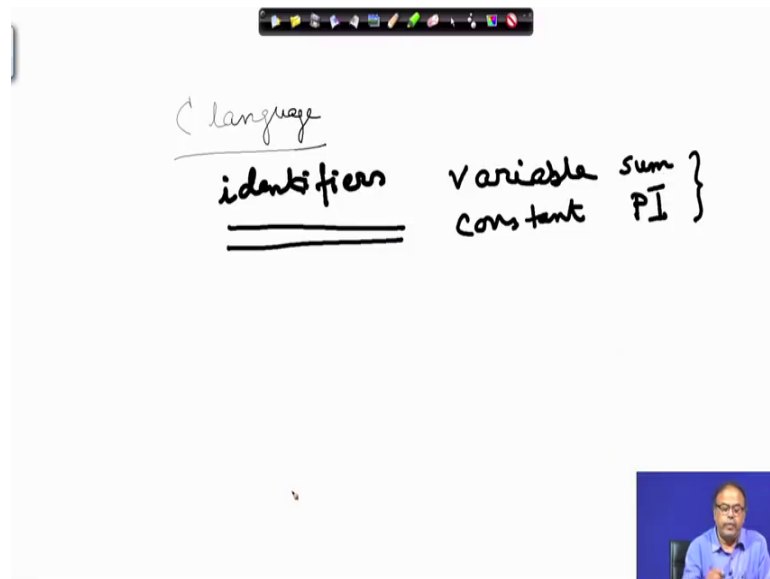


Problem Solving Through Programming In C
Prof. Anupam Basu
Department Of Computer Science & Engineering
Indian Institute of Technology, Kharagpur

Lecture – 10
Address and Content of Variables and Types

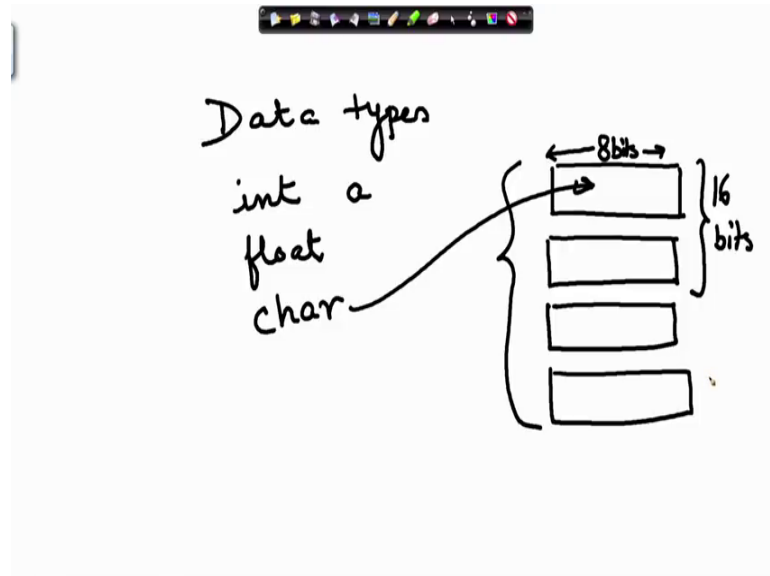
So, till now we have looked at how in C language.

(Refer Slide Time: 00:29)



We can write the identifier, this is becoming a little too thick, which identifiers are used for used for writing the variable names and the constant name. So, for example, PI or some variable name suggest sum all those things. Now we have seen what are the rules that the language c imposes on writing the names of variables or constants, that is how we can write the identifier alright.

(Refer Slide Time: 01:30)

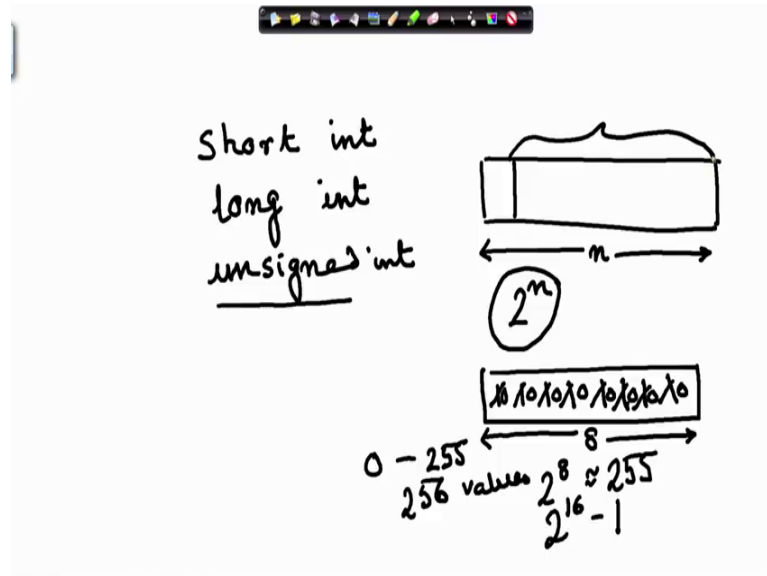


Also we have seen that there are different data types like int which stands for integer, and float which represents floating point numbers or real numbers, and char is used to specify some data which is of type character. Will see some more examples of this in the course of these lectures, now we also know that int means integer and whenever a particular variable `a` is declared to be an integer, then typically it also varies from machine to machine and compiler to compiler, 2 bytes or 16 bits are located for storing one integer alright.

Each of these boxes are that I am showing here are 8 bits wide. So, there are 2 such. So, 16 bits, for float we will have 2 more real numbers are stored using 32 bits whereas, characters are typically stored in 8 bits. Now this is not so sacrosanct as in some machine which is much more powerful and much more accurate having high resolution we can have 32 bits for storing integers 64 bits for storing floating point numbers and characters can be a bit 16 bits.

However depending on what is the type declaration, the amount of storage the amount of memory that is allocated to a particular variable varies ok.

(Refer Slide Time: 03:37)



We just like int float char we also have got some more like short int alright, long int or unsigned int. These are also different data types, will come across his in the course of this lecture. Short int means just if in an integer takes 2 bits or short int will take one bit if a long int if a int take 16 by 16 bits; that means, 2 bytes or long int can be made to consume 4 such bytes or 32 bits, but still that will be an integer. So, in this 4 bytes an integer that that will be stored; that means, a integer accuracy will be much larger.

So, depending on the number of bits I allocate to a particular variable, depending on the number of bits suppose I allocate n bits the range of values that I can represent varies for example, if there be n bits then I can go from I can have 2 to the power n distinct values stored for example, if there be 8 bits, then the maximum value that I can store is when all these 8 bits are once and that is your knowledge of binary arithmetic will tell you that this will be 2 to the power 8 right that is 256 it will be actually 255 alright. So, 255 and if I make everything 0 if each of them are made 0 all zeros will be 0 right, I can have the range from 0 to 255; that means, total 0 to 255; that means, I can store any of distinct 256 values 2 out of 256 values distinct values I can store any one of them. Now if this 8 would have become 16, then my maximum range would be 2 the power 16 minus one right that is the maximum value that I can store it.

Now signed and unsigned mean sometimes in our presentation we keep one bit for the sign part in that case of course, the range decreases, but if I go for unsigned, then we remaining within say 16 bits to bytes.

I can have a larger representation we will go will encounter these details as and when we need them.

(Refer Slide Time: 06:46)

Some Examples of Data Types

- int
0, 25, -156, 12345, -99820
- char
'a', 'A', '*', '/', '\'
- float
23.54, -0.00345, 25.0
2.5E12, 1.234e-5

int yourvar
char myvar
myvar = "a"
yourvar = 10
E or e means "10 to the power of"
myvar = ''

Now, let us come to some examples of the data types. You can see integers 0, 25 minus 156 all these are examples of integers. Now here I am for the first time showing some characters. Now the character values are you see the character values have got something special say I am declaring some variable char as type char my variable, I name that my var my var is of type character and I assign I want to assign to my var I want to assign that my variable will hold the character a.

Now, when I am assigning a character, then I have to put a single quote around this. For example, I had another variable int yourvar. Yourvar is another variable which is of type int. So, if I assign into that in yourvar then I can stay to say an integer value 10, but when I right onto a character a character constant has to be always in cap, I mean encapsulated with into single quotes right as here.

Now, this single quote within the single quote is a character slash, what about this this is just single quote single quote; that means, we win that there is a blank so; that means, I

am see if I say myvar is assigned this; that means, myvar will be assigned a blank character. Now you should remember that each of these characters, that we type in have got an ASCII value each of these characters have got an ASCII value.

(Refer Slide Time: 09:02)


Some Examples of Data Types

- int
0, 25, -156, 12345, -99820
- char
'a', 'A', '*', '/', ''
- float
23.54, -0.00345, 25.0
2.5E12, 1.234e-5

ASCII

a
b
c
A
B
C
1
2
9

E or e means "10 to the power of"



What is ASCII? ASCII stands for American standard code for information interchange. Now according to this table for every English character a b c d and capital A B C D and 1 2 two everything up to 9 or all of them have got some particular code American standard code and that is accepted in all the computers. So, whenever I type an a, when I strike the key on the keyboard when I strike the key a right whenever a strike a then actually when I as I press a what goes inside the computer is an ASCII code of a alright.

Now, this ASCII code of a will be store therefore, and the code for b the quote for capital A are all distinct. So, whenever I type in a character from the keyboard a particular ASCII code goes in whenever I assign some value to a variable for example, as I did right now myvar signed a this means that myvar will now, have done the ASCII code of a alright it will have the ASCII code of a.

(Refer Slide Time: 10:32)


Some Examples of Data Types

- int
0, 25, -156, 12345, -99820
- char
'a', 'A', '*', '/', ''
- float
23.54, -0.00345, 25.0
2.5E12, 1.234e-5

myvar = 'a'

E or e means "10 to the power of"

2.5×10^{12}
 1.234×10^{-5}



Now the third variety that is these three are very common float for example, 23.54 or minus 0.00345 25.0 or I can also write it in this way. 2.5 E12 what does this mean?

This means, it is 2.5 times 10 to the power 12, what does this mean? This means 1.234 times 10 to the power minus 5, because it is E minus 5 here it is E 12 I can use capital E or small e that really does not matter these are the examples of floating point constant ok.

(Refer Slide Time: 11:44)

Some Examples of Data Types


- int
0, 25, -156, 12345, -99820
- char
'a', 'A', '*', '/', ''
- float
23.54, -0.00345, 25.0
2.5E12, 1.234e-5

float x, y, z;
x = 23.54;

E or e means "10 to the power of"

y = 2.5e12
z = 1.234E-5

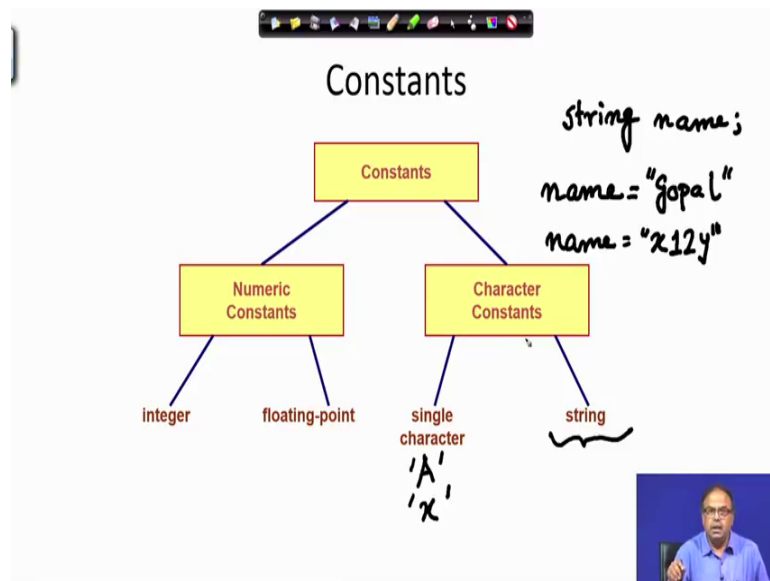
1.234×10^{-5}



So, if I have a variable like float x and I assign I can assign x to be say x is 1, y is 1, z is another one. So, I can assign x to be 23.54 semi colon or I can assign y to be 2.5 E 12 and z to be 1.234 E minus 5 alright.

That means now z will have the value 1.234 times 10 to the power minus 5 that is how we represent the floating point numbers given this. So, this part is clear that is how we write the variables now and the these are the examples of data types.

(Refer Slide Time: 12:47)



Now, coming to constants the constants can give integer constants on floating point constants just the once that we I was showing right now. But there is another type of constant character constant so on. We have already seen example of character constants of single character like we had say a sorry what is happening here; can have a single character like A or say x all these within a single quote a single characters, and there is another type of character constant which is a string.

For example I can have a string which is another type of character constant which is not a single character, but a string of characters. For example, I want to write down my name the name of a person. So, string type variable name to name and I can assign some value to the string like say g o p a l alright. So, this is a number of characters taken together is forming a string of characters, it could also be named to be x 1 2 y double quote. Now note that in this case I am using double quote where is for single character I was using

single quote. Now these are some of the rules of defining character constants on numeric constants in C.

(Refer Slide Time: 14:55)

The slide is titled "Integer Constants" and contains the following content:

- Consists of a sequence of digits, with possibly a plus or a minus sign before it.
 - Embedded spaces, commas and non-digit characters are not permitted between digits.
- Maximum and minimum values (for 32-bit representations)
 - Maximum :: 2147483647
 - Minimum :: -2147483648

Handwritten mathematical notes on the slide include:

- A circled "32" next to the text "(for 32-bit representations)".
- The expression 2^{32} with a vertical line and a "0" below it, indicating a range from 0 to $2^{32}-1$.
- The expression $2^{31}-1$ and -2^{31} .

At the bottom right of the slide, the number "25" is visible.

Next we move to you have seen the integer constant. Now couple of things to be just mentioned that the maximum and minimum number of values, that can be stored as an integer constant is dependent on how many bits are allocated for the presentation. For example, as I said that for 32 bit representation, I can have 2 to the power 32 different combination alright. So, if you compute this you will find that on one side here is 0 when everything.

If I take one bit to designate positive or negative, then I will be left with that 31 bits. So, the maximum I can have on the positive side is 2 to the power of 31 minus the 1 middle one is 0 alright and I can go up to this and on the other side I can go up to 2 to the power minus 2 to the power 31 right. So, this is the range, there is a maximum integers and the minimum integer that I can represent.

But; obviously, we need not be so concerned about it, because that varies with the number of bit representation in the machine. So, for a 64 bit representation; obviously, this size will be doubled; will be much larger I am sorry it will it will be much larger.

(Refer Slide Time: 16:32)

Floating-point Constants

- Can contain fractional parts.
- Very large or very small numbers can be represented.
23000000 can be represented as 2.3e7
- Two different notations:
 1. Decimal notation
25.0, 0.0034, .84, -2.234
 2. Exponential (scientific) notation
3.45e23, 0.123e-12, 123E2

Handwritten annotations:
- A box around 0.123 in 0.123×10^{-15}
- A box around 123 in a separate box
- A callout box: "e means '10 to the power of'"


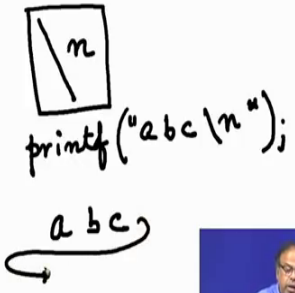
So, we have also seen floating point numbers just now. So, I do not need to repeat this, and where it why are we going for this this type of why are you going for exponential type of representation, because that enables us to represent much larger numbers and very small numbers also using less number of bits because I can always write 0.123 into 10 to the power minus whatever.

So, here minus 12 I could have written minus 15. So, it has got 2 num 2 parts one is the mantis apart that is this part I just put the decimal part 1 2 3 be present in binary somewhere and on this side I put some bits for the exponents. So, it can be minus 15 plus 15. So, using less number of bits I can increase the range and can go for a much larger range of a presentation.

(Refer Slide Time: 17:48)

Single Character Constants

- Contains a single character enclosed within a pair of single quote marks.
 - Examples :: '2', '+', 'Z'
- Some special backslash characters
 - '\n' new line
 - '\t' horizontal tab
 - '\'' single quote
 - '\"' double quote
 - '\\' backslash
 - '\0' null



This one we have already explained that single character constants, now here of course, you can see that this operator plus also has gotten ASCII quote every character has gotten ASCII representation whatever we have we find on the keyboard has gotten ASCII representation. Therefore, I can also have capital z or plus as a character now here is something that is a little new to you we have already encounter one of these as a friend earlier, here you can see that we are using a special character like backslash.



This backslash means that whatever is following a backslash is not the normal nature of that for example, if I write n, it really does not mean an a character n, but backslash n has got a different meaning alright. For example, suppose I was writing something printf say I write you have seen that example earlier printf suppose I am just writing a b c and then I put backslash n; that means, I am I will be painting a b c, but after that I will not print n, but since its backslash n, its a some other information it is telling us that go to the new line. So, immediately we go to the new line. Similarly we can see that backslash t this one is the horizontal tab.

(Refer Slide Time: 19:37)

Single Character Constants

- Contains a single character enclosed within a pair of single quote marks.
 - Examples :: '2', '+', 'Z'
- Some special backslash characters

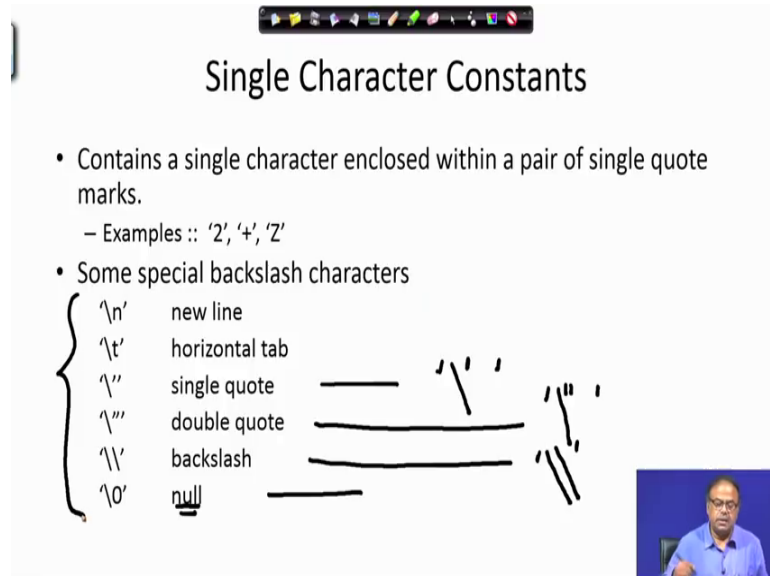
'\n'	new line	
'\t'	horizontal tab	←
'\''	single quote	←
'\"'	double quote	
'\\'	backslash	
'\0'	null	



So, if I have backslash t my cursor will move from here to some fixed tabular distances right.

Backslash now you know single quote or double quote; single quote if I put a charac if I just want to print the characters single quote how do I do it? I will do it because any character have to do it in this quote. Now if I put single quote here then it will be confused it will take these 2 and will take a blank character in between, because a blank character is represented as n is blank with into single quotes, but I really want that here not blank, but I want to print the single quote.


(Refer Slide Time: 20:29)



Single Character Constants

- Contains a single character enclosed within a pair of single quote marks.
 - Examples :: '2', '+', 'Z'
- Some special backslash characters

}	<code>\n</code>	new line	
	<code>\t</code>	horizontal tab	
	<code>\"</code>	single quote	<code>'</code> <code>'</code>
	<code>\"</code>	double quote	<code>"</code> <code>"</code>
	<code>\\</code>	backslash	<code>\</code> <code>\</code>
	<code>\0</code>	<u>null</u>	<code>0</code>



So, in that case what should I do? For this should take the signal quote and then backslash single quote back I mean single quote; that means, this single quote is different from these 2 single quotes. So, these are the boundaries of the character representation and what is the character? That is single quote similarly for double quote you can now very easily reason that I must enclose it with in single quote, and then backslash double quote followed by single quote .

Similarly, if I want to print backslash what should I do? Single quote then backslash; that means, it is something different, backslash single quote similarly backslash null is backslash 0 alright.

(Refer Slide Time: 21:38)

String Constants

- Sequence of characters enclosed in double quotes.
 - The characters may be letters, numbers, special characters and blank spaces.
- Examples:
"nice", "Good Morning", "3+6", "3", "C"
- Differences from character constants:
 - 'C' and "C" are not equivalent.
 - 'C' has an equivalent integer value while "C" does not.

Handwritten annotations:
A bracket under "3+6" in the examples list.
A bracket under "C" in the differences list.

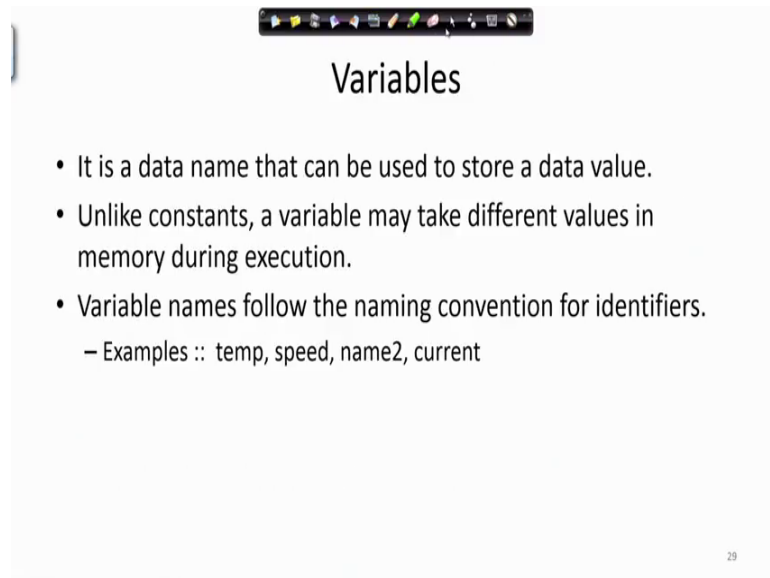
28

So, these are some special character constants that we may encounter during our programming practice. The other new things that we have learnt is string constant; now string constants are us are sequence of characters its a sequence of characters enclosed within double quotes.

Just like we wrote that the characters may be the characters within the double quote may be letters, numbers, special characters bank blank spaces like that for example, nice good morning this is a blank here, now what will happen with this. When I put this as a string do not think that it will be computed and printed as 9. It is just a string that will be printed. So, if I write in this way within double quote if I write three plus six then just 3 plus 6 that string will be printed. The difference between the with character constants is that, backslash I mean the single quote c this is a character this is a string and they are not equivalent.

Because their representations we will see we will be internally there will be represented in a different way this one has got an equivalent integer value that is ASCII code whereas, this does not have an ASCII code this is something different where they will be c and something more which will see later. So, string constants thing only thing to remember is, string constants are a sequence of characters which can be letters numbers expressions whatever this sort of operator special characters enclosed within double quotes alright that is a string character .

(Refer Slide Time: 23:40)



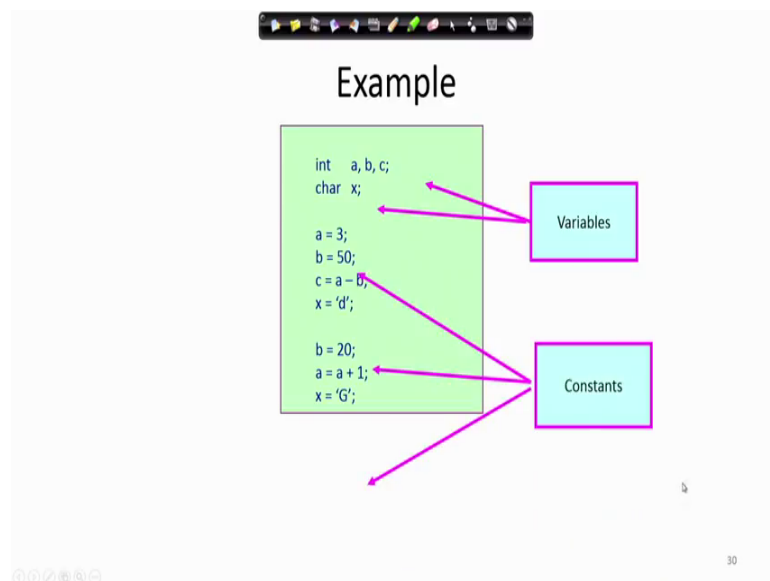
Variables

- It is a data name that can be used to store a data value.
- Unlike constants, a variable may take different values in memory during execution.
- Variable names follow the naming convention for identifiers.
 - Examples :: temp, speed, name2, current

29

Now, we already know what variables are. So, we do not need to repeat that.

(Refer Slide Time: 23:47)



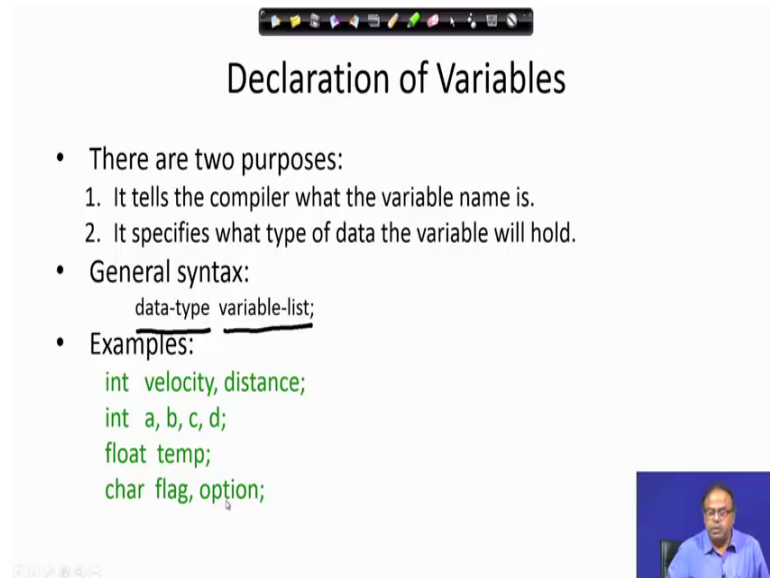
Example

```
int a, b, c;  
char x;  
  
a = 3;  
b = 50;  
c = a - b;  
x = 'd';  
  
b = 20;  
a = a + 1;  
x = 'G';
```

30


And we have we have seen the variables.

(Refer Slide Time: 23:51)



Declaration of Variables

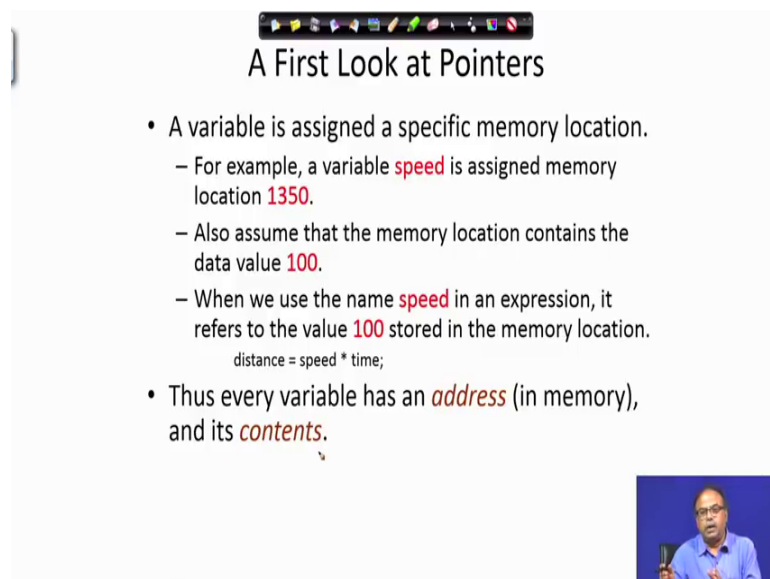
- There are two purposes:
 1. It tells the compiler what the variable name is.
 2. It specifies what type of data the variable will hold.
- General syntax:
`data-type variable-list;`
- Examples:
`int velocity, distance;`
`int a, b, c, d;`
`float temp;`
`char flag, option;`



We know that the variables are to be declared and the general syntax is a particular data type.


Sorry is it will be a particular data type followed by variable list right. So, like examples we have already seen, `int velocity distance` `int a b c d`, `a b c d` `velocity distance` all integer variables `temperature` is a `float temp` is a floating point variable, `flag option` these are character type of variables we have already seen them right.

(Refer Slide Time: 24:29)



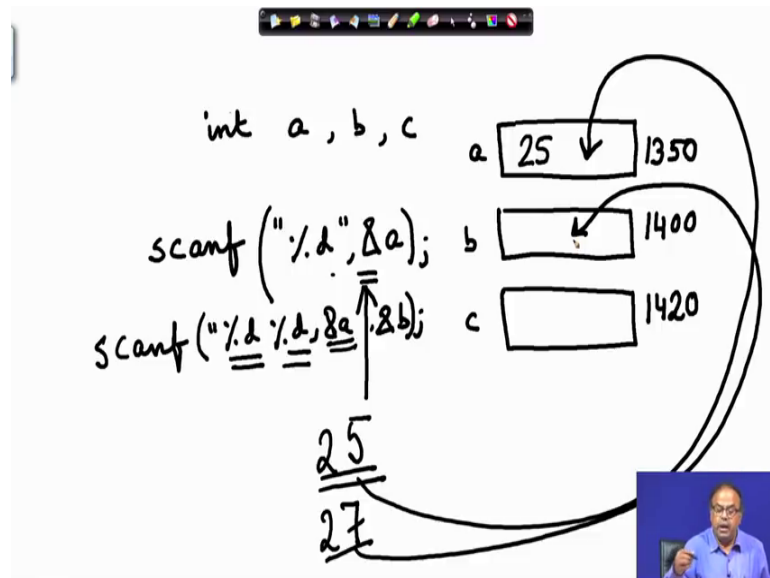
A First Look at Pointers

- A variable is assigned a specific memory location.
 - For example, a variable `speed` is assigned memory location `1350`.
 - Also assume that the memory location contains the data value `100`.
 - When we use the name `speed` in an expression, it refers to the value `100` stored in the memory location.
`distance = speed * time;`
- Thus every variable has an *address* (in memory), and its *contents*.



Now, we come to something that we evaded till now. Pointers have got big role in C programming, but we will just have a very simple look at the pointers. Here pointer means basically address alright. So, you please forget about the title for the time being, a variable is assigned a specific memory location that we know and that memory location is assigned by the compiler. So, if we have some variable say when we find out int a b c.

(Refer Slide Time: 25:14)



Then as we have discussed earlier a b c are three memory location, which are the sign by the compiler which of these memory locations actually have got an address right. So, the address can be say this is 1350 is an address just like our houses have an address just like your rooms have got some numbers, just as your drawers may have some levels. So, similarly might be this is 1400, this is say 1450 or 1420 suppose a b c has got this 3 addresses are right.

Now, when I read when I try to read something, we know that I need to scanf. Now in scanf what I did is percentage d and a; that means, I am trying to read the variable a, but I did not explain to you earlier why I put this. And this and means that and you know what is this percentage d. So, I have got some space some space to hold an integer and that space is the in the a variable, but when the you from the keyboard type in say the value 25.

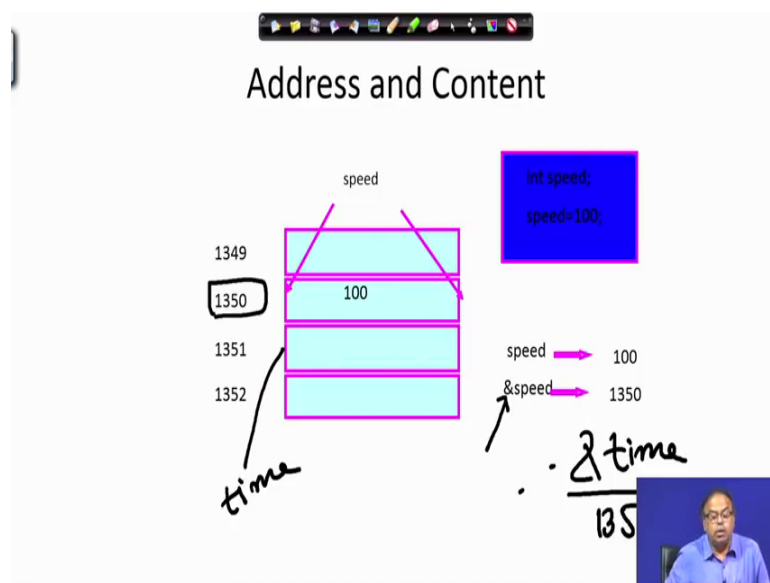
When will that value go? The value will go to the address of the variable a, what is the address of the variable a? 1350. So, it will go to 1350, 25 will come here similarly when

we write say scanf percentage d, percentage d comma and a and b then I am going to read 2 values and 2 integers, and the address of the first one upon suppose type in 25 and 27. So, for the add 25 will go to the address of a that is a 1350 and 27 will go to the address of b that is 1400.

Given this lets now read this a variable is a sign the specific memory location we know that. For example, of variable speed is assigned memory location 1350 and assume that the memory location contains the data value 100. So, when we use the name speed in an expression it refers to the value 100. So, for example, when we write distance is speed into time, then it will take this speed from this location 1350. Every variable has an address and its contents. So, we have seen a has got an address a is a variable a has got an address 1350.

And when I write the 25 into that 25 is a content. So, address and content we had earlier discussed also.

(Refer Slide Time: 29:06)



But you see here integers speed I think you can read it and. So, speed is this particular location that is in 1350 and when I right when I write speed equals speed assigned 100, then 100 is written over here alright when I assign it. So, speed is getting the value 100 whenever, but when I say what is and speed when I am asking the question what is the address of the variable speed, what is the address of the variable speed and the answer would be 1350.

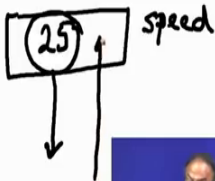
So, this and sorry this and operation this and operation is nothing, but asking for the pointer to speed or the address to the variable speed. So, this should be the answer. So, and of suppose here time is given here, if I just say and time what will that be returned what is and time? And time will be 1351 something of this sort alright. So, that is another thing that we needed to understand what is the purpose of this and.

(Refer Slide Time: 30:38)


Contd.

- In C terminology, in an expression
`speed` refers to the contents of the memory location.
`&speed` refers to the address of the memory location.
- Examples:

```
printf ("%f %f %f", speed, time, distance);  
scanf ("%f %f", &speed, &time);
```



The diagram shows a rectangular box containing the number '25'. An arrow points from the box to the word 'speed' written to its right. Another arrow points downwards from the bottom of the box.



A small video inset in the bottom right corner shows a man with glasses and a blue shirt, likely the presenter.

So, here in C terminology `speed` refers to the contents of the memory location, and `&speed` refers to the address of the memory location corresponding to the variable `speed`. So, let us come to this example `printf` percentage f percentage f percentage f; that means, I am going to print three floating point numbers and what are the variables c floating point values `speed` `time` and `distance`; that means, what am I going to print look here I am going to print the contents of the memory location `speed`, the content of the memory location `time`, the content of the memory location `distance`.

And when I am reading percentage f percentage f and `speed` and `time`; that means, what that I am reading where I am reading in the address of the variable `speed`, I am reading in the address of the variable `time`. So, this is required to be understood. So, basically when I have say `speed` I once again repeat, suppose `speed` is 25 and I print `speed`; that means, I am printing the content of this location `speed`, but whenever I am reading into `speed` where am I reading the value? I am reading into the address of `speed` that is this location.

That is the main difference between these 2 alright. Let us stop here in the next lecture, we will straight way move head to write some c expressions, because till now whatever we have learnt are the bits and pieces the tools of c that is how the how just like in the language how the word are written what are the some of the simple rules, but then we will have to learn writing the real sentences in a language. So, that it we will start from the next lecture

Thank you