Chapter 2: Problem Solving

- In this chapter you will learn about:
  - Introduction to Problem Solving
  - Software development method (SDM)
    - Specification of needs
    - Problem analysis
    - Design and algorithmic representation
    - Implementation
    - Testing and verification
    - Documentation

Introduction to Problem Solving

- Programming is a problem solving activity. When you write a program, you are actually writing an instruction for the computer to solve something for you.
- Problem solving is the process of transforming the description of a problem into a solution by using our knowledge of the problem domain and by relying on our ability to select and use appropriate problem-solving strategies, techniques and tools.

Case Study

- Problem: Compute the straight-line distance between two points in a plane.

Software Development Method (SDM)

- For programmer, we solve problems using Software Development Method (SDM), which is as follows:
  1. Specification of needs
     - State the problem clearly
  2. Problem analysis
     - Describe the input and output information
  3. Design and algorithmic representation
     - Work the problem by hand for a simple set of data
  4. Implementation
     - Develop a solution and convert it to a computer program
  5. Testing and verification
     - Test the solution with a variety of data
  6. Documentation
     - Document and record information for reference

Specification of Needs

- Specifying the problem requirements requires you to state the problem clearly and to gain the understanding of what to be solved and what would be the solution.
- When specifying problem requirement, we ask ourselves the following questions:
  - What the problem is.
  - What the solution should provide.
  - What is needed to solve it.
  - If there are constraints and special conditions.

Case Study

Problem: Compute the straight-line distance between two points in a plane.

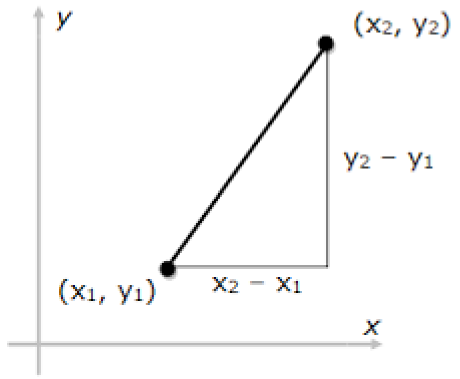
- What the problem is.
- What the solution should provide.
- What is needed to solve it.
- If there are constraints and special conditions.

Problem Analysis

- Analyzing the problem require us to identify the following:
  - Input(s) to the problem, their form and the input media to be used
  - Output(s) expected from the problem, their form and the output media to be used
  - Special constraints or conditions (if any)
  - Any formulas or equations to be used

Case Study

- Input?
  - Point 1 coordinate
  - Point 2 coordinate
- Output?
  - Distance between points
- Constraint/condition?
  - Nil
- Formula/equation?
  - Pythagorean theorem



Designing algorithm

- Designing algorithm to solve the problem requires you to develop a list of steps, arranged in a specific logical order which, when executed, produces the solution for a problem.
- Using top-down design (also called *divide and conquer*):
  - You first list down the major tasks
  - For each major task, you further divide it into sub-tasks (refinement step)
- When you write algorithm, write it from the computer's point of view.

Designing Algorithm cont..

- An algorithm must satisfy these requirements:
  - It may have an **input(s)**
  - It must have an **output(s)**
  - It should **not be ambiguous** (there should not be different interpretations to it. Every step in algorithm must be clear as what it is supposed to do)
  - It must be **general** (it can be used for different inputs)
  - It must be **correct** and it must solve the problem for which it is designed
  - It must execute and terminate in a **finite** amount of time
  - It must be **efficient** enough so that it can solve the intended problem using the resource currently available on the computer

Case Study

Major Task:

1. Read Point 1 coordinate
2. Read Point 2 coordinate
3. Calculate distance
4. Display the computed distance

However, looking at the above algorithm, we can still further refine step 3, by introducing the formula to calculate the amount to pay.

After refinement:

1. Read Point 1 coordinate
2. Read Point 2 coordinate
3. $Distance = \sqrt{(side\ 1)^2 + (side\ 2)^2}$
4. Display the computed distance

Remember, the order of the steps in algorithm is very important. Consider the following, will the result be the same?

1. Display the distance
2. Read Point 1 coordinate
3. Compute distance
4. Read Point 2 coordinate

Pseudocodes

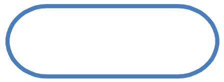
- A pseudocode is a semiformal, English-like language with limited vocabulary that can be used to design and describe algorithms.

- Criteria of a good pseudocode:
  - Easy to understand, precise and clear
  - Gives the correct solution in all cases
  - Eventually ends

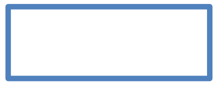
Flowcharts

- Flowcharts is a graph used to depict or show a step by step solution using symbols which represent a task.
- The symbols used consist of geometrical shapes that are connected by flow lines.
- It is an alternative to pseudocoding; whereas a pseudocode description is verbal, a flowchart is graphical in nature.

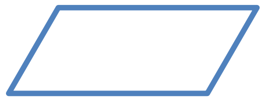
Flowchart Symbols



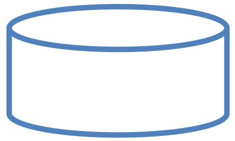
Terminal symbol - indicates the beginning and end points of an algorithm.



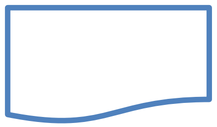
Process symbol - shows an instruction other than input, output or selection.



Input-output symbol - shows an input or an output operation.

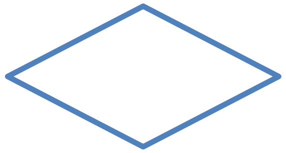


Disk storage I/O symbol - indicates input from or output to disk storage.

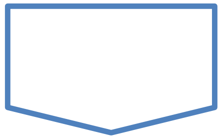


Printer output symbol - shows hardcopy printer output.

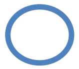
Flowchart Symbols cont…



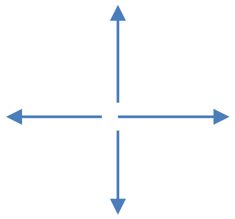
Selection symbol - shows a selection process for two-way selection.



Off-page connector - provides continuation of a logical path on another page.



On-page connector - provides continuation of logical path at another point in the same page.



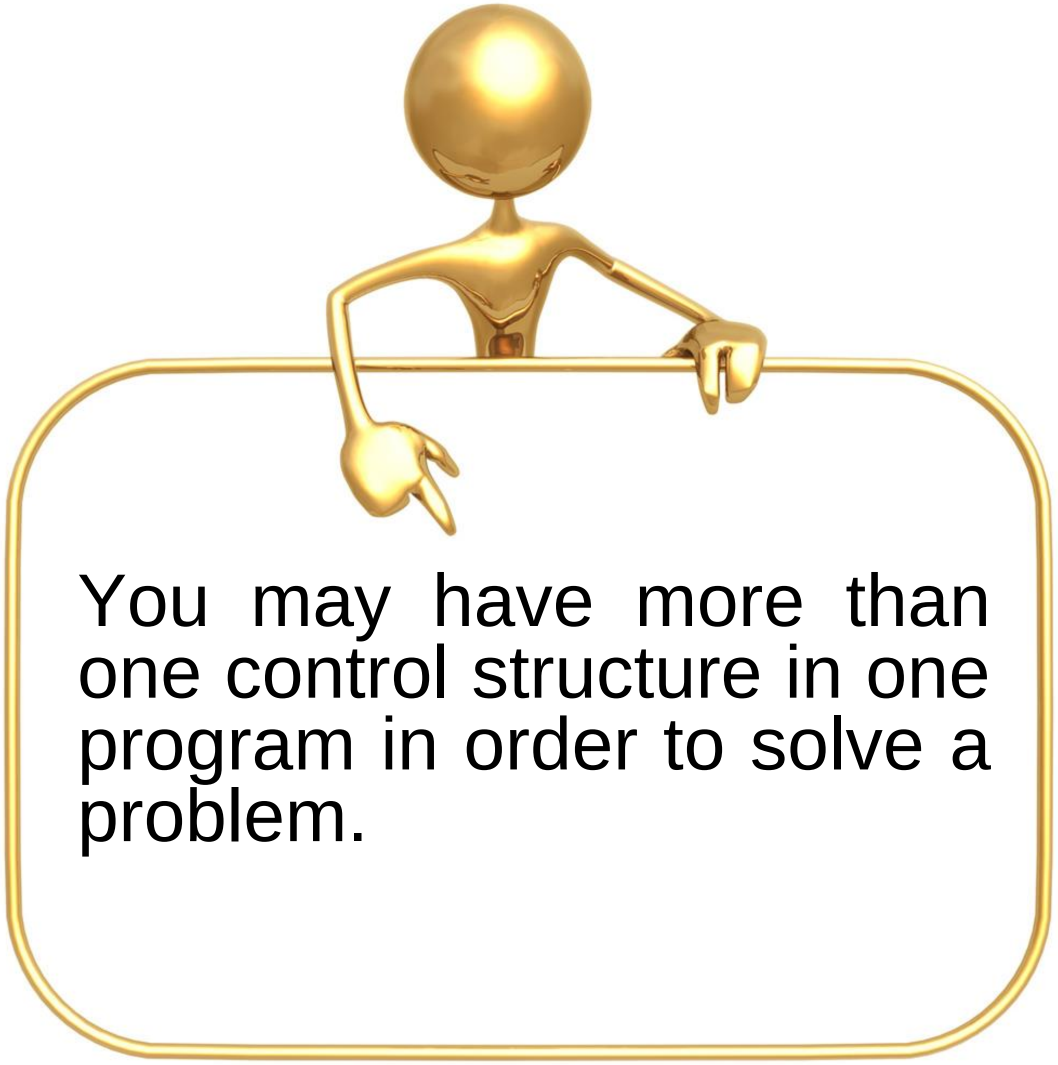
Flow lines - indicate the logical sequence of execution steps in the algorithm.

Control Structure

- An algorithm can be represented using Pseudocode or Flowchart.
- In 1966, two researchers, C. Bohn and G. Jacopini, demonstrated that any algorithm can be described using only **3 control structures**: sequence, selection and repetition.

Control Structure

- Sequence: A series of steps or statements that are executed in the order they are written in an algorithm.
- Selection: Defines two courses of action depending on the outcome of a condition. A condition is an expression that is, when computed, evaluated to either true or false.
- Repetition: Specifies a block of one or more statements that are repeatedly executed until a condition is satisfied.

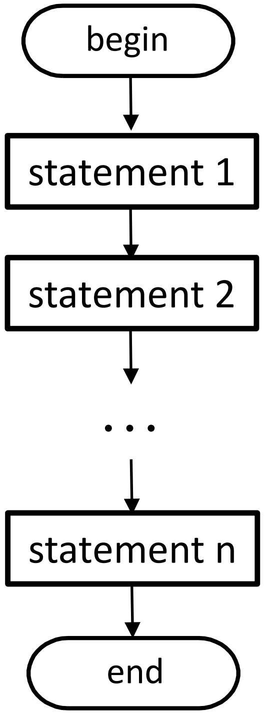


You may have more than one control structure in one program in order to solve a problem.

The **Sequence** control structure

- A series of steps or statements that are executed in the order they are written in an algorithm.
- The beginning and end of a block of statements can be optionally marked with the keywords *begin* and *end*.

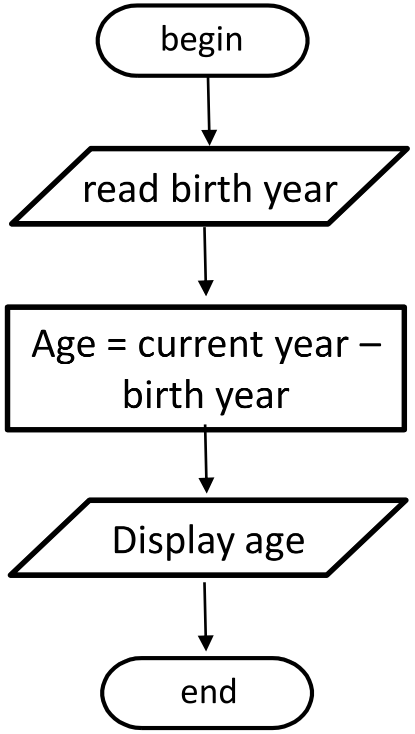
```
begin
    statement 1.
    statement 2.
    …
    …
    statement n.
end
```



The **Sequence** control structure

Problem: calculate a person's age

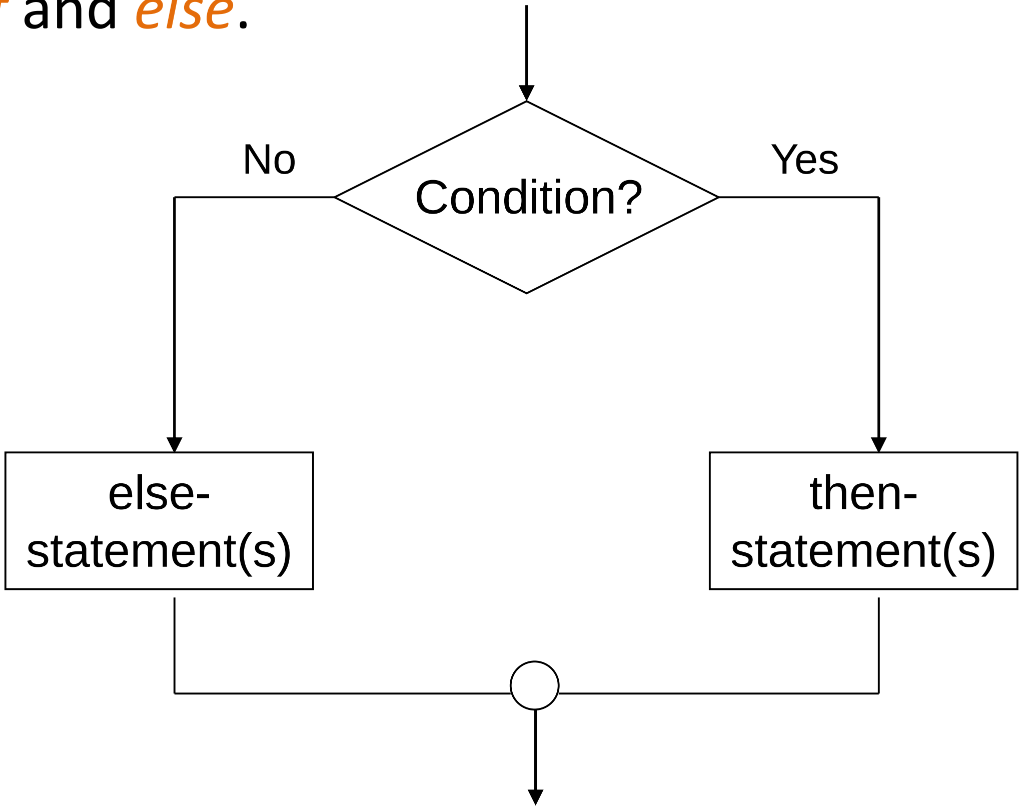
```
Begin
   read birth year
   age = current year – birth year
   display age
End
```



The **Selection** control structure

- Defines two courses of action depending on the outcome of a condition. A condition is an expression that is, when computed, evaluated to either true or false.
- The keyword used are *if* and *else*.
- Format:

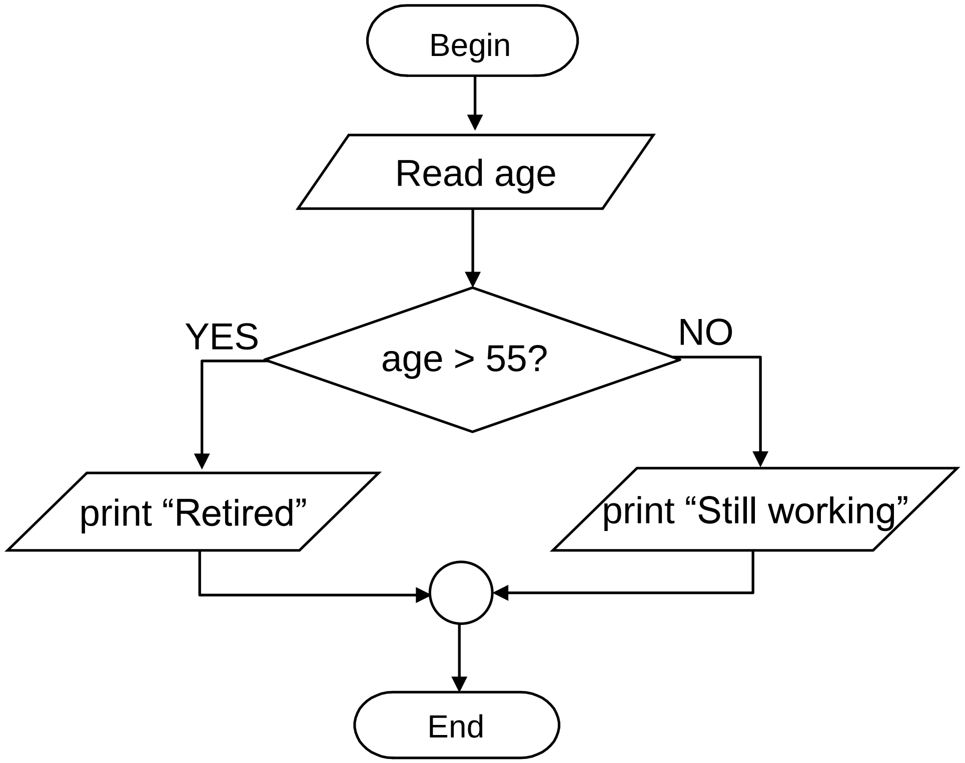
```
if (condition)
    then-part
else
    else-part
end_if
```



The **Selection** control structure

```
Begin
   read age
   if (age is greater than 55)
        print “Retired”
   else
        print “Still working”
   end_if
End
```

```
Begin
   read age
   if (age > 55)
        print “Retired”
   else
        print “Still working”
   end_if
End
```



Pseudocodes: The **Selection** control structure

- Sometimes in certain situation, we may omit the else-part.

```
if (number is odd number)
   print "This is an odd number"
end_if
```

Example 1

- Nested selection structure: basic selection structure that contains other if/else structure in its then-part or else-part.

```
if (number is equal to 1)
      print "One"
else if (number is equal to 2)
      print "Two"
else if (number is equal to 3)
      print "Three"
else
      print "Other"
end_if
```

Example 2

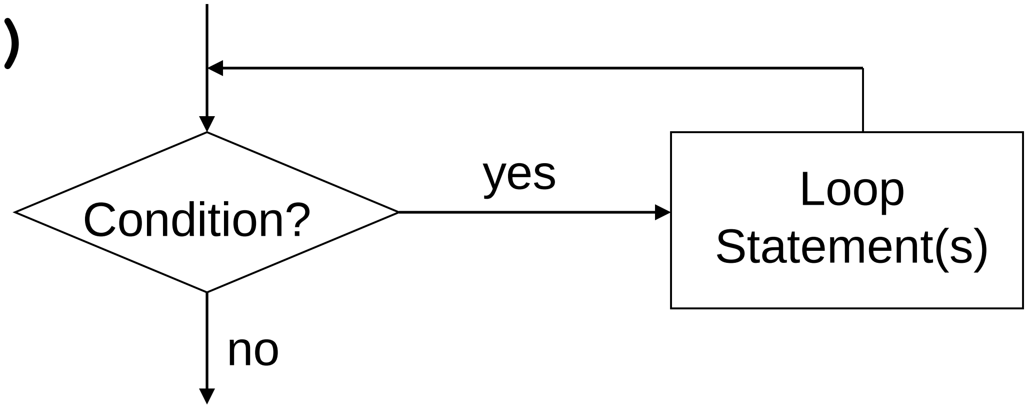
Exercise

Draw the flowchart diagram for Example 1 and Example 2

The **Repetition** control structure

- Specifies a block of one or more statements that are repeatedly executed until a condition is satisfied.
- The keyword used is *while*.
- Format:

```
while (condition)
    loop-body
end_while
```



Problem: Write a program that reads and displays the age of 10 people (one after another).

For this problem, we need a way to count how many people whose age have been processed (read and displayed). Therefore, we introduce a concept of *counter*, a variable used to count the number of people whose age have been processed by the program.

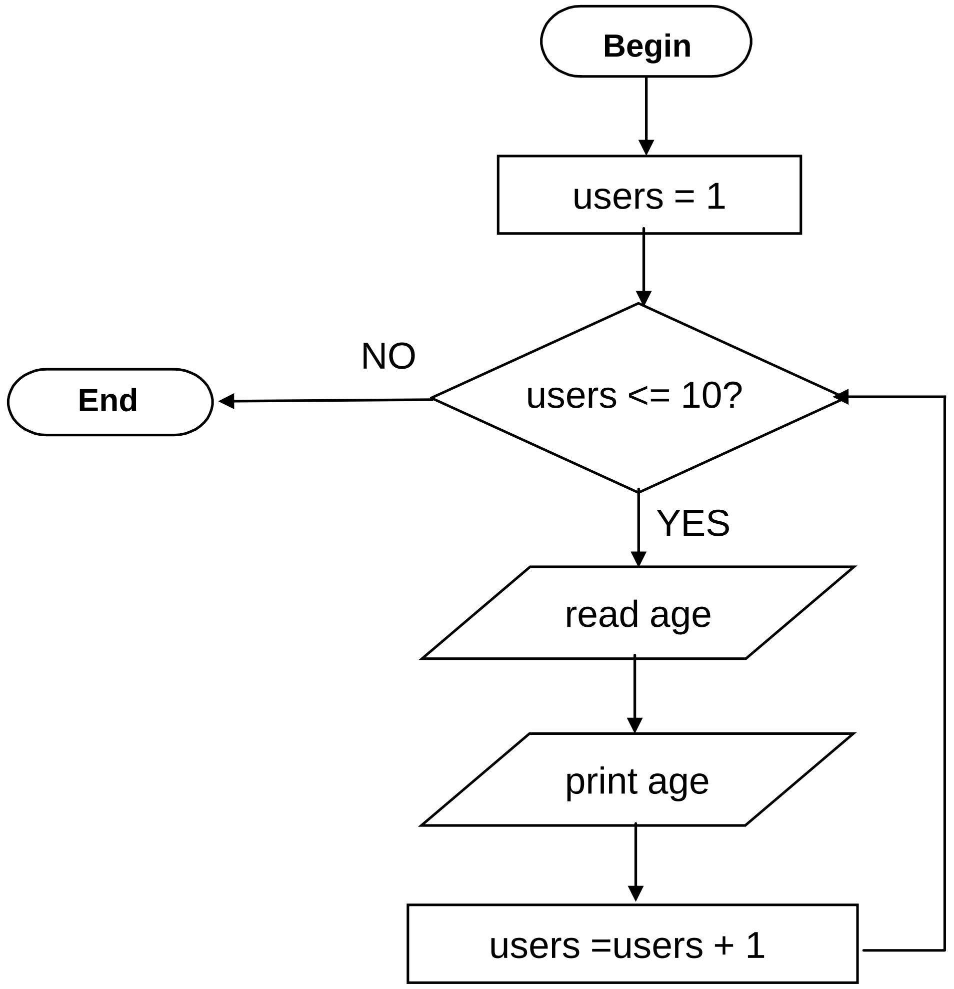
```
Begin
   number of users giving his age = 1
   while (number of users giving his age <= 10)
       read the age from the user.
       print the user age.
       number of user giving his age + 1
   end_while
End
```

Counter initialisation

Loop condition

Updating counter

```
Begin
   users = 1
   while (users <= 10)
       read age
       print age.
       users = users + 1
   end_while
End
```


Begin
users = 1
users <= 10?
NO
End
YES
read age
print age
users =users + 1

Subsequently..

```
Begin
   number of users giving his age = 0
   while (number of users giving his age < 10)
      read the age from the user.
      print the user age.
      number of user giving his age + 1
   end_while
End
```

You can start the counter with ZERO

The loop condition must less than the value it requires to stop

```
Begin
   users = 0
   while (users < 10)
      read age
      print age.
      users = users + 1
   end_while
End
```

Little extra…

- Now let us put together everything that you have learnt so far.
- Problem:

  Write a program that will calculate and print the age of 10 persons, given their birth year. If the age of the person is above 55, then the program will print “Retired”, otherwise, the program will print “Still working”.

```
Begin
   users = 1
   while (users <= 10)
        begin
            Read birth year
            age = current year – birth year
            print age
            if age > 55
                print "Retired"
            else
                print "Still working"
            end_if
            users = users + 1
        end
   end_while
End
```

Example 3

Note that in this example, we are using all the three control structures: sequence, selection and repetition

Exercise

Draw the flowchart diagram for Example 3

Implementation

- The process of implementing an algorithm by writing a computer program using a programming language (for example, using C language)
- The output of the program must be the solution of the intended problem
- The program must not do anything that it is **not** supposed to do

  (Think of those many viruses, buffer overflows, trojan horses**,** *etc. that we experience almost daily. All these result from programs* doing more *than they were intended to do)*

Testing and Verification

- Program testing is the process of executing a program to demonstrate its correctness
- Program verification is the process of ensuring that a program meets user-requirement
- After the program is compiled, we must execute the program and test/verify it with different inputs before the program can be released to the public or other users (or to the instructor of this class)

Documentation

- Writing description that explain what the program does.
- Important not only for other people to use or modify your program, but also for you to understand your own program after a long time *(believe me, you will forget the details of your own program after some time ...)*
- Can be done in 2 ways:
  - Writing comments between the line of codes
  - Creating a separate text file to explain the program

Documentation cont…

- Documentation is so important because:
  - You may return to this program in future to use the whole of or a part of it again
  - Other programmer or end user will need some information about your program for reference or maintenance
  - You may someday have to modify the program, or may discover some errors or weaknesses in your program
- Although documentation is listed as the last stage of software development method, it is actually an ongoing process which should be done from the very beginning of the software development process.

Exercise time!!!

Volume calculation

Write a pseudocode and a flowchart for a C program that reads the value of height, width and length of a box from the user and prints its volume.

Height estimation

Given the following formula to estimate height of a person based on femur length and humerus length, design an algorithm to estimate a person's height from the femur and humerus lengths.

Female height = femur length x 1.94 + 28.7
Male height = femur length x 1.88 + 32

Female height = humerus length x 2.8 + 28.2
Female height = humerus length x 2.9 + 27.9

Sum of 1 to 10

Write a pseudocode or flowchart for a program that would compute and print the sum of all integers between 1 and 10.

Summary

- This chapter introduced the concept of problem solving: a process of transforming the description of a problem into a solution.
- A commonly used method – SDM which consists of 6 steps
- 3 basic control structures : sequence, selection and repetition structures
- Pseudocode and Flow chart

T.H.E E.N.D