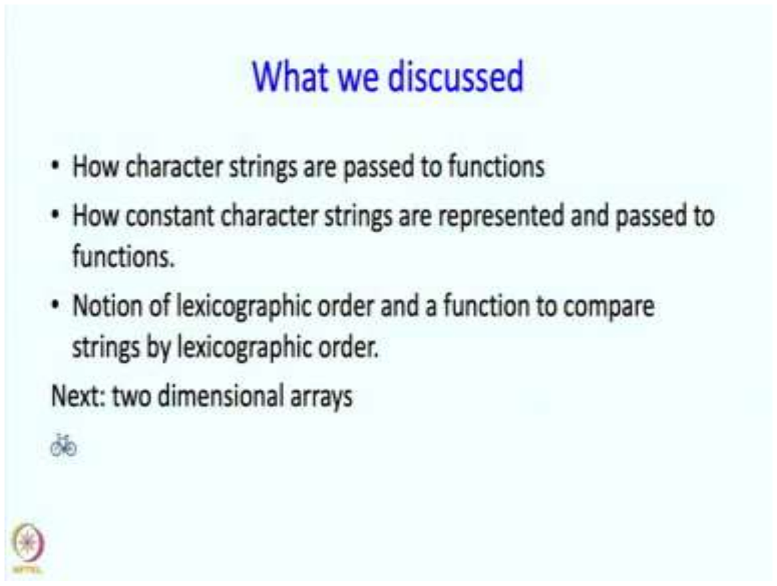
An Introduction to Programming through C++
Professor Abhiram G. Ranade
Department of computer science and Engineering,
Indian Information Technology Bombay
Lecture 17 Part - 3
More on Arrays
Two dimensional arrays

(Refer Slide Time: 0:20)



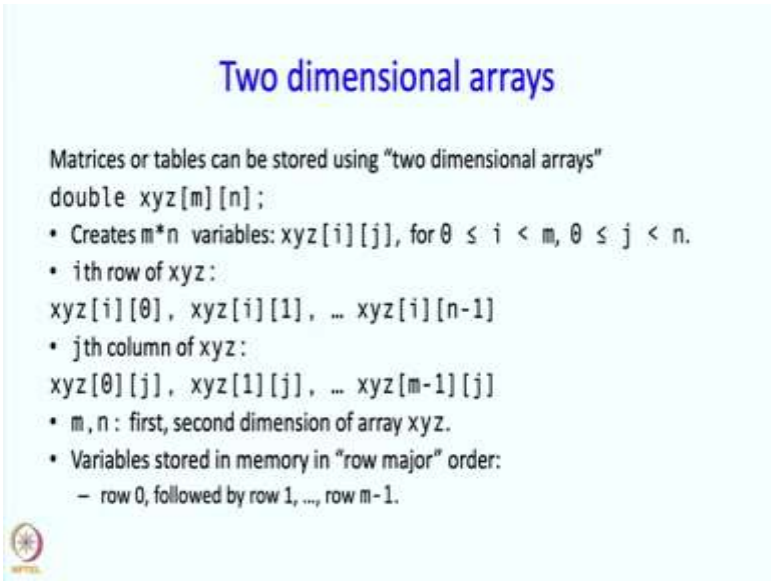
What we discussed

- How character strings are passed to functions
- How constant character strings are represented and passed to functions.
- Notion of lexicographic order and a function to compare strings by lexicographic order.

Next: two dimensional arrays

Welcome back. In the next, in the previous segment, we discussed how character strings are passed to functions. The notion of constant character strings, and we also did example of lexicographic order, a function to compare strings by lexicographic order. In this segment, we are going to discuss two dimensional arrays. Now, two dimensional arrays are quite nice, because we can store matrices or tables using two dimensional arrays and also other things, which we will see in a minute.

(Refer Slide Time: 0:49)



Two dimensional arrays

Matrices or tables can be stored using "two dimensional arrays"

```
double xyz[m][n];
```

- Creates m*n variables: xyz[i][j], for $0 \leq i < m$, $0 \leq j < n$.
- ith row of xyz:

```
xyz[i][0], xyz[i][1], … xyz[i][n-1]
```

- jth column of xyz:

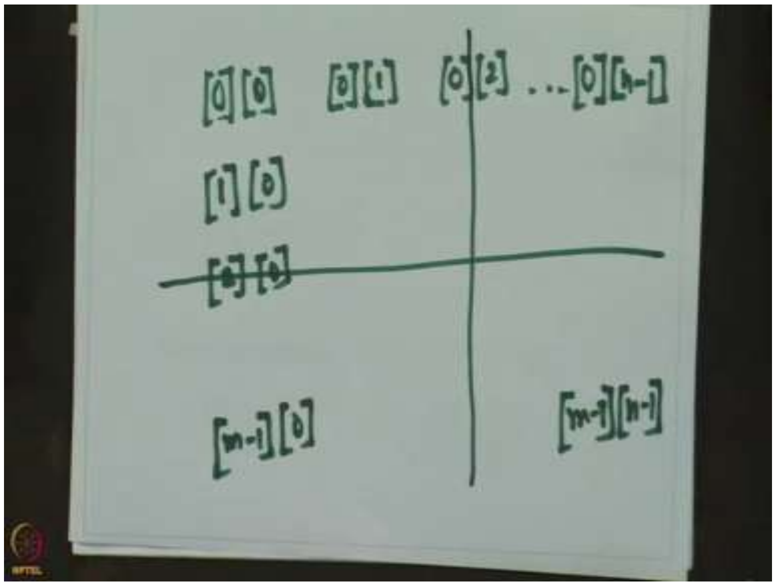
```
xyz[0][j], xyz[1][j], … xyz[m-1][j]
```

- m, n: first, second dimension of array xyz.
- Variables stored in memory in "row major" order:
  - row 0, followed by row 1, …, row m-1.

So, this is how you might declare a two dimensional array. So, this is a two dimensional array of doubles, what does it do? Well, it creates m times n variables. And these variables are accessed, or we can think they have names xyz[i][j], where i must be between 0 and M, and j must be between 0 and n the upper bounds not inclusive. So, what does this do, so it creates m times n variables, and you can think of this, think of these as organized in a table with rows and columns.

So, the ith row of xyz is the one in which the first. the first index is i, the other index can be anything at all. So, it can take all possible values 0 through n-1. So, so this second index can take all these values. So that is the ith row. And the jth column of xyz is what you get, if you hold the second index, fixed j and allow all the, the first index to take arbitrary values. So, in fact, it is natural to think of these elements as arranged in two dimensions, in a table, so to say.

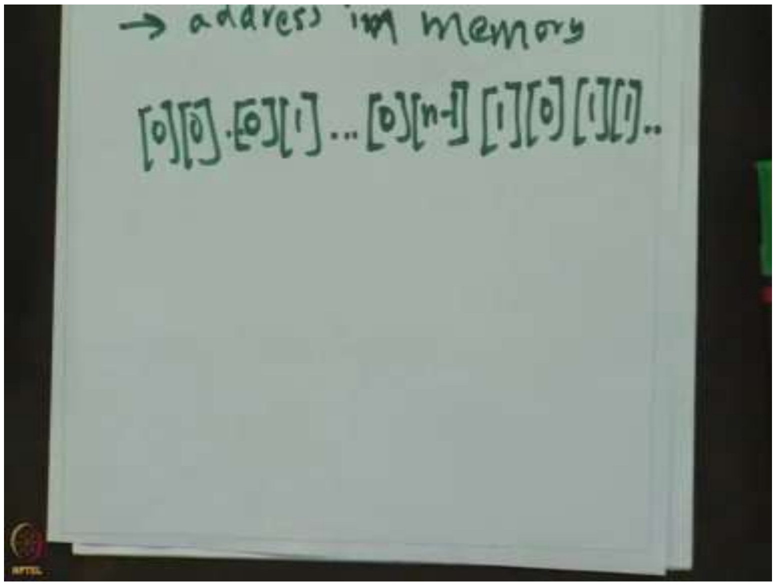
(Refer Slide Time: 2:40)



So, so this is the element [0][1], then this you can think 0, you can think of the element, sorry, this is the element [0][0], you can think of the element [0][1] as sitting over here, then [0][2] as sitting over here all the way till [0][n-1]. And in this direction, the first index is going to change, so this is going to be index element [1][0], this is going to be element [2][0] and this is going to be element [m-1][0]. And in the last column, similarly, you are going to get the element [m-1][n-1].

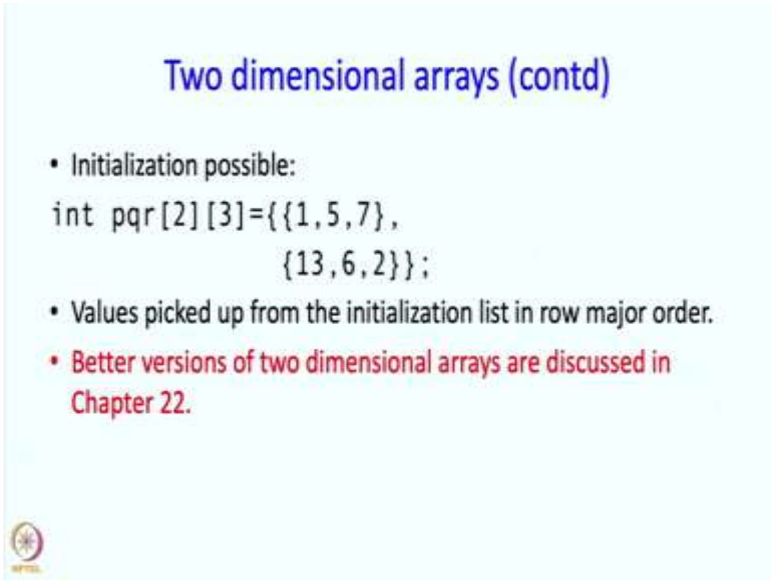
So, this will be the second row and this will be the second column, so that is what it is. So, on a computer you get m times n variables and for your visualization purpose, we are thinking of these being organized as rows and columns, but the computer does not really think about them as being rows and columns. Well you may ask, well how does the computer store them? So, the computer stores these variables in what is called a row major order.

(Refer Slide Time: 4:37)



So, so let me write that down. So, the computer will store these variables. So, row 0 first. So, say [0][0] and [0][1] all the way till [0][n-1], then it will store the first row. So this is [1][0] and [1][1], and so on, okay? So, so this is increasing address, address and memory. So, it allocates a block of memory. At the very beginning of the block, [0][0] is stored, then the next [0][1] is stored, so the row, all the elements of row 1 are stored, then all the element row 0 are stored, then all the elements of row 1 are stored, then all the elements of row 2 are stored, and so on. So, you do not really need to know, but this is how it is stored. For programming purposes, it is not quite necessary. Well, in a way, we will see in a way it is it is useful to know this.

(Refer Slide Time: 6:03)



Two dimensional arrays (contd)

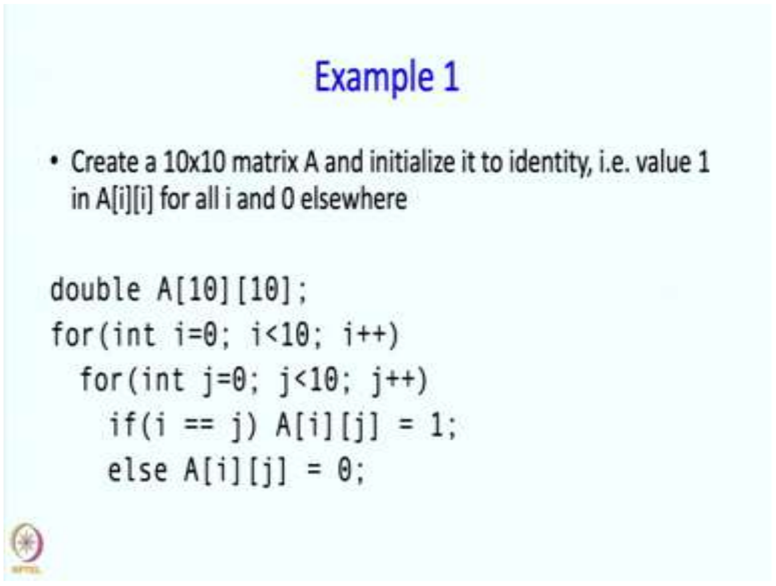
- Initialization possible:

```
int pqr[2][3]={{1,5,7},
               {13,6,2}};
```

- Values picked up from the initialization list in row major order.
- Better versions of two dimensional arrays are discussed in Chapter 22.

Okay, now, when you define two dimensional arrays you can simultaneously initialize them as well. So, for example, here you have an array pqr of 2 rows and 3 columns, so you give the initialization for 0th row and then 1st row, that is put itself put in braces and inside that you should supply 3 values because the second dimension is 3, so the number of columns is 3, so 3 and 3 given over here . I could have put them in the same, I could have put the whole thing one after another also but I just wanted to share you that this is the, this is 1 row and this is a second row, okay. All right. And, yeah, so the values are picked up from the initialization list also in row major order, the order in which the data is stored on the computer as well. Now, two dimensional arrays that I am describing right now, are a little bit, are not really all that useful, so we will see why in a minute. And later on in the course, we will talk about nicer two dimensional arrays, but anyway, many of these concepts like having two indices, or, yeah, or even the fact that a two dimensional array really can be thought of as a one dimensional array followed by another one dimensional array, are actually important and so, this is something that we get it from the language C, and we might as well learn it, because these ideas are important for the nicer ideas that will come later.

(Refer Slide Time: 7:49)



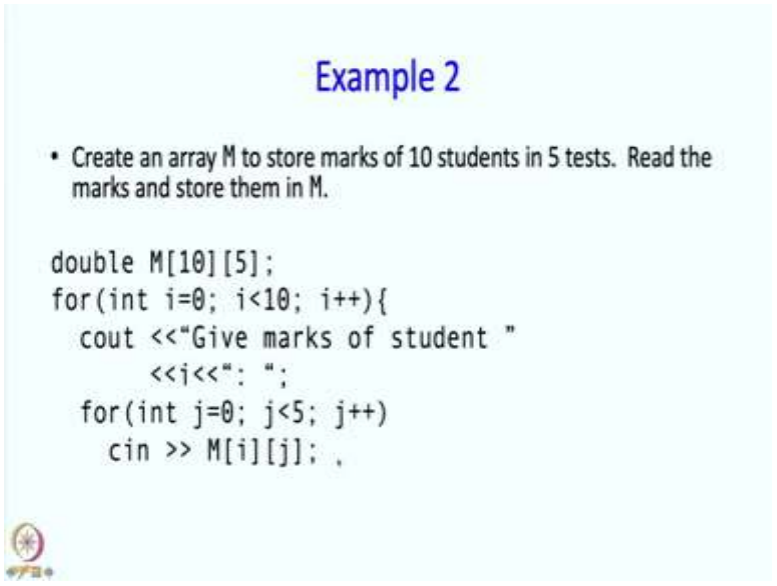
Example 1

- Create a 10x10 matrix A and initialize it to identity, i.e. value 1 in A[i][i] for all i and 0 elsewhere

```
double A[10][10];
for(int i=0; i<10; i++)
  for(int j=0; j<10; j++)
    if(i == j) A[i][j] = 1;
    else A[i][j] = 0;
```

Anyway, so some examples, so this says, create a 10 by 10 matrix and initialize it to the identity. Okay, so value 1 in those elements whose indices are the same? Okay, diagonal elements, and 0 elsewhere? Okay. So, first of all, the matrix will have to be declared in this manner. So 10 rows and 10 columns and then for i going from 0 to 10 and say, this is the row index, and this is the column index. And if I and j happened to be the same, then you are going to set A[i][j] to 1, if they happen to be different, you will set A[i][j] to 0. So, that is exactly what you want, when you want to create an identity matrix.

(Refer Slide Time: 8:39)



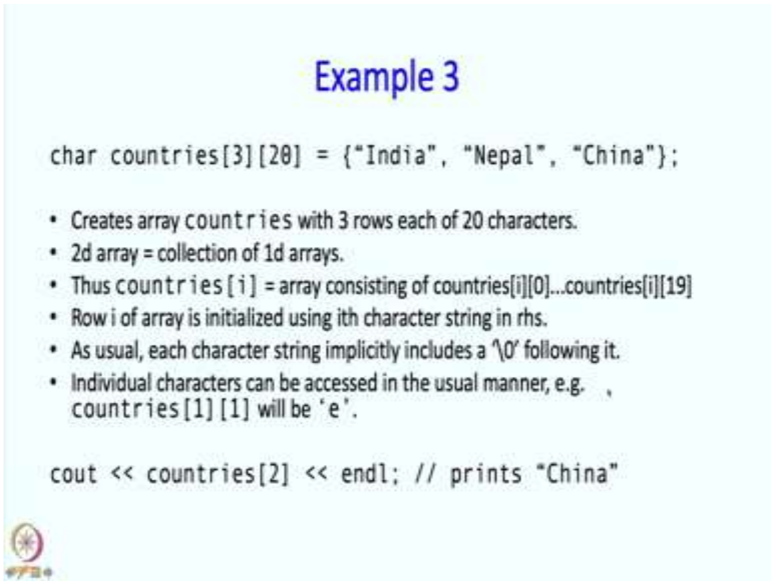
Example 2

- Create an array M to store marks of 10 students in 5 tests. Read the marks and store them in M.

```
double M[10][5];
for(int i=0; i<10; i++){
  cout <<"Give marks of student "
       <<i<<": ";
  for(int j=0; j<5; j++)
    cin >> M[i][j]; .
```

Here is another example. This time you want to create an array m to store the marks of 10 students in 5 testes and then you are asked the read the marks and store them in M. And you can ask that the marks be given in a convenient manner for you. So, so, we want marks of ten students in 5 tests, so 50 marks in all and 10 students 5 tests, so we should have a row in which there are five marks. So, a row for every student so 10 rows, so 10 rows, each row consisting of 5 columns. So, that is what we have here, then for every row or for every student, we are going to print them as say “Give me the marks of student ‘i’. And then we are simply going to read marks. So, there are 5 marks to be read. So, we are going to have another loop and this time we are reading the jth mark of student i and, so whatever comes in, whatever is typed in will be placed in M[i][j], that is it. Okay. One more example, so here we have an array of characters, okay. So, character arrays are again slightly different from other arrays, even 2 dimensional arrays are slightly different. So, these are different in the sense that, of course, they have 3 rows and 20 columns. And if we had not initialized then that would be the same as before but the initialization is slightly different.

(Refer Slide Time: 10:23)



Example 3

```
char countries[3][20] = {"India", "Nepal", "China"};
```

- Creates array `countries` with 3 rows each of 20 characters.
- 2d array = collection of 1d arrays.
- Thus `countries[i]` = array consisting of countries[i][0]...countries[i][19]
- Row i of array is initialized using ith character string in rhs.
- As usual, each character string implicitly includes a '\0' following it.
- Individual characters can be accessed in the usual manner, e.g. `countries[1][1]` will be `'e'`.

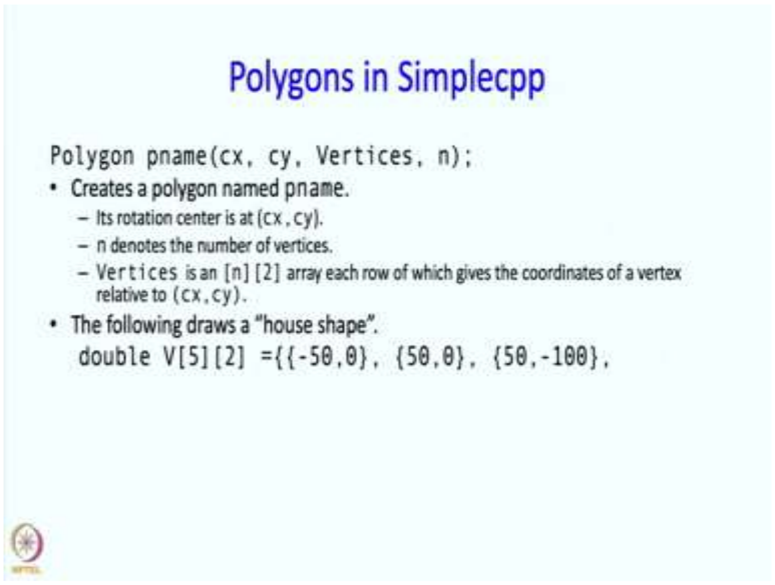
```
cout << countries[2] << endl; // prints "China"
```

So, here you are initializing it, so that you can supply all the data but again, just as we initialized 1 dimensional character arrays, the initialization is slightly different. So, here you are going to find as many character string constants as there are number of rows and each string constant is going to be initializing the corresponding row. So, this will create an array country, countries with 3 rows of 20 characters and an interesting idea is that a two dimensional array is a collection of one dimensional arrays. So, notice I said, if you remember when I said that the 2 dimensional arrays are stored in row major order. So they are stored 1 row at a time, one row following another row and so, each of those rows can be thought of as an independent array and that is sort of exploited in these char arrays, okay. So, countries[2] used to denote the third row and so this will be going to 0th row, this will go into the first row, this will go into the 2nd row, okay, countries i consist of countries i[0] to i[19] or ith row. So, row i initialized using the ith character string in this right hand side.

So, what happens so, row 0 gets India followed by the null character, row 1 gets Nepal followed by the null character and row two gets China followed by the null character. And inside each row, say the 0th character over here will go to this 0th column, this will go to the first column and so on. And we can access individual characters as well. So, if I write countries[i][j] then it will give me the jth character of the ith row or the jth character of the ith character string which is being stored, okay.

So, [1][1], for example, will mean so, the first word, so, this is first, this is 0, this is first, and the first character. So, this is the 0th character, this is the first character so it will mean this e. So, if I print, I can print countries[2]. So, although country has 2 indices, I can just use one of them and that will be in just that row, that entire row. So, the second row, or this is the 0 row, 1st row, 2nd row. So, the second row stores the character string China and therefore, if I say cout<<countries[2], that character will get printed followed by the new line.

(Refer Slide Time: 13:40)



Polygons in Simplecpp

```
Polygon pname(cx, cy, Vertices, n);
```

- Creates a polygon named pname.
  - Its rotation center is at (cx,cy).
  - n denotes the number of vertices.
  - Vertices is an [n][2] array each row of which gives the coordinates of a vertex relative to (cx,cy).
- The following draws a "house shape".

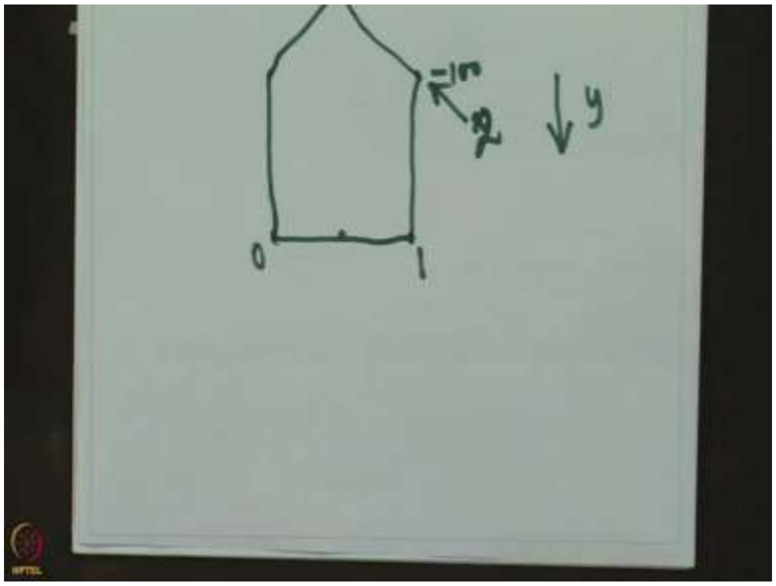
```
double V[5][2] ={{-50,0}, {50,0}, {50,-100},
```

Now two dimensional arrays are useful also because they allow us to define polygons in simplecpp. So here is the way polygons are excepted to be defined. So you write the key word polygon, then you write the name that you want give to the polygon and then you give two coordinates which are rotation center of that polygon. And then you give the vertices of the polygons, and this is going to be a two dimensional array and you will have to tell how many vertices there are? So, for that you specify n.

So, the vertices is an n by 2 array, but there is something interesting or something unusual about arrays in C++, which is that you really, you really cannot specify or it is not customary to specify the second dimension. You can, but usually, these two dimensional arrays are useful only when the second dimension is sort of a fixed number. So, here the second dimension is a fixed number, you know, that in on the plane, you always have to give two coordinates.

So, we are going to, so this is going to have n rows and each row is going to have two columns, one column which gives the x coordinate and other column which is going to go y coordinates. So, here the how many columns are there does not need to be given because we sort of know that, okay. So, whenever we, whenever we define this polygon you are expected to give an array of this kind. So, what is not known in this is n that is what you have to specify over here. So, here is an example. So, this is going to draw what is a house shape, so maybe we should draw a picture of this.

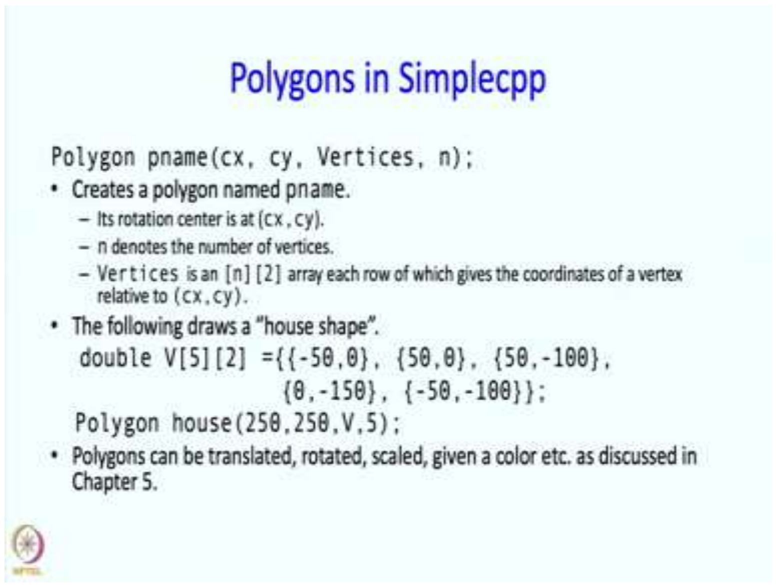
(Refer Slide Time: 15:43)



So, what does this look like? So, this is, let us say this is (cx,cy), which is the origin about which these coordinates have to be given. So, (-50,0) is going to be, so X coordinate is -50 with respect to this, so it will be say somewhere over here, and 0 because it is at the same level, the y coordinate is at the same level. Then the next point is (50,0), okay, so it is at the same level over here, but 50, on this side. So, this is, this is point 0, this is point 1. Okay.

Then the next point is (50,-100), so 50 is this and -100. Well, when we write coordinates, we have to remember that the y axis is actually going in this direction. So, this is 0, then this is -100, okay. So, this is the second coordinate. So, this is vertex 2. Then after that, we have (0,-50) so it is somewhere over here and -50 somewhere over here. So, this is the next coordinate and (-50,100) this is an X coordinate. So, the polygon that will get drawn is this. Okay, so, well, as you can see, it resembles sort of a very, very schematic house.

(Refer Slide Time: 17:35)



Polygons in Simplecpp

```
Polygon pname(cx, cy, Vertices, n);
```

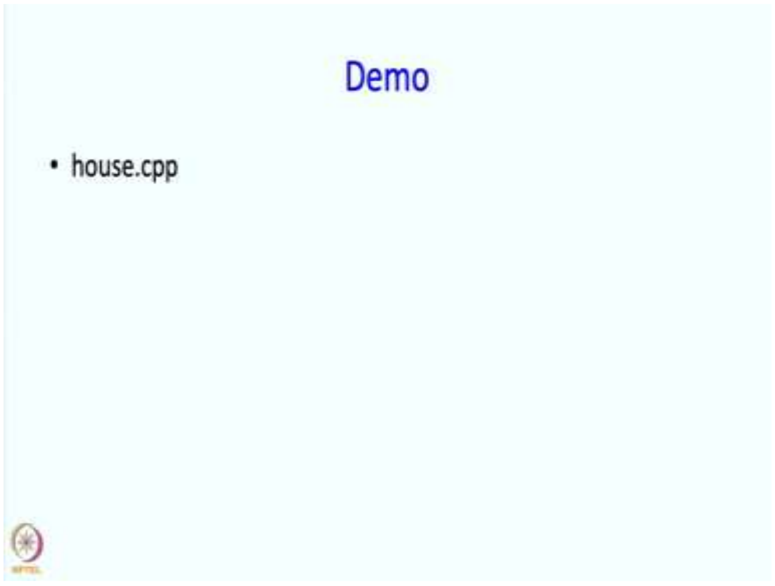
- Creates a polygon named `pname`.
  - Its rotation center is at `(cx,cy)`.
  - `n` denotes the number of vertices.
  - `Vertices` is an `[n][2]` array each row of which gives the coordinates of a vertex relative to `(cx,cy)`.
- The following draws a "house shape".

```
double V[5][2] ={{-50,0}, {50,0}, {50,-100},
                 {0,-150}, {-50,-100}};
Polygon house(250,250,V,5);
```

- Polygons can be translated, rotated, scaled, given a color etc. as discussed in Chapter 5.

So, this command, Polygon house(250, 250) says that I want this house to come up at screen coordinates (250, 250). So with respect to the screen coordinates, these coordinates are going to get measured. Okay. So, so this coordinate really is going to be (200, 250), if you look at the screen coordinates. Now polygons can be translated, rotated, scaled, given color and things like that, exactly as in chapter 5, okay, of the book.

(Refer Slide Time: 18:04)

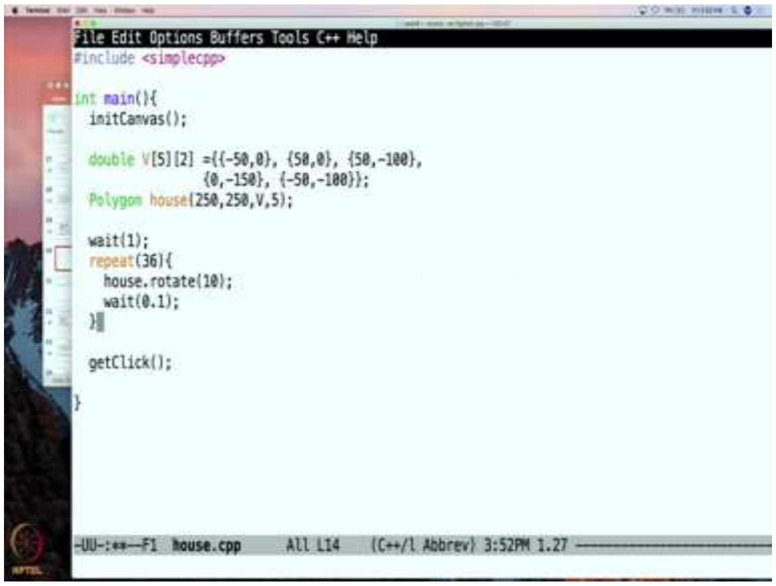


Demo

- house.cpp

Okay, so let us see this, let us do a quick demo of this.

(Refer Slide Time: 18:34)



File Edit Options Buffers Tools C++ Help

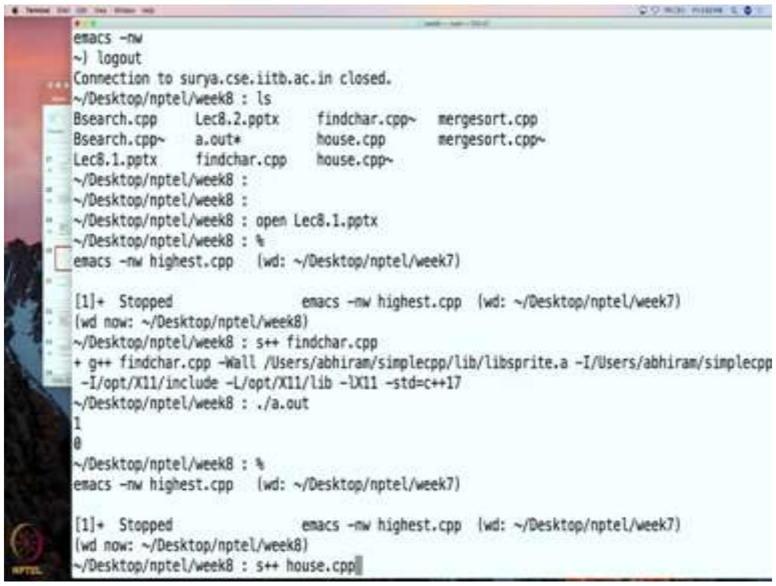
```
#include <simplecpp>

int main(){
  initCanvas();

  double V[5][2] ={{-50,0}, {50,0}, {50,-100},
                   {0,-150}, {-50,-100}};
  Polygon house(250,250,V,5);

  wait(1);
  repeat(36){
    house.rotate(10);
    wait(0.1);
  }

  getClick();

}
```

-UU-:**--F1 house.cpp   All L14   (C++/l Abbrev) 3:52PM 1.27

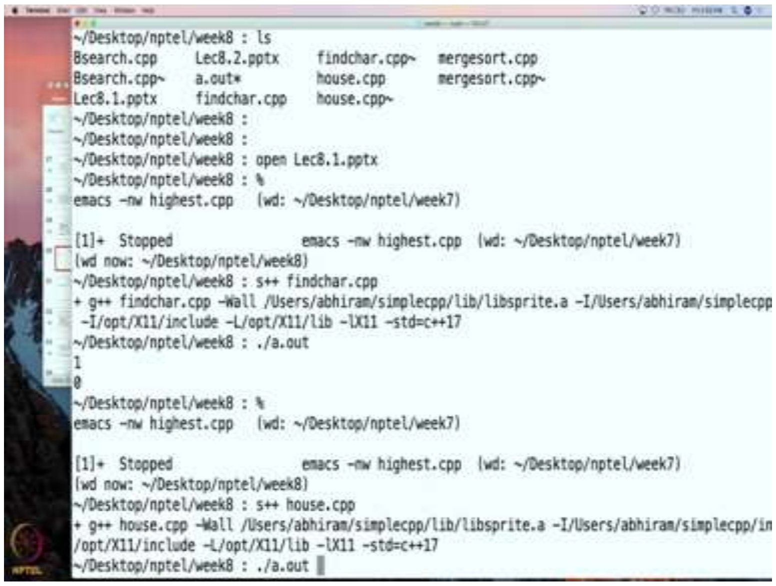
Okay, so this is our, our array, what we just had there, and this is the created polygon. And notice that I put initCanvas at the beginning, because otherwise, this would be an error. You need to have initCanvas called because that is where the polygon is going to be drawn. Okay, so let us do something interesting after that, just for fun. So, what we are going to do is, we are going to rotate this, so that will show you where the rotation center is.

So, maybe I will wait for one second so that you get the chance to see the original position. And then I am going to rotate. So, repeat, say, maybe say 36 times and what I will do is I will rotate, I will say house.rotate(10) and we will wait, say, for say 0.1 second so that it does not all happen to fast and that is it., okay? So, let us see, let us compile this and see what happens?

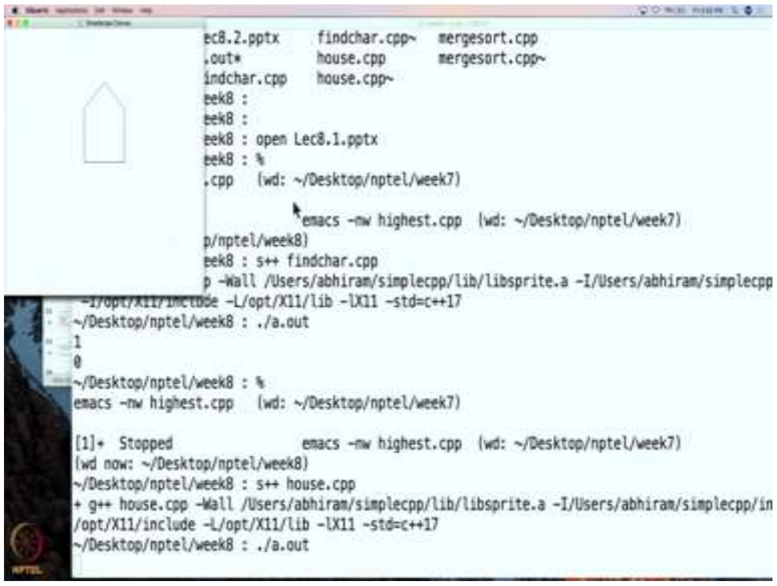
(Refer Slide Time: 19:30)



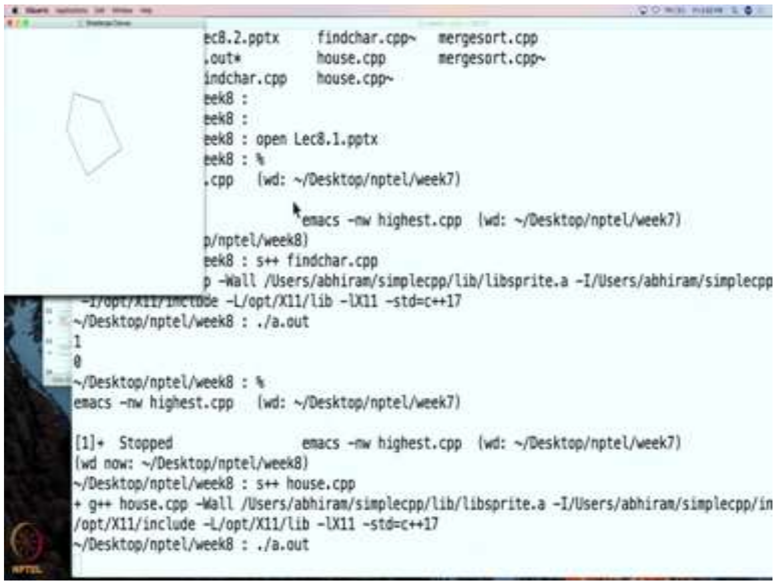
```
emacs -nw
~) logout
Connection to surya.cse.iitb.ac.in closed.
~/Desktop/nptel/week8 : ls
Bsearch.cpp     Lec8.2.pptx     findchar.cpp~   mergesort.cpp
Bsearch.cpp~    a.out*          house.cpp       mergesort.cpp~
Lec8.1.pptx     findchar.cpp    house.cpp~
~/Desktop/nptel/week8 :
~/Desktop/nptel/week8 :
~/Desktop/nptel/week8 : open Lec8.1.pptx
~/Desktop/nptel/week8 : %
emacs -nw highest.cpp   (wd: ~/Desktop/nptel/week7)

[1]+  Stopped                 emacs -nw highest.cpp  (wd: ~/Desktop/nptel/week7)
(wd now: ~/Desktop/nptel/week8)
~/Desktop/nptel/week8 : s++ findchar.cpp
+ g++ findchar.cpp -Wall /Users/abhiram/simplecpp/lib/libsprite.a -I/Users/abhiram/simplecpp
 -I/opt/X11/include -L/opt/X11/lib -lX11 -std=c++17
~/Desktop/nptel/week8 : ./a.out
1
0
~/Desktop/nptel/week8 : %
emacs -nw highest.cpp   (wd: ~/Desktop/nptel/week7)

[1]+  Stopped                 emacs -nw highest.cpp  (wd: ~/Desktop/nptel/week7)
(wd now: ~/Desktop/nptel/week8)
~/Desktop/nptel/week8 : s++ house.cpp
```



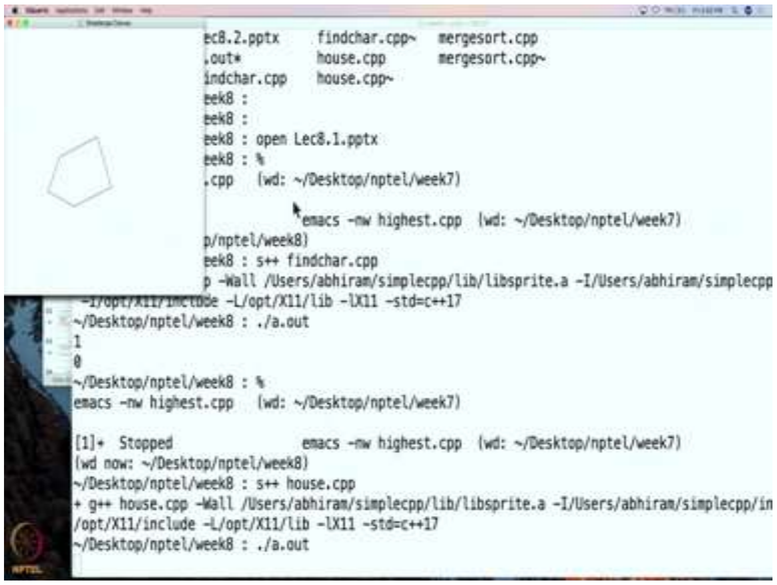
```
~/Desktop/nptel/week8 : ls
Bsearch.cpp     Lec8.2.pptx     findchar.cpp~   mergesort.cpp
Bsearch.cpp~    a.out*          house.cpp       mergesort.cpp~
Lec8.1.pptx     findchar.cpp    house.cpp~
~/Desktop/nptel/week8 :
~/Desktop/nptel/week8 :
~/Desktop/nptel/week8 : open Lec8.1.pptx
~/Desktop/nptel/week8 : %
emacs -nw highest.cpp   (wd: ~/Desktop/nptel/week7)

[1]+  Stopped                 emacs -nw highest.cpp  (wd: ~/Desktop/nptel/week7)
(wd now: ~/Desktop/nptel/week8)
~/Desktop/nptel/week8 : s++ findchar.cpp
+ g++ findchar.cpp -Wall /Users/abhiram/simplecpp/lib/libsprite.a -I/Users/abhiram/simplecpp
 -I/opt/X11/include -L/opt/X11/lib -lX11 -std=c++17
~/Desktop/nptel/week8 : ./a.out
1
0
~/Desktop/nptel/week8 : %
emacs -nw highest.cpp   (wd: ~/Desktop/nptel/week7)

[1]+  Stopped                 emacs -nw highest.cpp  (wd: ~/Desktop/nptel/week7)
(wd now: ~/Desktop/nptel/week8)
~/Desktop/nptel/week8 : s++ house.cpp
+ g++ house.cpp -Wall /Users/abhiram/simplecpp/lib/libsprite.a -I/Users/abhiram/simplecpp/in
/opt/X11/include -L/opt/X11/lib -lX11 -std=c++17
~/Desktop/nptel/week8 : ./a.out
```



```
ec8.2.pptx      findchar.cpp~    mergesort.cpp
.out*           house.cpp        mergesort.cpp~
indchar.cpp     house.cpp~
eek8 :
eek8 :
eek8 : open Lec8.1.pptx
eek8 : %
.cpp   (wd: ~/Desktop/nptel/week7)

                 emacs -nw highest.cpp  (wd: ~/Desktop/nptel/week7)
p/nptel/week8)
eek8 : s++ findchar.cpp
p -Wall /Users/abhiram/simplecpp/lib/libsprite.a -I/Users/abhiram/simplecpp
 -I/opt/X11/include -L/opt/X11/lib -lX11 -std=c++17
~/Desktop/nptel/week8 : ./a.out
1
0
~/Desktop/nptel/week8 : %
emacs -nw highest.cpp   (wd: ~/Desktop/nptel/week7)

[1]+  Stopped                 emacs -nw highest.cpp  (wd: ~/Desktop/nptel/week7)
(wd now: ~/Desktop/nptel/week8)
~/Desktop/nptel/week8 : s++ house.cpp
+ g++ house.cpp -Wall /Users/abhiram/simplecpp/lib/libsprite.a -I/Users/abhiram/simplecpp/in
/opt/X11/include -L/opt/X11/lib -lX11 -std=c++17
~/Desktop/nptel/week8 : ./a.out
```



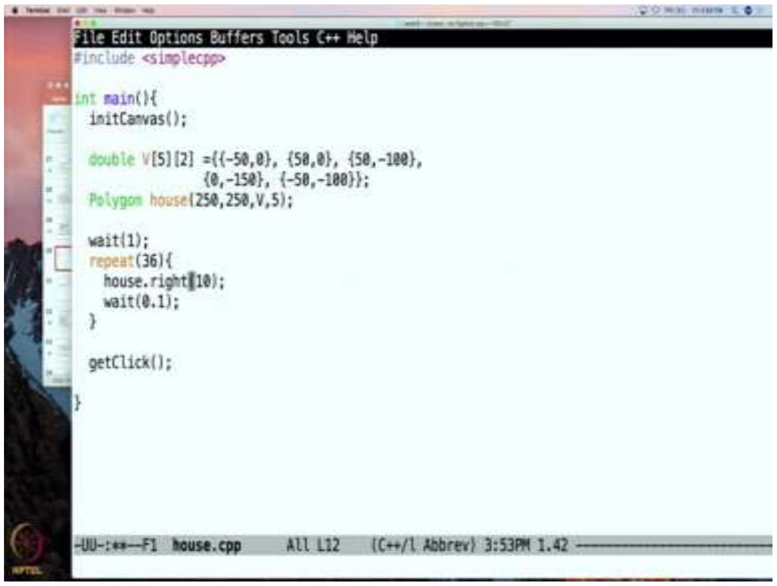
```
ec8.2.pptx      findchar.cpp~    mergesort.cpp
.out*           house.cpp        mergesort.cpp~
indchar.cpp     house.cpp~
eek8 :
eek8 :
eek8 : open Lec8.1.pptx
eek8 : %
.cpp   (wd: ~/Desktop/nptel/week7)

                 emacs -nw highest.cpp  (wd: ~/Desktop/nptel/week7)
p/nptel/week8)
eek8 : s++ findchar.cpp
p -Wall /Users/abhiram/simplecpp/lib/libsprite.a -I/Users/abhiram/simplecpp
 -I/opt/X11/include -L/opt/X11/lib -lX11 -std=c++17
~/Desktop/nptel/week8 : ./a.out
1
0
~/Desktop/nptel/week8 : %
emacs -nw highest.cpp   (wd: ~/Desktop/nptel/week7)

[1]+  Stopped                 emacs -nw highest.cpp  (wd: ~/Desktop/nptel/week7)
(wd now: ~/Desktop/nptel/week8)
~/Desktop/nptel/week8 : s++ house.cpp
+ g++ house.cpp -Wall /Users/abhiram/simplecpp/lib/libsprite.a -I/Users/abhiram/simplecpp/in
/opt/X11/include -L/opt/X11/lib -lX11 -std=c++17
~/Desktop/nptel/week8 : ./a.out
```



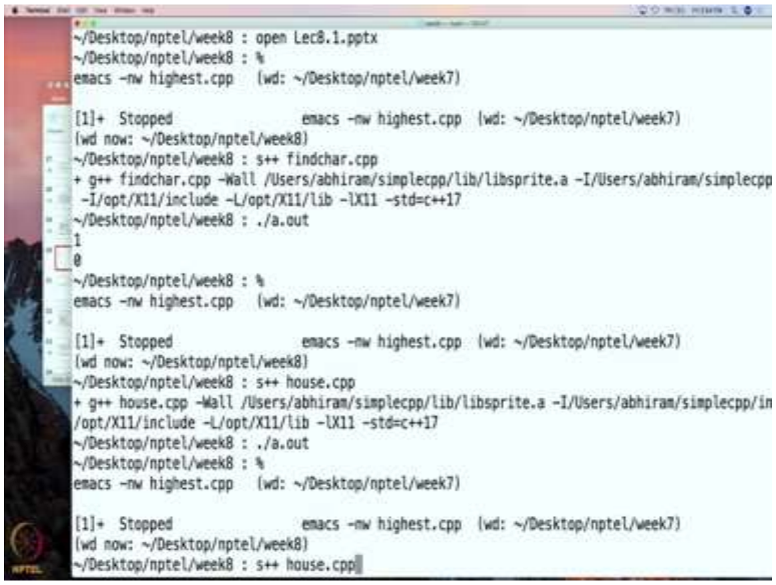
```
ec8.2.pptx     findchar.cpp~   mergesort.cpp
.out*          house.cpp       mergesort.cpp~
indchar.cpp    house.cpp~
eek8 :
eek8 :
eek8 : open Lec8.1.pptx
eek8 : %
.cpp   (wd: ~/Desktop/nptel/week7)
                      emacs -nw highest.cpp  (wd: ~/Desktop/nptel/week7)
p/nptel/week8)
eek8 : s++ findchar.cpp
p -Wall /Users/abhiram/simplecpp/lib/libsprite.a -I/Users/abhiram/simplecpp
 -I/opt/X11/include -L/opt/X11/lib -lX11 -std=c++17
~/Desktop/nptel/week8 : ./a.out
1
0
~/Desktop/nptel/week8 : %
emacs -nw highest.cpp   (wd: ~/Desktop/nptel/week7)

[1]+  Stopped                 emacs -nw highest.cpp  (wd: ~/Desktop/nptel/week7)
(wd now: ~/Desktop/nptel/week8)
~/Desktop/nptel/week8 : s++ house.cpp
+ g++ house.cpp -Wall /Users/abhiram/simplecpp/lib/libsprite.a -I/Users/abhiram/simplecpp/in
/opt/X11/include -L/opt/X11/lib -lX11 -std=c++17
~/Desktop/nptel/week8 : ./a.out
```

So, see it is rotating? So, see Okay, so see it is rotating. So, I think I am turning it a bit too much. But anyway, you see how it is rotating?

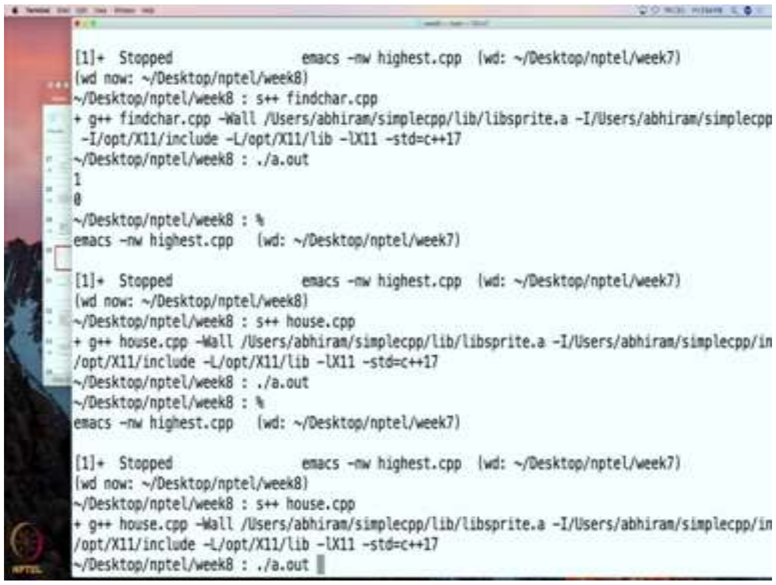
(Refer Slide Time: 19:55)



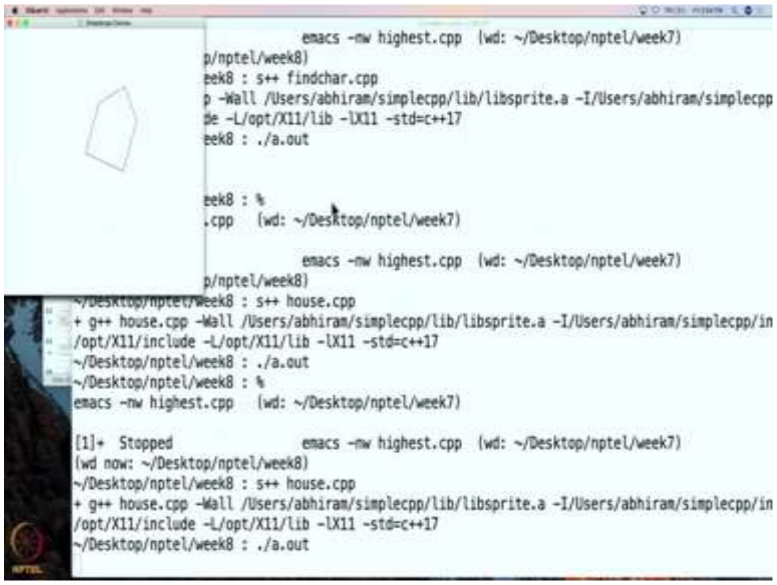
```
File Edit Options Buffers Tools C++ Help
#include <simplecpp>

int main(){
  initCanvas();

  double V[5][2] ={{-50,0}, {50,0}, {50,-100},
                   {0,-150}, {-50,-100}};
  Polygon house(250,250,V,5);

  wait(1);
  repeat(36){
    house.right(10);
    wait(0.1);
  }

  getClick();

}

-UU-:**--F1  house.cpp      All L12    (C++/l Abbrev) 3:53PM 1.42 ----------------------
```



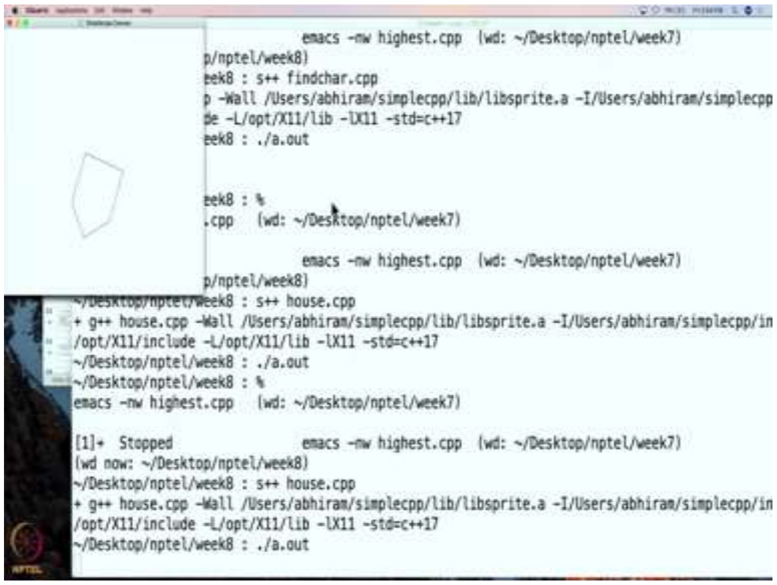
```
~/Desktop/nptel/week8 : open Lec8.1.pptx
~/Desktop/nptel/week8 : %
emacs -nw highest.cpp   (wd: ~/Desktop/nptel/week7)

[1]+  Stopped                 emacs -nw highest.cpp  (wd: ~/Desktop/nptel/week7)
(wd now: ~/Desktop/nptel/week8)
~/Desktop/nptel/week8 : s++ findchar.cpp
+ g++ findchar.cpp -Wall /Users/abhiram/simplecpp/lib/libsprite.a -I/Users/abhiram/simplecpp
 -I/opt/X11/include -L/opt/X11/lib -lX11 -std=c++17
~/Desktop/nptel/week8 : ./a.out
1
0
~/Desktop/nptel/week8 : %
emacs -nw highest.cpp   (wd: ~/Desktop/nptel/week7)

[1]+  Stopped                 emacs -nw highest.cpp  (wd: ~/Desktop/nptel/week7)
(wd now: ~/Desktop/nptel/week8)
~/Desktop/nptel/week8 : s++ house.cpp
+ g++ house.cpp -Wall /Users/abhiram/simplecpp/lib/libsprite.a -I/Users/abhiram/simplecpp/in
/opt/X11/include -L/opt/X11/lib -lX11 -std=c++17
~/Desktop/nptel/week8 : ./a.out
~/Desktop/nptel/week8 : %
emacs -nw highest.cpp   (wd: ~/Desktop/nptel/week7)

[1]+  Stopped                 emacs -nw highest.cpp  (wd: ~/Desktop/nptel/week7)
(wd now: ~/Desktop/nptel/week8)
~/Desktop/nptel/week8 : s++ house.cpp
```



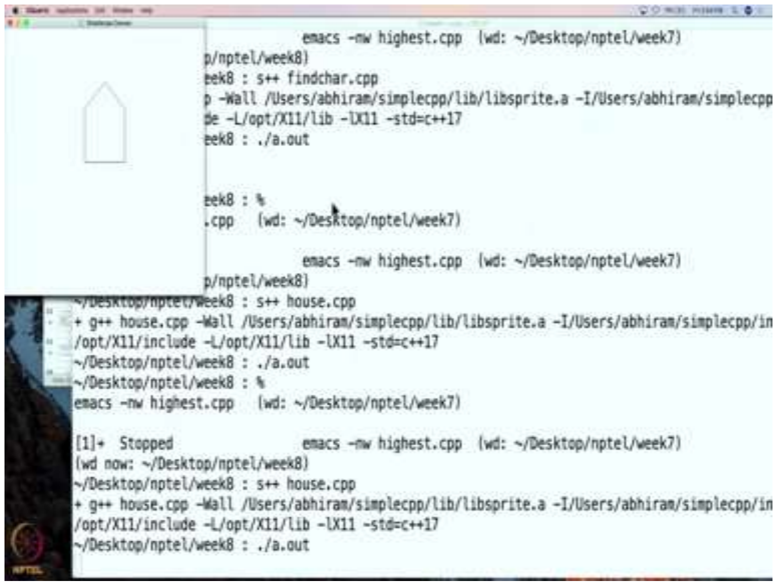
```

[1]+  Stopped                 emacs -nw highest.cpp  (wd: ~/Desktop/nptel/week7)
(wd now: ~/Desktop/nptel/week8)
~/Desktop/nptel/week8 : s++ findchar.cpp
+ g++ findchar.cpp -Wall /Users/abhiram/simplecpp/lib/libsprite.a -I/Users/abhiram/simplecpp
 -I/opt/X11/include -L/opt/X11/lib -lX11 -std=c++17
~/Desktop/nptel/week8 : ./a.out
1
0
~/Desktop/nptel/week8 : %
emacs -nw highest.cpp   (wd: ~/Desktop/nptel/week7)

[1]+  Stopped                 emacs -nw highest.cpp  (wd: ~/Desktop/nptel/week7)
(wd now: ~/Desktop/nptel/week8)
~/Desktop/nptel/week8 : s++ house.cpp
+ g++ house.cpp -Wall /Users/abhiram/simplecpp/lib/libsprite.a -I/Users/abhiram/simplecpp/in
/opt/X11/include -L/opt/X11/lib -lX11 -std=c++17
~/Desktop/nptel/week8 : ./a.out
~/Desktop/nptel/week8 : %
emacs -nw highest.cpp   (wd: ~/Desktop/nptel/week7)

[1]+  Stopped                 emacs -nw highest.cpp  (wd: ~/Desktop/nptel/week7)
(wd now: ~/Desktop/nptel/week8)
~/Desktop/nptel/week8 : s++ house.cpp
+ g++ house.cpp -Wall /Users/abhiram/simplecpp/lib/libsprite.a -I/Users/abhiram/simplecpp/in
/opt/X11/include -L/opt/X11/lib -lX11 -std=c++17
~/Desktop/nptel/week8 : ./a.out
```



```
                    emacs -nw highest.cpp   (wd: ~/Desktop/nptel/week7)
p/nptel/week8)
eek8 : s++ findchar.cpp
p -Wall /Users/abhiram/simplecpp/lib/libsprite.a -I/Users/abhiram/simplecpp
de -L/opt/X11/lib -lX11 -std=c++17
eek8 : ./a.out


eek8 : %
.cpp   (wd: ~/Desktop/nptel/week7)

                    emacs -nw highest.cpp   (wd: ~/Desktop/nptel/week7)
p/nptel/week8)
~/Desktop/nptel/week8 : s++ house.cpp
+ g++ house.cpp -Wall /Users/abhiram/simplecpp/lib/libsprite.a -I/Users/abhiram/simplecpp/in
/opt/X11/include -L/opt/X11/lib -lX11 -std=c++17
~/Desktop/nptel/week8 : ./a.out
~/Desktop/nptel/week8 : %
emacs -nw highest.cpp   (wd: ~/Desktop/nptel/week7)

[1]+  Stopped           emacs -nw highest.cpp   (wd: ~/Desktop/nptel/week7)
(wd now: ~/Desktop/nptel/week8)
~/Desktop/nptel/week8 : s++ house.cpp
+ g++ house.cpp -Wall /Users/abhiram/simplecpp/lib/libsprite.a -I/Users/abhiram/simplecpp/in
/opt/X11/include -L/opt/X11/lib -lX11 -std=c++17
~/Desktop/nptel/week8 : ./a.out
```



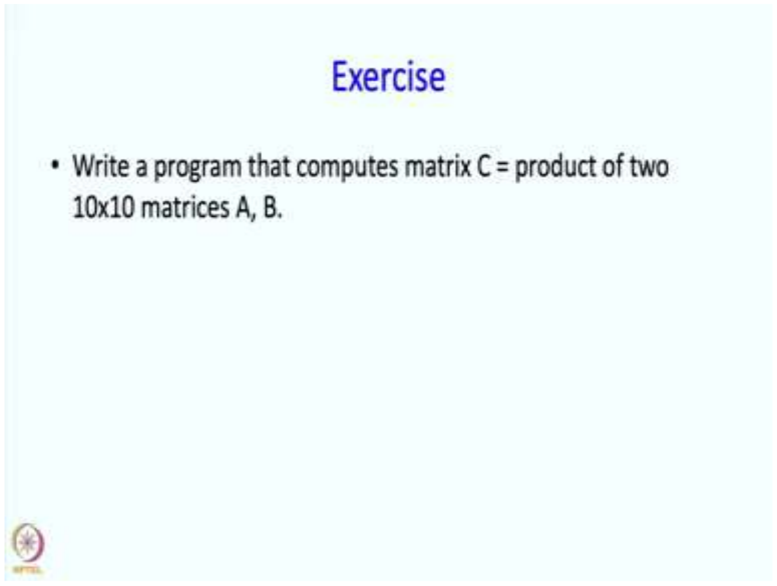
```
                    emacs -nw highest.cpp   (wd: ~/Desktop/nptel/week7)
p/nptel/week8)
eek8 : s++ findchar.cpp
p -Wall /Users/abhiram/simplecpp/lib/libsprite.a -I/Users/abhiram/simplecpp
de -L/opt/X11/lib -lX11 -std=c++17
eek8 : ./a.out


eek8 : %
.cpp   (wd: ~/Desktop/nptel/week7)

                    emacs -nw highest.cpp   (wd: ~/Desktop/nptel/week7)
p/nptel/week8)
~/Desktop/nptel/week8 : s++ house.cpp
+ g++ house.cpp -Wall /Users/abhiram/simplecpp/lib/libsprite.a -I/Users/abhiram/simplecpp/in
/opt/X11/include -L/opt/X11/lib -lX11 -std=c++17
~/Desktop/nptel/week8 : ./a.out
~/Desktop/nptel/week8 : %
emacs -nw highest.cpp   (wd: ~/Desktop/nptel/week7)

[1]+  Stopped           emacs -nw highest.cpp   (wd: ~/Desktop/nptel/week7)
(wd now: ~/Desktop/nptel/week8)
~/Desktop/nptel/week8 : s++ house.cpp
+ g++ house.cpp -Wall /Users/abhiram/simplecpp/lib/libsprite.a -I/Users/abhiram/simplecpp/in
/opt/X11/include -L/opt/X11/lib -lX11 -std=c++17
~/Desktop/nptel/week8 : ./a.out
```



```
                    emacs -nw highest.cpp  (wd: ~/Desktop/nptel/week7)
p/nptel/week8)
eek8 : s++ findchar.cpp
p -Wall /Users/abhiram/simplecpp/lib/libsprite.a -I/Users/abhiram/simplecpp
de -L/opt/X11/lib -lX11 -std=c++17
eek8 : ./a.out


eek8 : %
.cpp   (wd: ~/Desktop/nptel/week7)

                    emacs -nw highest.cpp  (wd: ~/Desktop/nptel/week7)
p/nptel/week8)
~/Desktop/nptel/week8 : s++ house.cpp
+ g++ house.cpp -Wall /Users/abhiram/simplecpp/lib/libsprite.a -I/Users/abhiram/simplecpp/in
/opt/X11/include -L/opt/X11/lib -lX11 -std=c++17
~/Desktop/nptel/week8 : ./a.out
~/Desktop/nptel/week8 : %
emacs -nw highest.cpp   (wd: ~/Desktop/nptel/week7)

[1]+  Stopped           emacs -nw highest.cpp  (wd: ~/Desktop/nptel/week7)
(wd now: ~/Desktop/nptel/week8)
~/Desktop/nptel/week8 : s++ house.cpp
+ g++ house.cpp -Wall /Users/abhiram/simplecpp/lib/libsprite.a -I/Users/abhiram/simplecpp/in
/opt/X11/include -L/opt/X11/lib -lX11 -std=c++17
~/Desktop/nptel/week8 : ./a.out
```

So, maybe let us just try, just try to change this a little bit. Okay, so how is that, how it rotates 10 degrees repeat 36 times, so let us maybe I know what happened. So, I should really be saying say house.left or house.right. Rotate takes the angle in radians so that is why it rotated too fast. By too much. See, this is rotating about the center and it will come back. Well houses are not supposed to rotate, but the point of this was just to show you that the center the rotation center was here. This is the point with which, with respect to which we specified the coordinates and that is the point which we can use to rotate as well. So, you can do fun things with this this kind of rotation if you want alright.

(Refer Slide Time: 21:06)

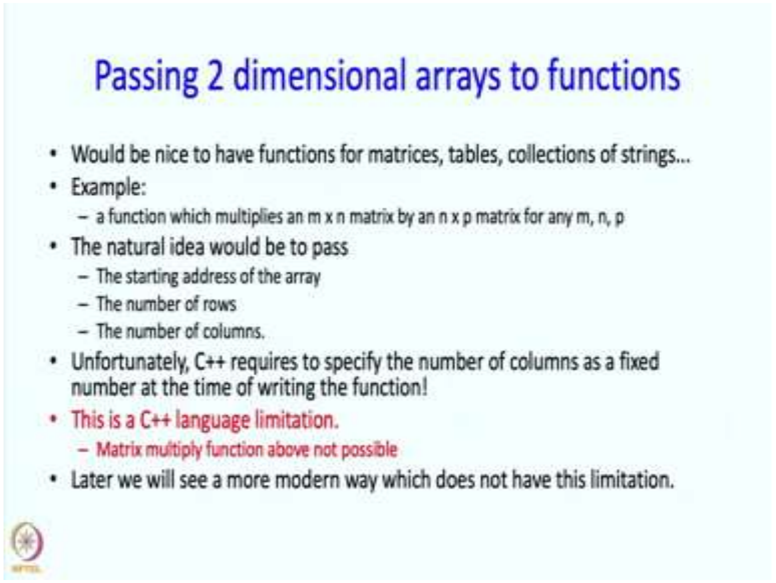


Exercise

- Write a program that computes matrix C = product of two 10x10 matrices A, B.

So, as an exercise in two dimensional arrays, I would like you to write a program which multiplies two matrices, say, A and B and computes their product in a matrix C. And for this you may remember the formula for matrix multiplication. So C_{ij} is equal to A_{ik} times B_{kj}. But, here to sum this over all, the values of k. Alright! So, what we are going to do, look at next is how do you pass two dimensional arrays to functions.

(Refer Slide Time: 21:44)



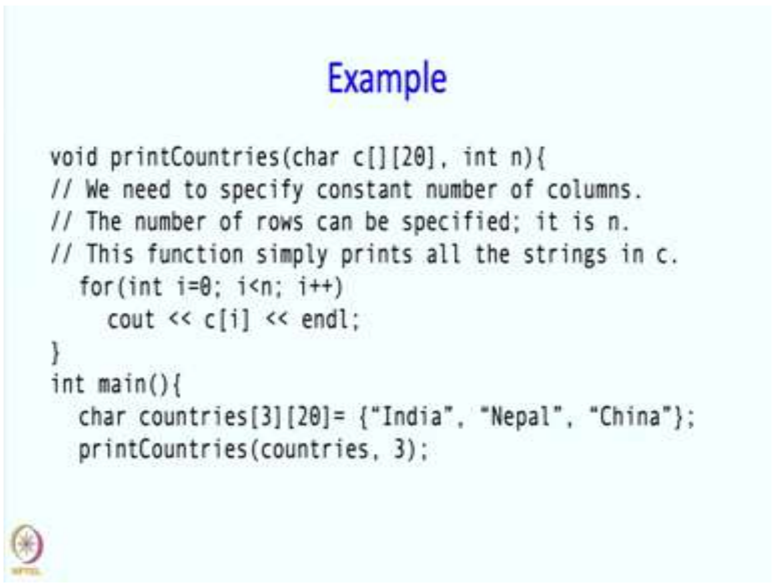
Passing 2 dimensional arrays to functions

- Would be nice to have functions for matrices, tables, collections of strings...
- Example:
  - a function which multiplies an m x n matrix by an n x p matrix for any m, n, p
- The natural idea would be to pass
  - The starting address of the array
  - The number of rows
  - The number of columns.
- Unfortunately, C++ requires to specify the number of columns as a fixed number at the time of writing the function!
- This is a C++ language limitation.
  - Matrix multiply function above not possible
- Later we will see a more modern way which does not have this limitation.

So, this is, of course, something that is desirable to have, because, of course, you could, you might want to have functions, say a function which multiplies and n by n matrix by n by p matrix, so for any m and p. And the natural idea would be to pass the starting address of the array, the number of rows, the number of columns. Unfortunately, C++ does not quite do this, right. So it requires you to specify the number of columns as a fixed number at the time of writing the function.

So, there is a technical reason for this. And I am not going to explain to explain that to you, because it is kind of a silly reason. It is kind of a complicated, there is a complicated reason for it. But the upshot is that the second, the second dimension, okay, the number of columns has to be given in a fixed manner. Okay, so I will give you an example of this just to make it clear. This is definitely a C++ language limitation, but the more modern thing that we will later on will not have this limitation and so this will be taken care of later on.

(Refer Slide Time: 23:00)

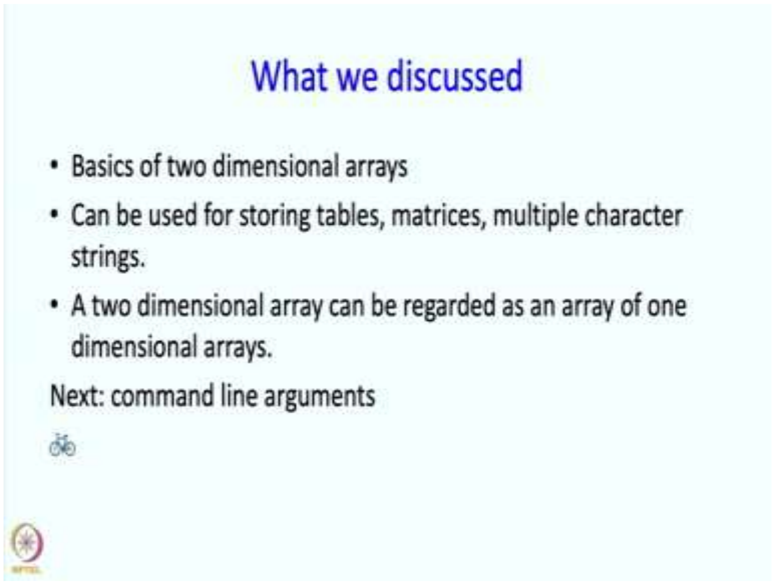


Example

```
void printCountries(char c[][20], int n){
// We need to specify constant number of columns.
// The number of rows can be specified; it is n.
// This function simply prints all the strings in c.
  for(int i=0; i<n; i++)
    cout << c[i] << endl;
}
int main(){
  char countries[3][20]= {"India", "Nepal", "China"};
  printCountries(countries, 3);
```

Just to tell you what I mean this is what it looks like, so the second dimension has to the constant. So, if I want to write it, if I want to have a function for different length 25 over here, sorry, this function will not work, if I want to pass an array with different length this function will not work. Anyway let us just see what this might be doing. So, here we have, what are we doing, we are getting an array with some unknown number of columns but there is an argument 'n' which is going to specify how many columns that is and the function then simply prints all the strings in C. So this is the main function, this is our old array, so it has 3, 3 strings in it, and if you call it in this manner, then all the three names will get printed.

(Refer Slide Time: 24:01)



What we discussed

- Basics of two dimensional arrays
- Can be used for storing tables, matrices, multiple character strings.
- A two dimensional array can be regarded as an array of one dimensional arrays.

Next: command line arguments

So. what did we discuss in this segment? So, we discussed the basics of 2 dimensional arrays We said that they can be used for storing tables, matrices and multiple character strings. We also said that a 2 dimensional array can be regarded as an array of 1 dimensional arrays. Next, we are going to come to command line arguments, but we will take a break first.