Problem Solving Procedures

- **Problem Identification.** Clearly explain and present a problem.

- **Problem Analysis.** Find out the type of input data, and the processing and output requirement. Break the large problems into a set of smaller tasks separated from each other.

- **Algorithm Design.** Design a set of steps which can be used to handle the required tasks. (May be written in flowchart or pseudocode.)


** Algorithm. A finite number of steps with a set of well defined rules solving a problem.**

- **Solution Development.** Write down the steps in the algorithm in a computer programming language.

- **Debugging and Testing.** Locate and correct any errors (bugs) and ensuring that all data are processed properly.

- **Documentation.** Orderly present of the complete purpose, function and history of a program.

Problem Identification

- Usually, the user should be asked more and more questions to define the problem more precisely.

- One of the best ways of problem definition is to formulate a problem using clear and concise statements.

- The statement(s) may usually start with “The problem is to …”. e.g. “The problem is to calculate the wages of employees based on their weekly working hours.”

Problem Analysis

- It involves two major steps:
  i. identifying inputs, processing steps and outputs (IPO chart)
  ii. split the large problem into a series of sub-problems (structured diagram).

- Examples of IPO charts
  i. Example 1 (p 7):

  Problem: Calculate the average height of students in a class.

<table>
<thead>
<tr>
<th>Input</th>
<th>Process</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>The height of each</td>
<td>i. Read the height of each student</td>
<td>The average height of the students</td>
</tr>
<tr>
<td>student</td>
<td>ii. Count the total number of students</td>
<td></td>
</tr>
<tr>
<td></td>
<td>iii. Calculate the average height</td>
<td></td>
</tr>
</tbody>
</table>

```
ii. Example 2 (p 7):

Problem: Accept two characters, sort them in ascending order and displayed them on screen.

<table>
<thead>
<tr>
<th>Input</th>
<th>Process</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two characters</td>
<td>i. Prompt the user to input two characters</td>
<td>Two characters in ascending order</td>
</tr>
<tr>
<td>(char 1, char 2)</td>
<td>ii. Get the two characters</td>
<td></td>
</tr>
<tr>
<td></td>
<td>iii. Arrange the two characters in ascending order</td>
<td></td>
</tr>
<tr>
<td></td>
<td>iv. Display the two characters</td>
<td></td>
</tr>
</tbody>
</table>

Algorithm Design

- An algorithm is a procedure in problem solving. It involves a series of steps.
- Computer can only solve problems step by step.
- So before a computer can operate on a problem, an algorithm must be developed.
- Problems with writing algorithm using ordinary language:
  i. The steps for solving a problem are so complicated that they are difficult to be written down in essay form.
  ii. After writing the steps down, it is not easy for anyone else to understand them. ‘.’ difficult for coding.
  iii. Danger of making errors.

Pseudocode

- Pseudocode is a sequence of English-like statements that represent an algorithm.
- Features of pseudocode:
  i. There are no formal syntax rules and guidelines.
  ii. It is written in plain English.
  iii. It does not depend on the programming language which is to be used in program coding.
  iv. A pseudocode program can be easily encoded to a procedure-oriented language such as Pascal,. It is easy to change the pseudocode if there is any logical error in the solution.

Flowchart

- A pictorial representation of an algorithm. (Refer to the notes of “Simple PASCAL programming”)

Solution Development

- We use a programming language or a suitable software developing tool to communicate with the computer.
- If the algorithm developed in the previous stage is sound and the representation of the algorithm is complete, the process in coding is simple and straight forward.
Top-down approach. (p.15)

i. In case a particular sub-problem is still too difficult to tackle as a whole, we may further split it into several sub-problems. The strategy is called divide and conquer.

ii. A structure diagram may me used. It contains modules, each of which represents a problem, either a big or a small one, represented by a block in the diagram.

iii. Stepwise Refinement. The process of breaking down a problem into its separated parts; these parts may be further broken down into even smaller parts, and so on.

iv. Example,

0. Travel from the Hong Kong University to the Chinese University of Hong Kong

Step 1
1. Walk to Sheung Wan Station
2. Take the MTR to Kowloon Tong Station
3. Take a train to Chinese University Station
4. Walk to the Chinese University

Step 2
2.1 Take the MTR from Sheung Wan Station to Central Station
2.2 Change at Central Station
2.3 Take the MTR from Central Station to Mongkok Station
2.4 Change at Mongkok Station
2.5 Take the MTR from Mongkok Station to Kowloon Tong Station

figure 1

** Pre-defined process symbol ( ). It indicates that the whole process is described elsewhere.

Here is another example for stepwise refinement,

Find the volume \( V \) of a right pyramid whose base is an equilateral triangle of side \( s \) and whose height is \( h \).

i. The result of refinement is shown in fig. 2

ii. If any step still cannot be managed, it may be further refined. For example step 2.1 may be refined as shown in fig. 3.

0. Find the volume of a pyramid.

1. Input the side \( s \) of the triangular base and the height of the pyramid
2. Calculate the area \( A \) of the triangular base
3. Calculate the volume \( V \) of the pyramid
4. Output the volume \( V \) of the pyramid

2.1 Calculate the altitude \( a \) of the triangular base
2.2 Substitute the altitude \( a \) and the base \( s \) in the formula for finding the area of a triangle
2.3 Perform multiplication and division to obtain the area of the triangle
3.1 Derive the formula for finding the volume of a pyramid
3.2 Substitute the area \( A \) of the base and the height \( h \) in the formula
3.3 Perform multiplication and division

3.1.1 Dissect of triangular prism into three pyramids
3.1.2 Show that the three pyramids are equal in volume
3.1.3 Write down the formula for finding the volume of a pyramid

figure 2
2.1.1 Draw the altitude
2.1.2 Show that the altitude bisects the base of the triangle
2.1.3 Apply the Pythagoras theorem

![figure 3]

- **Bottom-up approach.**
  
i. Start from the component level, particularly if most or all components are ready. Software developers will integrate the basic modules to form a larger module.
  
ii. Top-down approach is suitable for larger problems while bottom-up approach is suitable for smaller problems.

- **Trial-and-error approach.**
  
i. There is no obvious way of systematically developing a solution.
  
ii. A proposed solution is made and a trial of it is performed to see is it is successful. If not a proposed solution is modified and another trial is performed.

**Program Debugging and Testing**

*Program Debugging (p. 10)*

- There are three main types of errors to be debugged.

  - **Syntax Errors**
    
i. Every language has its own rules of grammar called **syntax**.
    
ii. Syntax errors occur when the program statement violates one or more rules of syntax.
    
iii. e.g.  
  
  ```  
  a = a+7;  (* use '=' instead of ':=' *);  
  int1 := num1 + num2;  (* It will be wrong if num1 or num2 are declared to be real type but int1 to be integer types *)  
  ```

  - **Run-time Errors**
    
i. When a program is run and producing results that the computer cannot handle and cannot continue execution, run-time occurs.
    
ii. e.g. division by zero, accessing file not on disk, writing to a write-protected disk.

  - **Logical Errors**
    
i. These are errors made in the process of devising the algorithm or in translating the algorithm into a computer language.
    
ii. Unlike the other two types of errors, the computer cannot detect and provide clues for correcting logical chosen test data with unknown results.
    
iii. Another way is to “trace” program, using the automatic trace feature in the system using.

*Program Testing (Ch. 24)*

- **Purpose.** To guarantee that the program produces correct and meaningful results under all circumstances.
■ A **dry run** is used to examine the logic and code of a program by simulating the execution of the program on paper.

■ A **trace table** is used to list the values of the variables when we run through a program one line at a time. It gives us a better understanding of the actual performance of a program. (E.g. p.94-95)

■ The testing should include the following items,
  i. Typical valid data.
  ii. Boundary valid data. (It is a set of critical values that affect result of the logical paths.)
  iii. Special and unusual valid data, which are exceptional cases and need special processing.
    *(Note that valid data should be able to test all parts of a program (every branch / every module).)*
  iv. Invalid, incomplete or meaningless data, which are used to test the error handling abilities of the programs.

**Program Documentation** (p. 11)

■ The need of documentation
  i. To help memory.
  ii. To inform others, i.e. to communicate with other people who maintain the program.
  iii. Full details, especially details of the program’s relation to other program.

■ There are two types of documents: user manuals (for common users) and program manuals (for programmers).

■ **User manual.** A user’s guide explains to users who have no programming experience on how the program should be used.

■ **Guidelines for program manual.**
  i. **Title page.** General information such as, the program’s title, the programmer’s name and the date.
  ii. **Program description / specification.** It should be,
    a. not using technical terms (jargon)
    b. a detail specification of the task
    c. a guideline to user about the work done by the program
  iii. **Algorithm / flowchart**
  iv. **Variable dictionary.** The following table is an example of variable dictionary,

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Coefficient of $x^2$ in the quadratic equation</td>
</tr>
<tr>
<td>B</td>
<td>Coefficient of $x$</td>
</tr>
<tr>
<td>C</td>
<td>The constant term</td>
</tr>
<tr>
<td>D</td>
<td>The discriminant</td>
</tr>
<tr>
<td>X1, X2</td>
<td>The two roots of the equation</td>
</tr>
</tbody>
</table>

v. **Program listing.**

vi. **Test data and sample output.**
  a. Appropriate test data should be prepared for different situations.
  b. The expected results are determined before the program is run.

** The six phases of problem solving are not necessarily performed in the order we have described. Sometimes we may need to return to previous steps. It also should be reminded that the phases are in a cycle. **