PROGRAM DEVELOPMENT AND COMPOSITION

Lubna Ahmed

Introduction

Four steps:

- Problem analysis and specification
- Algorithm development
- Program coding
- Program execution and testing

Top down process

Top down design is the process of starting with a large task and breaking it down into smaller, more easily understandable pieces that perform a portion of desired task.

Top down process cont'd

Steps involve in solving any problem:-

1. Clearly state the problem that you are trying to solve.
2. Define the inputs required by the program and the outputs to be produced by the program.
3. Design the algorithm that you intend to implement in the program.
4. Turn the algorithm into Fortran statements.
5. Test the resulting Fortran program.

Top down process cont’d

1. Clear Statement of Problem

Programs are usually written to fill some perceived need, but that need may not be articulated clearly by the person requesting the program.

2. Define Input and Output of a Problem

Review the problem in order to determine the

Input	**Information given**
Output	**Information to be produced to solve the problem**

Top down process cont'd

Example:

John is a nuclear physicist and is conducting research with the radio-active element polonium. The half-life of polonium is 140days. John would like to know how much polonium will remain after running his experiment for 180 days if 10 milligrams are present initially.

Input	**Output**
Initial amount: 10 mg **Half life: 140 days** **Time Period : 180 days**	**Amount remaining**

Top down process cont'd

3. Algorithm Development

Algorithm is a step-by-step procedure for finding the solution to a problem. In this stage, designer divides the problems into **subtasks.** This process is called **decomposition.** If the subtasks are themselves large, the designer can break them up into even smaller sub-subtasks. This process continues until each piece does a simple, clearly understandable job.

Each Subtask is then refined through **Stepwise refinement**. Stepwise refinement is usually done by **Pseudo code.**

4. Algorithm to Fortran Statement

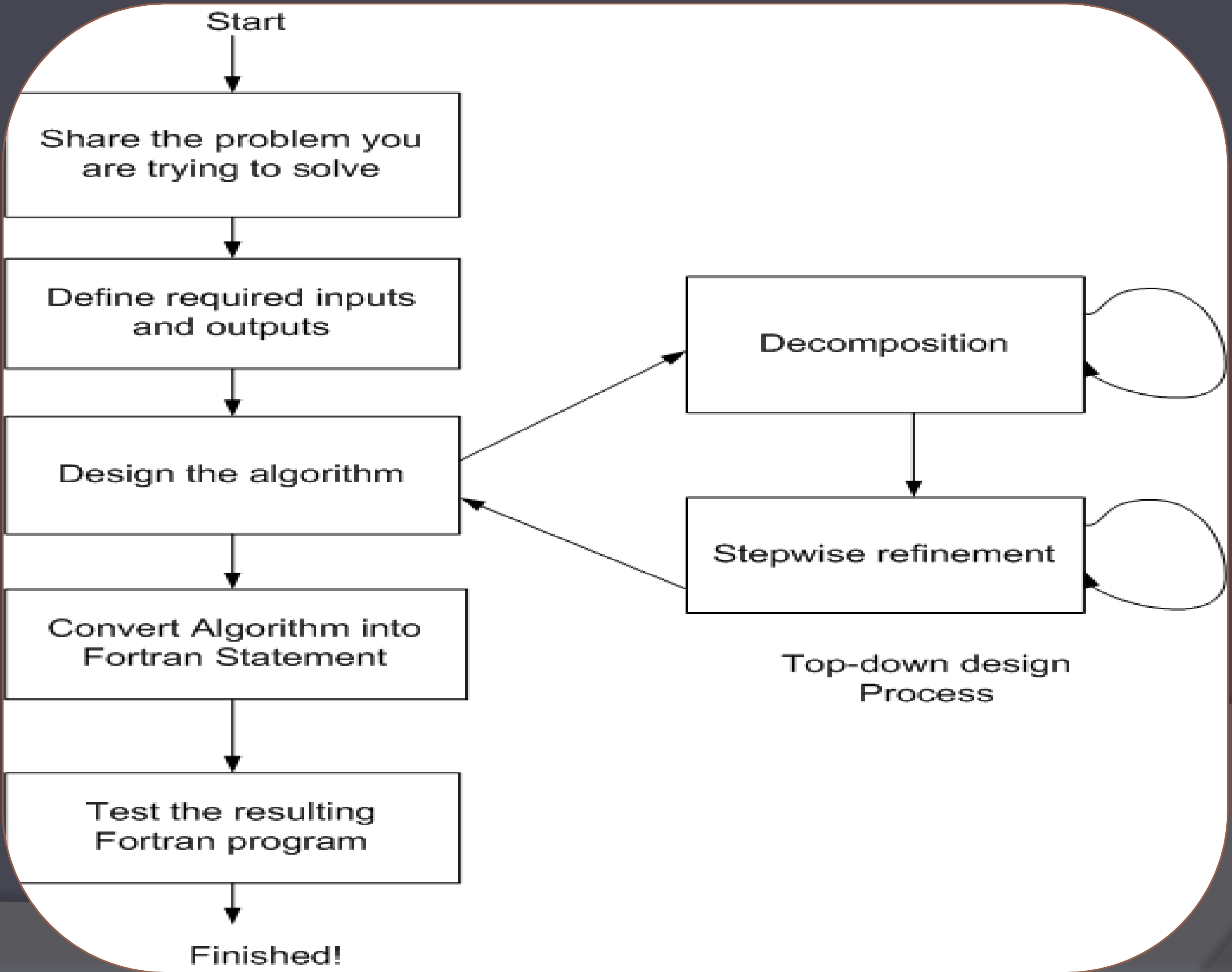
If the **decomposition** and **refinement** process was carried out properly, this step will be very simple. All the programmer will have to do is to replace **pseudo code** with the corresponding Fortran statements on a one-for-one basis.

Top down process cont'd

5. Test the Program

- Test the component of the program.
- Test the program for all legal input data set.
- Program involving different branches must be tested for each individual branches.
- Release for use.

Top down process cont'd


Start
Share the problem you are trying to solve
Define required inputs and outputs
Design the algorithm
Convert Algorithm into Fortran Statement
Test the resulting Fortran program
Finished!
Decomposition
Stepwise refinement
Top-down design Process

Algorithm Development

- Once the problem has been specified, a procedure to produce the required output from the given input must be designed.
- Such a procedure is called algorithm.
- The relationships between the inputs and outputs are necessary in this step.
- Algorithm should be a *Pseudocode* description of the actual code to be written.
- Some programmers also use graphical representation of algorithms in addition to or in place of pseudocode description.
- Most common one is the flowchart.

Pseudo code &Flowchart

The constructs used to build algorithms can be described in two different ways:

1. Pseudo Code

2. Flowchart

Pseudo Code:

- Pseudo code is a kind of structured English for describing algorithms. It is a hybrid mixture of English and Fortran.
- It allows the designer to focus on the logic of the algorithm without being distracted by details of language syntax.

Pseudo code &Flowchart cont’d

Example:

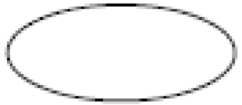
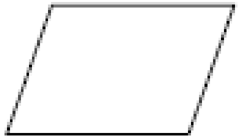
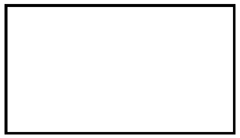
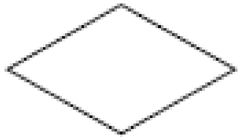
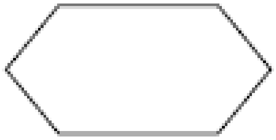
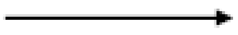
Pseudo Code for Fahrenheit to Kelvin conversion

```
1.Prompt user to enter temperature in degrees Fahrenheit
2.Read temperature in degrees Fahrenheit (temp_f)
3.temp_k in Kelvin $\longleftarrow$ (5./9.)*(temp_f-32)+273.15
4.Write temperature in degree Fahrenheit
```

Pseudo code &Flowchart cont’d

Flowchart is a type of diagram that represents an algorithm or process, showing the steps as boxes of various kinds, and their order by connecting these with arrows. This diagrammatic representation can give a step-by-step solution to a given problem.

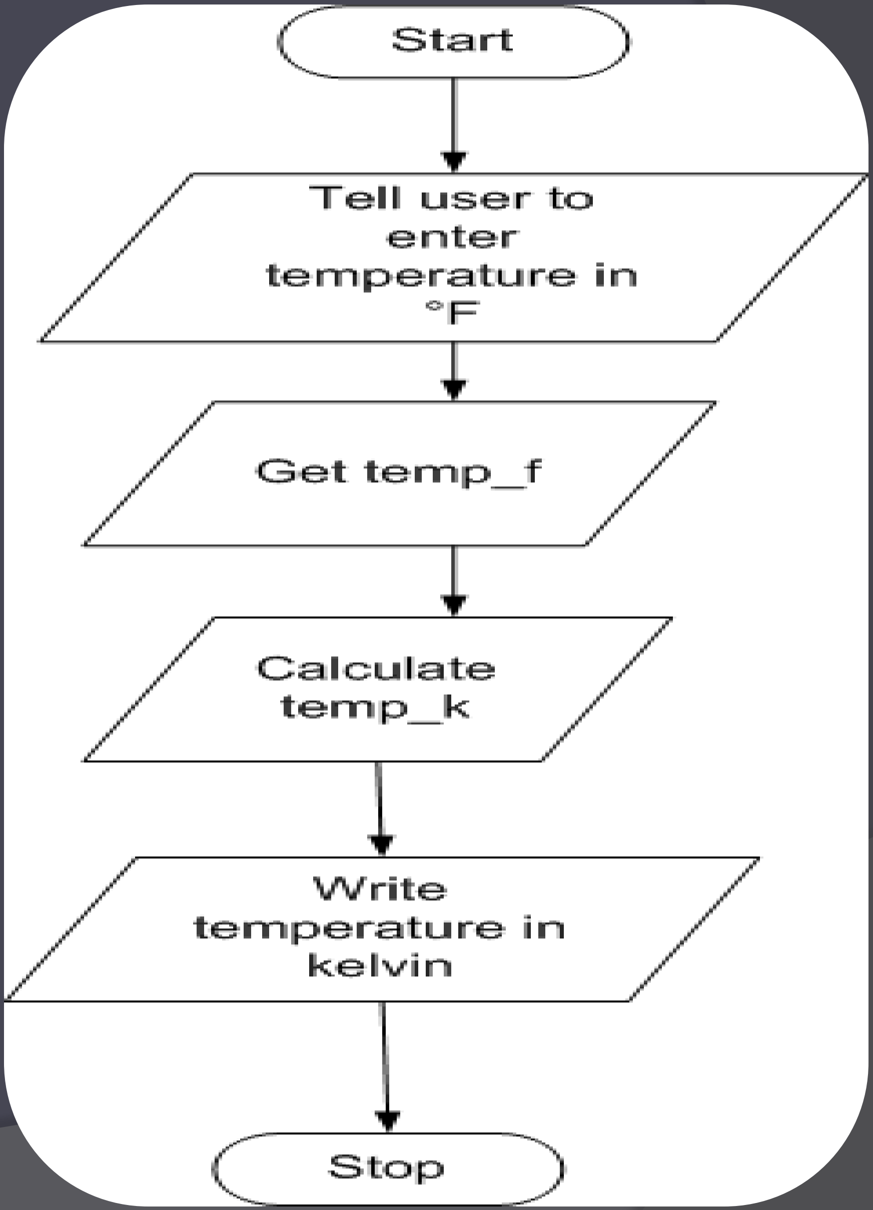
Pseudo code &Flowchart cont’d

	An oval is used to indicate beginning or end of an algorithm
	A parallelogram indicates the input or output of information
	A rectangle indicates the assignment of values to variables; also used to indicate computations
	A diamond indicates a point in an algorithm where a selection is made
	A hexagon indicates the beginning of a repetition structure
	An arrow, called a flow line, indicates the order in which the steps of algorithm are to be carried out

Pseudo code &Flowchart cont'd

Example:
Flowchart of converting temperature from degree Fahrenheit to Rankin scale

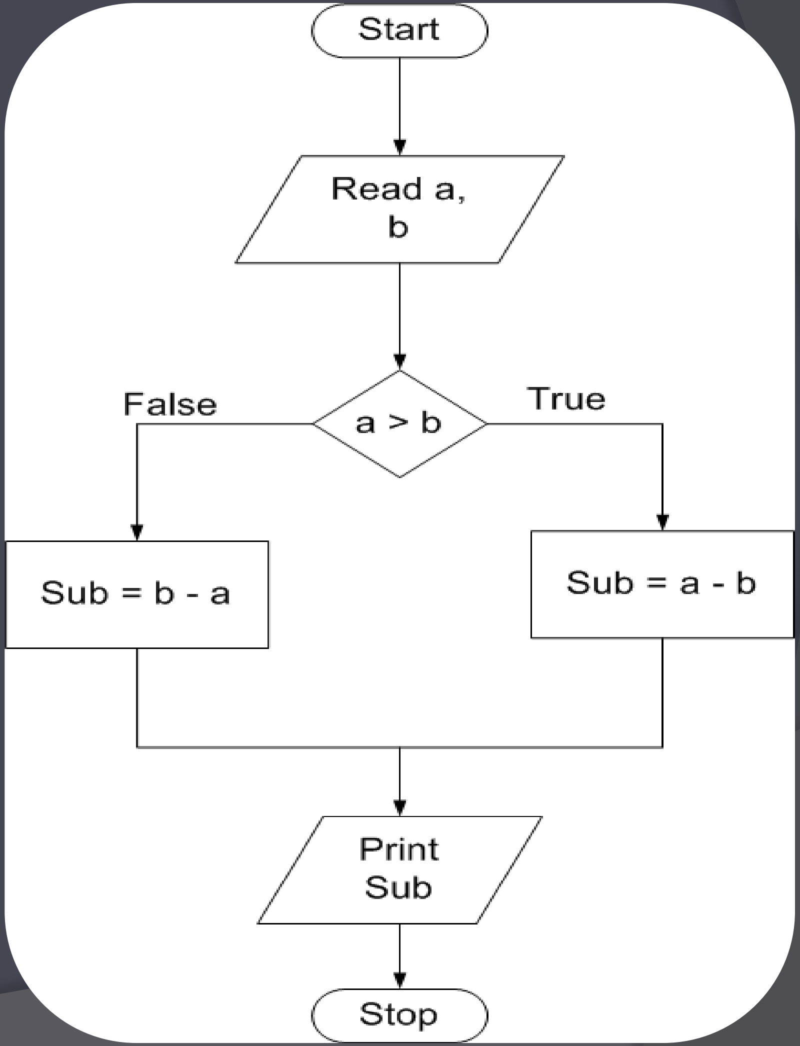
This is an example of SEQUENTIAL STRUCTURE


Start
Tell user to enter temperature in °F
Get temp_f
Calculate temp_k
Write temperature in kelvin
Stop

Pseudo code &Flowchart cont'd

Example:
Flowchart to determine the difference between two real numbers

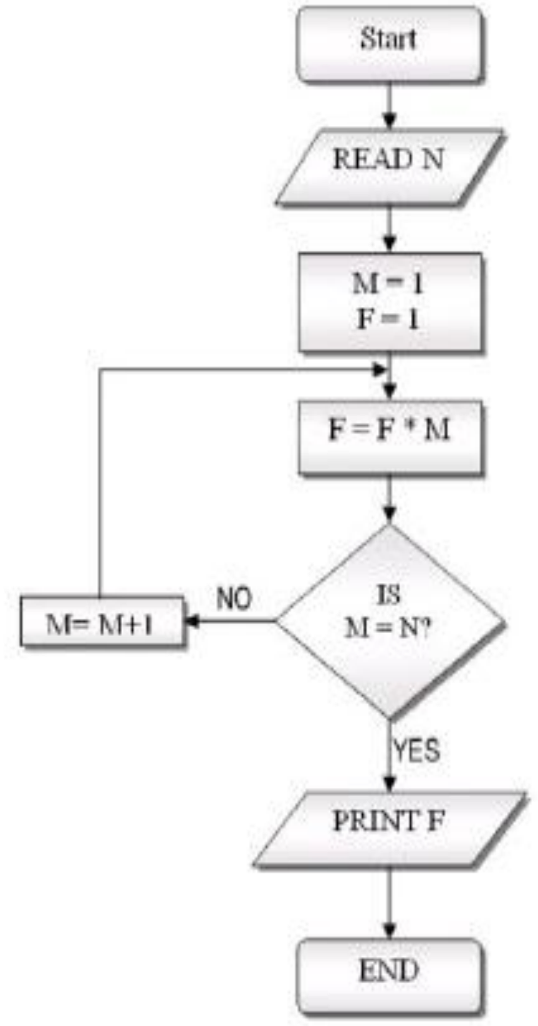
This is an example of SELECTIVE STRUCTURE



Pseudo code &Flowchart cont'd

Example:
Flowchart to determine factorial of any number

This is an example of REPETITIVE/ ITERATIVE STRUCTURE



Program Coding

If the problem has been carefully analyzed and if complete and clear algorithms have been developed, the third step of program coding is usually straightforward

Program Coding cont'd

Simple Structure of Fortran Programming

Program name. Opening documentation	Heading
Declare variables and structures. Assign values to variables.	Specification Part
Process data. Print results. End program.	Execution Part

Program Coding cont’d

```
 PROGRAM DECAY
 IMPLICIT NONE
!This program calculates the amount of a radioactive
!substance that remains after a specific time, given
!an initial amount, and its half-life. Variables used
!are:
    !INIT        : initial amount of substance
    !HFLIFE      : half-life of the substance
    !TIME        : time at which the amount remaining
                   ! is calculated
   !RESID        : amount remaining
REAL INIT, HFLIFE, TIME, RESID
       PRINT*,’ENTER INITIAL AMOUNT, HALF-LIFE, AND TIME’
       READ*,INIT,HFLIFE,TIME
       RESID=INIT*0.5**(TIME/HFLIFE)
       PRINT*,’AMOUNT REMAINING=’, RESID
       END
```

Program Testing

<u>Errors in Program</u>

- Syntax error or compile-time error
- Run-time error
- Logical error

<u>Syntax Error</u>

- Incorrect system commands causes syntax errors.
- Errors in the program's syntax, such as
  - Incorrect punctuations, or
  - Misspelled key words will be detected during compilation.

<u>Example:</u>

```
PRINT*, 'AMOUNT REMAINING=.RESID
```

Program Testing cont’d

- Runtime Error

 Errors that may not be detected until execution of the program fall in this category.

 These errors must be corrected by replacing the erroneous statements with the correct ones.

Example:

An attempt to divide by zero in an arithmetic expression

Program Testing cont'd

- Logical Error :

Logical errors arise in the design of the algorithm or in the coding of the program that implements the algorithm.

Example:

If the statement

```
RESID = INIT*0.5**(TIME/HFLIFE)
```

was mistakenly entered as

```
RESID = INIT*0.5*(TIME/HFLIFE)
```

no error would occur during the compilation or execution of the program. But the result of the program would be Incorrect

Program Testing cont'd

Rounding Error

At times a program will give numerical answers to a problem which appear in explicably different from what we know to be the correct mathematical solution. This can be due to rounding error.

Example:

Run the following program extract:

```
X = 0.1
DO
X = X + 0.001
PRINT*, X
IF (X == 0.2)  EXIT
END DO
```

This program never stops, X never reaches to 0.2 exactly. X misses the value of 0.2 by about 10^{-9}.

Program Testing cont’d

- Thus, *“it is important that the user run a program several times with input data for which the correct results are known in advance”.*
- A program cannot be considered to be correct until it has been validated with several sets of test data.
- The test data should be carefully selected so that each part of the program is checked