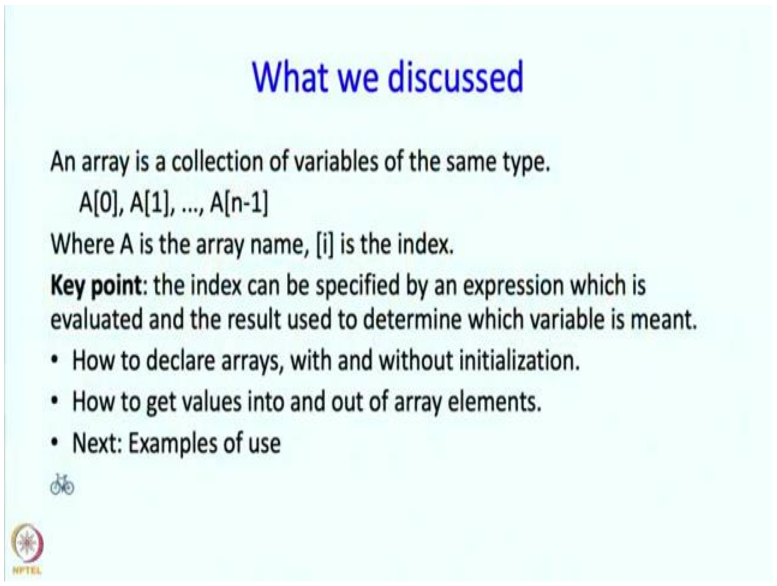
An Introduction to Programming through C++

Professor Abhiram G. Ranade
Department of Computer Science and Engineering
Indian Institute of Technology Bombay

Lecture No. 15 Part - 2
Array Part – 1
Marks averaging problem

(Refer Slide Time: 0:38)



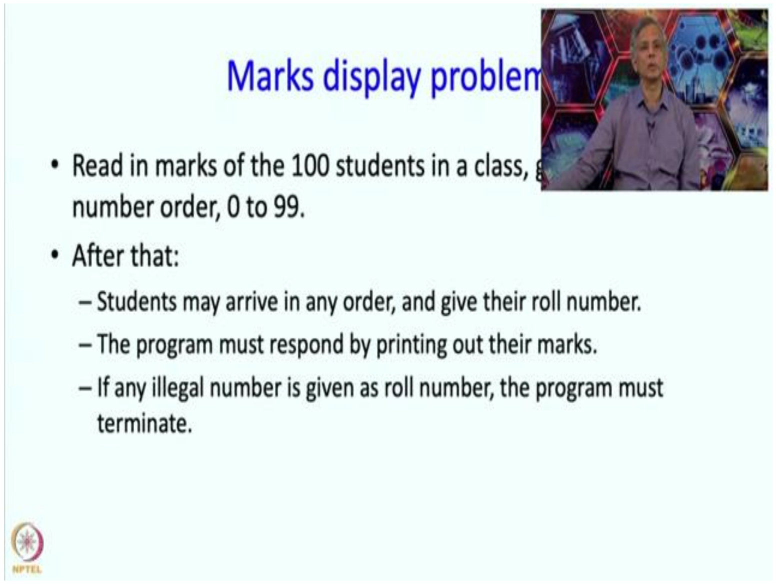
What we discussed

An array is a collection of variables of the same type.
A[0], A[1], ..., A[n-1]
Where A is the array name, [i] is the index.
Key point: the index can be specified by an expression which is evaluated and the result used to determine which variable is meant.

- How to declare arrays, with and without initialization.
- How to get values into and out of array elements.
- Next: Examples of use

Welcome back, in the previous segment we saw how to declare arrays with and without initialization and how to access the elements, how to use them in computation, how to read and read values into it and print the values.

(Refer Slide Time: 0:41)



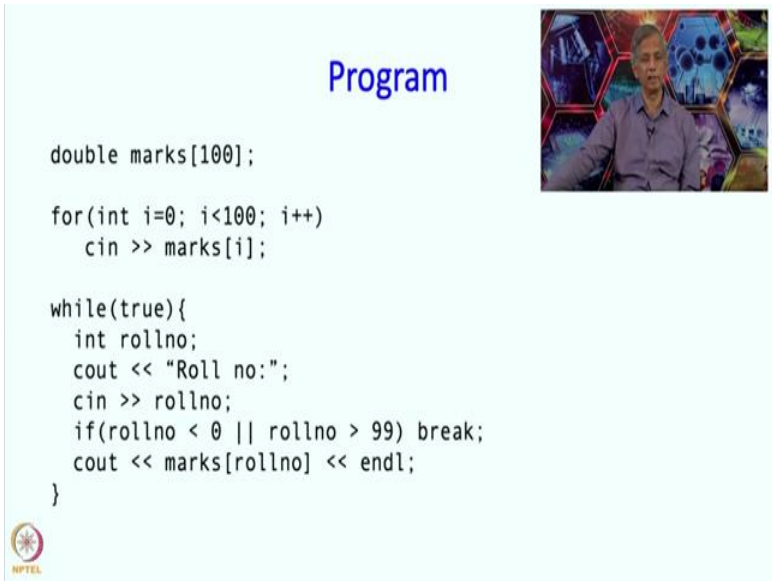
Marks display problem

- Read in marks of the 100 students in a class, g number order, 0 to 99.
- After that:
  - Students may arrive in any order, and give their roll number.
  - The program must respond by printing out their marks.
  - If any illegal number is given as roll number, the program must terminate.

Now we are going to see some larger examples of using arrays. So we are going to start with relatively simple familiar sounding problem. So the problem is you are supposed to write a program that reads in marks of 100 students in a class, and the marks are given in the order of the roll number. And let us say in this class, the roll numbers happened to go from 0 to 999. So yes, I know that the numbers usually do not start at 0, but just, just for fun let us say they do start over 0, over 0. So in this class the roll number, there are 100 students and their roll numbers go through, go from 0 through 99. So that is the first part of what is the program is supposed to do.

After that students may arrive in any order, and type their roll number using the keyboard. The program must reply to that by printing out their marks. So maybe first the student with the roll number 43 comes, then the program is supposed to print out what marks student number 43 got, then there will be 67 so marks for 67 will be printed, then there maybe 6 then the marks for the student with roll number 6 should be printed, and so on. Now, if at any time an illegal number is given then the program must terminate, so this is this specification, this is what our program is expected to achieve.

(Refer Slide Time: 2:20)



Program

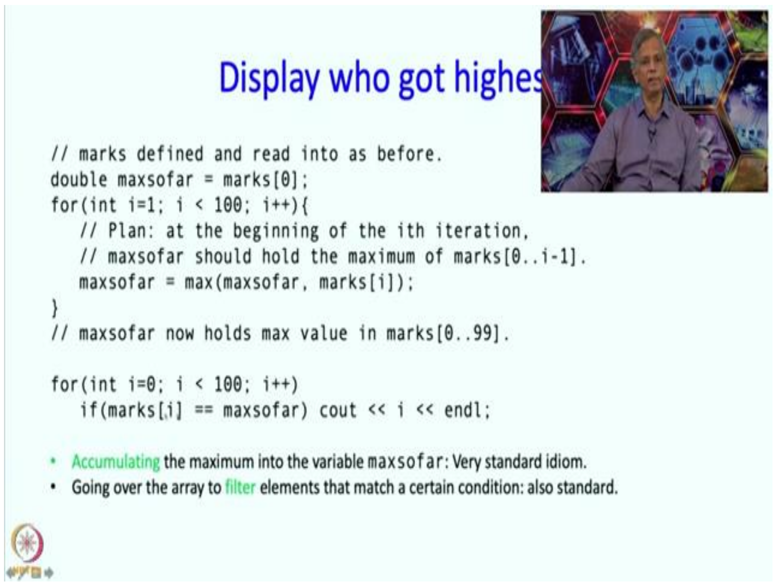
```
double marks[100];

for(int i=0; i<100; i++)
   cin >> marks[i];

while(true){
  int rollno;
  cout << "Roll no:";
  cin >> rollno;
  if(rollno < 0 || rollno > 99) break;
  cout << marks[rollno] << endl;
}
```

So, what does the program look like? So we clearly should have an array in which we should be storing the marks and so let us say we have an array of doubles and of 100 elements, we need 100 elements because we have 100 students. Now the first step is to read in all the values, all the marks that the students have received. And we were told that the marks will be typed in this order. So first the marks were 0, then the marks were 1, then the marks were 2, so it is natural to write this for loop, where in the ith iteration we are reading in the marks obtained by student i and placing that in marks of i.

And then there is the loop in which we are going to wait for students to come and type their roll numbers. So for this lets say we have a variable called roll number and we will print out a message saying roll number, so this will be a queue to the students to type their roll numbers. So then we will read in whatever the students are typing and now if the students have typed a valid roll number, then we should print out the corresponding marks.

So, if roll number less than 0 or roll number greater than 99 that is a invalid number, so in that case we are going to stop execution so we are going to break out of this loop. Otherwise, we know that the roll number that was typed in lies between 0 and 99, so in fact it is a valid index as far as the array is concerned, so we are just going to print out marks of roll number and that is the end of the loop. So if you break out of it, the program will terminate and that would be the end.

(Refer Slide Time: 4:25)



Display who got highes

```
// marks defined and read into as before.
double maxsofar = marks[0];
for(int i=1; i < 100; i++){
   // Plan: at the beginning of the ith iteration,
   // maxsofar should hold the maximum of marks[0..i-1].
   maxsofar = max(maxsofar, marks[i]);
}
// maxsofar now holds max value in marks[0..99].

for(int i=0; i < 100; i++)
   if(marks[i] == maxsofar) cout << i << endl;
```

- Accumulating the maximum into the variable maxsofar: Very standard idiom.
- Going over the array to filter elements that match a certain condition: also standard.

Now let us do a slightly more interesting version of this where we are going to read in everything, but we are going to print out the roll numbers of those students who got the highest marks. So here is the program, so let us say that the marks array was defined as before and we have read in the marks exactly as before. So now, we want to figure out first what is the, what are the highest marks.

So for this we are going to have a variable called maxsofar, so maxsofar we are going to initialize to 0 and the way to interpret this is that at the beginning of the ith iteration of this loop we are going to make sure that maxsofar holds the maximum of the marks between 0 and i or 0 and i minus 1. So indeed, before entering the loop, maxsofar is the value of marks[0], so it is indeed trivially the maximum amongst all say marks[0] all the way till marks[i-1] which is 0 itself, so it is in fact.

So there is only one element in this set marks of 0 through i minus 1 because both of these will be 0 and indeed maxsofar is that value. But this plan says that not only should this be true at the beginning of the (ith), of the first iteration when i is 1, but it should be true for all iterations. So if it is true for all iterations, when our body of the loop should make it be so. So, what do we need to do?

Well so when we enter this loop, we know that maxsofar is the maximum amongst 0 through i minus 1. So, if the next number marks of i is bigger than this then our marks, our max should change. Otherwise, our max should be as before. So all that we need to do is to write maxsofar is the max of maxsofar, whatever we have seen so far and whatever we see next, in the next marks value.

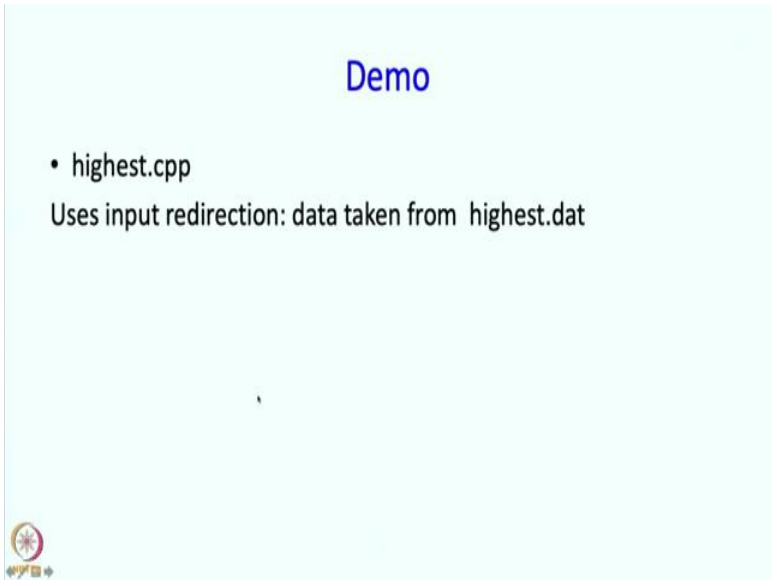
So this max is a statement, is a function which is already defined in C++. And in fact we know that C++, functions that C++ defines for math purposes are in this include in this header file called cmath, so indeed max comes from that header file cmath, that is it. So what would be, what would be true at the end of it? So on the 99th iteration, what would be true? So on the 99th iteration what would be true? So i equals 1 and i equals 99, so at that point maxsofar should hold the maximum of everything from 0 through 99 and indeed that means maxsofar will have value of the maximum possible marks. So at this point we have identified what the maximum marks so far are. So maxsofar now holds max value in the entire array. Now, we just have to check which are the marks or which are the roll numbers really who have got that many marks.

So again go over all the elements this time even including 0 and we are going to check are the marks[i] equal to maxsofar? If so then we should print out i so that is it. So there are two ideas which have been used in this, which are worth noting. So in this first part, we are doing something which might be called accumulation.

So maxsofar is accumulating the maximum value, it is keeping track of the maximum value and this is a very standard idiom, here we are keeping track of the maximum value, but you may keep track say of the sum of all the values. Again it is going to be something like that instead of max of maxsofar, and max i, it might be something like plus of something and something, something and marks[i], so that is one idiom.

And then in the second part we are going over all the elements again, but we are printing out only some subset that subset which satisfies this condition. And in other words we are filtering out things. And therefore, this idiom is often called a filtering idiom.

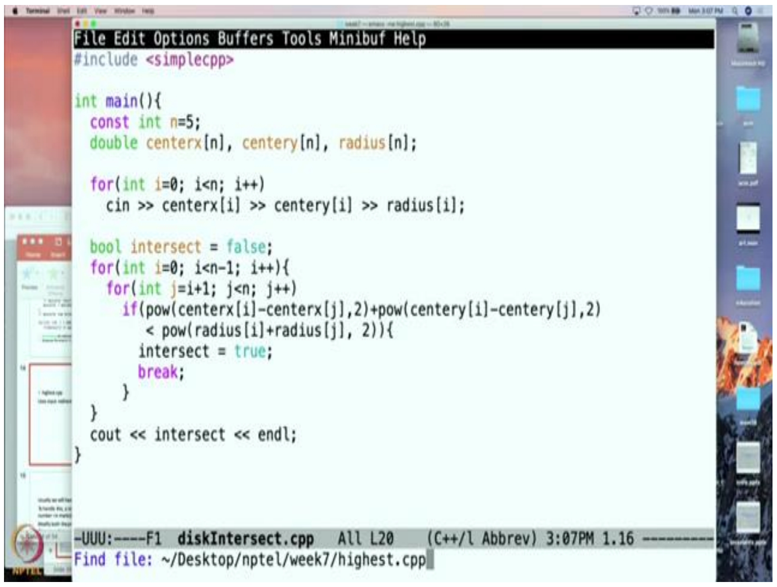
(Refer Slide Time: 9:25)



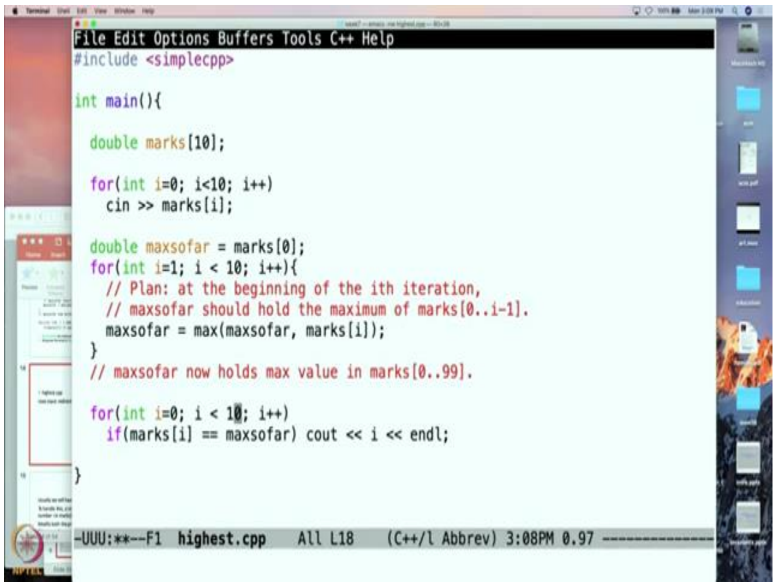
Demo

- highest.cpp

Uses input redirection: data taken from highest.dat



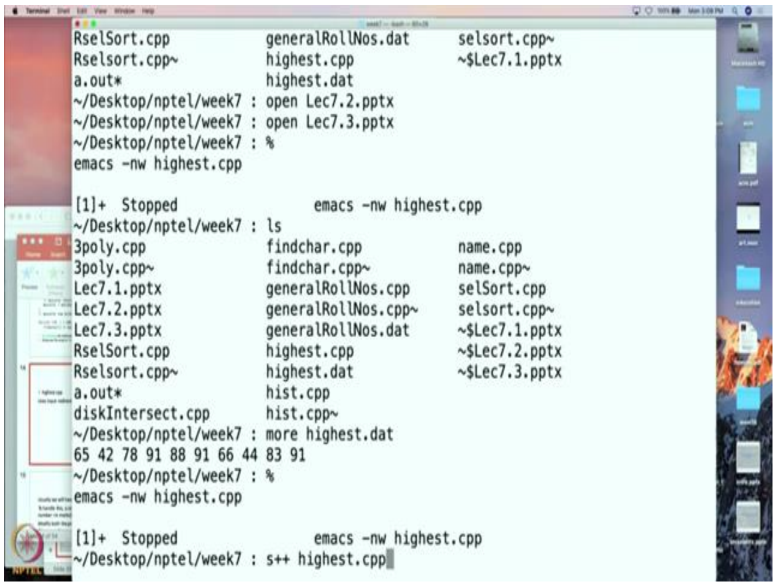
```
File Edit Options Buffers Tools Minibuf Help
#include <simplecpp>

int main(){
  const int n=5;
  double centerx[n], centery[n], radius[n];

  for(int i=0; i<n; i++)
    cin >> centerx[i] >> centery[i] >> radius[i];

  bool intersect = false;
  for(int i=0; i<n-1; i++){
    for(int j=i+1; j<n; j++)
      if(pow(centerx[i]-centerx[j],2)+pow(centery[i]-centery[j],2)
         < pow(radius[i]+radius[j], 2)){
        intersect = true;
        break;
      }
  }
  cout << intersect << endl;
}

-UUU:----F1  diskIntersect.cpp    All L20    (C++/l Abbrev) 3:07PM 1.16 --------
Find file: ~/Desktop/nptel/week7/highest.cpp
```



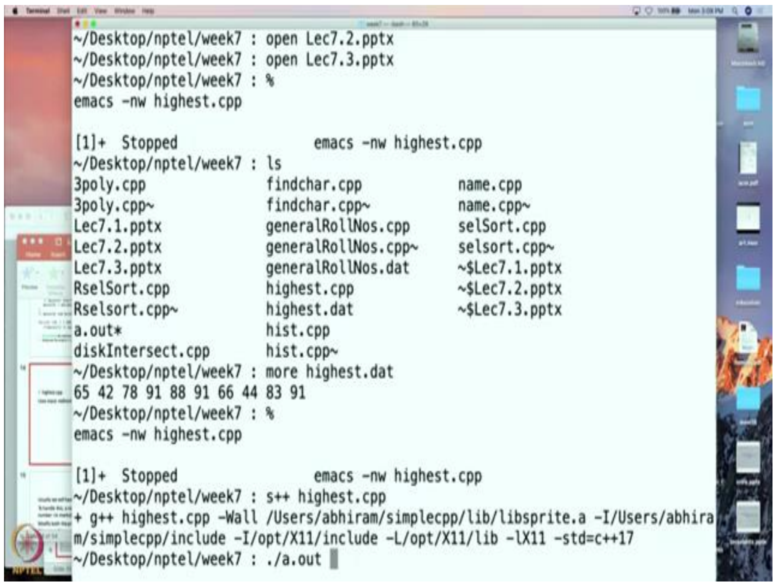
```
File Edit Options Buffers Tools C++ Help
#include <simplecpp>

int main(){

  double marks[10];

  for(int i=0; i<10; i++)
    cin >> marks[i];

  double maxsofar = marks[0];
  for(int i=1; i < 10; i++){
    // Plan: at the beginning of the ith iteration,
    // maxsofar should hold the maximum of marks[0..i-1].
    maxsofar = max(maxsofar, marks[i]);
  }
  // maxsofar now holds max value in marks[0..99].

  for(int i=0; i < 10; i++)
    if(marks[i] == maxsofar) cout << i << endl;

}

-UUU:**--F1  highest.cpp     All L18    (C++/l Abbrev) 3:08PM 0.97 ------------
```

So let us see a demo of this and this is going to be in file highest.cpp which I have created already, but I am going to show it to you. So let me get to that file, so highest.cpp is this. So you can see that we have used marks[100] array as before and then we are doing cin marks, all of this, and this is our max array. Now this is the program that you have seen, now I am going to make a change it to it because I do not want to type all the (100 element) 100 numbers. So I am going to change it so that I will only use 10 numbers so I should really change all the 100's to 10's.

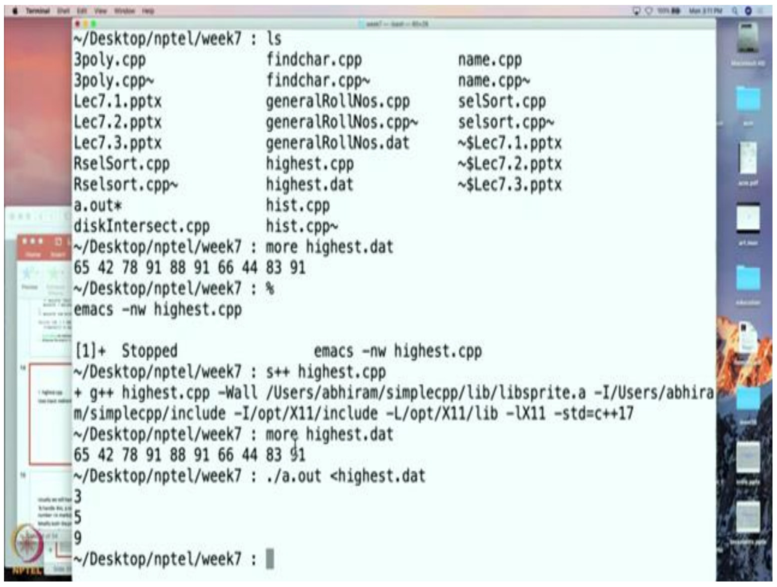
(Refer Slide Time: 10:38)



```
RselSort.cpp            generalRollNos.dat      selsort.cpp~
Rselsort.cpp~           highest.cpp             ~$Lec7.1.pptx
a.out*                  highest.dat
~/Desktop/nptel/week7 : open Lec7.2.pptx
~/Desktop/nptel/week7 : open Lec7.3.pptx
~/Desktop/nptel/week7 : %
emacs -nw highest.cpp

[1]+  Stopped                 emacs -nw highest.cpp
~/Desktop/nptel/week7 : ls
3poly.cpp               findchar.cpp            name.cpp
3poly.cpp~              findchar.cpp~           name.cpp~
Lec7.1.pptx             generalRollNos.cpp      selSort.cpp
Lec7.2.pptx             generalRollNos.cpp~     selsort.cpp~
Lec7.3.pptx             generalRollNos.dat      ~$Lec7.1.pptx
RselSort.cpp            highest.cpp             ~$Lec7.2.pptx
Rselsort.cpp~           highest.dat             ~$Lec7.3.pptx
a.out*                  hist.cpp
diskIntersect.cpp       hist.cpp~
~/Desktop/nptel/week7 : more highest.dat
65 42 78 91 88 91 66 44 83 91
~/Desktop/nptel/week7 : %
emacs -nw highest.cpp

[1]+  Stopped                 emacs -nw highest.cpp
~/Desktop/nptel/week7 : s++ highest.cpp
```



```
~/Desktop/nptel/week7 : open Lec7.2.pptx
~/Desktop/nptel/week7 : open Lec7.3.pptx
~/Desktop/nptel/week7 : %
emacs -nw highest.cpp

[1]+  Stopped                 emacs -nw highest.cpp
~/Desktop/nptel/week7 : ls
3poly.cpp               findchar.cpp            name.cpp
3poly.cpp~              findchar.cpp~           name.cpp~
Lec7.1.pptx             generalRollNos.cpp      selSort.cpp
Lec7.2.pptx             generalRollNos.cpp~     selsort.cpp~
Lec7.3.pptx             generalRollNos.dat      ~$Lec7.1.pptx
RselSort.cpp            highest.cpp             ~$Lec7.2.pptx
Rselsort.cpp~           highest.dat             ~$Lec7.3.pptx
a.out*                  hist.cpp
diskIntersect.cpp       hist.cpp~
~/Desktop/nptel/week7 : more highest.dat
65 42 78 91 88 91 66 44 83 91
~/Desktop/nptel/week7 : %
emacs -nw highest.cpp

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



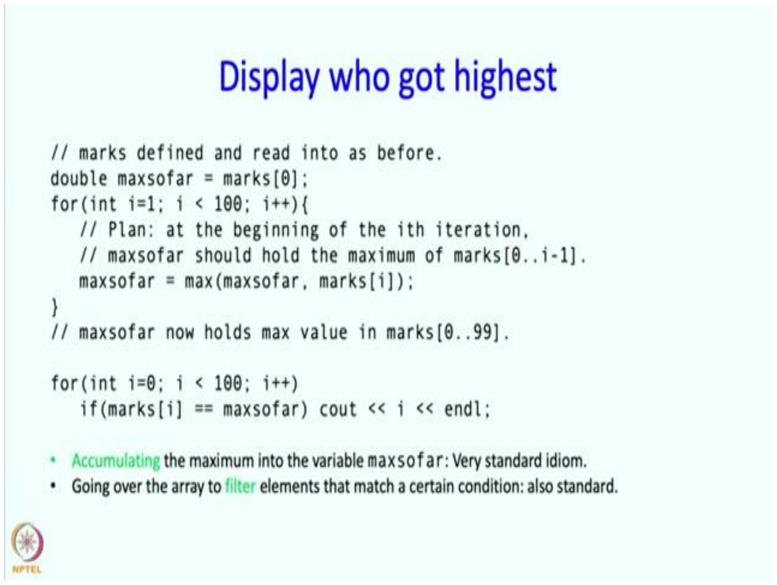
```
~/Desktop/nptel/week7 : ls
3poly.cpp               findchar.cpp            name.cpp
3poly.cpp~              findchar.cpp~           name.cpp~
Lec7.1.pptx             generalRollNos.cpp      selSort.cpp
Lec7.2.pptx             generalRollNos.cpp~     selsort.cpp~
Lec7.3.pptx             generalRollNos.dat      ~$Lec7.1.pptx
RselSort.cpp            highest.cpp             ~$Lec7.2.pptx
Rselsort.cpp~           highest.dat             ~$Lec7.3.pptx
a.out*                  hist.cpp
diskIntersect.cpp       hist.cpp~
~/Desktop/nptel/week7 : more highest.dat
65 42 78 91 88 91 66 44 83 91
~/Desktop/nptel/week7 : %
emacs -nw highest.cpp

[1]+  Stopped                 emacs -nw highest.cpp
~/Desktop/nptel/week7 : s++ highest.cpp
+ g++ highest.cpp -Wall /Users/abhiram/simplecpp/lib/libsprite.a -I/Users/abhira
m/simplecpp/include -I/opt/X11/include -L/opt/X11/lib -lX11 -std=c++17
~/Desktop/nptel/week7 : more highest.dat
65 42 78 91 88 91 66 44 83 91
~/Desktop/nptel/week7 : ./a.out <highest.dat
3
5
9
~/Desktop/nptel/week7 :
```

Alright, so let us see, so let us compile this and let us run it. Now, when I am going to run this, I am going to do something slightly unusual, something that you may not have seen so far, you have not seen so far in this course. So normally I am expecting, so when I run this program C++ expects me to type the values from the keyboard. But, if I have to test a program often then typing the values, especially 10 values even is a bit cumbersome.

So what I am going to do instead, is that I have already typed in those values in this file highest.dat. So maybe I should show you that first. So, if I do this highest.dat, it will show you that I have already typed in these values, so these are my 10 marks, so marks for student 0, roll number 0 are 65, for student roll number 1 42, 2 78 and so on. So these numbers I have already typed.

Now if I do ./a.out and I say <highest.dat, then instead of taking input from the keyboard this program will take the input from that file, so let us see that. So it took the input from the file and then it has printed those roll numbers for which the marks were the highest possible, so let us check that. So, if we look at this which were the highest marks? Well 91 is the highest mark and where did this appear? So this is roll number 0, this is roll number 1, 2, 3 so indeed roll number 3 got 91, 4, 5 so indeed 5 got 91, 6, 7, 8, 9 so roll number 9 also got 91, so it has indeed printed out things correctly.

(Refer Slide Time: 12:54)



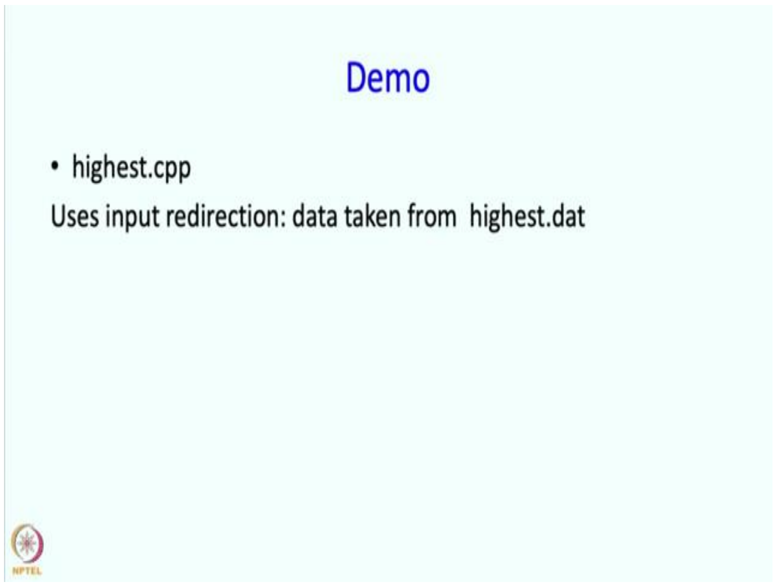
Display who got highest

```
// marks defined and read into as before.
double maxsofar = marks[0];
for(int i=1; i < 100; i++){
   // Plan: at the beginning of the ith iteration,
   // maxsofar should hold the maximum of marks[0..i-1].
   maxsofar = max(maxsofar, marks[i]);
}
// maxsofar now holds max value in marks[0..99].

for(int i=0; i < 100; i++)
   if(marks[i] == maxsofar) cout << i << endl;
```

- Accumulating the maximum into the variable `maxsofar`: Very standard idiom.
- Going over the array to filter elements that match a certain condition: also standard.

So here we saw that our program this program was executed, but we changed the number of students so that only 10 students were used.

(Refer Slide Time: 13:10)



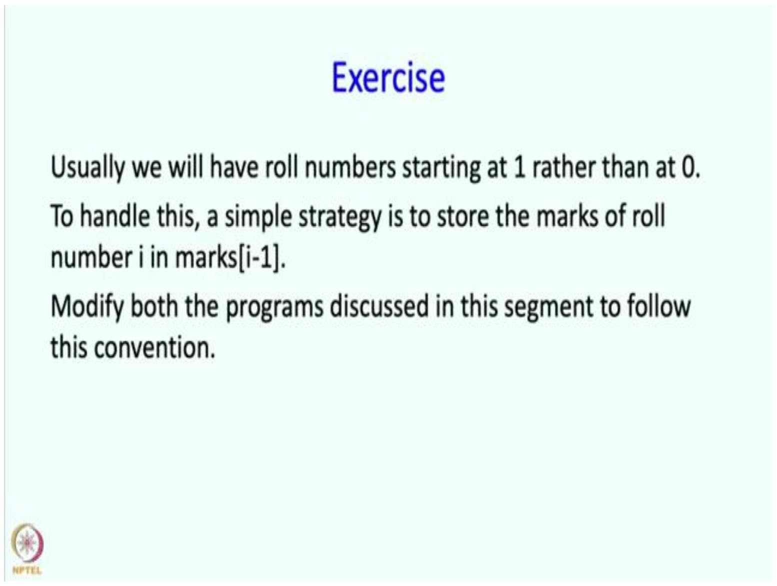
Demo

- highest.cpp

Uses input redirection: data taken from highest.dat

And here the input data, sorry the data was taken from the file rather than by rather than through the keyboard because we wrote <highest.dat.

(Refer Slide Time: 13:28)



Exercise

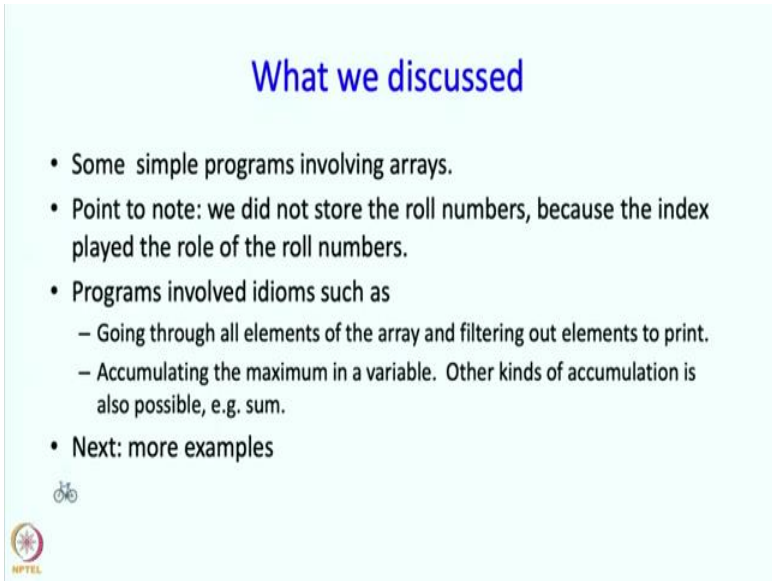
Usually we will have roll numbers starting at 1 rather than at 0.

To handle this, a simple strategy is to store the marks of roll number i in marks[i-1].

Modify both the programs discussed in this segment to follow this convention.

So a small exercise for you, usually we will have roll numbers starting at 1 rather than 0, to handle this a simple strategy is to store the marks of roll number i in marks[i-1]. So we have written two programs so far, so this exercise says modify both the programs so that they follow this convention. So they follow the more usual convention that student roll numbers start at 1 and not at 0.

(Refer Slide Time: 14:04)



What we discussed

- Some simple programs involving arrays.
- Point to note: we did not store the roll numbers, because the index played the role of the roll numbers.
- Programs involved idioms such as
  - Going through all elements of the array and filtering out elements to print.
  - Accumulating the maximum in a variable. Other kinds of accumulation is also possible, e.g. sum.
- Next: more examples

So what have we discussed in this segment? Well, we saw a couple of simple programs involving arrays. And an important point to note is that although roll numbers are involved, we did not actually store the roll number anywhere. Because the index played the role of the roll numbers. So because we stored the marks of roll number 0 first, then the marks of roll number 1, the marks of roll number 2, we sort of implicitly made a correspondence between the role number and the mark.

And then the program involved a few important idioms. So going through all the elements of an array and filtering out the elements to print and also the accumulation the maximum of all the elements in a variable. And we will continue with more examples, but we will take a quick break.