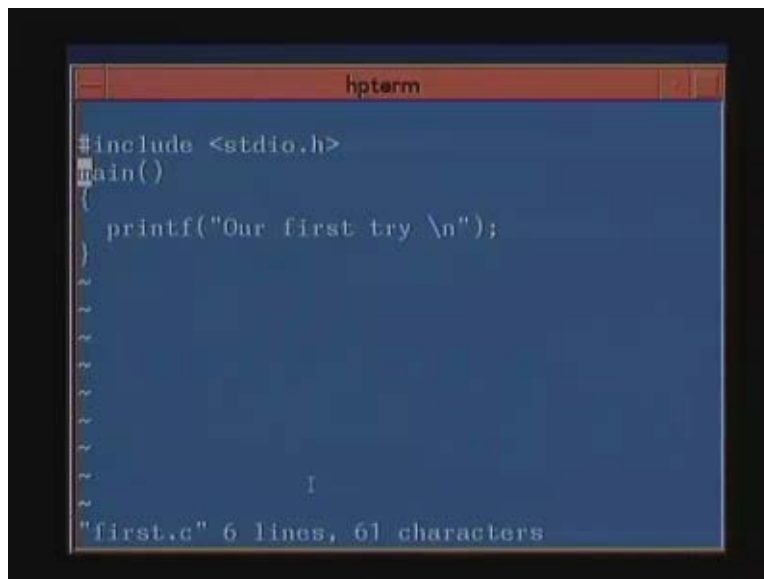


Programming and Data Structure
Dr. P.P.Chakraborty
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur
Lecture # 2
C Programming - I

Hello and today will be the second lecture of this course and we will take two lectures, initial two lectures to understand what is meant by simple C programming. We will assume that most of us have done some programming in a language which may not necessarily be a C language. So we will utilize some time to just get acquainted with the constructs of C language. And C language is a language which provides a large number of facilities to the programmer, a large number of high level facilities for data structuring as well as programming methodology as we discussed in the first class. And today we will see some of the simple constructs, later on we will move into the more sophisticated and complex constructs of C language.

So today will be an introduction to C language which we shall see by running certain example programs which are there on the computer. I will assume that you are aware of some programming language and therefore we will just take a set of greater examples so that we go through how to do each and every aspect of programming in C. So let's go and see what happens. We will be executing it on the HP machine out here and we will see how to, some examples.

(Refer Slide Time 02:45 min)



```
hpterm
#include <stdio.h>
main()
{
    printf("Our first try \n");
}
~
~
~
~
~
~
~
~
~
~
"first.c" 6 lines, 61 characters
```

The first example that we will see is this. So a C programming language will have a main function declaration. So this is unlike other languages like Fortran where you need not require to name the main function or the main program. In C you have to call it main. The arguments which can be given here are certain arguments which we shall see later and we shall leave them blank

for the time being. Now here we include a file and this is a library which is the input output library of C.

Now C has got a huge set of libraries and along with C language, you have an input output library which is called the standard io.h. So we call it in short std io.h and for all our input and output programs, we use this std io.h, so we include it in our programs. C is a block structured language and in every block can be defined by these braces, this is one brace and this is the brace which is, the opening brace is here and the corresponding closing brace is here. And we will see the concept of blocks, every main program must have a block. So here we define a block for the main program and inside this block, we have a sequence of statements. So this is the structure of a simple C program.

Now in this example, we just print something. So the print function is called printf, this prints into the standard output if you don't give any other arguments. You can also use file printing function which are also available, we shall come to it later on. Now in a print statement whatever text item has to be printed will be given within these codes, this is one quote and this is the ending quote here. So within the quotes whatever you want to get printed will be have to be printed and other than, here we have written our first try. So we want to print our first try and this backslash n is a control sequence **we** into the print statement. This control sequence means here backslash n means new line. So **you** after printing our first try it will give a new line. So let us see how this program works. So one we run it, it gives our first try.

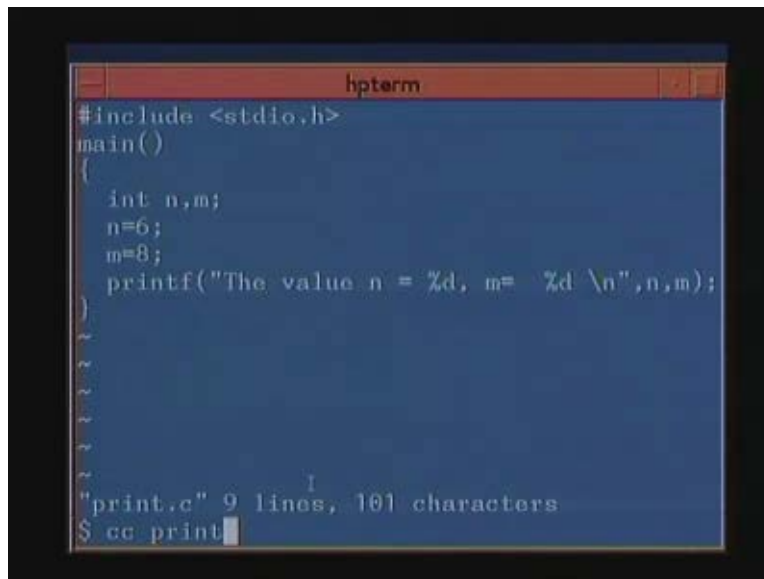
Now we could output another backslash here and here then what would happen is that after printing our, it could give a new line and then print first try in the next line. So if you execute this, here am compiling the program with the cc compiler, cc is the compiler and it creates a dot, a dot out file which is executed. So this data, a dot file has to run and you will see our is now printed first, our and then first try in the next two lines. If you look our program again, we will see there as several other than this backslash n you can give other constructs, one of them is t which gives the tab. So if you get backslash t, it will give you a tab between our and first. So you see how this executes our and then there is a gap for tabbing and then first try. So this is the simple program, we just print something and hopefully we will be using the print statement again and again.

So, the first thing in any programming language is to write and read, so here we have learnt how to write in C. Now we will see the next is how to write up the value of a variable. So let us take the second program. Here we are going to write down the value of a variable n. So we again included stdio.h, we got a main program within curly braces and here we are using a variable called n. Now this variable has to be declared unlike Fortran where depending on how the variable is named, the type of the variable is assumed and there is a default type. There you can also declare a type of a variable. In C language you have to declare the types of each and every variable and we have declared n to be an integer and the format for declaring n to be an integer is int n.

Similarly we will see how to declare a real number, how to declare a character, how to declare a strings and how to declare arrays a bit later but now we will just work on the integers for the time being. And here we have assigned n equal to 5 and we have printed the value of n. So we have

written printf, the value of n is equal to... Now here is a percentage d, this percentage d is another control construct. It gives you the format in which the variable will be printed and percentage d means it will be printed as an integer, backslash n, n. So this is the, between the two quotes from here to here is the control sequence of the complete control of the print f statement. It says first t h e will be printed after the blank, v a l u e will be printed, blank n equal to then an integer will be printed, after the blank there will be a new line. And corresponding to this integer we will print this value of n. So let's see how this program will run.

(Refer Slide Time 11:19 min)



```
hpterm
#include <stdio.h>
main()
{
    int n,m;
    n=6;
    m=8;
    printf("The value n = %d, m= %d \n",n,m);
}
~
~
~
~
"print.c" 9 lines, 161 characters
$ cc print
```

It prints the value of n is equal to 5. Now we could also change the value of n say to 6 here and we could include another variable say m, declare it and assign m is equal to 8 and the print the value of m here also and for this we will have to give another percentage d. So this means it will print the value of n is equal to, it will print the value of n and in the next place it will print the value of n. So if you want to write m also then you will write like this. So it will print the value of n equal to and it will give the value of n here and then m equal to the value of m here because first n and then m will be printed. So this is the program, sorry there is some mistake. There is some i out here which is incorrect, this is syntax error. So it is given a syntax error during compilation.

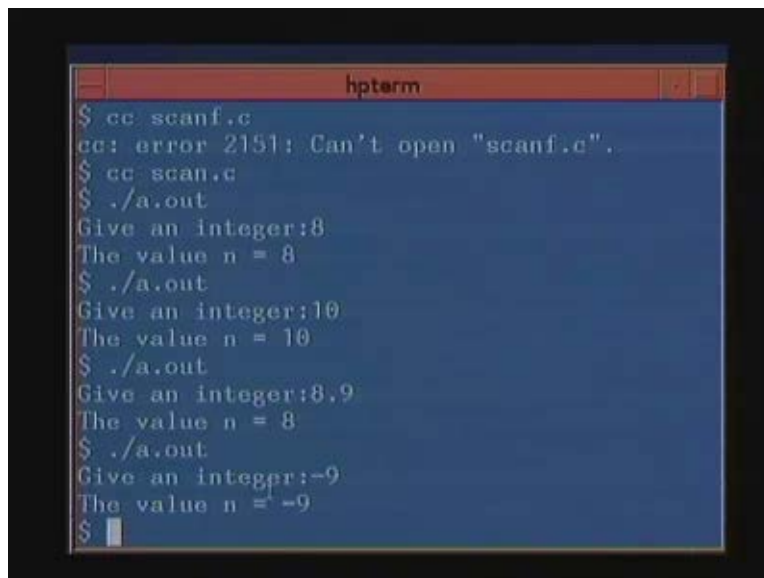
So now it's okay, so it now prints the value of n equal to 6, m is equal to 8. And if you want them into different lines, you can put the control n and the other sequences as and when they are required. So now let us take the next program. The next program, so we have seen a program to simply print. You have seen another program to declare variables and print their values. The next program that we will see is reading, how to read the value of a variable. In the previous program that we did, we just assigned the value of a variable to a fixed value and then printed it.

Now we will read it and print it because reading and writing is the first thing that you should know well anywhere in any language whether it's Fortran or English. So this is a program to read an integer and print its value. So it says hash include stdio.h mean this is the main braces, we

declare an integer int n, we print saying give an integer and here there is no backslash n, so it will wait here for you to give an integer and this is the function to read a value like printf, scanf is the library function which can be used to read the value. And just like print f it has the control within these quotes which says that I am going to read in an integer percentage d. And when I give the integer very finally I have with the variable something called AND n. If I give only n, the results will not come. So, I have to have to give something called AND n to actually read in the value of n. we will understand why we have to give AND n a little later on. Now let us see how this problem works first. So if we have to read in a value n, we have to write in and an integer, we have to write in scanf, percentage d and n and then print f we have just print in the value of n as you can see in the normal manner.

So let's see how this program runs. So we compile it, scan dot c that is the name of the file and the executable is a dot out. So it says give an integer and if you give 8, it prints the value of 8 again 10, it will give the value of 10 and if you give 8.9, it will give you 8 because it is supposed to read in an integer and if you give a decimal number it will truncate the decimal number and read it.

(Refer Slide Time 15:11 min)



```
hpterm
$ cc scanf.c
cc: error 2151: Can't open "scanf.c".
$ cc scan.c
$ ./a.out
Give an integer:8
The value n = 8
$ ./a.out
Give an integer:10
The value n = 10
$ ./a.out
Give an integer:8.9
The value n = 8
$ ./a.out
Give an integer:-9
The value n = -9
$
```

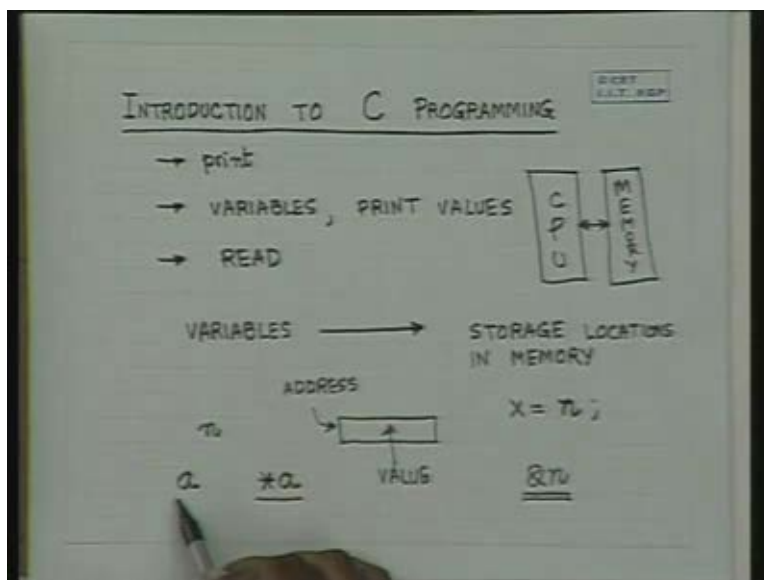
So we will see what percentages format mean when you give a real number and it is supposed to be read as an integer, it is just truncated as you can see in this example. It does not matter whether you give a negative value or a positive value, all of them will be read. So let's go back to the program and see what we were doing. So we were reading something and printing, now let's come back to the intelligent question on this AND. This AND is related, see this scanner is a function and this AND is related to how parameters are passed in C language. We shall come to how parameters are passed a bit later on, for this information and n means the address of the location n.

Now let us understand this a better before we proceed further. In the computer you will see that variables are stored finally, if you have variables they are after compilation during execution

they are storage locations in the memory of the CPU. So, the computer has got the simple one von Neumann architecture then can be looked upon as the CPU and the memory and here the program as well as the data is stored.

Now corresponding to the variables, data corresponding to the variables are stored in the memory. So, corresponding to every variable n , you have a memory location. Now this memory location has got two items, one is the address of this location and the other is the value inside this. Now whenever in C programming language, whenever you write x is equal to n then this means the value of x is assigned the value of n or in other words the value of n is copied into the value of x . So whenever we use value statements, we mean whenever we use a simple variable we mean the value of a variable.

(Refer Slide Time 18:46 min)



Whenever we write $\&n$, we mean the address location of the variable n . So $\&n$ means the address location of any variable n and suppose somebody passes an address location a to you and if you want to know the value of this address location, you write $*a$. So a is an address location, if you want to find out the content or the value of that address you write $*a$. So there are two things about the variable. If it is a variable then $\&a$, if it's a variable n and a then $\&a$ of it will give you the address location. If it's an address location then $*a$ of it will give you the value. So these are two things which are used extensively in C and here let us come back to the program now.

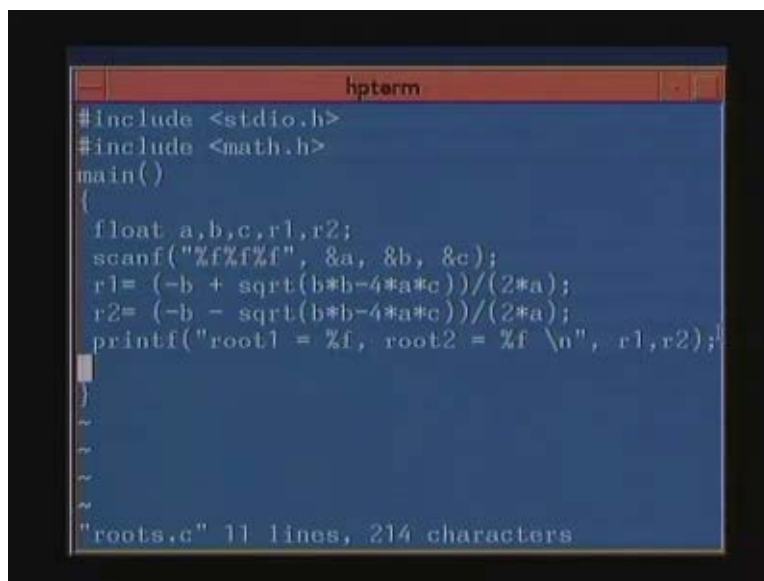
In the program, this $\&n$ means you are passing the address of the location n . Why you have to pass the address of the location n is a question. We shall answer a bit later when we come to functions. This is related to the parameter passing rule and once we come to functions and pass parameters and change variables, we will realize why it is like this in C. So for the time being, let us just know that whenever we have to write a scanf function and we have to read in the value of a variable, we have to pass the address of that variable and the address of that variable is passed by the $\&n$. Any question? So now we move on to a more involved exercise where we are

using some assignments statements. So this is a program which finds out the roots of an equation.

Now we will be using functions that is why I have included the, another library other than the io library, I have included the math library. The math library includes some functions in C. So I have got only one main program, we have not come to subroutines yet, so inside this main program I have now declared float a b c r1 and r2, float declares a real number, a floating point number, like in declares an integer, float declares a real number. Semicolon as I did not mention, semicolons break up statements. Float a b c r1 r2, a b c are expected to be the values of the coefficients of a quadratic equation which is given as $a x^2 + b x + c$ and r1 and r2 are the roots that we are expected to find and we will take in some real numbers. So first thing we have to read in is the coefficients a b and c. So how do we read in the coefficients? Scanf and since they are floating point numbers, we read them by the percentage f format. So, percentage f, percentage f, percentage f if you give a blank here then you will have to read in a blank as well. If you give a comma here, you will have to read in a comma as well.

So, normally they will accept it with blanks and if you explicitly give a blank here you also have to explicitly give a blank. If you give a comma here like this then when reading in we will have to give a comma, we will avoid all those things. So percentage of three floating point numbers and as I mentioned earlier we are going to read in AND a, AND b and AND c.

(Refer Slide Time 22:26 min)



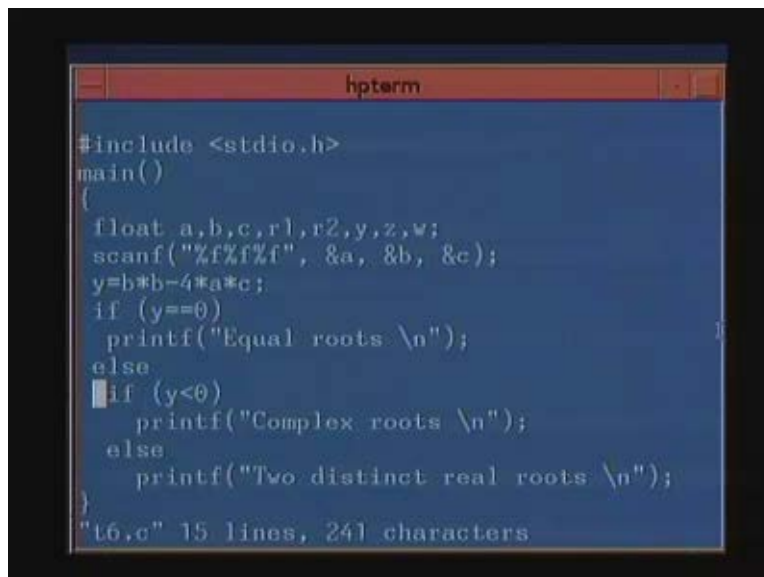
```
hpterm
#include <stdio.h>
#include <math.h>
main()
{
    float a,b,c,r1,r2;
    scanf("%f%f%f", &a, &b, &c);
    r1 = (-b + sqrt(b*b-4*a*c))/(2*a);
    r2 = (-b - sqrt(b*b-4*a*c))/(2*a);
    printf("root1 = %f, root2 = %f \n", r1,r2);
}
~
~
~
~
"roots.c" 11 lines, 214 characters
```

So to read in a b and c, we have to pass the addresses of a b and c. And then we compute the roots r1 and r2 assuming that complex roots will not be there and other issues will not be there, assuming the inputs will come correctly. We give r1 is equal to minus b plus square root of b square minus 4 a c divided by twice a and r2 is minus b minus square root of b square minus 4 a c by twice a and then we print out root 1 and root 2. So, the first thing when we run this program is you are supposed to give in the three values.

So let's compile it first and then run it. **Sorry, this will not be the correct.** I have to compile it with the math library included, so there is in Unix we have to compile it with minus lm format that is the... so let's give the three values 1.0 2.0. So with the inputs 1 2 and 1, the roots are 1 and minus, the two roots are equal roots and both of them are minus 1. And you can give other values to see what you will get. With a equal to 1, b equal to 3 and the other one you will get this. So this is the simple statement where we have used assignments, we have used the square root function out here sqrt, you have got a whole lot of other in built mathematical functions sin, cos and everything else is available like in any other language. Only thing we will have to include the math library and we have to compile it for that the math library is included properly and is loaded.

In Unix we have to compile to the minus lm option and in the windows or the dos you have to compile it appropriately. So however this program does not take care of the issues when the roots are equal, separating out the cases when the roots are equal, when the roots are complex, when the roots are unequal. That is the two real roots are there, complex roots are there and the equal roots are there. So now let us take, let us see how we will separate out the three parts. In order to separate out the three parts we need the conditional statement or the if statement. So there is an if statement in C language as well. Here is the program which does that. So here I have included stdio.h, I did not compute the roots, so I did not include the math library.

(Refer Slide Time 26:00 min)



```
hpterm
#include <stdio.h>
main()
{
    float a,b,c,r1,r2,y,z,w;
    scanf("%f%f%f", &a, &b, &c);
    y=b*b-4*a*c;
    if (y==0)
        printf("Equal roots \n");
    else
        if (y<0)
            printf("Complex roots \n");
        else
            printf("Two distinct real roots \n");
}
"t6,c" 15 lines, 241 characters
```

Float a b c r1 r2 y z w, y is equal to scanf percentage f percentage f a b and c. y is b square minus 4 a c, if y is equal to 0 printf equal roots else if y is less than 0 complex roots else there are two distinct real roots. So this is the program which uses the if then else statement and a nested one that too and I am sure all of us understand what is the nested if then else statement. So I am not going into the details of what nesting means and what nested if then else means, the else is related to the closest if, that's the standard practice. But please note here, the equality is not y equal to 0. You must give two equal signs like I have given here to test for equality and if you give y equal to 0, it will assign y to 0.

In C language you can make an assignment even inside a conditional. So if I here get an y equal to 0, instead of y double equal to 0, it would have assigned y to 0 also. So the equality testing is y equal to equal to 0 else y less than 0 else the two distinct roots. So this is the simple if then else format and I am sure you understand the if then else format as well. So now let us see the complete program with the printing of the equal roots, printing of the complex, complex won't print, printing of the distinct roots. So let's see the, so let's run this program first. Let us run this program. So if you give 1, we will get equal roots. We can also give integers, if you give integers they will be converted to floating point. For example you can give also this, there is no problem it will read it properly.

(Refer Slide Time 28:20 min)

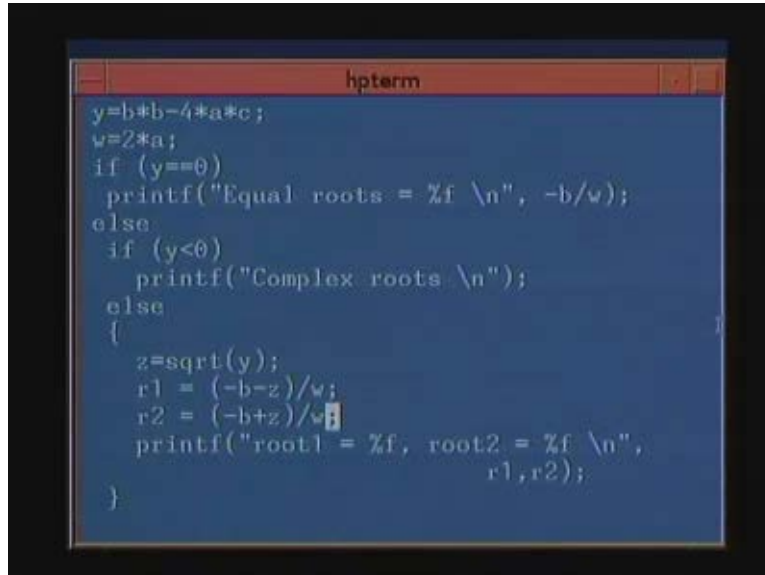
```

hpterm
    printf("Complex roots \n");
else
    printf("Two distinct real roots \n");
)
"t6.c" 15 lines, 241 characters
$ cc t6.c
$ ./a.out
1.0 2.0 1.0
Equal roots
$ ./a.out
1 3 0
Two distinct real roots
$ ./a.out
1 1 1
Complex roots
$

```

So if you give 1 3 0 it will print two distinct roots and if you give say things like 1 1 1, it will give you complex roots because $b^2 - 4ac$ is now negative and the complete program will look like this. So I have done a scanf, I have computed the $b^2 - 4ac$, I have computed twice a out here and if y is equal to 0, I print equal roots. And I print the value of the two equal roots which is $-b \pm \sqrt{b^2 - 4ac}$ because $b^2 - 4ac$ is 0, therefore it must now be $-b \pm 2a$ and I have pre computed twice a out here to get so... $b \pm w$ should give me the equal, the value of the root and there are two equal roots. If else if y is less than 0 is in the same format as the previous program, just the computation is done. it just prints complex roots else it does the square root of y , y was $b^2 - 4ac$ if you notice here and so there is a square root of y and here $b \pm z$ by w , w is twice a computed here.

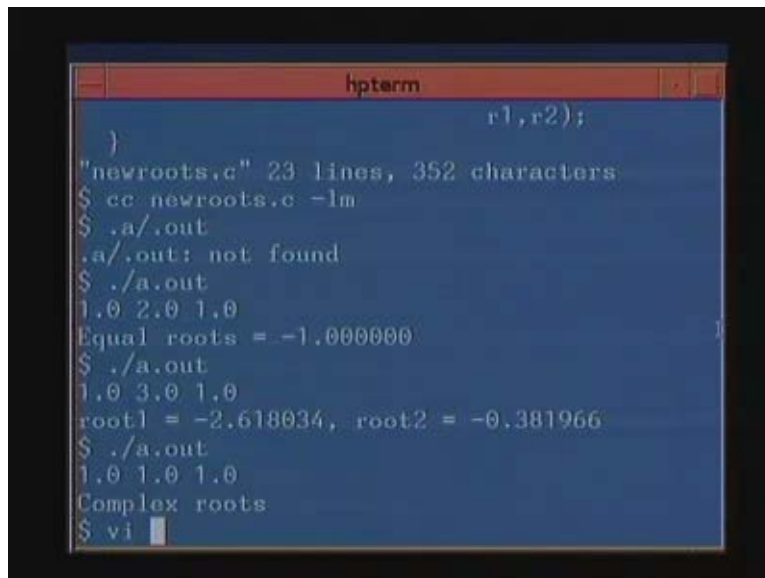
(Refer Slide Time 29:34 min)



```
hpterm
y=b*b-4*a*c;
w=2*a;
if (y==0)
    printf("Equal roots = %f \n", -b/w);
else
    if (y<0)
        printf("Complex roots \n");
    else
    {
        z=sqrt(y);
        r1 = (-b-z)/w;
        r2 = (-b+z)/w;
        printf("root1 = %f, root2 = %f \n",
               r1,r2);
    }
}
```

And here it is b plus z by w and the two roots are computed that prints equal roots, that prints two distinct roots, with 1 3 0 we got the two distinct roots. Here we did not compute because in the program it just printed complex roots and did not do anything else.

(Refer Slide Time 30:38 min)

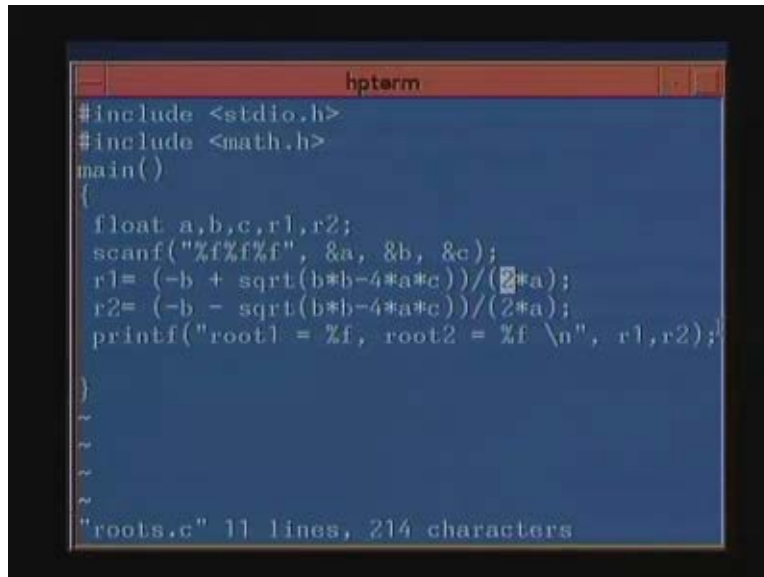


```
hpterm
    r1,r2);
}
"newroots.c" 23 lines, 352 characters
$ cc newroots.c -lm
$ ./a.out
./a.out: not found
$ ./a.out
1.0 2.0 1.0
Equal roots = -1.000000
$ ./a.out
1.0 3.0 1.0
root1 = -2.618034, root2 = -0.381966
$ ./a.out
1.0 1.0 1.0
Complex roots
$ vi
```

Another point I would like to say that is go back to the original roots program is a very simple issue but I would like to raise this issue right at this point, so that we remember it as we go ahead, is here in computing in the two statements I have done this square root in twice. I have done this b square minus 4 a c twice, I have computed twice a twice, 2 star a. and this is going to put in more effort and obviously it is going to take more time to do this computation because we

are repeating computation. And since, efficiency is something that we are seeking to achieve in a good program. We should try and see that we are not doing such things repeatedly. Therefore it would have been much better if you had used the variable because one or two variable will not make much of a difference unless you are using too much space which is not there.

(Refer Slide Time 31:53 min)



```
hpterm
#include <stdio.h>
#include <math.h>
main()
{
    float a,b,c,r1,r2;
    scanf("%f%f%f", &a, &b, &c);
    r1= (-b + sqrt(b*b-4*a*c))/(2*a);
    r2= (-b - sqrt(b*b-4*a*c))/(2*a);
    printf("root1 = %f, root2 = %f \n", r1,r2);
}
~
~
~
~
"roots.c" 11 lines, 214 characters
```

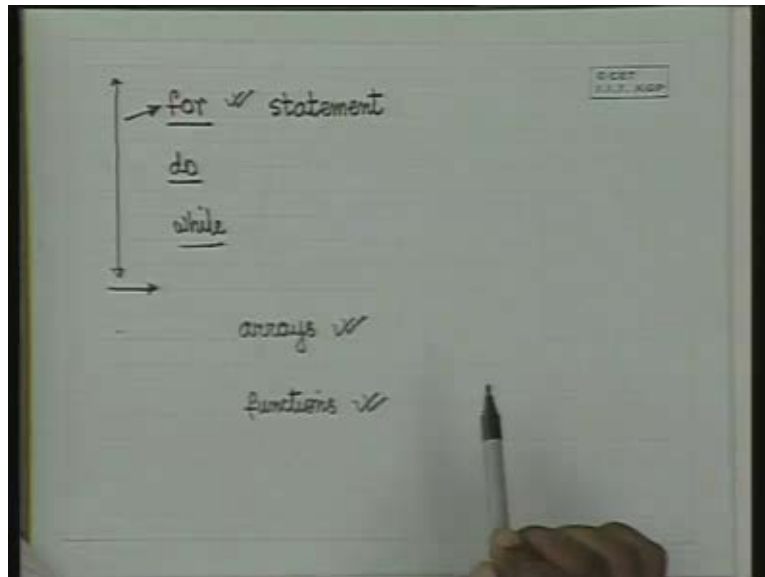
And here we will use most space to say one time and we will declare a variable and assign it, precompute this $b^2 - 4ac$. When we will precompute this square root of that and then we can precompute twice a and then use it as and when required. And that's exactly what we did in the final program. Here we precomputed the value of $b^2 - 4ac$, we precomputed the value of twice a and here because y was to be checked, so this value of $b^2 - 4ac$ and twice a were to be used repeatedly once here and square root was to be also used twice, so we computed it only once, stored it up and we used it. It is not the computer is so fast that we will not be possible to show the difference in the two at this point of time but I hope you all understand that there is going to be quite amount of difference in these two things.

So this is another simple program to work out the roots of an equation and we have seen what it means to write an if then else statement and we have also seen how to do an assignment, declaring of integers as well as floating point number. And the other conditional construct that everyone uses in any programming languages is the loop. So we will now see an example to find out to use the loop.

So the next construct that we will do is the for statement. There are many ways of looping in C language and the for statement is a very flexible and quite a general purpose routine. It depends on your taste of which loop we will use, there is a do statement and there is a while statement also where three such looping constructs are available in C. this is the standard looping construct where you have got a lot of control.

For example you want to go from i to n in steps of one, steps of two or steps of four then the for statement for fixed looping is very useful whereas for lot of conditional looping, the do and the while statements are useful but the for statement can also be use the conditional looping. If the for statement is quite flexible and so we will see the for statement for looping next.

(Refer Slide Time 34:53 min)



The next after seeing the for statement for looping we shall have, we shall see how arrays are declared **and how** and the finally we shall see how functions are declared in C language. So, we will stop this lecture here at the conditional statement and in the next lecture we will start off with the loops specially with the for statement. Then we will see examples with arrays and then we will see how functions are working.