Computer Programming– Three Stages

- Computer programming can be divided into three distinct areas of activity:
  1. Problem definition and problem solving
  2. Creating a structured solution (or algorithm)
  3. Coding (e.g. C, Java, C++)
- Consider an example as case of customer while marketing
- Specify the problem
- Solve the problem
- Specify solution in structured format

Problem Solving, Algorithm Development and Coding

- In problem solving, algorithm development and coding we tend to view problems at different levels of abstraction.
  - **Problem solving** is done at a high level of abstraction. Much of the detail is omitted at this stage in order that the problem can be more easily specified and solved.
  - **Algorithm development** works at a lower level of abstraction than problem solving. It contains much of the detail that will eventually become the program. However, it still omits the finer detail required in the final compute program
  - **Coding** works at a very low level of abstraction. Programming code must be written with every single detail a computer needs to carry out a task included.

Contd...

- Problem solving, algorithm development and coding are also different in terms of the language that is used to complete the activity:
  - Problem solving is generally done using **use natural, everyday language**
  - Algorithm development is done using a specialist, semi-structured language (**pseudo code**)
  - Coding is done using a highly-structured (**programming**) language that is readable by machines

What is a Problem?

- There are many different kinds of problems:
  - How do I get to Oxford from London?
  - Why is it wrong to lie?
- For a problem to be solvable through the use of a computer, it must generally:
  - Be technical in nature with **objective** answers that can be arrived at using **rational** methods
  - Be **well-structured**
  - Contain **sufficient information** for a solution to be found
  - Contain little, if any, **ambiguity**.

How do We Solve Problems?

- We need to THINK!
- We need to engage in one or more of the following types of thinking:
  - **Logical reasoning**
  - **Mathematical reasoning**
  - **Lateral thinking**
- Problem solving is easier if we employ a **problem solving framework** and an appropriate **problem solving strategy to** aid us.

A Problem Solving Framework

- An outline framework for problem solving:
  1. Understand the problem
  2. Devise a plan to solve the problem
  3. Carry out the plan
  4. Assess the result
  5. Describe what has been learned from the process
  6. Document the solution.

- In Vickers, this framework is called the **How to Think Like a Programmer** (HTTLAP) approach to problem solving.

Understanding the Problem

- To understand a problem
  - We need to read and reread it till we understand every detail
  - We need to dissect the problem into its component parts (e.g. problems and **sub-problems**)
  - We need to remove any **ambiguity**
  - We need to remove any information that is **extraneous** to the problem
  - We need to determine our **knowns** and our **unknowns**
  - We need to be aware of any **assumptions** we are making.

Devise a plan to the solve the problem

- If a problem contains a set of sub-problems, in what order are you going to solve them?
- How are you going to represent the problem:
  - Numerically?
  - Graphically?
  - Tabular data?
  - Natural language?
- Does the problem lend itself to a particular problem solving strategy or strategies:
  - Working backwards?
  - Logical reasoning?
  - Finding a pattern?
  - Accounting for all possibilities?

Carry Out the Plan

- Consider the following problem:

  In a room with ten people, everyone shakes hands with everyone else exactly once. In total, how many handshakes are there?

- How would you represent and solve the problem?
- Different strategies to be used

Assessing the Results

- It is very unusual when solving complex problems to achieve the correct result first time round.
- To verify our solutions are correct, we need to take a few steps backwards:
  - Was our understanding of the problem correct?
  - Did we overlook anything?
  - Did we choose the correct strategy?
  - Did we employ that strategy correctly?
  - Have we made any incorrect or unwitting assumptions?
- However, it is often very difficult to spot our own mistakes. It is often better, therefore, to have somebody else verify our solutions for us.

Contd...

- Sometimes solutions appear correct, but are in fact wrong, due to an initial misunderstanding of a problem (What Vickers calls, **errors of the third kind**).
- If you have misunderstood a problem, it does not matter how good a coder you are, your program will not work as it is supposed to.
- Therefore getting the problem-solving part of programming right is absolutely essential if we are to build programs that work as they are supposed to work.

Describing What you have Learned

- You can only become a good problem solver by reflecting on your experiences of problem solving.
- Keeping a record of problems you have attempted, your success, failures, the approaches you have used, etc. will:
  - Broaden your problem solving repertoire
  - Help you to recognize similarities/patterns in problems
  - Help you identify and fix logical or implementation errors
  - Help you to solve problems faster and more effectively

Documenting the Solution

- Documenting a solution will help you to **generalize** your approach to other similar problems.
- Do you recognize the following type of problem? How would you solve it?

Carlos and his friends have a fantasy football league in which each team will play each other three times. The teams are Medellin, Cali, Antigua, Leon, Juarez, Quito and Lima. How many games will be played in all?

- It will also prevent you forgetting how you arrived at your solutions.
- In the long run, it will make you a better problem solver and eventually a better programmer.

Computer Problem-Solving

Algorithm Development Phase

Analyze	Understand (define) the problem.
Propose algorithm	Develop a logical sequence of steps to be used to solve the problem.
Test algorithm	Follow the steps as outlined to see if the solution truly solves the problem.

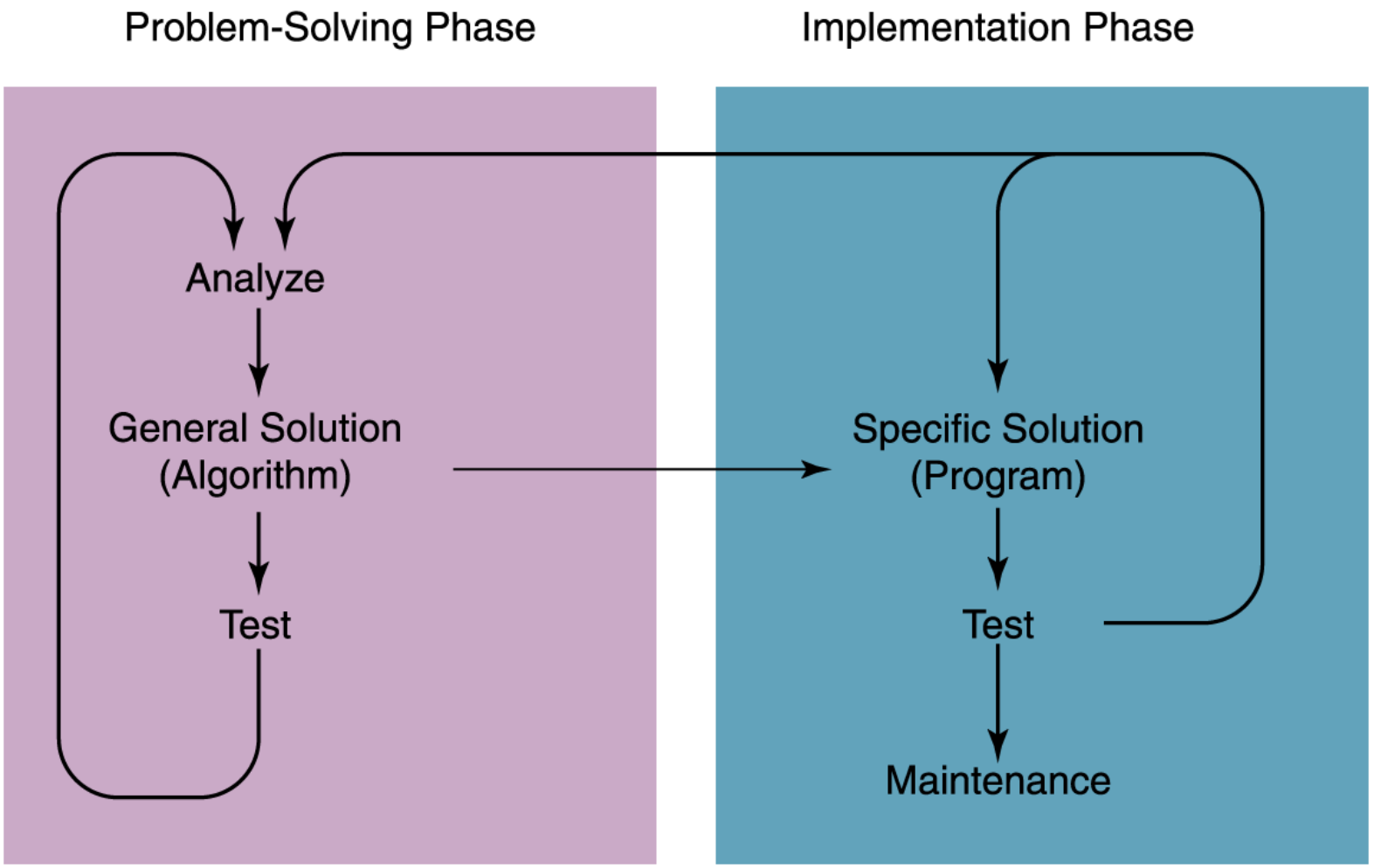
Implementation Phase

Code	Translate the algorithm (the general solution) into a programming language.
Test	Have the computer follow the instructions. Check the results and make corrections until the answers are correct.

Maintenance Phase

Use	Use the program.
Maintain	Modify the program to meet chaining requirements or to correct any errors.

The Interactions Between Problem-Solving Phases



Conveying solution in a formal language

Purpose of program planning

- To write a correct program, programmer must write each and every instruction in the correct sequence.
- Logic (instruction sequence)of the program can be very complex.
- Hence program must be planned before they are written to ensure program instructions are:
- Appropriate for the problem
- In the correct sequence