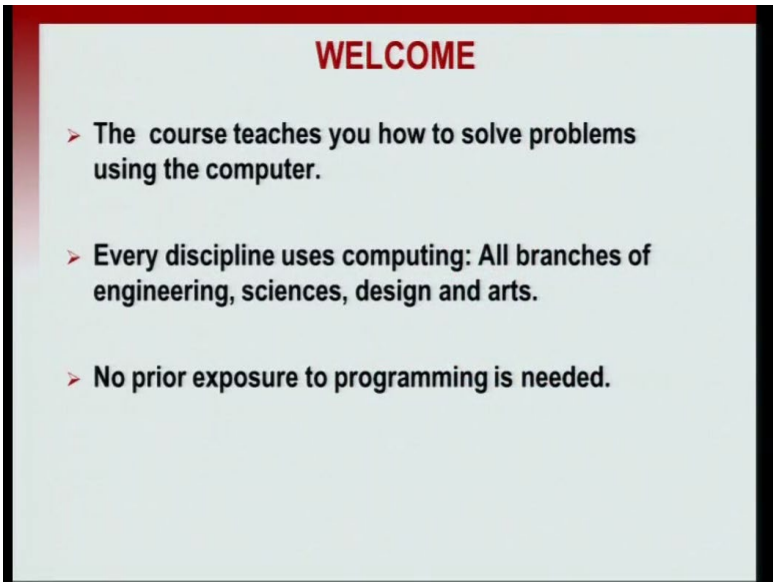
Introduction to Programming in C
Department of Computer Science and Engineering

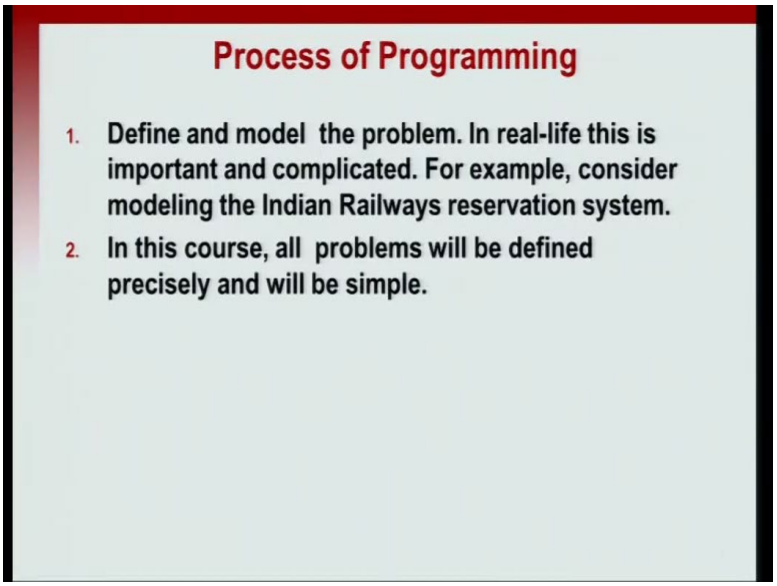
Welcome to the introductory programming course on NPTEL MOOCs. The goal of this is to learn how to code basic programs in the C programming language.

(Refer Slide Time: 00:18)



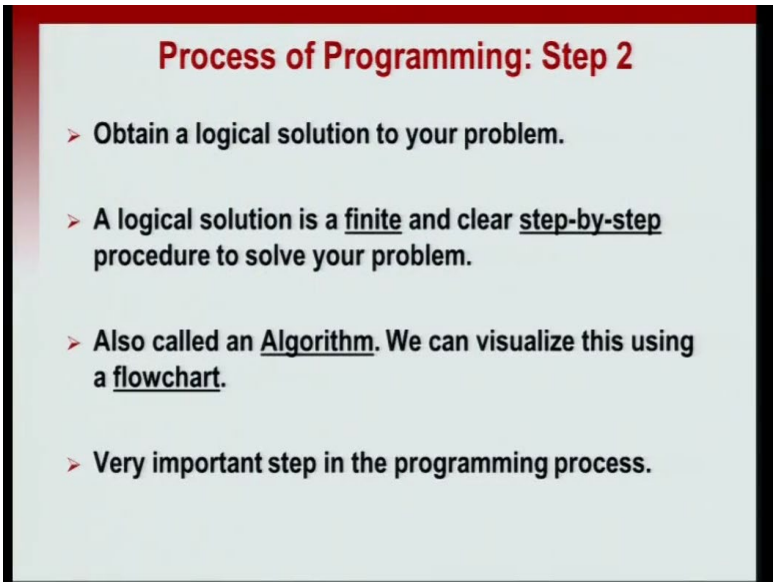
Basically the aim of this course is to teach you how to solve problems using a computer. And by the end of this course, we will hope that you can write medium-sized programs – maybe running to a couple of 100 lines of code comfortably in the C programming language. Programming nowadays is considered a basic skill similar to mathematics that is needed across all disciplines like engineering, in the sciences, and nowadays even in the arts. So, little bit of programming skill is an enhancement to any other skillset that you might already you have. This course we will start from the ground up; we do not assume any prior experience in programming whether in C or in any other language. So, the focus will be to start from the basics; and to use C as a medium of program.

(Refer Slide Time: 01:21)



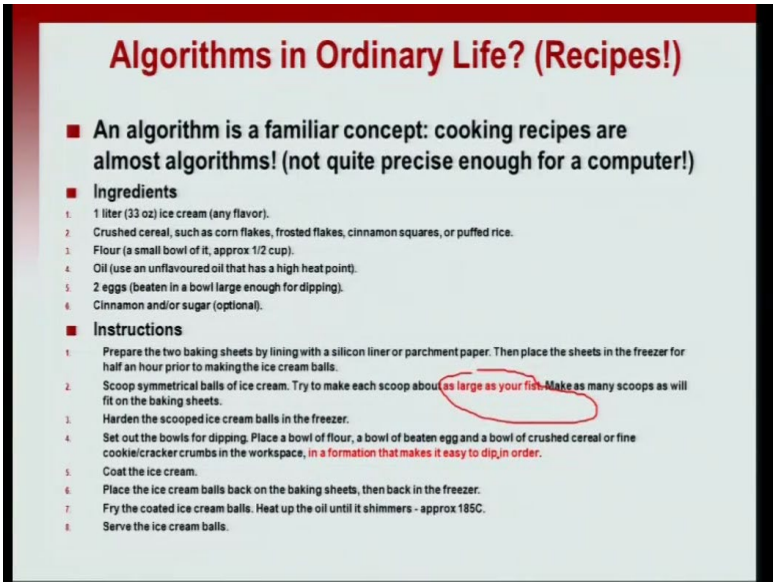
A couple of words about the process of programming; it involves two basic steps. One is to define the problem; often you get real-world problems, which are not precise enough to write a program for. So, the first step would be to define and model the problem. And this is a very important step in large scale software development; however we will not focus on this as part of this course. During this course, you will not write large software system like the Indian railways reservation system; those are extremely complex problems involving multiple programmers. In this course, we will assume that the problem is well-defined and already provided to you. So, they will be precise and they will be fairly short and simple. So, this is the first step of programming, which is definition of the problem, which you can assume will be given.

(Refer Slide Time: 02:22)



Now, comes the second step, which is to obtain logical solution to your problem. And what do we mean by a logical solution? A logical solution is a finite sequence of steps; do this first, do this next; if a certain condition is true do this; otherwise, do something else. This is called an algorithm. So, an algorithm is basically a finite step-by-step procedure to solve a problem. One way to visualize an algorithm is using a flowchart. If you are new to programming, it is recommended that, you draw flowcharts to define the solution to your problem. Experienced programmers very rarely draw flowcharts, but that is not a reason for beginning programmers to avoid flowcharts.

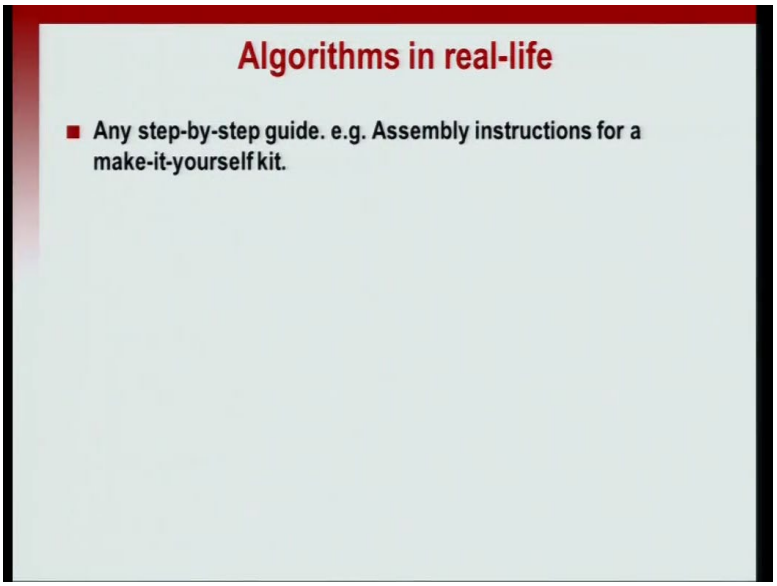
(Refer Slide Time: 03:29)



So, defining a problem is there; then the process of coming up with an algorithm. This is a very important step in the programming process. And followed by this, there is a third step, which is to implement the algorithm in a usual programming language. So, is the concept of an algorithm a new concept? I would claim that, it is not. An algorithm is a very familiar concept; the most important example that you can think of are cooking recipes. Now, cooking recipes are written in a way that, they are almost algorithms. They are not quite precise enough for a computer, but they come quite close. For example, let us take an unnamed dish – a desert and let us look at how things specified in a recipe. And we will see that, this analogy is quite deep. There is a very strong similarity in the way that recipe is written and a program is written. So, usually, they will have a list of ingredients upfront. For example, you have ice cream, crushed cereal and so on. And then once you have all the ingredients in place, then you have instructions to say how do you start and how do you end up with the dish.

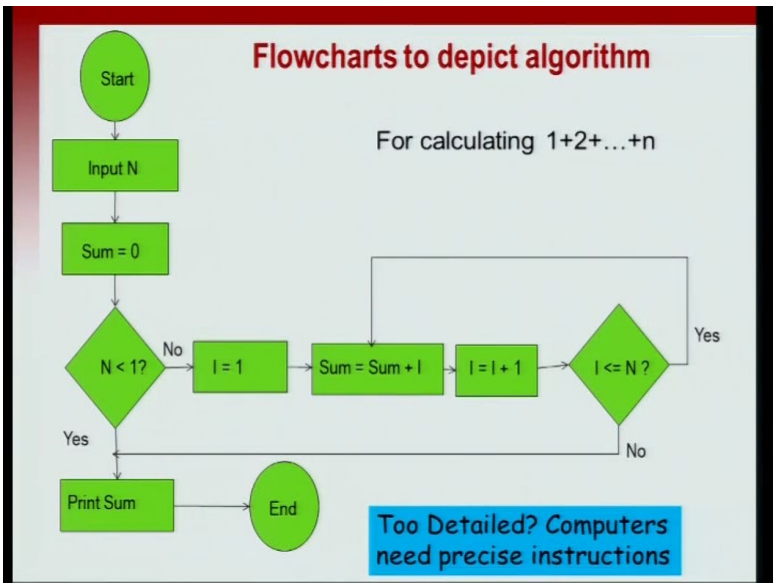
Now, those instructions will be fairly precise; of course, you assume that, the person preparing the dish is a fairly-experienced cook, so that certain instructions need not be given in very precise detail. For example, you can say do this, heat oil and so on. And it is assumed that, a person knows how to heat oil. Even so you will see that, certain recipes are fairly vague and other recipes are fairly detailed. And in any recipe, you can see certain things, which are vague and will cause confusion to most people. For example, here is a term, which says try to make each scoop about as large as your fist. Now, that of course, is a vague term, because my fist could be a different size than yours. And then you will see that, in a formation that makes it easy do dip in order. So, this is fairly vague and it is not very helpful to a cook, who is making this for the first time. So, think of algorithms as similar to recipes, but mentioned in a more precise manner.

(Refer Slide Time: 06:12)



Another way you can be familiar with algorithms is when you have the – when you buy a make it yourself kit for a furniture or something like that; and you will be provided with a step-by-step instructions on how to assemble the kit. Often when you buy disassembled table or something like that, it will come with a sheet telling you how to start with the components and build a table. Those are also similar to an algorithm.

(Refer Slide Time: 06:41)

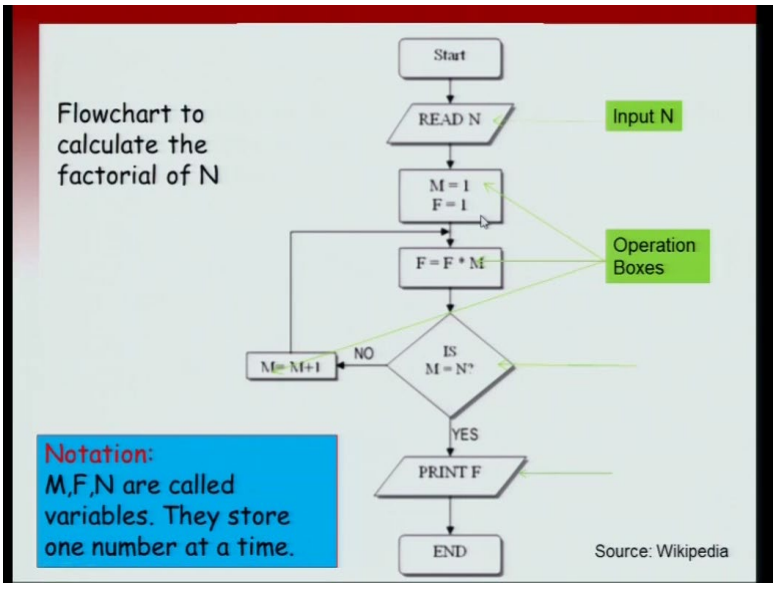


So, let us look at a flowchart to depict a mathematical algorithm and we will use this flowchart to explain certain conventions about how algorithms can be described. So,

every flowchart will have a start and an end; and it will have a finite number of boxes. So, this is the finite number of instructions that I was talking about. There are certain conventions in drawing flowcharts; the start and the end are often described in circles. Then there are ordinary boxes and then there are diamonds. We will shortly describe what they mean. So, suppose you want to write an algorithm for adding the first n numbers; all of you know how to do it. The point is how do you describe this step-by-step to somebody who does not know it already.

So, first you have to take what is the upper limit N and then you have to sum them up. So, one way to sum them up is start with an initial sum of 0 and then add numbers one by one. So, increment a counter from 1 all the way up to n. So, you start with I equal to 1 and then add the I-th number to the sum; and then increment I; if I is already N, then you are done; if I is not N, then you go back and do the sum all over again until you hit an I. When you reach I equal to N, you come out the program; print the sum; and end the program. So, this is a very simple flowchart. So, initially, if N is less than 1, you have nothing to do; if N is greater than 1, you start a counter from I equal to 1 to n and add the numbers one by one until you hit the N-th number.

(Refer Slide Time: 08:56)



If you wanted to compute a slightly different problem, which is let us say the factorial of N, which is just a product of the first N numbers, the flowchart will look fairly similar; the only difference is that instead of adding numbers, you will multiply them. So, this

flowchart is similar to the previous flowchart; you will first input in N, and then increment N until you hit N equal to M. If so you will finally, print the factorial; otherwise, you go back to the loop. So, here are the conventions used. The start symbol is often ((Refer Time: 09:45)) as a circle or an oval; the input symbol and the output symbol are often represented as parallelograms; and the normal operation boxes are represented as rectangles; and the test box to see whether you have hit a limit to test some condition in general, they are represented as diamonds.