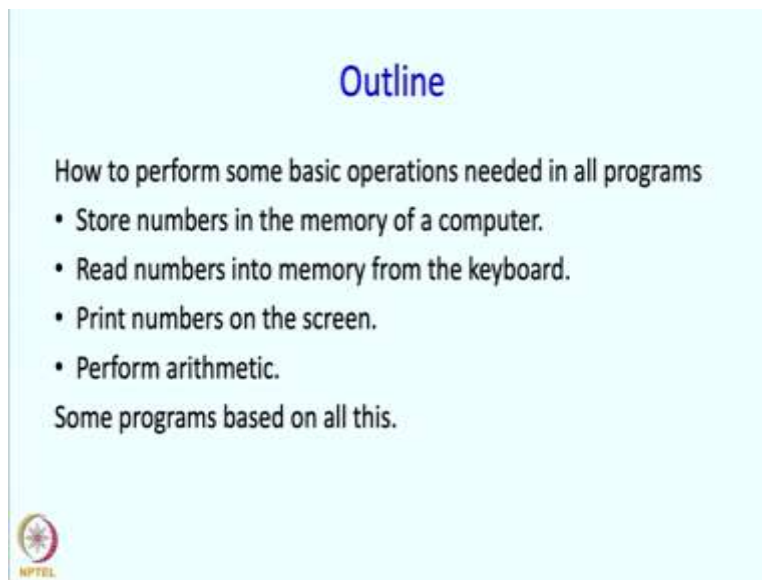


Introduction to Programming through C++
Professor Abhiram G. Ranade
Department of Computer Science and Engineering
Indian Institute of Technology Bombay
Lecture No. 3 Part – 1
Basic Elements of Program
Variables and data types, Introduction.

Hello, and welcome to this second week first lecture sequence of the NPTEL course on an Introduction to programming through C++. What I am going to do today is something like the following.

(Refer Slide Time: 0:34)




Outline

How to perform some basic operations needed in all programs

- Store numbers in the memory of a computer.
- Read numbers into memory from the keyboard.
- Print numbers on the screen.
- Perform arithmetic.

Some programs based on all this.




I am going to talk about how to perform some basic operations that are needed in all programs. So the first amongst these is using the memory of the computer, how do you store numbers in the memory of a computer? Then, how you read numbers from the keyboard into the memory? How do you print the numbers in memory on to the screen? How do you perform arithmetic? Clearly this operations are required in all programs and they sort of form the base of whatever here we are going to the study next and we are going to talk about all this today. Okay, and during the lecture and towards the end we will see programs based on all of this.

(Refer Slide Time: 1:19)

Reserving memory for storing numbers

- Before you store numbers, you must explicitly reserve space for storing them.
 - “space” : region of memory
- This is done by a “variable definition” statement.
- **variable**: name given to the space you reserved.
 - “Value of a variable”: value stored in the variable
- You must also state what kind of values will be stored in the variable: “data type” of the variable.




So the first task is reserving memory for storing numbers. So before you want to store the numbers into the memory, you must explicitly reserve space for storing them. And in this course whenever we talk about “space” we will mean region of memory. And this reservation done by a “variable definition” statement. The term ‘variable’ is used to denote the space that we have reserved. It is the name of a the region of a memory, and you get to decide this name, how? We will tell you in a minute. The value of variable likewise means the value stored in that region of memory that is just reserved. Then, in addition to defining the variable while giving its name you also must state what kind of values will be stored in the variable. So basically these value are indicative of the type of the data and we will see this in a minute.

(Refer Slide Time: 2:34)

Variable creation/definition

Statement form:
data-type-name variable-name;

- Example from chapter 1:
int nsides;
- int : data type name. Short for "integer".
 - Reserve space for storing integer values, positive or negative, of a "standard" size.
 - Standard size = 32 bits on most computers.
 - Two's complement representation will typically be used.
- nsides : name given to reserved space, or the created variable.




So, a statement for creating variables or defining variables. So this has the form 'data-type-name' and 'name-of-the-variable'. We have already done this in chapter 1 or in lecture 1 and the example that we had there was `int nsides`. 'int' here is the data type name and it is short for "integer", it indicates at values that we are talking about storing are going to be integers. So this statement is going to reserve space for storing integer values positive or negative, of a standard size. And the standard size these days tends to be the 32 bits on most computers. And the values that are being stored in this variable are going to be interpreted as Two's complement values. So this sort of like having the sign bit and then a magnitude but it is not quite, but if you think about it as sign bit and magnitude that is okay. `nsides` is the name to be given to this region of memory that has been reserved for us, or, the region of memory is also called the variable that we just created and so `nsides` is the name of that variable.

(Refer Slide Time: 4:02)

Variable names: "Identifiers"

- Sequence of 1 or more letters, digits and the underscore "_" character
 - Should not begin with a digit
 - Some words such as `int` cannot be used as variable names. Reserved by C++ for its own use.
 - case matters. `ABC` and `abc` are distinct identifiers
 - Space not allowed inside variable name
- Examples: `nsides`, `telephone_number`, `x`, `x123`, `third_cousin`
- Non-examples: `#sides`, `3rd_cousin`, `3 rd cousin`
- Recommendation: use meaningful names, describing the purpose for which the variable will be used.

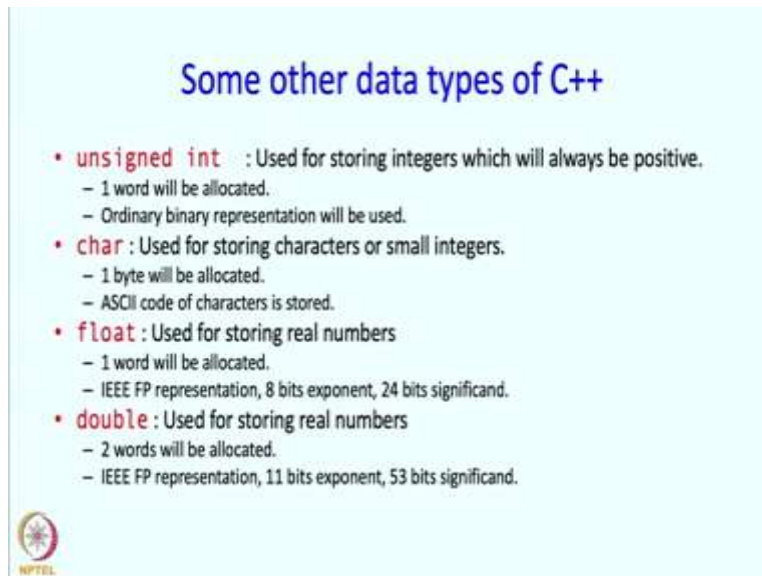


Now, names in C++ including names of variables are called "identifiers". So an identifier can be a sequence of 1 or more letters, digits and the underscore character. Now, a name cannot begin with a digit. And some words such as `int` cannot be used as variable names. So these terms `int` and so on are reserved by C++ for its own use and so they cannot be used as identifiers.

In C++ case is important, so whether you write something in the lower case or upper case does makes a difference. So I can have capital `ABC` as an identifier or I can have little `abc` as an identifier and these will be considered different identifiers. And the only characters used to make up a name and once mentioned above - letters, digits and underscore. So this means for example that you cannot have space inside the name of a character or once you give a name to a character you cannot break it up. If you break it up it might mean something else, you cannot break it up by putting spaces. So here are some examples, `nsides` is an example we have already seen, you can have a name called '`telephone_number`', you can have a name called '`x`', you can name called '`x123`', you could have a name or an identifier called '`third_cousin`'. What is not allowed however as identifiers or names of variables are something like '`#sides`' this is not allowed because hash is not allowed in identifiers. '`3rd_cousin`' looks okay but begins with a digit and we have to abide by the rule that names of identifiers cannot begin with digits. And '`3 rd cousin`' is not a valid identifier because it contain spaces.


Now this gives you a lot of choice in what to select as names and you are urged to use meaningful names, so basically name should describe the purpose for which the variable is going to be used.

(Refer Slide Time: 6:42)



Some other data types of C++

- **unsigned int** : Used for storing integers which will always be positive.
 - 1 word will be allocated.
 - Ordinary binary representation will be used.
- **char** : Used for storing characters or small integers.
 - 1 byte will be allocated.
 - ASCII code of characters is stored.
- **float** : Used for storing real numbers
 - 1 word will be allocated.
 - IEEE FP representation, 8 bits exponent, 24 bits significand.
- **double** : Used for storing real numbers
 - 2 words will be allocated.
 - IEEE FP representation, 11 bits exponent, 53 bits significand.



Besides the int data type, C++ has several other data types. So for example there is the ‘unsigned int’ data type, this is used for storing integers which are guaranteed to be positive, or alternately whatever is stored in such a variable will be of course a bit sequence as always but it will be interpreted as a non-negative value. And typically one word will be allocated, or 32 bits will be allocated. And in these 32 bits whatever will be stored interpreted as an ordinary binary number.

Char is also kind of integer, but it is small variable and it is used for storing characters or small integers. So it only has 1 byte width so it only consist of 8 bits and often the value which is stored in it will be interpreted as the ASCII code of some character so will see this in a minute.

Float is a data type which is use for storing real numbers and typically 1 word will be allocated. And this word will be used as per the IEEE floating point representation, using 8 bits from the word as exponent and 24 bits for the significant.

The double is another data type of C++ and this is also for storing real numbers, in this case 2 words will be allocated. And this will be floating point and 11 bits will be exponent and 53 bits will be used for significant.

(Refer Slide Time: 8:35)

Examples

```
unsigned int telephone_number;  
float mass, acceleration;
```



- OK to define several variables in same statement.
- Keyword **long**: says, "I need to store bigger or more precise numbers, so give me more than usual space."

```
long unsigned int cryptographic_password;
```

- Likely 64 bits will be allocated.

```
long double more_precise_acceleration;
```

- Likely 96 bits will be allocated



So here for example is a name a 'telephone_number' which I can give for an unsigned int variable. I might want to have variables to store mass and acceleration and these clearly will be real numbers and I could define them by writing 'float mass, acceleration'.

So here, in a single statement I am defining several variables of the same type though. Now, along that these definitions I can attach a keyword long and long basically says that I really want a bigger variable or I want more precise numbers so something like that, so give me more than the usual space. So I could have long unsigned 'int cryptographic_password' . And this might very likely give me 64 bits of storage. So unsigned int, so I really get to stored 64 bit binary numbers in such variable.

'long double more_precise_acceleration', defines a variable which I have just decided to call more precise acceleration. And since double is already 64 bits long double will probably give me something like 96 bits. So the exact values of this you will know by looking at the manual for the language, or really the manual for the compiler that you are using, or the standard.

(Refer Slide Time: 10:22)

Variable initialization



- A value can be stored in a variable at the time of creation.

```
int i=0, result;  
float vx=1.0, vy=2.0e5, weight;
```

- `i, vx, vy` given values as well as defined.
- `2.0e5` is how you write 2.0×10^5
- Although the computer uses binary, you write in decimal.

```
char command = 'f';
```

- `'f'` is a "character constant". It represents the ASCII value of the quoted character.

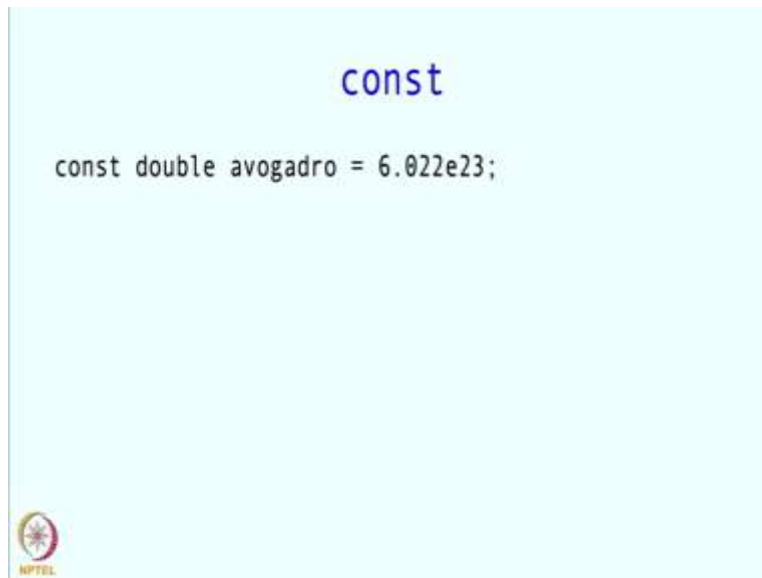


Now along with defining a variable we can also ask that a certain value be stored in it. So this value will be stored right at the time the variable is created. So for example, I could write `'int i=0, result;'` in this the variable `i`, I have chosen to initialize, whereas `result` is uninitialized, I can also write `'float vx=1.0, vy=2.0e5, weight;'`.

So in this `vx` and `vy` have been initialized and note the `2.0e5` is how you write the floating point number, or the number `2.0` into `10` to the power `5` of scientific notation. So that value will get stored in `vy`. `Weight` will not get an initial value. Now, as we can see although inside the computer the binary representation is used then you communicate with the computer when you tell what to store you are just using ordinary decimal notation, scientific notation or whatever it is, okay.

Here is another initialization, I set variable called `'command'` to the character `'f'` while defining it, so here I am defining it as a type of character and inside it, I am storing the character `'f'`. Now, this `'f'` is something called a character constant and these three characters together, this string of characters really simply means the ASCII value of the quoted character. So this happens to be `102` and so really what is going to be stored is this ASCII value.

(Refer Slide Time: 12:34)



There is also keyword called 'const' and I can attach it to a variable definition and here for example I am saying 'const double Avogadro=6.022e23', so this says that I do want space of type double to store Avogadro, but, I am not going to modify this value once I have stored. So normally value stored in a variable can be changed, but here I am just telling I am just declaring beforehand that look, I do not intend to change this value and therefore, if by mistake I change it then the compiler will warn.

(Refer Slide Time: 13:25)

The slide is titled "Reading values into variables" in a blue, sans-serif font. To the right of the title is a small video inset showing a man in a suit speaking. Below the title is a list of bullet points:

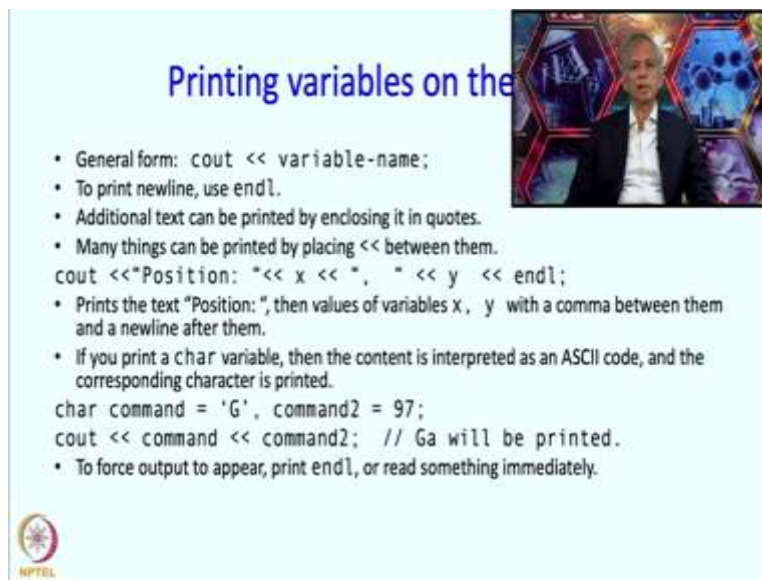
- We did this in chapter 1:
`cin >> nsides;`
- Can read into several variables one after another
`cin >> vx >> vy;`
- User expected to type in values consistent with the type of the variable into which it is to be read.
 - "Whitespace" = space characters, tabs, newlines, typed by the user are ignored.
 - newline/enter key must be pressed after values are typed
- If you read into a char type variable, the ASCII code of the typed character gets stored.
`char command;`
`cin >> command;`
- If you type the character 'f', its ASCII value, 102, will get stored.

In the bottom-left corner, there is a small circular logo with a starburst pattern and the text "NPTEL" underneath it.

I can read values into the variables and we have seen one example in chapter 1, okay. So I can write `'cin>>nsides'` so this will wait for the user to type in value from the keyboard and that type value will be stored into `nsides`. And I do not have to just read one value, I can read several values in the same statement. So I could write `'cin>>vx>>vy'` so this will expect that the user will type in two values. The first of those values will go into `vx`, the second will go into `vy`.

Now, when the user types something when the computer is waiting for `vx` or `vy` or the computer is waiting for a floating point number, it is expected that the user will type a floating point number or the user will type an integer. Well, when a computer is waiting for a number it really does not matter what, what kind of number a user types, it is okay so long as the user types a number. But the user should not type other characters. So the user should not type `'a'` because `'a'` does not mean anything as a number. So it is important that when such a command executes, the user types a number of a value of the correct type. While typing in values the user can put in characters which are often called white space characters, and these are simply the space character, tabs, newlines. And these serve to delimit two values. So after typing the value for `vx`, the user can type a space or the user should type a space before typing `vy`. But the user could type as many spaces as he or she wants, or the user could type in a newline or several lines so how many new lines, how many spaces are type is ignored. But, some white space has to be there between two values. Inside the typing of a single value of course, you cannot put in any white space. And this is a little bit of a tricky point for beginners initially, just because I typed `1.0` does not mean that that `1.0` value is accepted, to make the computer accept that value you have to hit the newline after this. If you are typing in several values the new line could be hit after typing the last one that is perfectly fine. And finally, if you are reading values into a char type variables then it is acceptable to type a character and what goes into the variable is the ASCII value of that character. So, if I define a char variable called `command` and if I the execute statement `'cin>>command'`, then C++ well wait for the user to type in a character, even here if the user types white space that will be ignored. But anything other than these white space characters if the user types, the ASCII value of that character will go into this variable called `command`. So, if for example if you type the character `f`, its ASCII value, `102` will get stored into `f`.

(Refer Slide Time: 17:19)



Printing variables on the screen


- General form: `cout << variable-name;`
- To print newline, use `endl`.
- Additional text can be printed by enclosing it in quotes.
- Many things can be printed by placing `<<` between them.

```
cout <<"Position: "<< x << ", " << y << endl;
```

- Prints the text "Position: ", then values of variables `x`, `y` with a comma between them and a newline after them.
- If you print a `char` variable, then the content is interpreted as an ASCII code, and the corresponding character is printed.

```
char command = 'G', command2 = 97;  
cout << command << command2; // Ga will be printed.
```

- To force output to appear, print `endl`, or read something immediately.



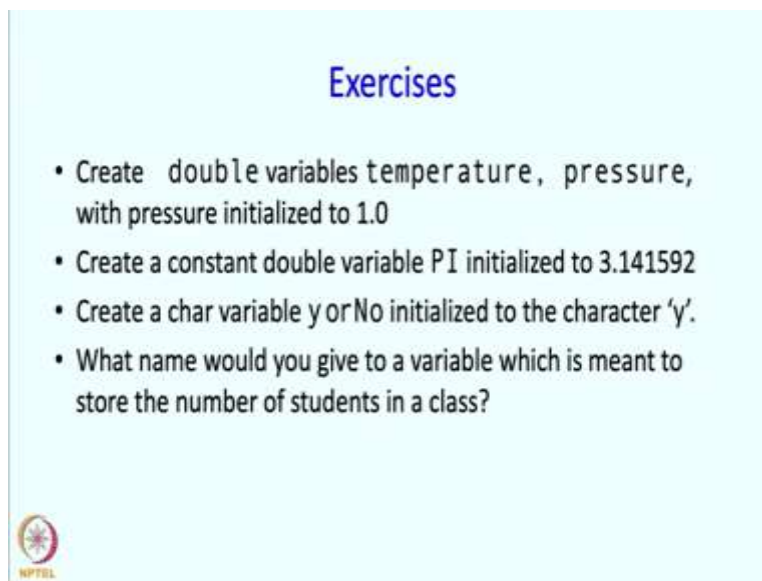
Printing variables on the screen is reasonably simple again we have seen an example of it in chapter 1. The general form most `cout` less than less than and the name of a variable, or it could be a message as well, okay. And, if I want a newline to be printed, I use the reserve keyword `endl`, e-n-d-l. And I can print a message and I just have to enclose that text inside quotes, okay. And, if I want to print several things I can do that I just have to put this less than less than between those things that I want to print. The less than less than as well as the greater than greater than is actually some kind of an operator. So I might say letter on that they less than less than operator or the greater than greater than operator.

So here for example, is a statement which is going to cause the message 'position' to be printed and of course the quotes will not be printed. After that the values of the variables `x` and `y` will be printed with the comma in between them, and then a newline will be printed. If you print a `char` variable, then the content is interpreted as an ASCII code, not as a number, and the corresponding character is printed.

So here for example, I am creating the variable 'command' in which I am storing letter 'G' and a second character variable called 'command2' into which I am storing the number 97, but if you remember 97 is the ASCII code of the little case a, okay. So I could have written `command2` equal to 'a' as well. In any case, if I print out these two characters, by writing '`cout<<command<<command2`', then whatever is the content of these variables that content will


be interpreted as an ASCII code, and the corresponding letter will be printed. So of course, command contains the ASCII code of G and so the little g will be printed, the command2 contains the ASCII code of letter little a and therefore, 'Ga' will be printed on the screen. Now, when you want output to appear on the screen, you really need to put an endl, and only a endl forces the output to appear. Until then C++ may just keep collecting whatever you are asking it to print, but C++ is forced to print it, if you print a newline so that is the sort of the signal to the C++ saying that whatever we have been collecting print now. Or, C++ will print whatever it is collecting. If you decide to read something, so if you are giving a message and then reading a something yes, the message will appear and then you can type after the message.

(Refer Slide Time: 21:02)



Exercises

- Create double variables temperature, pressure, with pressure initialized to 1.0
- Create a constant double variable PI initialized to 3.141592
- Create a char variable y or No initialized to the character 'y'.
- What name would you give to a variable which is meant to store the number of students in a class?



So here is quick exercise for you, so create double variables temperature, pressure and initialize pressure to 0. Create a constant double variable PI initialized to 3.141592 and the idea here is that from then on you can use capital PI without having to remember what the value of PI is. Create a char variable y or No which is initialized to the character y. And here is something that you need to think about often-what name should I give and here is the question is what name would you give to a variable in variable which is meant to store the number of students in a class.

(Refer Slide Time: 21:51)

What we discussed

- How to define variables
- How to initialize variables
- How to read a value into a variable and print the value of a variable.



Okay, so what have we discussed in this segment of this lecture sequence? We first discussed how to define variables. We discussed how to initialize variables. We discussed how to read a value into a variable and print the value of a variable. So we will take a break here.