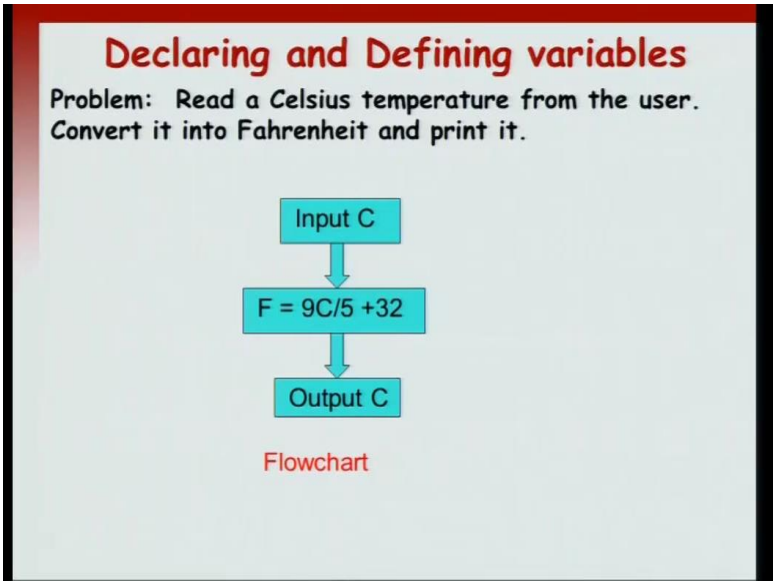
Introduction to Programming in C
Department of Computer Science and Engineering

(Refer Slide Time: 00:13)



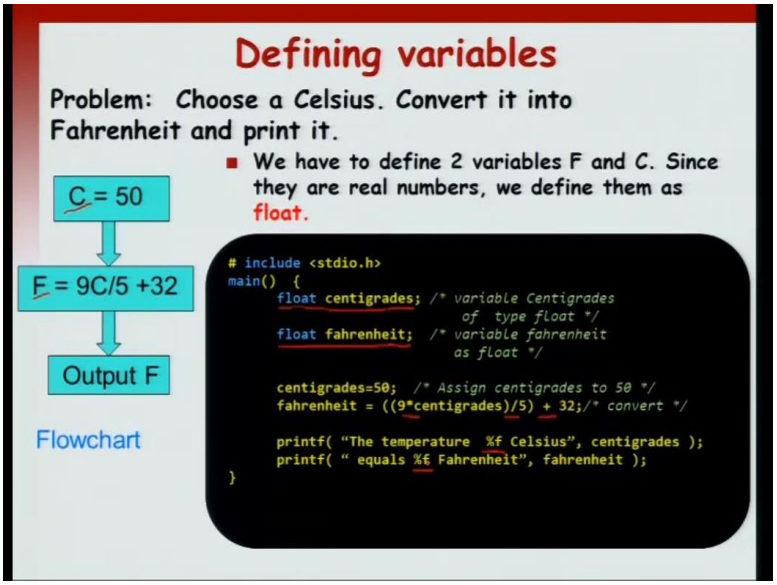
In this session we will see slightly more sophisticated programs. Recall that in our discussion about flowcharts, we talked about variables, which were conceptually seen as little boxes in which you can hold values. So, let us see how to write simple C programs in which we make use of variables. So, we will illustrate with the help of a sample program.

So, we have this following program, which is very simple, read a Celsius temperature and convert it into the equivalent Fahrenheit temperature. This is something that all of you must know. So, the flowchart is very simple, you have an input C, which is the current Celsius that you want to convert. Then you apply the formula F, which is 9C/5 + 32. In this session, we well see how to write simple C programs, which makes use of variables.

Recall that in our discussion about flowcharts we talked about variables, which were conceptually seen as little boxes in which you can hold values. So, let us illustrate a simple C program making use of variables with the help of a program. So, we have a small problem, which is convert a Celsius temperature into the equivalent Fahrenheit temperature. This is the formula that all of you must know. So, let us write a C program for it.

So, we will draw the simple flowchart for doing the program. You input the temperature in C, in Celsius, convert it into Fahrenheit according to the formula 9C/5 + 32. Once you have done that, the variable F holds the Fahrenheit value, so you output the F. So, here is the simple flowchart that we want to implement.

(Refer Slide Time: 02:03)



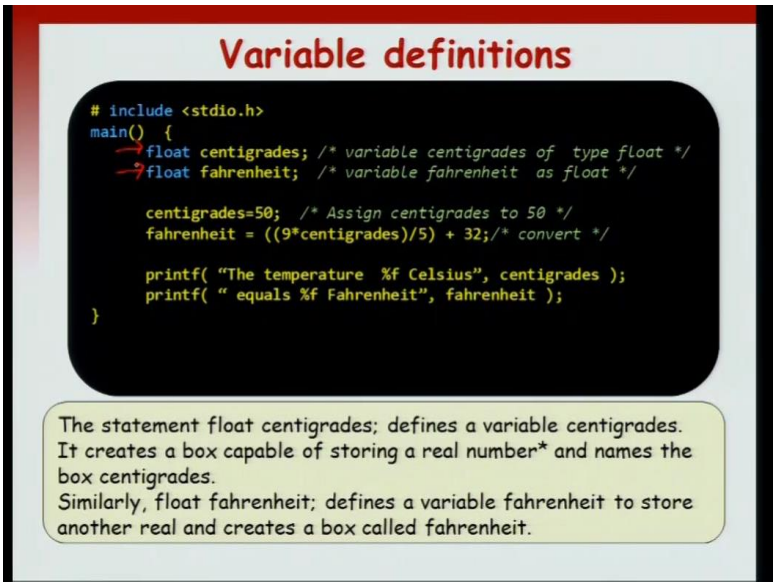
Now, how do we write the equivalent C code. So, we, in the flowchart we have seen, that we have two variables, C and F. These are the variables that we would want to implement in the C code. So, let us see how to do it. So, we write the following C program in which now we have two more components, one is the variable declaration, float centigrade and then the second is another variable, float Fahrenheit. So, centigrade corresponds to C in the flowchart and Fahrenheit corresponds to F in the flowchart. So, I write the following code, which is supposed to implement the flowchart on the left.

So, let us say, that the input is 50 degree Celsius and then the Fahrenheit, the formula is exactly the same as what we have in the flowchart. We have 9C/5 + 32. Notice here, that these are arithmetic operators. So, the * arithmetic operator stands for multiplication, / stands for division and + stands for addition. So, this is exactly as in the flowchart except, that here in the flowcharts multiplication symbol is being swallowed, but in C you have to specify it using a * operator. So, Fahrenheit equal to 9C/5 + 32 is exactly

similar to the analogous line in the flowchart.

And finally, for outputting we will use printf statement. So, here is something new in the printf statement. We use what are known us format specifiers, this %f symbols are new and we will describe them shortly.

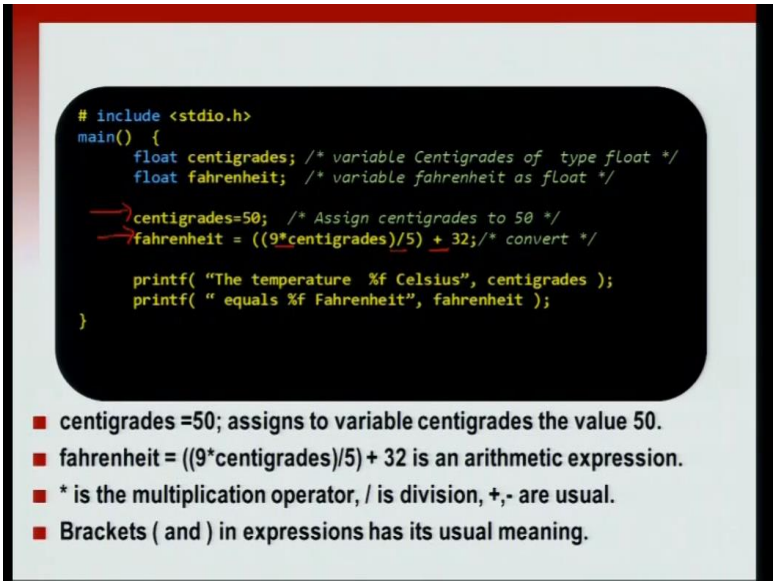
(Refer Slide Time: 04:05)



So, let us look at the program in little more detail. So, we have two statements, which are of interest, in the beginning of the code, which are what are known as the definition of the two variables. Recall from are discussion on flowcharts that variables are boxes and each box has a name associated with it. So, you have two concepts associated with variable as far as flowcharts were concerned, one was the box and the second was the name of the box. Now, when we come to C, we will associate a third concept with variable, which is the type of the box.

So, if you look at the first statement it says, float centigrade;. Now, this defines a variable centigrades. It creates a box capable of storing a real number and names the box the centigrades. So, the box is of type float. Type float means, that box can hold real number. Similarly, fahrenheit is also a box, which can hold real number. So, you declare that the type of that variable is float. So, these are supposed to be the first two lines of the code.

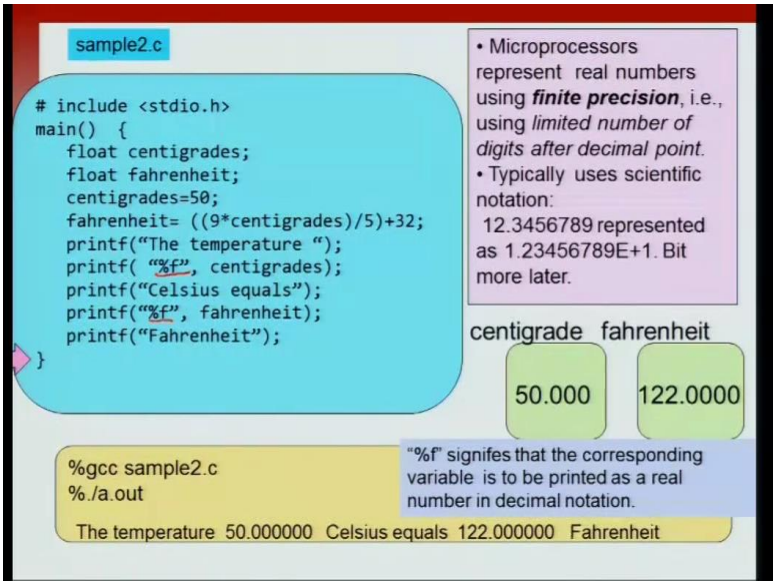
(Refer Slide Time: 05:29)



Now, centigrades = 50 that is the line, which assigns the value 50 to the variable centigrades. So, once you execute the code, the box associated with name centigrade will hold the value 50 followed by the line, which computes the value of fahrenheit. So, fahrenheit equal to 9C/5 + 32. It is an, it associates an arithmetic expression. So, it evaluates an arithmetic expression, takes its value and stores it in the box associated with the fahrenheit. And as we just saw before, * is the multiplication operator, / is the division operator and + is the additional operator.

Now, the brackets in an arithmetic expression are just like brackets in mathematics. So, they group together a particular thing. Now, let us just trys to the program. Let us see what happens step by step when we run the program.

(Refer Slide Time: 06:48)



Let us say that we save the file as sample2.c and then run it as ./a.out. So, first we will have two boxes created, one for centigrade and one for fahrenheit. These can store float numbers. Now, what are float numbers? Basically, they are real numbers, which are saved by the microprocessor. Now, the microprocessor can store variable real number only using finite precision. So, this is different from actual real numbers that we encounter in mathematics. So, we have only a limited number of digits after the decimal point, but other than that you can think of them as real numbers. We will see floating point number later in the course in greater detail, right. For now, think of them as the machine representation of a real number.

So, once you finish the declaration statements what you have are two boxes one first centigrade one for Fahrenheit and because you declare the types to be float it is under stood that those boxes will hold real number. So, let us execute the first executable assignment here. centigrades = 50 and you will see that the box contains 50.000 something. Even though we specified it as an integer, it will convert it into real number, floating pointing number and store it. Then this is followed by the calculation of the Fahrenheit value, and let us say that you compute 9 times 50 divided by 5 plus 32, it comes out as 122. After that line is executed, the box associated with Fahrenheit will contain 122.

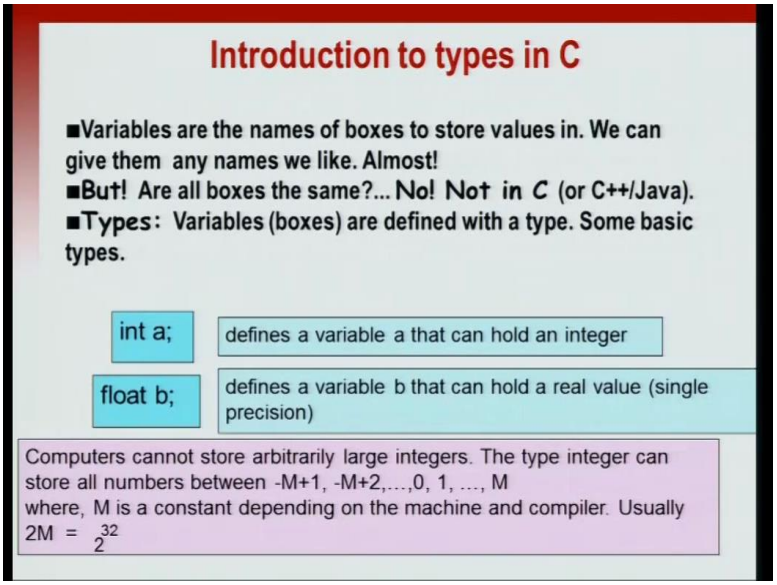
Then, the next line says, print, printf the temperature, there is no new line, so the next printf will start from where this printf ended and here you see something new, which is the %f symbol. So, these are what are known as format specifiers. So, the %f symbol say, that take the corresponding variable, which is given centigrades here. Now, print it as a float, print it as a real number.

So, notice the difference between first printf and the second printf. The first printf just had a string between ‘‘, the second printf has two arguments, one is a string between ‘‘ and the string is %f and then the second argument is centigrades. So, it says take value of the centigrade and print it as a floating point number. So, it does that and you see 50.000 in the output. There is no new line. So, the next printf starts from the previous line where the previous printf left off, 50 Celsius equals, it prints that.

And now, you have another format specifier. It says printf %f fahrenheit. fahrenheit is 122 and it will print it as floating point number or as real number. So, it will print it as 122.000 printf fahrenheit. So, the finial massage, that well be printed will be the temperature, 50 centigrade, 50 Celsius equals 122 Fahrenheit.

So, the new thing we have seen in the program include variable definitions, how they have an associated type and similarly, how do we print these variables. So, we do not want to print the names of the variables, we want to print the, we want to print the content of the variable. We want to print what is stored in the box. For that we use the format specifiers like these %f.

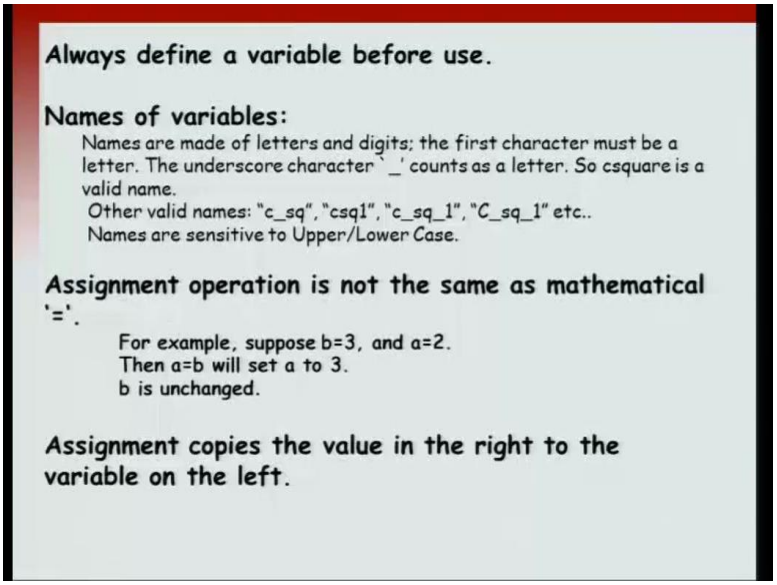
(Refer Slide Time: 10:59)



So, let us briefly introduce what are types in C. So, variables are the names of the boxes in which to store values, but these boxes are special. Certain boxes can hold only certain kinds of values, so all boxes are not the same. There are different kinds of boxes. Now, types are basically saying, that a particular box can hold a particular kind of data. So, variables are defined with an associated type and we will use some basic types during the course of this program language tutorial.

One of the two common type, two of the common types that we see in this program are int, which stands for an integer and float, which stands for a floating point number, which stands for real number. Notice, that machine can hold only a fix number of bits. So, that does not mean, that the integer can go from minus infinite to infinite. It goes from a certain vary small negative number to a very large positive number. Similarly, floating point also is limited by a particular range. This is because machines cannot represent arbitrary values. The type of integer can store all numbers from -M+1,...0,1,...,M. So, there will be some large M, for which, for which defines the upper limits and the lower limits of the particular machine. Now, that limit may depend on which particular machine that you use. On a, on a 32-bit machine it will be 2^{32} , M will be 2^{32}.

(Refer Slide Time: 13:03)



A few final words about variables. Just like in a cooking recipe, you well never mention a step, which involves ingredient without mentioning, that ingredient is needed in the first place. So, you will never say, that use salt and if you look at the list of ingredients, you were, you will see, that there is no salt in the list of ingredients. Such recipes are considered bad. So, when you write a typical recipe, you list out all the ingredients first and then write the steps for the cooking. Similarly, in a program you define whatever variables that you need before those variables are used by any statement in the program. Always define a variable before use.

Now, a word about names of variables in the C programming language. The names are consisting of numbers, letters and an underline symbol, an underscore symbol. And there is a particular convention that a variable cannot start with a number. So, the initial letter has to be a letter or an _, it cannot be number, but further can be either capital letter, small letter or numbers or an _. So, there are valid names like c_sq, csq1, c_sq_1. So, all these are valid. One thing to note is, that the names are sensitive to upper and lower case. So, for example, capital C Centigrade is different from a centigrade, which starts with a small c. So, these are two distinct variables that is the common source of errors when we start programming.

Another thing to note or to watch out for is, that the assignment operation, which is equal to is not the same as mathematical equal. So, when mathematically we you say a equal to b, it means, that a and b are the same quantity. So, a equal to b is the same as saying b equal to a. This is not true in C. For example, let us say, that you have the statements b = 3; and then later you have a = 2; and further you have the statement a = b; So, the statement a equal to b will set a to b's value. So, b's value is 3 and that value will be copied to a. So, it will set a to 3 and b we will be unchanged. So, watch out for this.

If, if you were expecting the mathematical operator, after the operation a = b, a and b we will have the same value, but that is not the case. The meaning of the symbol equal to is, that take the value on the right hand side of the expression and copy that into the box specified by the life side. So, copy the value in the right hand side to the variable on the left.