5 Introduction to Problem Solving

In This Chapter

5.1 INTRODUCTION

There is a famous quote by Steve Jobs, which says, "*Everyone in this country should learn to program a computer, because it teaches you to think*".

This quote itself is proof-enough to say that in order to program a solution for a problem, you should think in a specific way. And this is what we are going to talk about in this chapter.

In this chapter, you will learn about problem solving, *i.e.*, analysing a problem, designing algorithms using tools like flowcharts and pseudocode and problem solving using decomposition. So, let us begin.

5.2 PROBLEM SOLVING CYCLE

Programs are not quick creations. In order to create efficient and effective programs, you should adopt a proper problem solving methodology and use appropriate techniques. In fact, problem solving methods follow a cycle – the *problem solving cycle*. In the coming lines, we are going to discuss the same.

Broadly problem solving requires *four* main steps :

1. Identify and analyse the problem.
2. Find its solution and Develop algorithm of the solution.
3. Code the solution in a programming language.
4. Test and Debug the coded solution.

And finally **implement and maintain** it.

The above mentioned steps are four major steps in a problem solving cycle. Each step contains many sub-steps. Let us talk about these sub-steps in order to understand the problem solving cycle. The aim of a problem solving cycle is to create a working program for the solution.

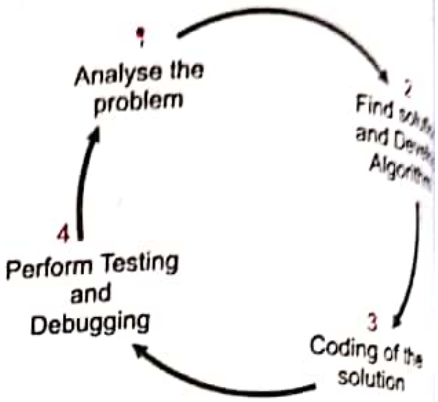


Figure 5.1 Steps in Problem solving cycle

The sub-steps of a problem solving cycle to create a working program are :

Analyze the problem	1.	*Understand the problem well*	It is very important to understand the problem minutely because sometimes the problem is not that appears but something else that is causing it.
	2.	*Analyze the problem*	Problem analysis is very important as the outcome of this will convert to the success of final solution. While analyzing, ■ identify processing components ■ identify the relationships among processing components.
Developing the Algorithm	3.	*Think of possible solutions*	As per the problem analysis, think of possible solutions by trying this ■ think of different ideas for solutions ■ check your ideas for aptness ■ finally, zero on most appropriate solution.
	4.	*Follow Modular approach while designing most appropriate solution*	Many small logically related modules or functions must be preferred over a big problem. It is called *Modular approach* wherein we divide a big program into many smaller and more manageable and understandable modules. Design the final program by ■ deciding step by step solution ■ breaking down solution into simple steps.
	5.	*Identify operations for solution*	Program coding involves identification of constituent operations required to get the desired output. One must decide about the minimum but simple operations required. Identify : ■ minimum number of inputs required ■ arithmetic and logical operations required for solutions ■ simple and efficient data structures suiting the need.

Coding	6.	*Code program using appropriate control structures*	The next step is to code the program as per finding of previous step. Coding is the technical word for writing the program. This step is to translate the algorithm into a programming language. You should use most appropriate control structures out of available options. Thus, it is important to know the working of different control structures and their suitability in different situations. ■ Use appropriate control structures such as conditional or looping control structures. ■ Think program's efficiency in terms of speed, performance and effectiveness.
Testing and Debugging	7.	*Test and Debug your program*	Testing is the process of finding errors in a program and debugging is the process of correcting errors found during the testing process. Thus, this phase involves : ■ finding errors in it. ■ rectifying the errors.
	8.	*Complete your documentation*	Documentation is intended to allow another person or the programmer at later date, to understand the program. Documentation might also consist of a detailed description of what the program does and how to use the program.
Implement and Maintain	9.	*Implement your code*	After testing and documentation, implement your program for actual use on site. Now, the real users can use your programs.
	10.	*Maintain your program*	Maintaining programs involves modifying the programs to remove previously undetected errors, to enhance the program with different features or functionality, or keep the program up-to-date as government regulations or company policies change.

5.2.1 Problem Solving using Decomposition

The previous section talked about problem solving cycle. The first five steps of problem solving cycle are very important as rest of the steps are based on them.

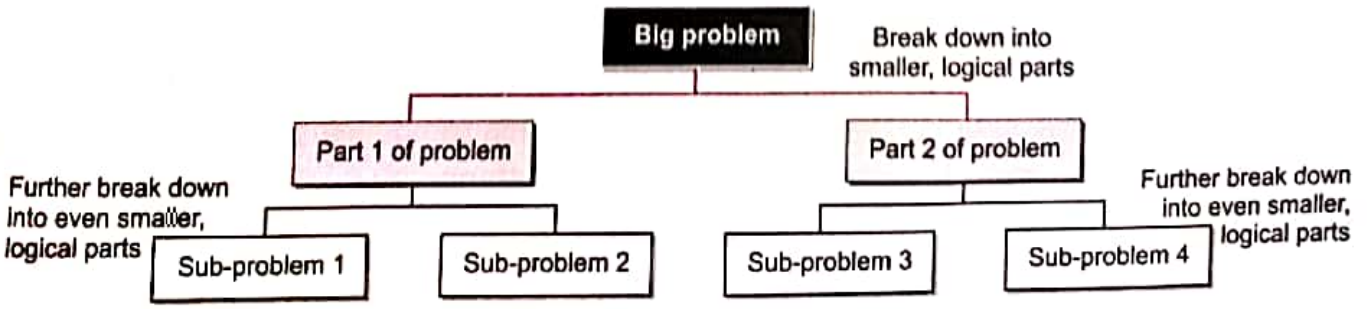
The first five steps involve understanding the problem, analyzing it, and thinking of possible solution with sub-module and operations in it. All this is effectively carried out using decomposition. Let us know what decomposition means.

DECOMPOSITION

The process of breaking down a big or complex problem into a set of smaller sub-processes in order to understand a problem or situation better, is known as **decomposition**.

Decomposition is the process of breaking down a big or complex problem into a set of smaller sub-processes to allow us to describe, understand, or execute the problem better. Decomposition involves :

- Dividing a task into a sequence of subtasks.
- Identifying elements or parts of a complex system.



Consider some examples of decomposition :

- **Everyday example.** Making cookies is a complex task that can be broken down into smaller, simpler tasks such as mixing up the dough, forming into shapes via cookie cutters, and baking.
- **Academic example.** Writing an essay is a complex task that can be broken down into smaller tasks such as developing a thesis, gathering evidence, and creating a bibliography page.
- **Engineering example.** Designing a solution to construct a bridge by considering site conditions, technology available, technical capability of the contractor, foundation, etc.
- **Computer Science example.** Writing a computer program/software by determining a well-defined series of smaller steps (mostly in the form of modules and functions) to solve the problem or achieve a desired outcome.

EXAMPLE 1 *Decompose the task of creating mobile app.*

SOLUTION To decompose the task of creating a mobile app, we would need to know the answer to a series of smaller problems :

- what kind of app is to be created.
- who the target audience for the app is.
- what the user interface of the app will be (what all screens, type of input etc.).
- what the app's graphics will look like.
- what audio will be included.
- what software/ platform will be used to build the app.
- how the user will navigate the app.
- what additional services will be required for the app, *e.g.*, database etc.
- how the app will be tested.

This list has broken down the complex problem of creating an app into much simpler problems that can now be worked out.

leed for Decomposition

Decomposition is the process of breaking a large problem into more manageable sub-problems. It is very important to decompose a problem into smaller sub-problems, reason being that large problems are disproportionately harder to solve than smaller problems. It's much easier to write two 500-line programs than a single 1000-line program. Larger a problem is, harder and more difficult it is to program as compared to a smaller problem (*see figure*).

Once you know how you can decompose a problem in smaller steps, you can create its solution by designing algorithms for it.

NOTE

Decomposing a problem into smaller sub-problems is important because large problems are disproportionately harder to solve than smaller problems.