ r < M**.
- 25 (mod 7) = 4   25 (mod 5) = 0   3 (mod 8) = 3

Congruence

- ***a ≡ b (mod M)*** if and only if ***M*** *divides* ***a – b***

  22 ≡ 2 (mod 5)

Mathematical Notation and Functions

Integer and Absolute Value Function

Let **x** be any real number. The integer value of **x**, written ***INT(x)***, converts **x** into an integer **y** deleting (truncating) the fractional part of the number.

INT (3.14) = 3   INT (-8.5) = -8   INT (7) = 7

The ***absolute value*** of the real number **x**, written **ABS(x)** or **|x|**, is defined as the greater of **x** and **–x**.

| -15 | = 15 | 3 | = 3 | -8.5 | = 8.5 | 8.5 | = 8.5

| 0 | = 0

Mathematical Notation and Functions

Factorial Function

- n! = 1 . 2 . 3 …. (n-2) . (n-1) . n
- 4! = 1.2.3.4 = 24   0! = 1

Fibonacci Function

- The Fibonacci sequence is as follows: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, … …

  Here, $F_0 = 0$ and $F_1 = 1$ and each succeeding term is the sum of two preceding terms.

Mathematical Notation and Functions

Permutation and Combination

- ${}^{n}P_{r} = n! / (n-r)!$
- ${}^{n}C_{r} = n! / r!\,(n-r)!$

Summation

- **$1 + 2 + 3 + \ldots\ldots + n = n\,(n+1) / 2$**

Mathematical Notation and Functions

Exponents and Logarithms

- $a^m = a . a . \dots . a$ (m times).
- $a^0 = 1$

- $b^y = x \quad y = \log_b x$
- $\log_2 8 = 3 \qquad \log_{10} 100 = 2$

- *Common logarithm* (base 10), *Natural logarithm* (base e), *Binary logarithm* (base 2)
- *The term log x shall mean* $\log_2 x$ *unless otherwise specified*

Algorithmic Notation

An **algorithm** is a finite step-by-step list of well-defined instructions for solving a particular problem.

Algorithm 2.1: (Largest Element in Array) A nonempty array DATA with N numerical values is given. This algorithm finds the location LOC and the value MAX of the largest element of DATA. The variable K is used as a counter.

```
Step 1. [Initialize] Set K:= 1, LOC:= 1 and MAX: = DATA[1].
Step 2. [Increment counter] Set K:= K + 1.
Step 3. [Test counter] If K > N, then:
            Write: LOC, MAX, and Exit.
Step 4. [Compare and update] If MAX < DATA[K], then:
            Set LOC:= K and MAX:= DATA [K].
Step 5. [Repeat loop] Go to Step 2.
```

The format for the formal presentation of an algorithm consists of *two* parts. The *first part* is a ***paragraph*** which tells the purpose of the algorithm, identifies the variables which occur in the algorithm and lists the input data. The *second part* of the algorithm consists of the ***list of steps*** that is to be executed.

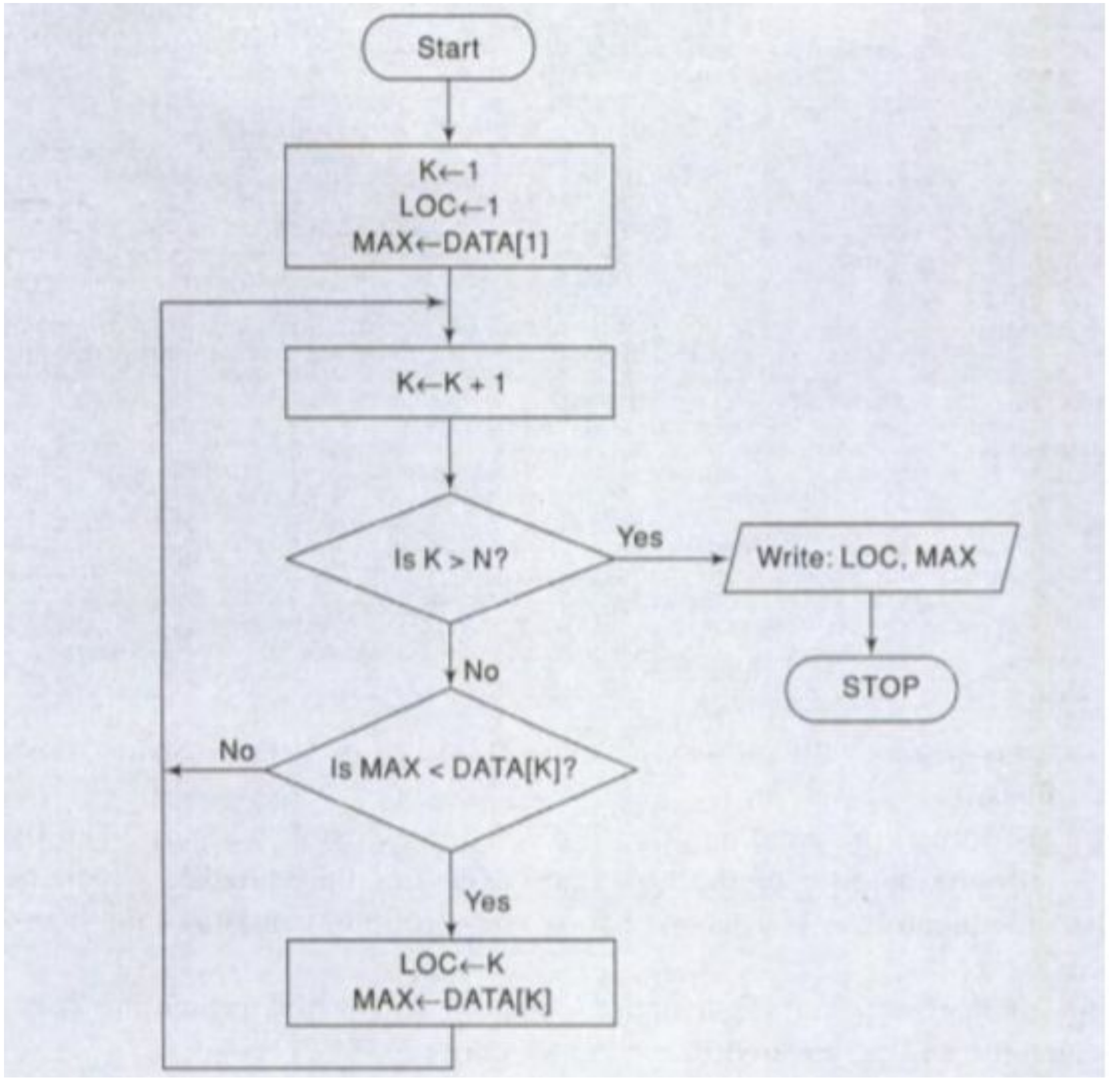
Algorithmic Notation

The following summarizes certain conventions that we will use in presenting our algorithms.

- **Steps, Control, Exit:** The steps of the algorithm are executed one after the other, beginning with Step 1, unless indicated otherwise. Control may be transferred to Step n of the algorithm by the statement “Go to Step n”. If several statements appear in the same step, then they are executed from left to right. The algorithm is completed when the statement ***Exit*** is encountered.
- **Comments:** Each step may contain a comment in brackets which indicates the main purpose of the step.
- **Variable Name:** Variable names will use capital letters. Single-letter [both uppercase and lowercase] names of variables used as counters or subscripts.
- **Assignment statement:** Dots-equal notation : = is used as assignment operation. Some texts use backward arrow ← or the equal sign = for this operation.
- **Input and Output:** For input ***Read:*** variables names is used. For writing, ***Write:*** Messages and/or variable names.

Flowchart

The *graphical representation* of an algorithm is called a **flowchart**. For the flowchart of the above algorithm see the following flowchart:



CONTROL STRUCTURES

There are **three** types of logic, or flow of control, called

a. Sequence logic or sequential flow

b. Selection logic or conditional flow

c. Iteration logic or repetitive flow

Sequence Logic (Sequential Flow)

- Unless instructions are given to the contrary, the modules are executed in the obvious sequence. The sequence may be presented explicitly, by means of numbered steps, or implicitly, by the order in which the modules arc written. Most processing, even of complex problems, will generally follow this elementary flow pattern.

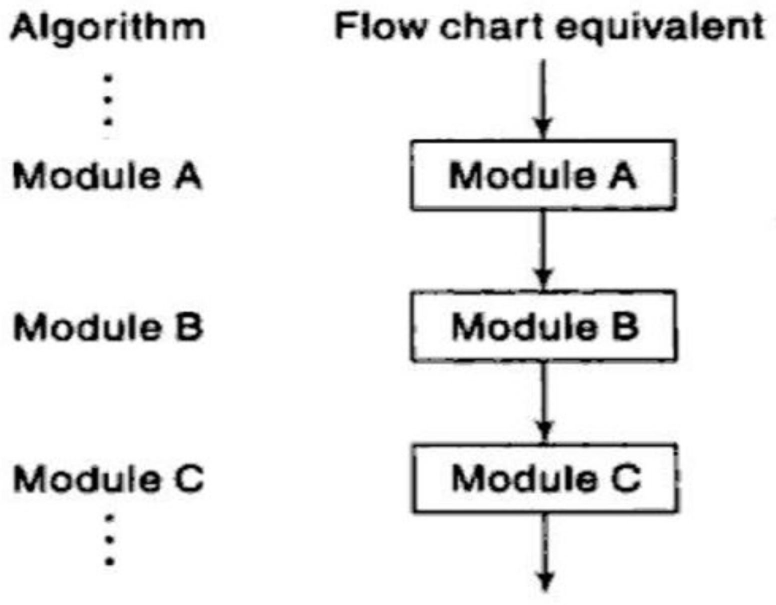


Fig. 2.3 Sequence Logic

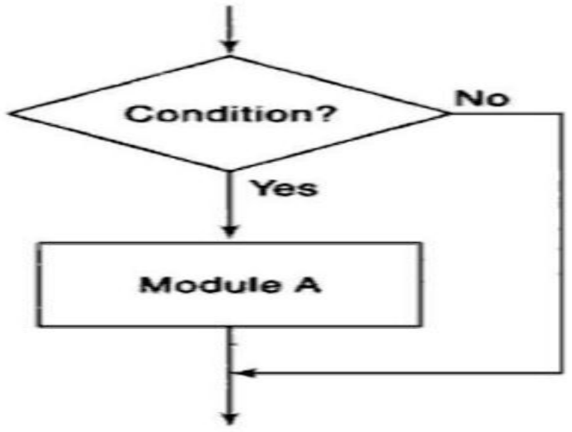
Selection Logic (Conditional Flow)

Selection logic employs a number of conditions which lead to a selection of one out of several alternative modules. The structures which implement this logic are called conditional structures or If structures. These conditional structures fall into three types, which are discussed below.

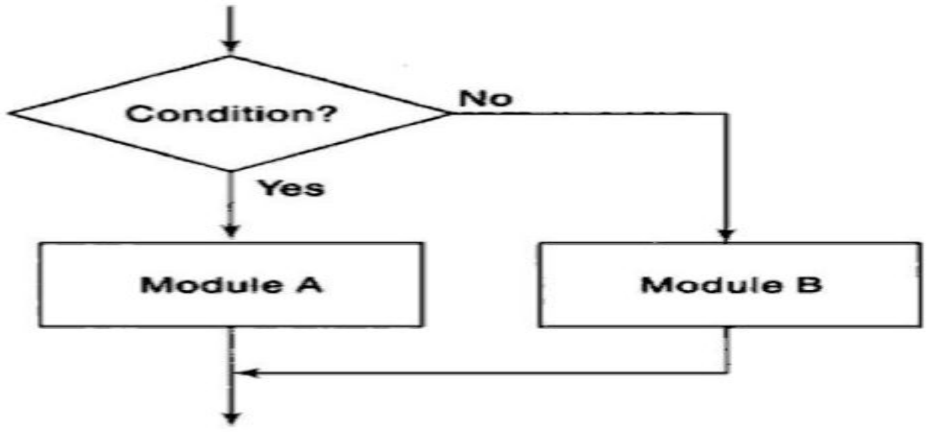
i) **Single alternative:** This structure has the form

```
If condition, then:
    [Module A]
[End of If structure]
```

Here, if the condition is TRUE, then Module A is executed, otherwise, Module A is skipped and control transfers to the next step of the algorithm.



(a) Single alternative.



(b) Double alternative.

Selection Logic (Conditional Flow)

ii) **Double alternative:** This structure has the form

```
If condition, then:
    [Module A]
Else:
    [Module B]
[End of If structure]
```

In this logic, if the condition is TRUE, then Module A is executed; if it is FALSE then Module B is executed.

iii) **Multiple alternatives:** This structure has the form:

```
If condition (1) then:
    [Module $A_1$]
Else if condition (2) then:
    [Module $A_2$]
........
Else if Condition (M) then:
    [Module $A_M$]
Else:
    [Module B]
[End of If structure]
```

The logic of this structure allows only one of the modules to be executed. If the condition (1) is TRUE then Module A_1 is executed, if it is false then condition (2) is checked and if it is TRUE, then Module A_2 is executed, and so on. If all the conditions are FALSE then Module B is executed.

Quadratic Equation

Algorithm: (Quadratic Equation) This algorithm inputs the coefficients A, B, C of a quadratic equation and outputs the real solutions, if any.

```
Step 1. Read: A, B, C.
Step 2. Set D:= $B^2 - 4AC$.
Step 3. If D > 0, then:
            (a) Set X1 := $(-B + \sqrt{D})/2A$ and X2:= $(-B - \sqrt{D})/2A$.
            (b) Write: X1, X2.
        Else if D = 0, then:
            (a) Set X:= $-B/2A$.
            (b) Write: 'UNIQUE SOLUTION', X.
        Else:
            Write: ' NO REAL SOLUTIONS'
        [End of If structure]
Step 4. Exit.
```

❖***Home task***: Draw the flowchart for the above algorithm.

Iteration Logic (Repetitive Flow)

There are two types of iteration logic. Each type begins with a Repeat statement and is followed by a module, called the body of the loop.

i) The ***repeat-for loop:***

```
Repeat for K = R to S by T:
    [Module]
[End of loop]
```

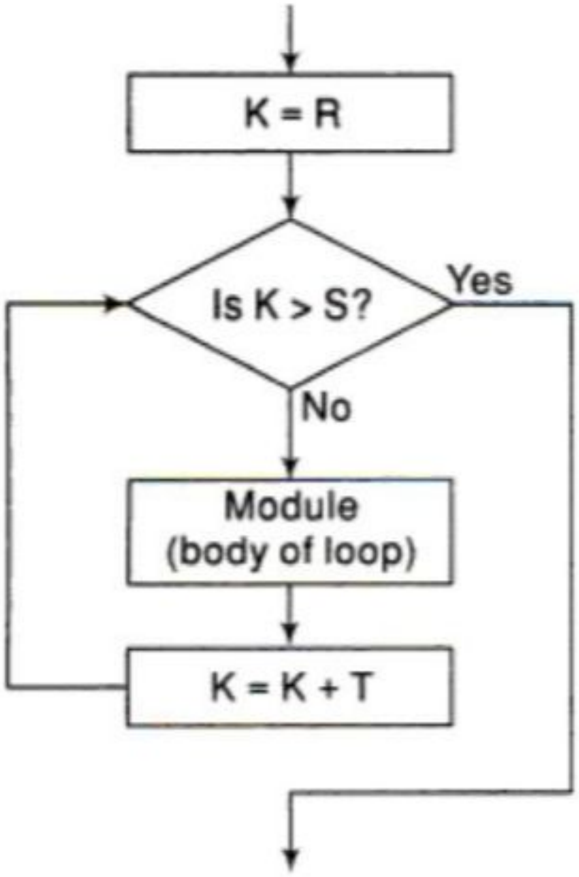
Here, R is called the *initial* value, S is the *end value* and T is the *increment*. K is called *control variable*.

ii) The ***repeat-while loop***:

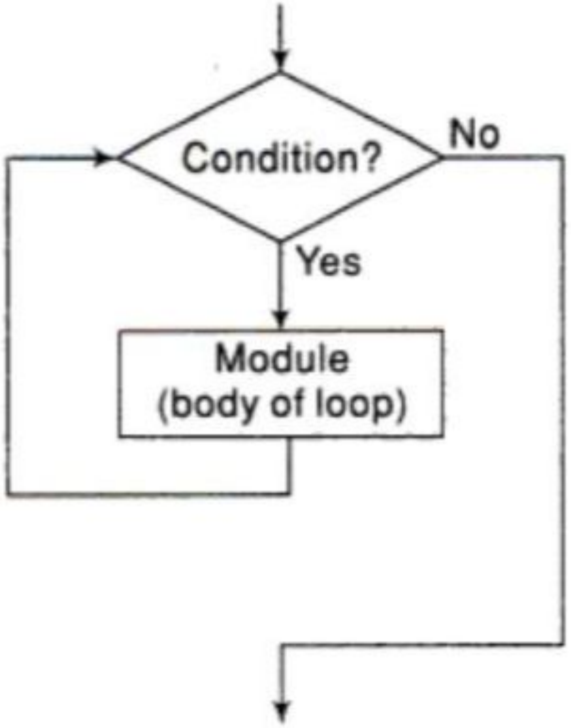
```
Repeat while condition:
    [Module]
[End of loop]
```

Here the loop is executed until the condition is FALSE.

Iteration Logic (Repetitive Flow)



(a) Repeat-For structure.



(a) Repeat-While structure.

Largest Element in Array

Algorithm 2.3: (Largest Element in Array) given a nonempty array DATA with N numerical values, this algorithm finds the location LOC and the value MAX of the largest element of DATA.

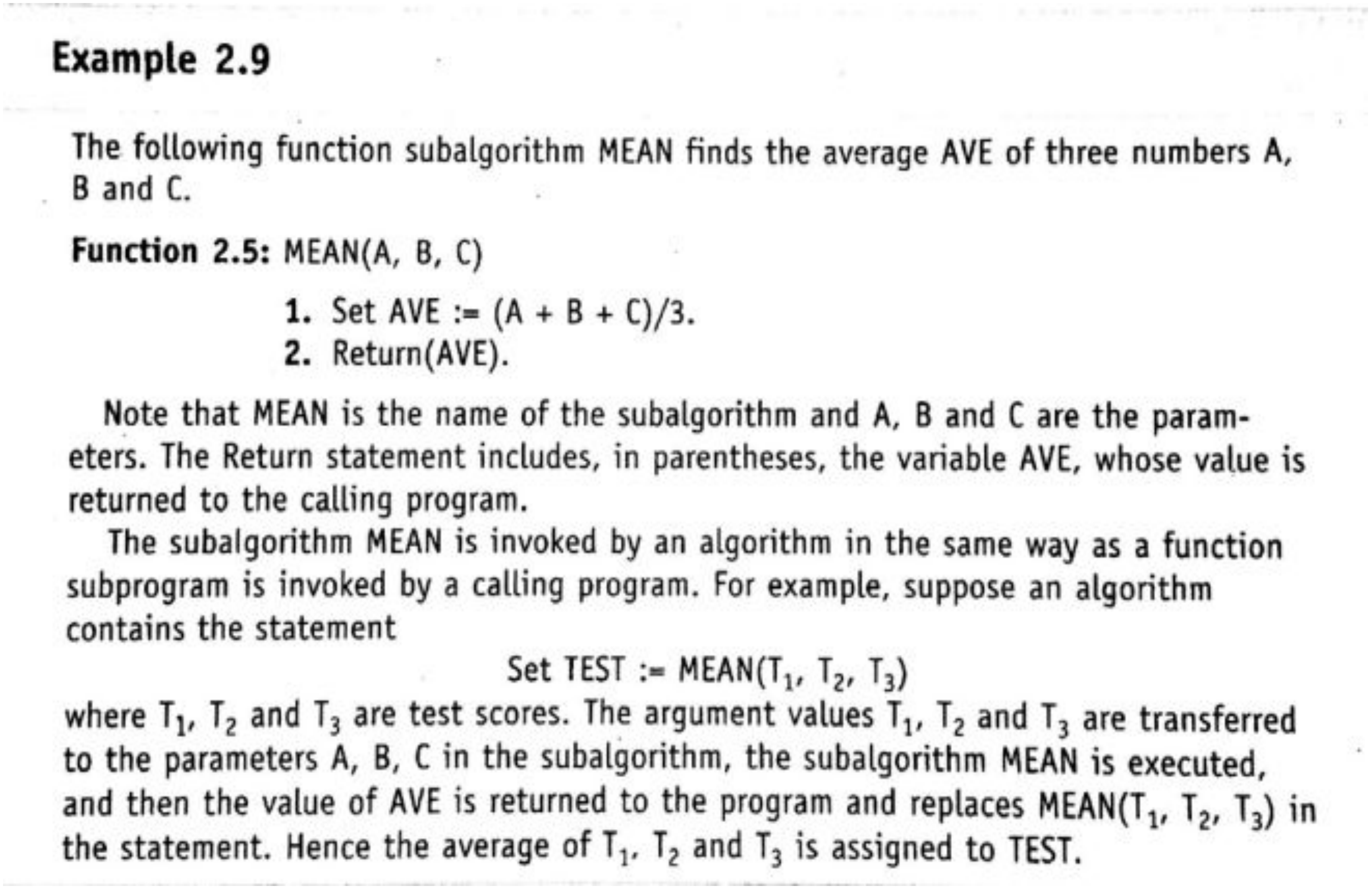
```
1. [Initialize] Set K:=1, LOC:=1 and MAX:= DATA[1].
2. Repeat Steps 3 and 4 while K <= N:
3.   If MAX < DATA [K], then:
           Set LOC: = K and MAX: = DATA [K].
4.   Set K: = K+1.
   [End of Step 2 loop.]
5. Write: LOC, MAX.
6. Exit.
```

❖***Home task***: Draw the flowchart for the above algorithm.

SUBALGORITHMS

- A ***subalgorithm*** *is* a complete and independently defined algorithmic module which is used (or *invoked or called*) by some main algorithm or by some other subalgorithm. A subalgorithm receives values, called *arguments*, from an originating (calling) algorithm; performs computations; and then sends back the result to the calling algorithm. The subalgorithm is defined independently so that it may be called by many different algorithms or called at different times in the same algorithm. The relationship between an algorithm and a subalgorithm is similar to the relationship between a main program and a subprogram in a programming language.
- The main *difference* between the format of a subalgorithm and that of an algorithm is that the subalgorithm will usually have a heading of the form NAME (PAR_1, PAR_2, … , PAR_K).
- Another difference is that the subalgorithm will have a ***Return*** statement rather than an ***Exit*** statement.
-

SUBALGORITHMS



Example 2.9

The following function subalgorithm MEAN finds the average AVE of three numbers A, B and C.

Function 2.5: MEAN(A, B, C)

```
1. Set AVE := (A + B + C)/3.
2. Return(AVE).
```

Note that MEAN is the name of the subalgorithm and A, B and C are the parameters. The Return statement includes, in parentheses, the variable AVE, whose value is returned to the calling program.

The subalgorithm MEAN is invoked by an algorithm in the same way as a function subprogram is invoked by a calling program. For example, suppose an algorithm contains the statement

Set TEST := MEAN(T_1, T_2, T_3)

where T_1, T_2 and T_3 are test scores. The argument values T_1, T_2 and T_3 are transferred to the parameters A, B, C in the subalgorithm, the subalgorithm MEAN is executed, and then the value of AVE is returned to the program and replaces MEAN(T_1, T_2, T_3) in the statement. Hence the average of T_1, T_2 and T_3 is assigned to TEST.

SUBALGORITHMS

Example 2.10

The following procedure SWITCH interchanges the values of AAA and BBB.

Procedure 2.6: SWITCH(AAA, BBB)

```
1. Set TEMP := AAA, AAA := BBB and BBB := TEMP.
2. Return.
```

The procedure is invoked by means of a Call statement. For example, the Call statement

Call SWITCH(BEG, AUX)

has the net effect of interchanging the values of BEG and AUX. Specifically, when the procedure SWITCH is invoked, the argument of BEG and AUX are transferred to the parameters AAA and BBB, respectively; the procedure is executed, which interchanges the values of AAA and BBB; and then the new values of AAA and BBB are transferred back to BEG and AUX, respectively.

Home Task

Problem:

1. Write a program to calculate the roots of the quadratic equation $ax^2 + bx + c = 0$ where a, b and c are known.
2. Write a program to find the largest number from a given list of integers.