Comparative Study of Computer Languages

PROBLEM ORGANISATION

Part II

R. Ramaswamy

Any problem which is to be computerised must be properly analysed and organised. Computer time is very precious. One cannot afford to waste the computer time by experimenting on the methods of solving the problem. In fact, the computer user has to spend more time in his study room in preparing the final problem, than he actually spends with the computer to get the calculations done.

Any computer problem must be carefully analysed in advance. All the steps that have to be followed must be carefully set forth in the correct sequence before preparing the instructions for the computer. The computer user must have a thorough grasp of the problem, a clear idea of the available input data and the expected output, as well as the possible alternative processes to be followed under different conditions. Aimless wandering will be ruinous to both the user and the computer. In this article we will study how a problem can be analysed and organised with the help of algorithms and flowcharts.

Problem analysis

Problem analysis consists in finding or deriving the procedural steps involved in getting the solution to a problem. The procedural steps must come within the realm of computer capabilities. It has already been pointed out that the computer can follow only elementary arithmetic logic instructions and so it is the user's job to convert every complex problem into procedural steps containing only elementary arithmetic logic operations.

The famous systems philosophy says that any complex situation is capable of being resolved into a large number of simpler ones and by solving all the simpler ones and integrating their solutions, the complex ones get solved. This philosophy is based on the simple truth that no problem is born big in this world but only grows just like a baby grows into a man or a woman.

How can one go about discovering the elementary steps involved in getting the solution to a complex problem? Unfortunately there is no set method for doing this, but one has to use one's robust common sense and ingenuity to arrive at the procedure. Let us consider a small example to see how to dissect the problem of finding the square root of a number.

Analysing square root problem

Suppose we are asked to find the procedure for finding the square root of a number. To the best of our knowledge, there is no single formula which gives the square root of a number. Obviously, the method involves the use of more than one formula. Let us now ask, what do we look for when we try to get the square root of a number.

In this problem of finding the square root of a number, say A, we look for some number S (called the divisor), which when divided into the given number A (called the dividend), gives a quotient Q equal to the divisor. This wordy description can be written in the mathematical form as

$$Q = A/S$$

If S is equal to Q, then S is the square root of A.

Let us now start the game by making a guess for the square root, say S1. To check whether S1 is the square root, divide A by S1 to get the quotient Q1. Then

$$Q1 = A/S1$$

Now check whether Q1 is equal to S1. If so, it was lucky, since the first guess was the correct square root. If not, we have to make a second guess. This time, the guess should be based on some logic.

Common sense tells us that the next best guess will be the average of S1 and Q1. So the second guess S2 will be written as

$$S2 = (S1 + Q1)/2$$

$$\text{or } S2 = (S1 + A/S1)/2$$

Now check whether S2 is the correct square root by comparing S2 with the quotient Q2 obtained by dividing A by S2. Q2 can be calculated using the formula

$Q2 = A/S2$

If Q2 is equal to S2 we have got the answer, otherwise we continue the same process by making the third guess as the average of S2 and Q2. In following this process it will be found that the values of S2 and Q2 approach towards each other and will have to merge sooner or later giving the much needed solution to the problem.

Now, the generalised relation between the ith guess and the (i+1)th guess can be written as follows

$S_{i+1} = (S_i + A/S_i)/2$

By a successive application of the above formula, the generated sequence of numbers will come closer and closer to the correct value, finally arriving at the correct value. While man may show signs of tiredness in this method, computers do not show any exhaustion in tackling the problem in this way. Thus, this method is prescribed for the machine, since it is in formula form.

The generalised formula given above is called the algorithm for the square root problem. When this formula is repeatedly applied, until the values of two successive guesses become equal, we get the solution. Readers can test this algorithm by taking the first guess for the square root of 4 as 1. The successive guesses come out as 1, 2.5, 2.05, 2.001 and so on.

Where do we actually stop this process of successive approximation or iteration as it is called? Theoretically, we can stop when the values of two successive guesses become equal. Very often it may not come about because of the limited accuracy of the machine. So it can be decided to stop when the absolute difference between two successive guesses becomes less than or equal to a pre-assigned small value, say .00001 or any other small value depending on the accuracy required. The main point is that we have dissected the problem of finding the square root of a number into a series of elementary steps which can be executed by the computer. That is, we have found the algorithm for the square root problem.

For the square root problem, the algorithm was found fairly comfortably by the so-called intution method. In fact, the above algorithm can be mathematically derived by using the Newton-Raphson method of successive approximation. Mathematical logic is thus nothing but a common sense approach to problems.

One can use specific mathematical techniques available for finding algorithms for various types of problems and one need not always depend on intuition as in the above example. Except in the case of intricate scientific problems, finding the algorithm will prove to be fairly simple.

Flowcharting

Computer users would like to represent the algorithmic steps in a more appealing way, called the pictorial way. The pictorial representation of an algorithm is called the flowchart. Flowcharts are very useful for coding the algorithmic steps into a computer program. Flowcharting is another way of organising a problem for computer implementation. This approach is consistent with the notion that a picture is worth more than thousand words. A flowchart represents the sequence of the different operations to be followed for solving a problem.

A flowchart consists of a number of boxes with certain geometrical shapes joined together by means of flowlines marked with arrows, The direction of the flowlines marked by arrows indicates the flow of logic. Each box represents a specific function and the instructions to perform the program function will be written inside the box in ordinary English language for easy understanding.

A flowchart helps the systematic analysis and organisation of a problem. It also provides a means of experimenting with the various approaches to the problem by a simple re-arrangement of boxes and arrives at the best choice suited to the situation.

A flowchart is a tool to assist the programmer in preparing the program. Its use by the programmer is analogous to the use of a blueprint by an architect. Just as an architect draws a blueprint before a building is constructed, similarly a programmer draws a flowchart before a program is written. A blueprint illustrates the arrangement of the different units to be constructed in a building and their relationship to one another. So also a program flowchart is a drawing which shows the arrangement of the different functions to be performed and their logical interrelationship.

A program flowchart is an essential and integral part of the documentation of the program. By documentation it is meant those explanations or descriptions (pictorial or wordy) which are needed for a complete understanding of a problem by a programmer as well as the user.

It has been pointed out that a flowchart consists of different boxes which represent different program functions connected by flowlines depicting the sequence in which these functions are to be performed. The common boxes used for the different program functions are shown in Fig. 1. Let us now consider four examples for flowcharting and the solutions to these problems.

Flowchart illustration 1

It is better to organise one's thoughts and actions whether one is a computer user or not. In this connection, flowcharting is a good exercise for sharpening one's thinking faculties For example, let us consider the problem of a person, who wants to get a meal he likes at a price he can afford. The orderly steps he has to adopt in achieving his objective is shown as a flowchart in Fig. 2.

The flowchart gives the different steps he has to follow and also the sequence in which these steps have to be followed. Obviously he cannot order for the dish and eat before he enters the hotel and sits comfortably. From the flowchart we find that there are three decision making situations represented by the diamond boxes, when he has to think for

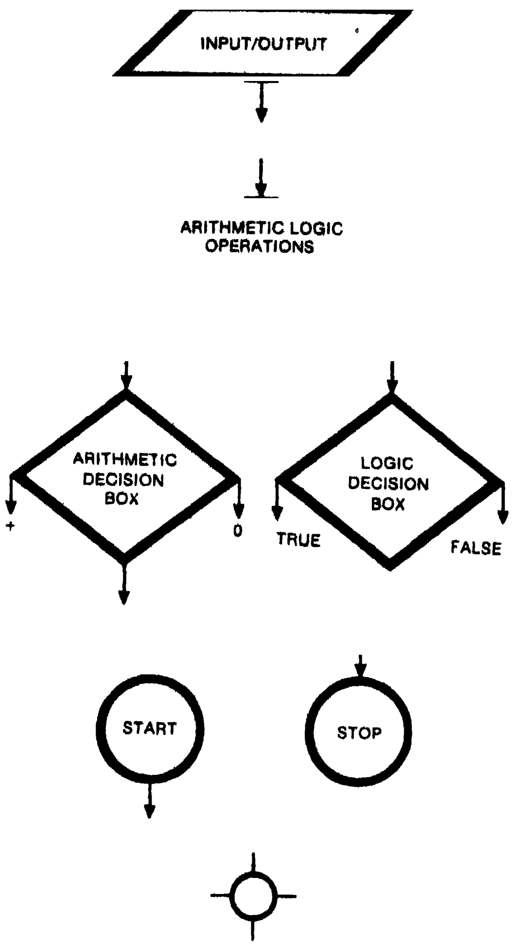

Fig. 1: Common boxes used for different program functions.

a moment and branch off along any one of the alternate paths.

Another important point we have to note is, that one end of the decision box gives instructions to go to another point and repeat the operations over again. This is called looping. Looping is an important technique used in computer programs to perform repetitive type of operations.

Flowchart illustration 2

Three positive numbers A, B and C are given. It is required to flowchart the procedure for locating and printing the smallest of the three numbers.

The normal procedure is to first compare A and B. If A is less than B, then A and C are compared and the smaller of the two is declared as the smallest of the three numbers. If A is greater than B, then B and C are compared and the smaller of the two is declared as the smallest of the three numbers.

The above wordy description of the procedure is very cumbersome and it will be more so, if there is a large number of data to be compared and processed. The procedure for finding the smallest of three numbers is illustrated in the flowchart shown in Fig. 3.

The flowchart tells the procedure to be adopted to achieve this task. First, we read the three numbers. This is called as reading the input data. Then to compare A and B for finding the smaller number, we put the comparison operation in the decision box. If the answer is yes, we go out via the left exit, otherwise we go out via the right exit. There are two more decision boxes, one at the right and another at the left, where the comparisons of B with C and A with C respectively are made. The result is any one of the possible four alternatives, which may be outputted. Fig. 3 is self-explanatory.

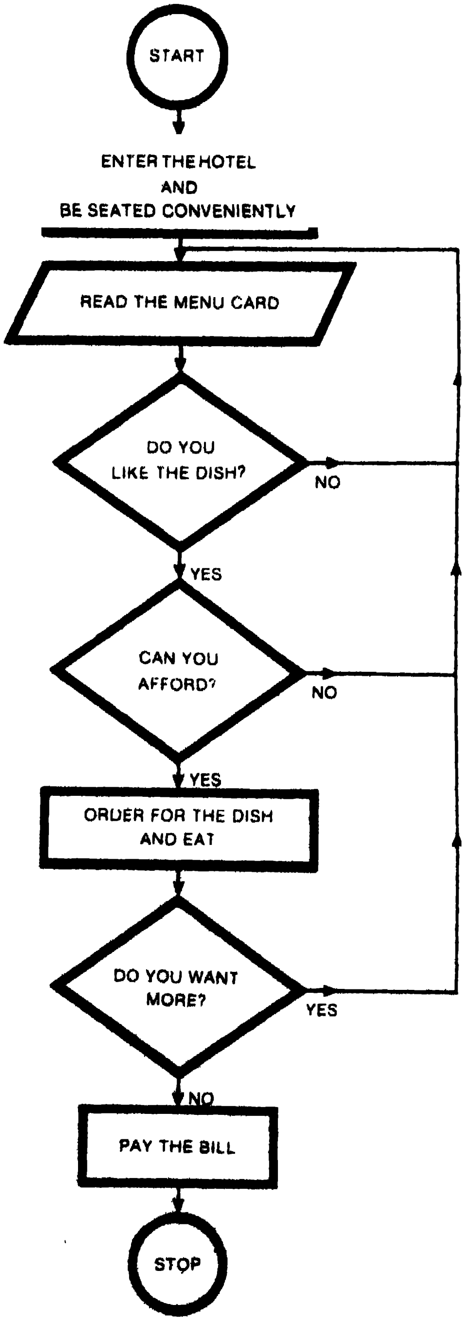

Fig. 2: Schematic diagram for flowchart illustration 1.

Flowchart illustration 3

There are 500 employees in a firm. Draw a flowchart to output the net wage payable to each employee, given the following particulars:

1. The basic pay and the deductions for each employee.

2. The allowances are calculated as follows: (i) HRA is calculated at 23% of the basic, (ii) DA is calculated as follows: DA is Rs 40 for pay up to 299, Rs 50 for pay between Rs 300 and Rs 999 and Rs 60 for pay above Rs 999.

In this problem we have to output the results for exactly 500 employees. So we have to make some provision for counting. This is done by setting up a counter which is automatically incremented by 1 as soon as it processes and prints the result for one employee. We initialise a counter COUNT equal to 0 in the beginning. After printing the result for one employee, we write COUNT = COUNT + 1. This means that the old value of COUNT is incremented by 1 and stored in the same location COUNT.

Then, every time an employee data is processed, we check whether COUNT is still less than 500. If it is so, we go on to read the data of the next employee, otherwise we come out of the loop and stop the operations. The above steps are illustrated in Fig. 4.

Flowchart illustration 4

500 students take a degree examination consisting of six papers. Draw a flowchart to find the average mark obtained by each candidate and declare the result on the basis of the average mark as follows:

1. Declare pass with honours if the average is 75% and above.

2. Declare pass in I class if the average is below 75% but 65% and above.

3. Declare II class if the average is below 65% but 50% and above.

4. Declare III class if the average is below 50% but 40% and above.

5. Declare FAIL if the average is below 40%.

In this problem we set up two counters, one for counting the passes and another for counting the total students. After processing each student we check whether the COUNT is

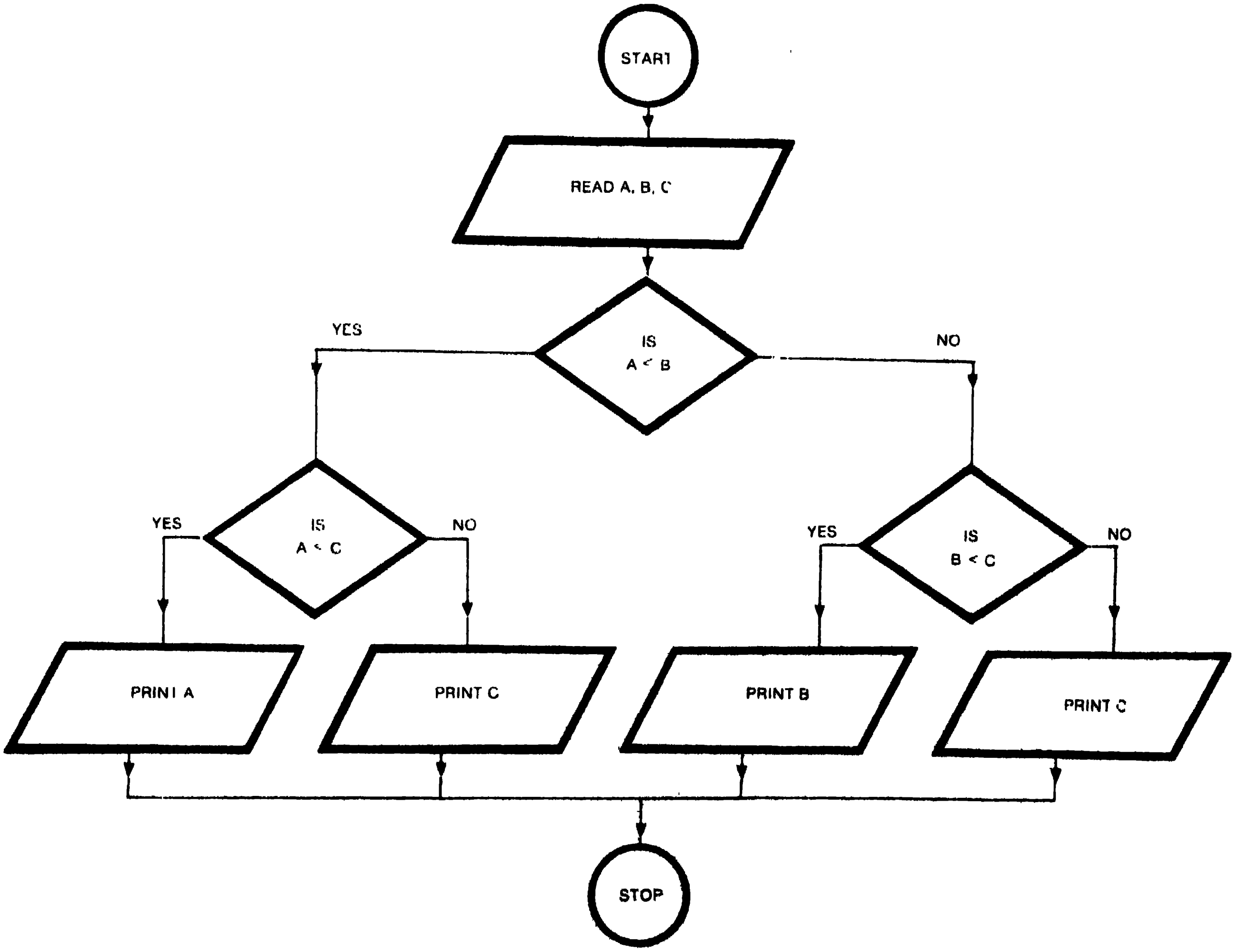

Fig. 3: Schematic representation of flowchart illustration 2.

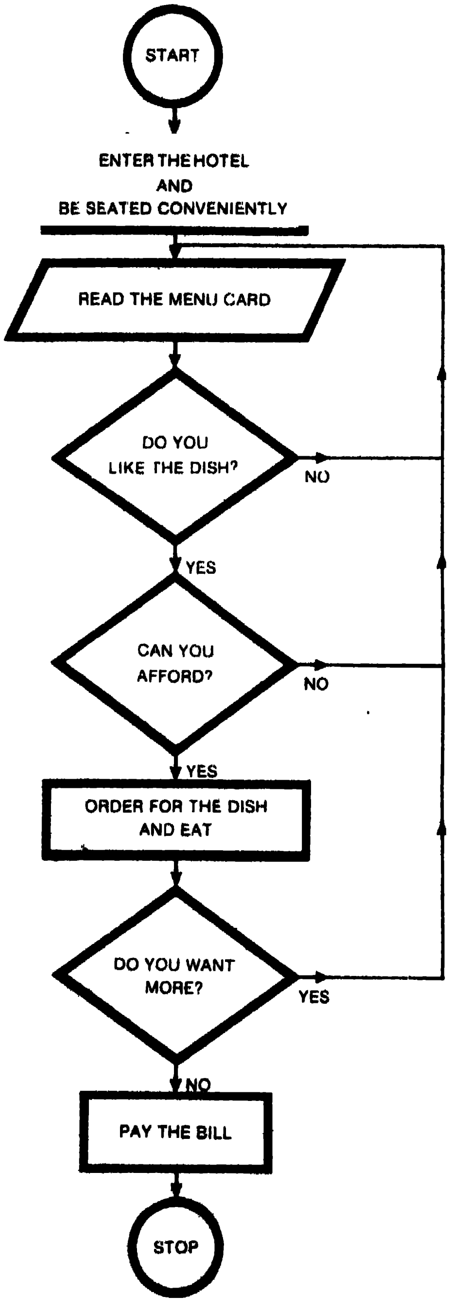

Fig. 4: Schematics of flowchart illustration 3.

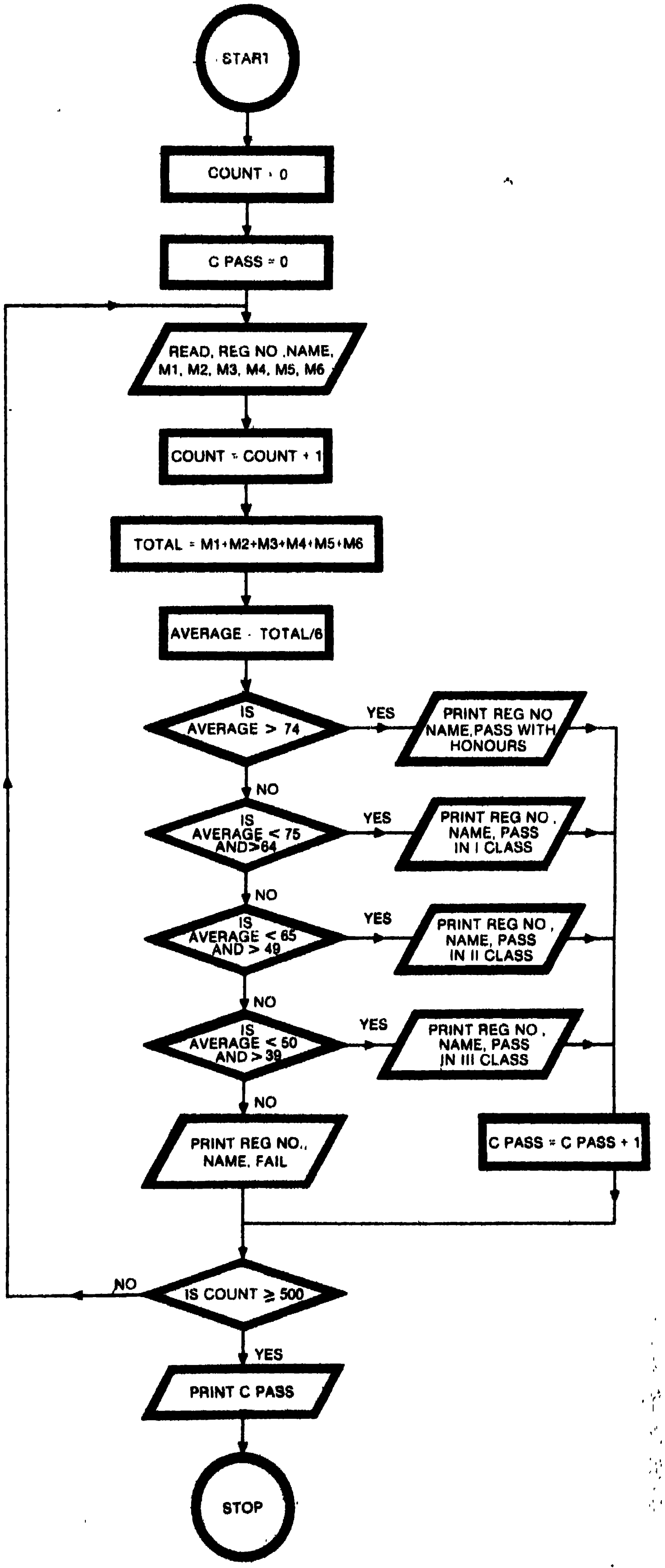

Fig. 5: Schematic of flowchart illustration 4.

less than 500. If it is so, we go to process the next student, otherwise we stop the operations after outputting the count of successful students. Carefully follow the steps shown in Fig. 5 and see how the objective is achieved.

Summary

Deriving the algorithm is the first and the foremost step before preparing any problem for computerisation. This process involves the splitting of the problem into smaller ones and finding the procedure for solving the smaller ones. This is nothing but a commonsense approach for solving complex problems. This is the famous 'systems approach' for solving problems which is based on the philosophy: 'when the foe is formidable, divide and conquer'.

Next, one must try to represent the different steps pictorially by what are called flowcharts: Flowcharting is an important technique used by programmers to analyse and depict the logic of a computer problem. Its importance cannot be over emphasised. Very many complex logic can be easily resolved if one tries to analyse it with the help of flowcharts. Writing computer programs for problems will become very comfortable if one has prepared the correct flowchart.

Algorithmising and flowcharting are the two important ways of analysing and organising a problem for computer implementation.

(To be continued)