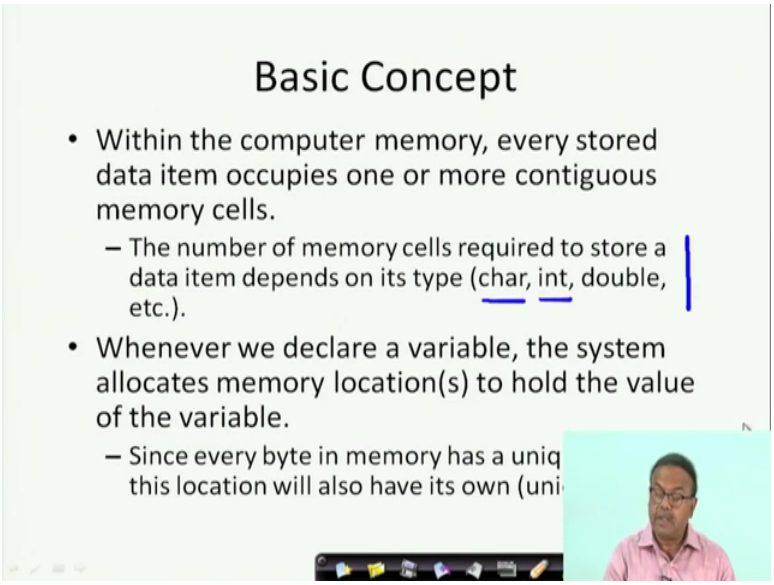
Problem Solving through Programming In C
Prof. Anupam Basu
Department of Computer Science & Engineering
Indian Institute of Technology, Kharagpur

Lecture – 58
Pointer

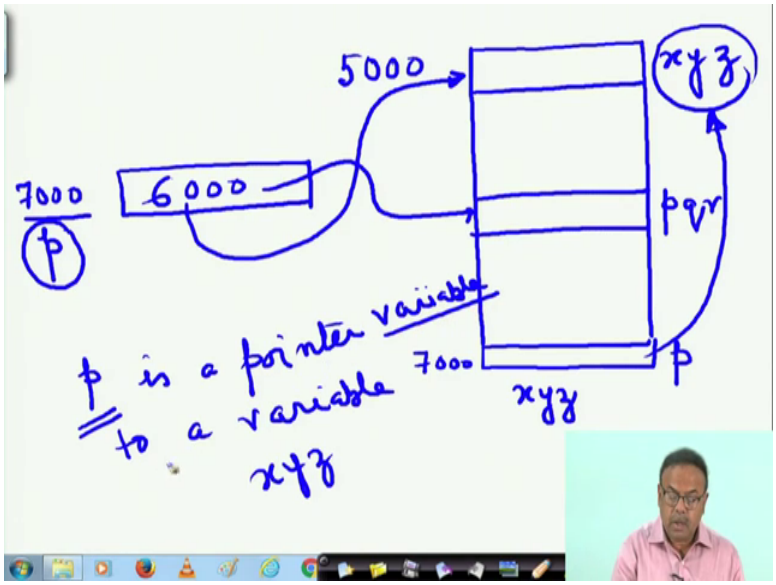
In today's lecture we will look at very important concept of programming, it is a required to conceptualize and understand the thing very well so, that you can have more flexibility with programming. We have visited this idea the concept of pointers, earlier in the context of our discussions of call by value and call by reference.

(Refer Slide Time: 00:48)



So, if you recall at that time, we had we had talked about the variables which are in the memory right.

(Refer Slide Time: 00:59)



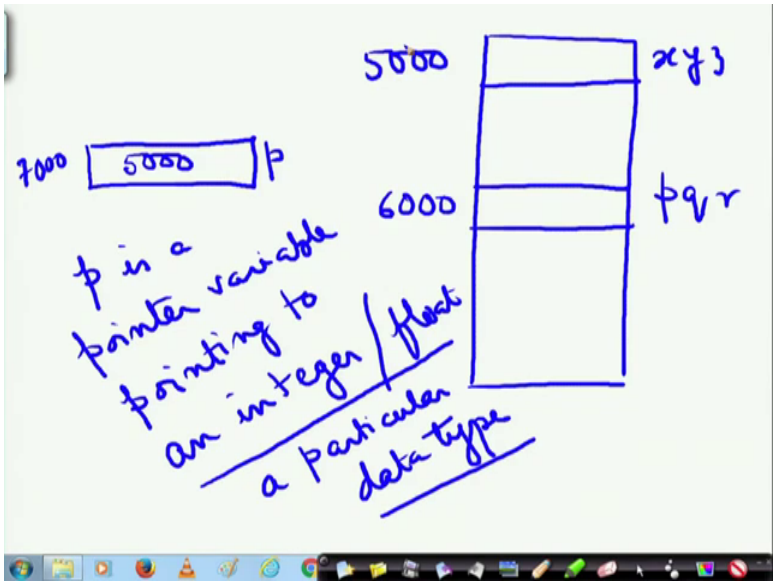
So, suppose I am talking of a variable x y z is the name of a variable. So, that variable has got some address, that address maybe say 5000 in the memory. And who has allocated this address that address has been allocated by the compiler.

Now, if I have another memory location, another memory location say a part of this part of this which I am not I am just drawing separately whose address is say 7000 say this one. 7000 and inside this location 7000 as this content, I write 5000. And I say that whatever is the content of this location 7000 that is the variable I am interested in.

So, can I say that if I now give a symbolic name p to this 7000, I can say or say this is p that this p is pointing to x y z because p is containing the address of x y z. Therefore, p is a pointer to a variable x y z, we also say that p right now it is pointing to x y z, but if I just change the value of this location 7000 and make it say 6000 then it will probably point to some other location here.

So, p is therefore, a pointer variable, but when I say that it is a pointer variable, then it is not pointing to any particular data it can point to a type of data. So, instead of making this statement that p is a pointer variable to a variable x y z, say this one is p q r say. Now, as I change this now p is a pointer variable to a variable p q r.

(Refer Slide Time: 04:02)

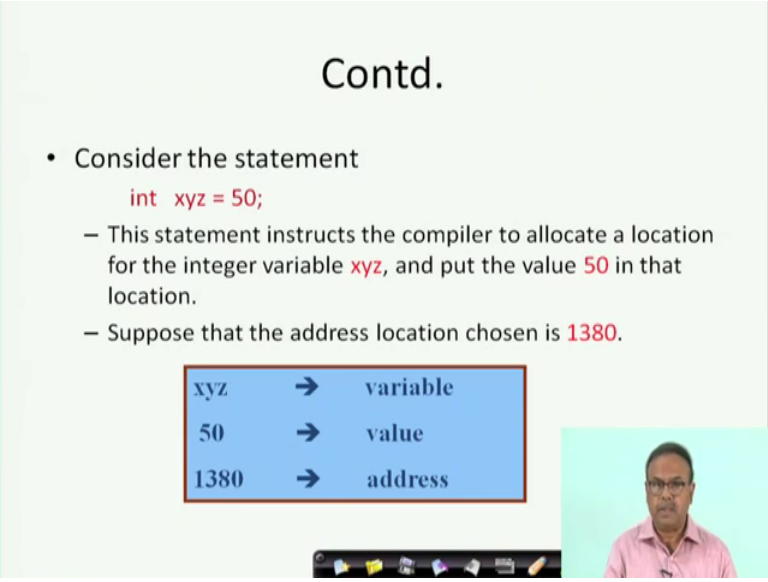


So, if I generalized it I will say that in this situation where I have got x y z at 5000 and p q r at 6000, I can say and 7000 is a pointer p and I can say p is a pointer variable pointing to say x y z and p q r are both integers to an integer or it could be a float or it could be a some other data type. So, pointed to a particular a particular data type why I specify the data type will be clear in some time from now. So, we have got a pointer the concept the most important concept here is that p is a pointer p is p is a variable and variable is holding some value, and that value is nothing, but an address of another variable all right address of another variable.

So, now let us look at this the basic concept every stored data items occupies one or more memory cells, whenever we declare a variable the system allocates memory locations to that variable we know that very well; so, need not spend more time on that. The number of memory cells required now this is important, you already know that the number of memory cells required to store a data item depends on its type typically for char we need one byte for int we need two bytes for float we need four bytes etcetera etcetera.

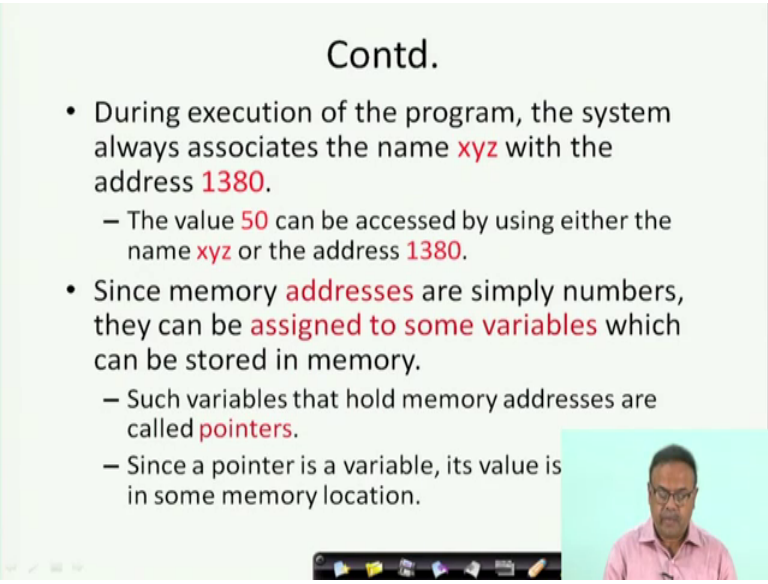
So, since every byte in memory has an unique address, this location will also have its unique address every element will have a unique address.

(Refer Slide Time: 06:38)



So, let us see here the same example, consider the statement int x y z assigned 50 this statement means that the compiler will allocate for this x y z some location and put the value 50 in that location. Suppose the address is address of x y z is 1380. Here x y z is a variable and 50 is the value and 1380 is the address of that variable we know that we have discussed it earlier.

(Refer Slide Time: 07:18)

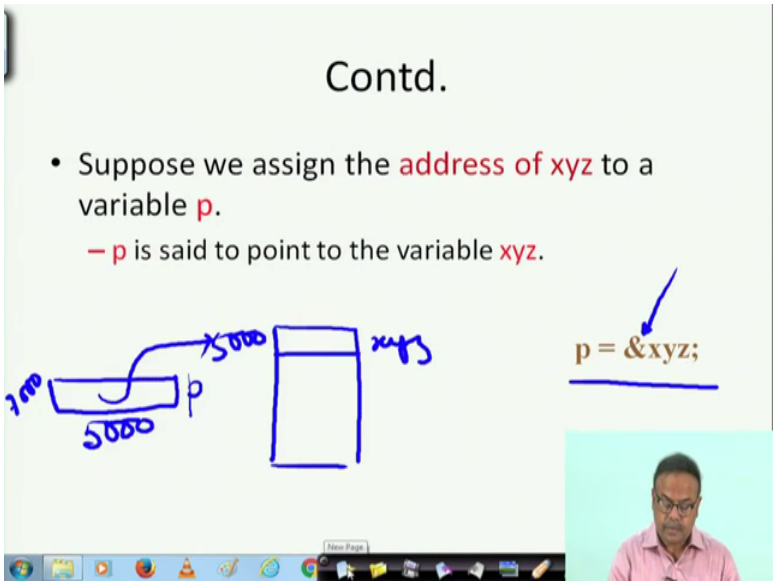


Now, during execution of the program, when the program is being executed the system always associates the name x y z with the address 1380. So, whenever in the program we

find x y z the variable x y z being referred, it will go to the memory location thirteen eighty and fetch that. So, the value 50 can therefore, be accessed by going to the location 1380 and accessing it. Now, the variables which are holding these addresses are known as the pointers. Now, memory address is a just numbers.

So, I can also store them in some variables and these variables which are for example, the address of the variable that we will hold the address of variable x y z is the pointer to x y z. So, it is also naturally stored in some memory location.

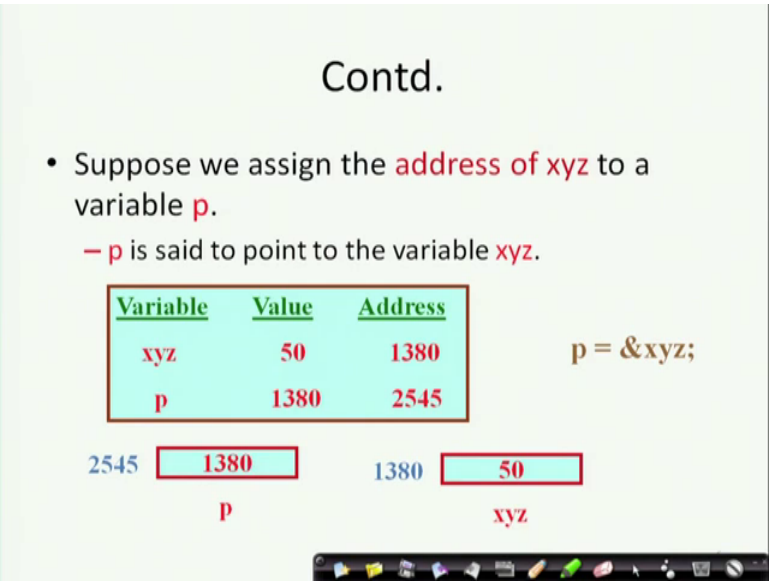
(Refer Slide Time: 08:25)



So, this is just the example that I was discussing right now. Suppose we assign the address of x y z to a variable p, then p is said to point to the variable x y z.

Now, how do I? I was just drawing this I was comfortably drawing this that there is 5000 here x y z and 5000 here and I was simply saying that this variable which was in location 7000 was pointing to this ; that means, this was being loaded with 5000 how is that being done how is 5000 being written inside this location? The statement is just like any other assignment will be that these variables name is p, p is assigned the address of x y z. So, p assigned and x y z right.

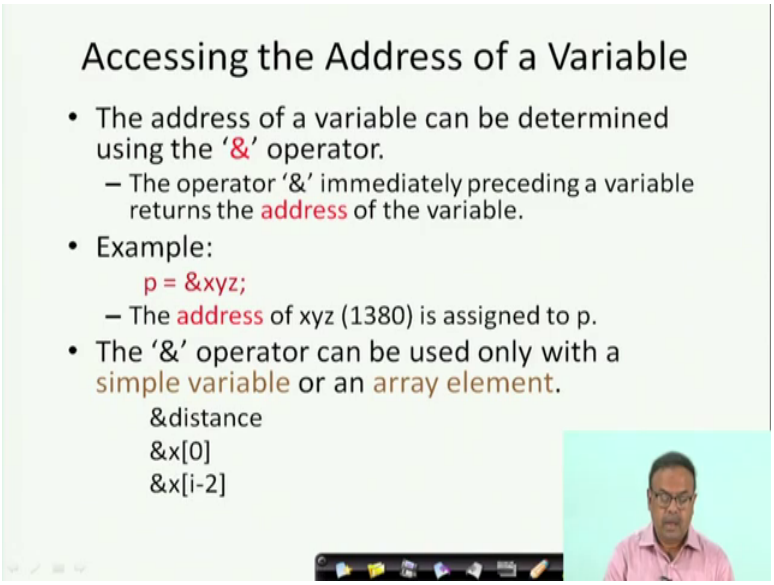
(Refer Slide Time: 09:32)



Now so, here we can see that the variable x y z is the address is 1380 value is 50 p is a pointer variable whose value is 1380 because it is pointing to x y z.

In our diagram that I was drawing by hand, this p was holding the value 5000 and its address here is say 2545 in my diagram it was 7000. So, this is a picture 1380 is the address of 50, and p which is located as 2545 is holding the address 1380 clear. Now this concept whenever you find difficulty in dealing with pointers my suggestion always to the student says, draw a piece of diagram and then you make the whole picture clear in front of you.

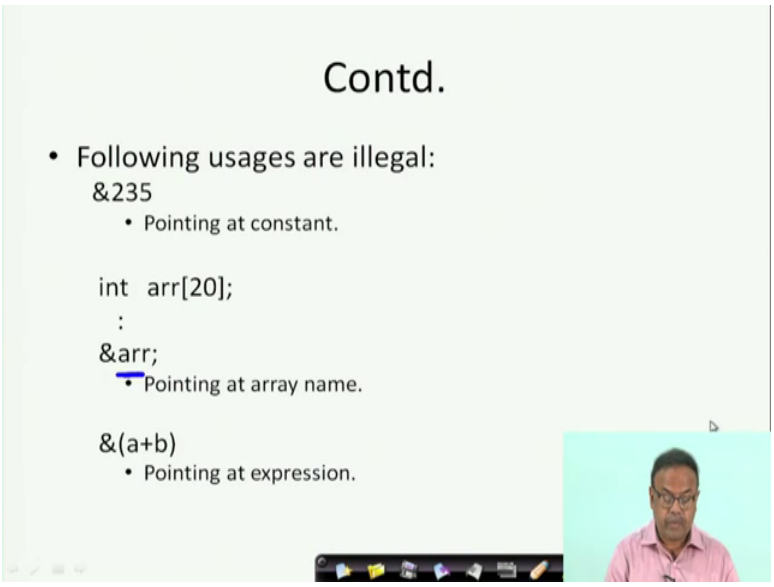
(Refer Slide Time: 10:33)



Thus in the address of a variable now you know that if I put the operator and ampersand immediately before the variable that will return the address of the variable right and x y z will give me the say the value 1380, which is the address of x y z.

The address of x y z is assigned. This operator and can be used only with a simple variable or with an array element for example, and distance and x 0; that means, the address of the first location of the array x and of x i minus 2 all this things are possible.

(Refer Slide Time: 11:23)

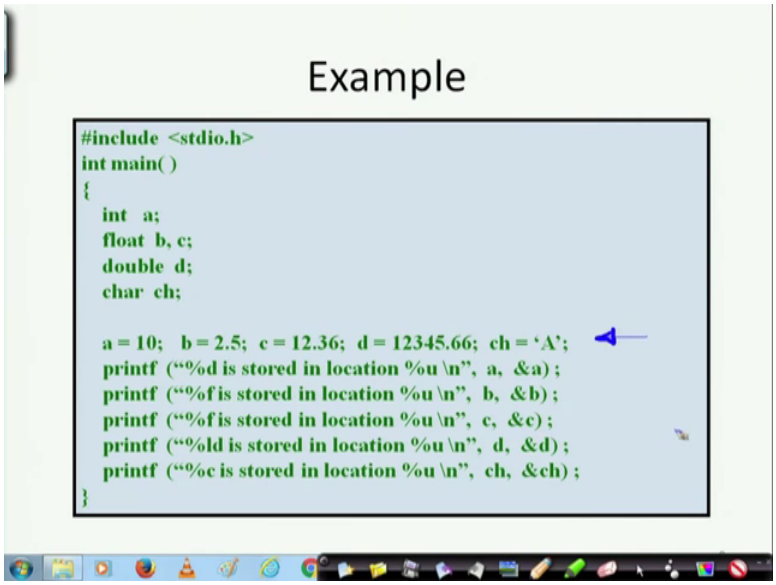


Now what is illegal? This is illegal; the reason is obvious two thirty 5 is not a variable it is a constant.

So, it does not have any fixed position in the memory. So, it is not a memory location therefore, it does not have address that is meaningless. Pointing to at a constant that is not possible say int I cannot say and arr, because this shows that arr is a particular variable, but it is an array therefore, I have to I cannot show it like this it is just pointing at an array name it is not pointing at the array ok.

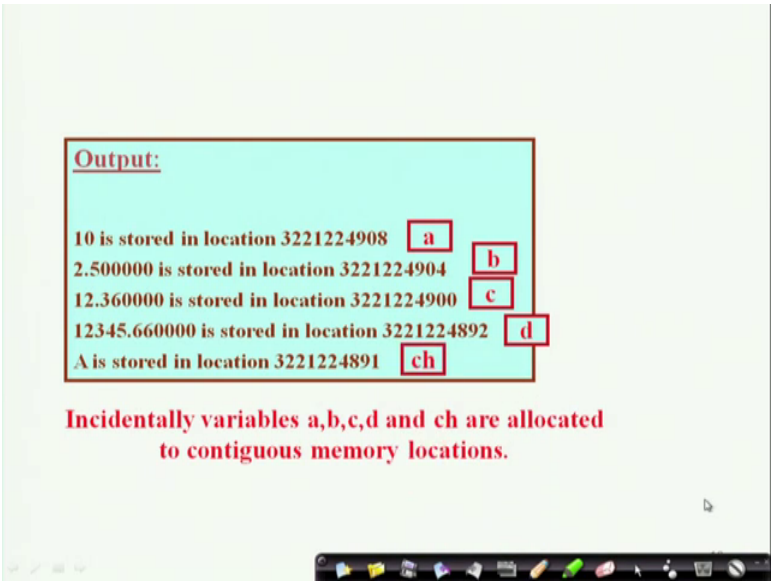
We should we cannot also do this and a plus b that is also not possible because this will be is a (Refer Time: 12:19) if I have a as some value and b has some value then I add that and that will be a value and a value does not have any address, that is pointing at an expression.

(Refer Slide Time: 12:26)



So, here is a quick example I think I had shown this to you earlier say I have got three variables a number of variables one character, one double floats. So, if I put say I am assigning some values to these here I am putting some values, and print f a particular variable a is stored in the address and a b is stored in address and b, c is stored in address and c so and so forth therefore, if we run it, we will get the addresses doming out.

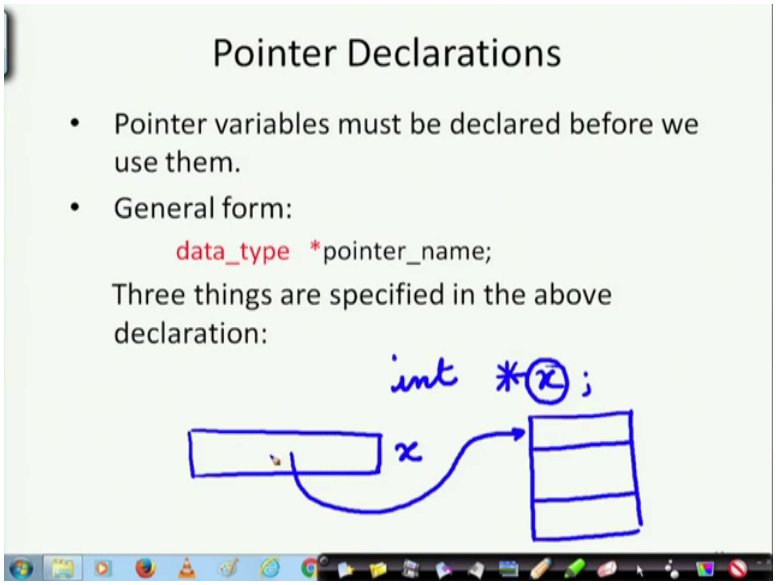
(Refer Slide Time: 13:12)



So, 10 is stored in location so and so, 2.5 is stored in location so and so, all right.

Now, here incidentally is just incidental that just for the sake of example, all these are contiguous locations, but they may not be contiguous locations also all right. So, 10 is stored in location, a is a was having the value 10 is stored in location so, and so like that it goes on.

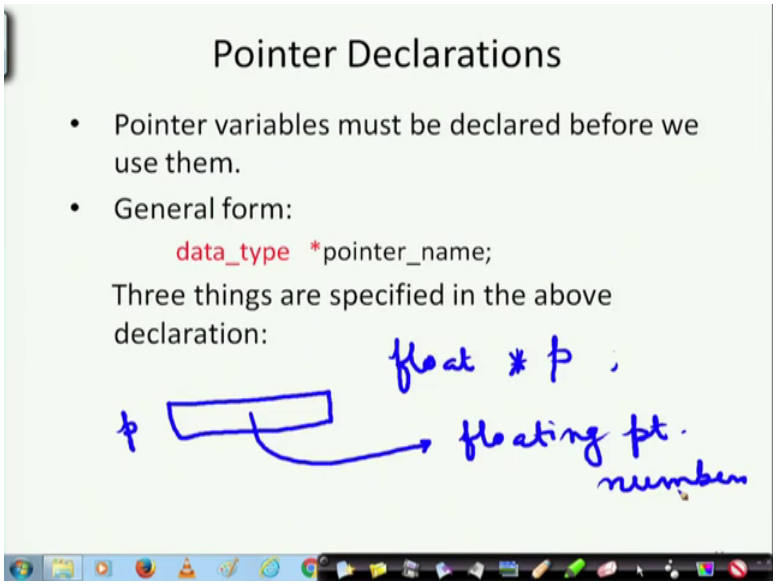
(Refer Slide Time: 13:46)



So, pointer declarations when we declare some variable as a pointer, the typical the standard form is this, data type shown in red that is very important and this star this is

something new that you also saw this in when we are discussing call by reference, for example, int is a type star x y z x what does it mean? It means that x is a variable of type integer, x is a variable sorry. So, I am sorry absolutely sorry x is a pointer, which is enabled or which is allowed to 0.2 variables of type integer only integers can be pointed out pointed to by x.

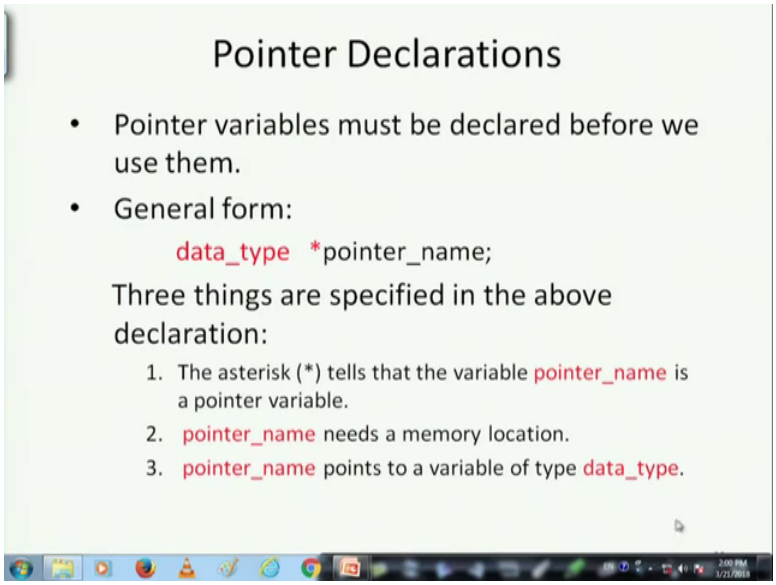
(Refer Slide Time: 15:02)



So, this is similarly I could have said float star p ; that means, p will be a pointer, that can only point to floating point numbers only all right.

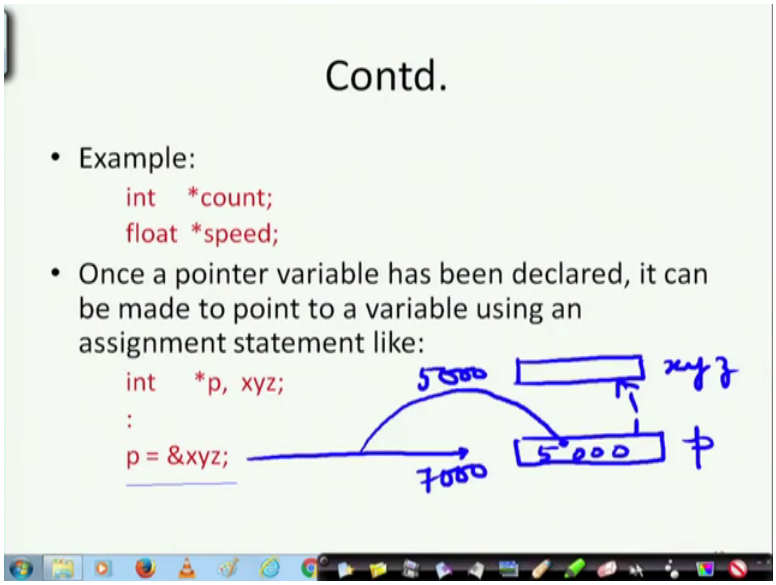
So, three things are specified in this disk in this declaration one is that.

(Refer Slide Time: 15:33)



This star tells that the variable pointer name or p whatever you call it, is a pointer variable not a normal variable and asterisk is. So, is telling that it is a pointer variable. Pointer name is a variable therefore, it needs a memory location and pointer name points to a variable of the specified data type. So, these three things you must remember when you are handling with the pointer.

(Refer Slide Time: 16:08)

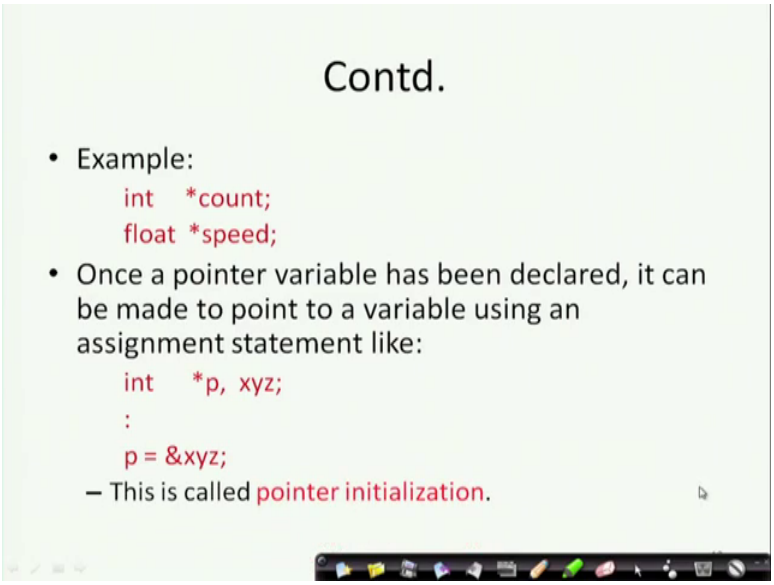


So, here is an example int star count what does it mean? That means, count is a pointer variable why a pointer variable? Because this is preceded with the star and where can

this pointer variable 0.2? To all data type data variables of type integer speed is again a pointer because it has got this asterisk and where can speed point to? Speed can point to variables of type float. Once a pointer variable has been declared, it can be made to point to a variable using an assignment statement like this int star p x y z.

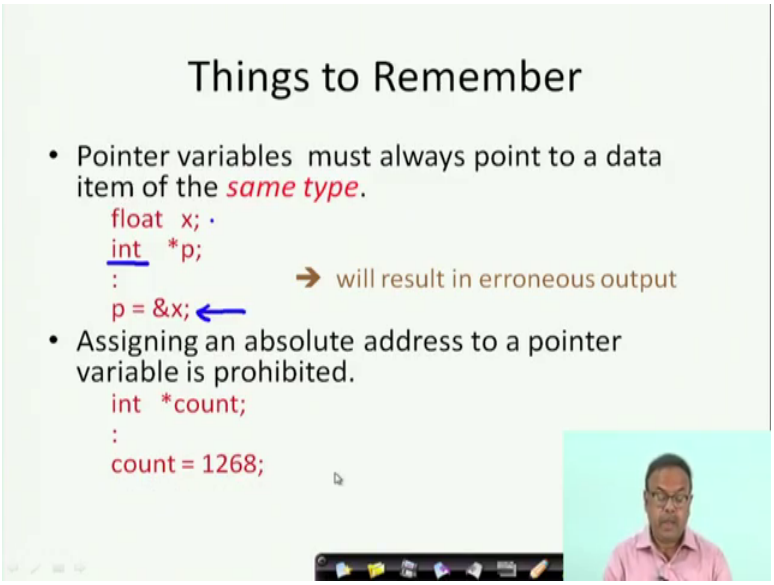
So, int star p is a point p is a pointer variable, x y z is an integer. So, you see by the same declaration I have declared two things one is a pointer to an integer and an integer. So, if I make such an assignment like p is assigned and x y z. So, x y z is here and its location can be 5000 and p is a pointer variable and so, when I do this assignment, this might be in the location 7000, but when I do this what it does is it loads this 5000 here. So, now, p is pointing to x y z. I hope it is clear now all right.

(Refer Slide Time: 17:50)



So, this is called pointer initialization.

(Refer Slide Time: 17:51)

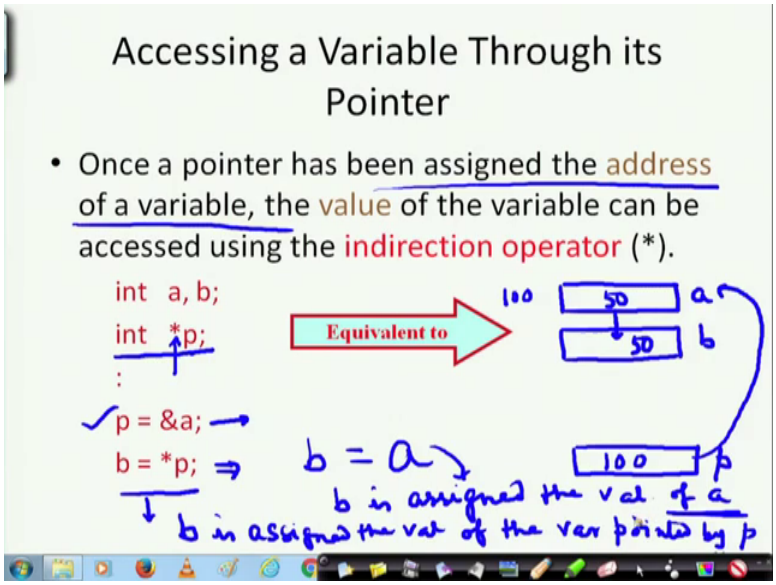


The things to remember is the pointer variables must always point to an item of the same type one pointer variable, either it points to an integer or it points to a float or it points to an array of characters whatever it is.

Assigning an absolute address now so, here for example, this is an error why it is this is an error? x is of type floating point, x is a variable and p is a pointer which is allowed to point two only integer, but here I have made an assignment where I am assigning to p, the address of the floating point number that is not allowed.

So, therefore, I am forcing p to point to x, but the type of p and the type of x are different. So, assigning an absolute address to a pointer is prohibited you cannot do this, you cannot you cannot do this you cannot force a pointer to a constant value, that you must keep in mind.

(Refer Slide Time: 19:00)



So, how do I access a variable through a pointer? Once a pointer has been assigned the address of a variable, the value of the variable can be accessed through the indirection operation. Now here you have to think of again we are talking of stars. So, let us give an example first.

For example, a b are two integers let us draw this I am sorry, suppose a is a location, b is another location both of them are integers and p is a variable, which is allowed to point to integers and what I do? I assign to p the address of a. So, the address of a let us say is 1000 or 100. So, 100 is written here. So, it comes here.

Now, I say. So, this what this statement do? 100 was the address of a that has been assigned to p. Now what is being done by this? B is being assigned star p; that means, whatever is b is getting the content of where p is pointing at suppose this was 50 then b is getting 50; So, little bit of confusion can occur because of these two star ps. Please understand that this star is just telling you that p is a pointer because it is coming in a declaration statement. On the other hand here is not a declaration statement is an assignment statement I have already declared p to be a pointer. So, p is a pointer.

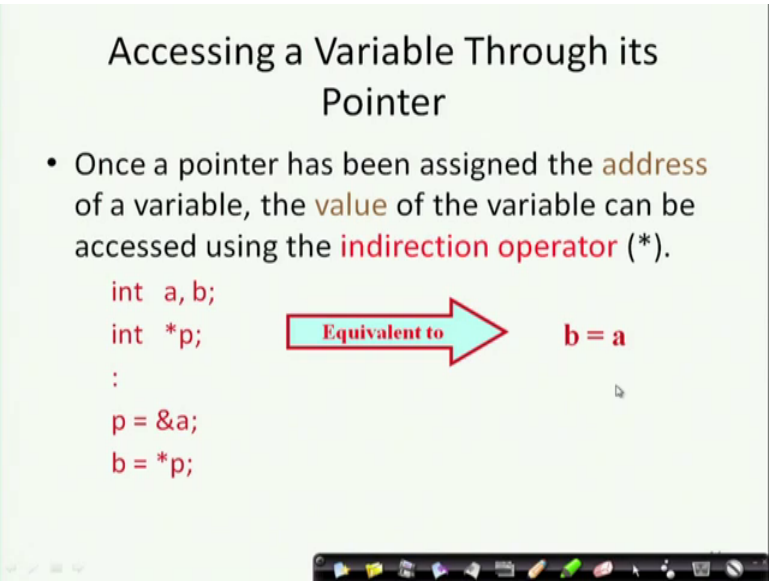
So, once a pointer has been assigned the address of a variable, once the pointer has been assigned the address of the variable that is done in this step the value of the variable value of the variable can be accessed using the indirection operation; that means, which variable say I could have done this p assigned a, but I am not doing the sorry p assigned

b assigned a. Instead of doing that what I am saying if I makes just note my two English statements. One is the value of a is being assigned to value to b statement number 1. Statement number 2 is our b is being assigned the value of a statement number 1; statement number 2 is b is being assigned the value of the variable that is being pointed at by p clear.

So, here one is b is assigned the value of a one statement here that is this statement. And this statement is b is assigned the value instead of a, I am saying is being assigned the value of the variable pointed by p.

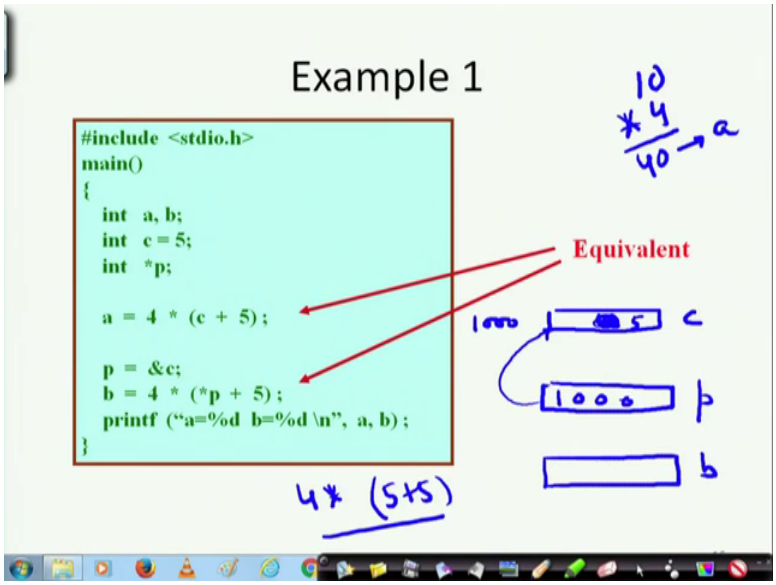
So, this is an indirect way of saying that. So, this is something that you must very clearly understand look at this once again.

(Refer Slide Time: 23:21)



So, I can now say that this thing is equivalent to b assigned a, but I have done in a indirect way.

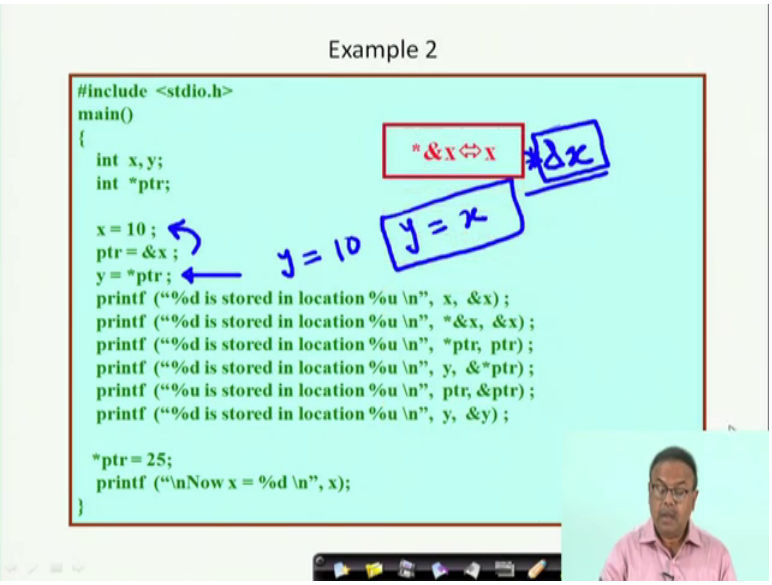
(Refer Slide Time: 23:21)



If you have understood this then pointer should be clear to you, here look at one example integer a b c assign 5 and p is a pointer to integers. Now, in this statement what has been done a is being assigned 4 times c plus 5. So, c is being added to 5. So, it is becoming 10 and 4 times 10 is 40, 40 is being assigned to a. Here what is being done? P is being assigned c is address. So, wherever c was c was say address 1000. So, that is being assigned to a variable p. So, p is becoming 1000 all right.

Now, b what is b being assigned? b is being assigned 4 times star p plus 5 what is star p? Star p sorry c is a star p is nothing, but the variable c is pointing to that and whatever was its value suppose it was 10, it was sorry it was 5. So, that value is being taken and 5 is being added to that. So, it becomes 10 times this. So, these two are essentially equivalent all right.

(Refer Slide Time: 25:17)

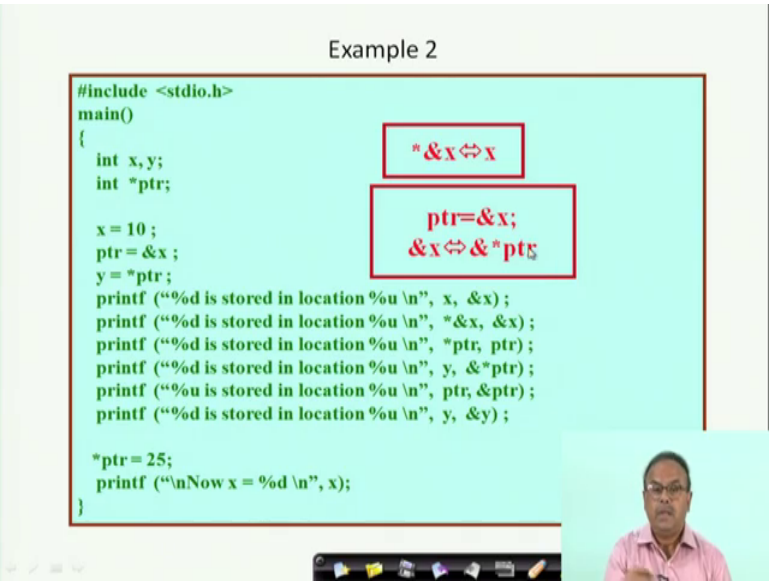


So, here is another example you have seen this example very similar to this that I am defining a pointer p t r is a pointer, x is 10 you just think of what it should print. X is 10, p t r is the address of x, p t r is the address of x here and y is being assigned star p t r what does this mean? P t r is pointing to 10 and y is being assigned to star p t r; that means, y is being assigned to 10 or y is being rather I should say y is being assigned x and x was 10. So, y is become becoming 10.

Now, if I print percent is stored x x x instead of x what will be printed? Instead of x the value of x will be printed 10 is stored in the address this. Similarly now what is this star and x what does this mean? And x is the address of x and where that is pointing to these two should be same.

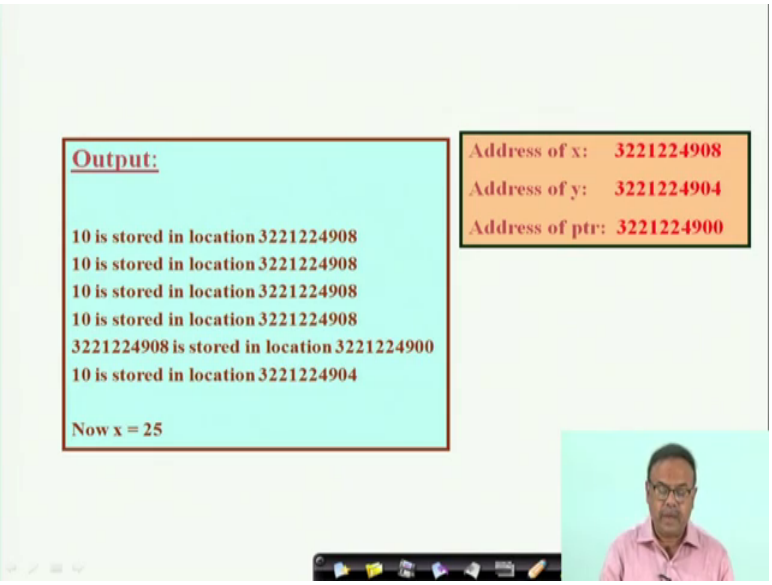
So, if you look at if you look at the result this is equivalent why? And x is the address of x and star just like star p; that means, the content of where the pointer is pointing to. So, these two are same right.

(Refer Slide Time: 27:18)



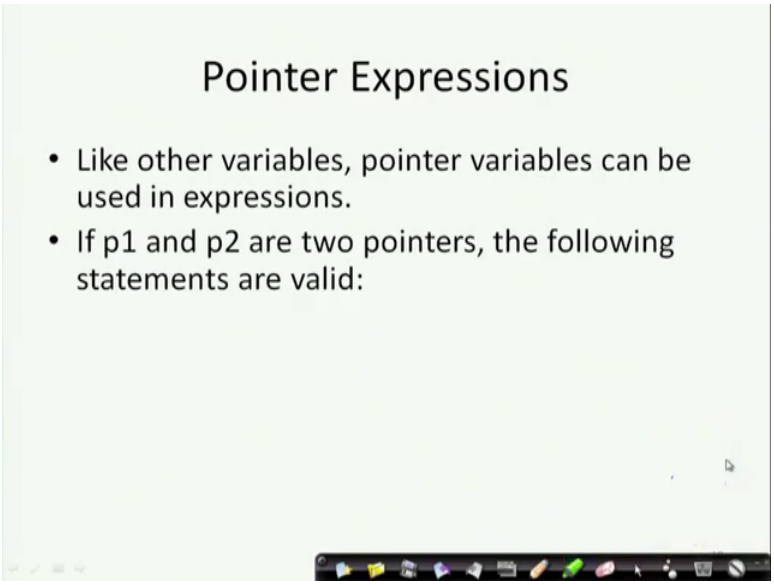
So, ptr is and x ptr has is the same and x is nothing, but ptr. So, these two are all equivalent ok. If I say and x; that means, I am taking the address of star ptr.

(Refer Slide Time: 27:37)



So, this is something you can toy with yourself and you will get a printout like this. So, these two are the same.

(Refer Slide Time: 27:46)



Now, here we are coming to something more which is known as pointer expressions and we will deal with that in the next lecture.