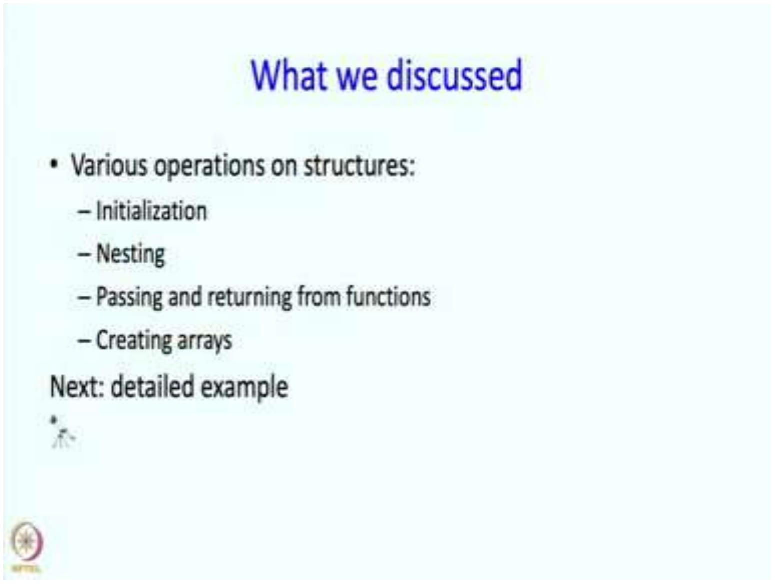
An Introduction to Programming through C++

Professor Abhiram G. Ranade
Department of computer science and Engineering,
Indian Information Technology Bombay
Lecture 19 Part 3
Structures
An example program

Welcome back. In the previous segment we discussed various operations on structures.

(Refer Slide Time: 0:18)



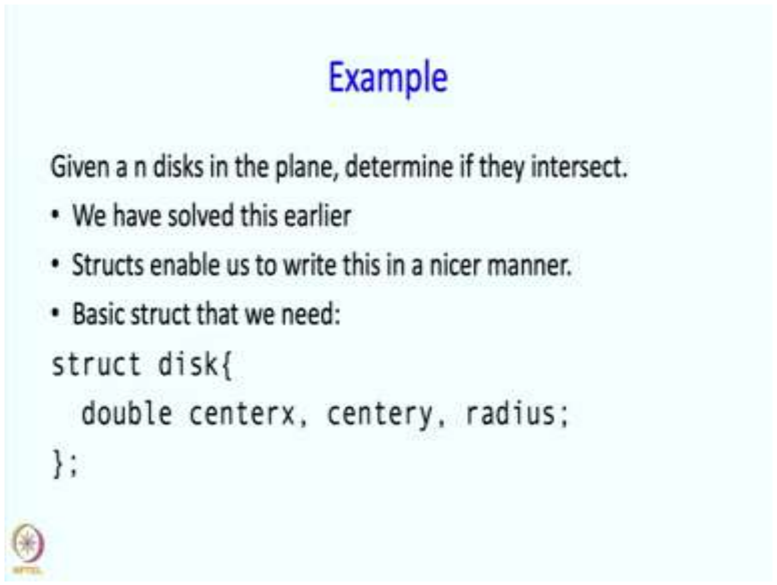
What we discussed

- Various operations on structures:
  - Initialization
  - Nesting
  - Passing and returning from functions
  - Creating arrays

Next: detailed example

Now, we are going to put together all these things into a somewhat longish example.

(Refer Slide Time: 0:29)



Example

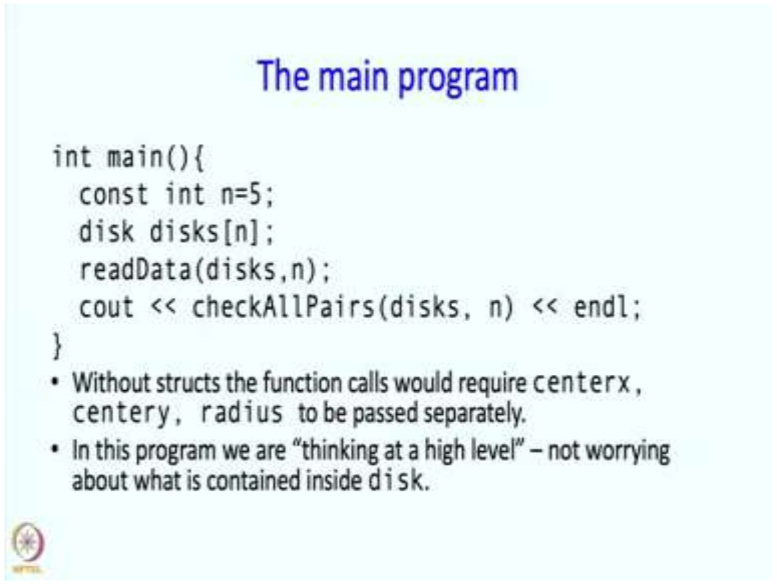
Given a n disks in the plane, determine if they intersect.

- We have solved this earlier
- Structs enable us to write this in a nicer manner.
- Basic struct that we need:

```
struct disk{
  double centerx, centery, radius;
};
```

So, here is the problem that we are going to solve. So, given n discs in the plane determine if they intersect? You may remember that we have written code for this earlier. Now, with structs, we are going to write the same code or code for the same job, but you will see that in some way that a code will be nicer. So, let us start with the basic struct that we need. So this is a struct disk and instead of having point, I could have had points as well, I could have just put in center x center y and radius as its 3 members.

(Refer Slide Time: 1:10)



The main program

```
int main(){
  const int n=5;
  disk disks[n];
  readData(disks,n);
  cout << checkAllPairs(disks, n) << endl;
}
```

- Without structs the function calls would require `centerx`, `centery`, `radius` to be passed separately.
- In this program we are "thinking at a high level" – not worrying about what is contained inside `disk`.

So, let me get to the main program. So, we are going to have five discs just for simplicity and so I have put in constant int n to denote the number of discs. And now I am just going to create just a number of array of disks. And so by the way note that in earlier case what we had to do in the main program was to create not just a single array but an array to store the coordinate another array to store the Y coordinate and another array to store the radius.

So, that makes the program verbose and furthermore, it does not really tell you that something is being done about discs, unless you write a comment, but here very compactly you understand that, yes, that we are doing something about discs. The first operation that we are going to perform is to read data. You will note that in the program that we wrote earlier, we put the code for reading right here, but now the code is elsewhere.

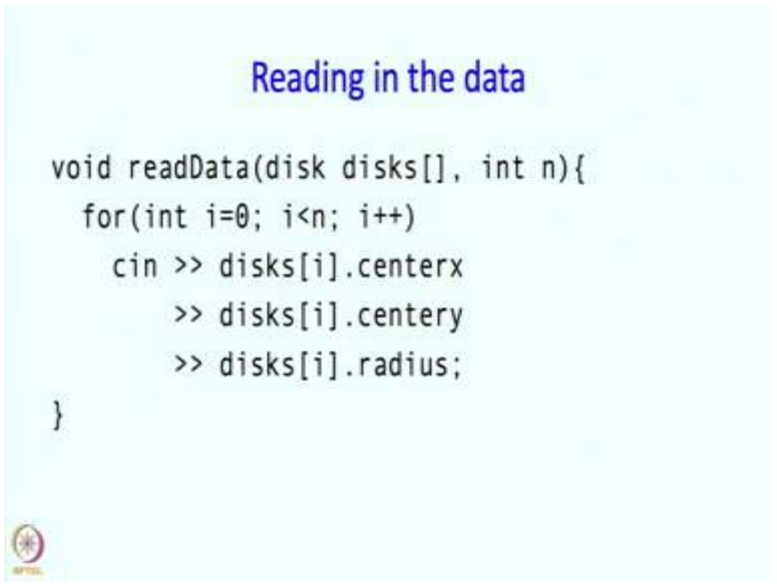
But here in the main program itself, if you know what that code is supposed to be doing is reading data and reading and in particular this looks like we are reading the various attributes of the discs. So, again this makes for better readability. Then we have another function which says check all pairs. So, you can tell easily that look we are going to do something with all pairs and the result is going to be printed.

So that is it. Main program is quite short and sweet and it has steps which have been given names and these steps are functions and we are passing data to the functions but the data is not, it does not look like a lot of data. I mean, there are not many names, there are not many parameters, arguments that we are passing. We are just clearly passing the main argument which is the disc the number of elements in it.

So there is a certain kind of compactness or neatness to this entire program. So, without structs, so you would have to you have lots of arrays being passed. So, that would just look a bit more cluttered. And in some sense, in this program, we are encouraging, thinking at a high level. So, here when we write read data we are saying, “Oh, somehow it will be read.”

I do not want to think about how exactly that is going to happen, but I know that it is going to happen and similarly, check all pairs. How is it going to happen? It is given somewhere else but this is what I really want to happen. So, this sort of tells you what is going at a higher level and does not force you to read all the details and then figure out. So, let us go to this function. First of all, reading the data.

(Refer Slide Time: 4:25)

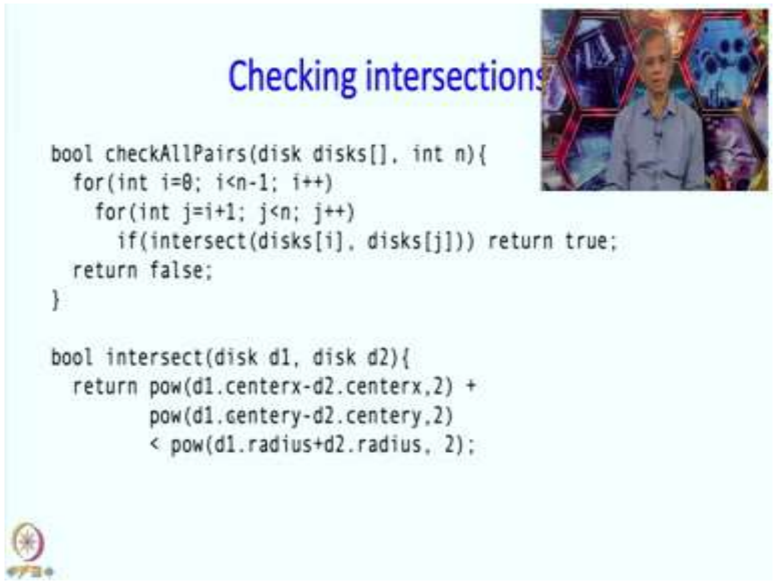


Reading in the data

```
void readData(disk disks[], int n){
  for(int i=0; i<n; i++)
    cin >> disks[i].centerx
        >> disks[i].centery
        >> disks[i].radius;
}
```

Reading the data is fairly natural. You go over all the elements in the array and read data into it.

(Refer Slide Time: 4:30)



Checking intersections

```
bool checkAllPairs(disk disks[], int n){
  for(int i=0; i<n-1; i++)
    for(int j=i+1; j<n; j++)
      if(intersect(disks[i], disks[j])) return true;
  return false;
}

bool intersect(disk d1, disk d2){
  return pow(d1.centerx-d2.centerx,2) +
         pow(d1.centery-d2.centery,2)
         < pow(d1.radius+d2.radius, 2);
```

Then we are going to check intersections. So, check all pairs or maybe I could have called it check all intersections that might have been even nicer name. Anyway, so there are n discs which are passed as arguments and we go over all pairs and we discussed this last time, when we solved this problem earlier. That we have to have 2 indices, i and j, and i should go from 0 to n-2 really, and therefore, we have written i less than n-1. And j should go from i+1 to n-1, and for each such pair we need to check whether those pairs intersect? If they interest we return true. If the loops execute without finding an intersection, then the controls will reach this point over here and at that point we know that no intersection was found and so we came in return false. Now, this function contains really just one kind of an idea. So, it says, “Look, I want to check intersection between every pair.” So, check between every pair is what is happening over here. How exactly the checking for a single pair happens that is described in intersect function.

So, this comes over here and notice that the intersect function only talks about 2 discs. It does not have to know about the entire array. So, the point is that when we write each function, our viewpoint is somehow limited, and that helps us think about our job, think about whatever code is doing a little bit better. So, here our rule was that we are going to look at the distance between the centers, and so this this whole thing is the square of the distance between the two centers.

And this should be smaller than the square of the sum of the radii. So, that is what this check that is the check that is happening over here and the result is being returned.

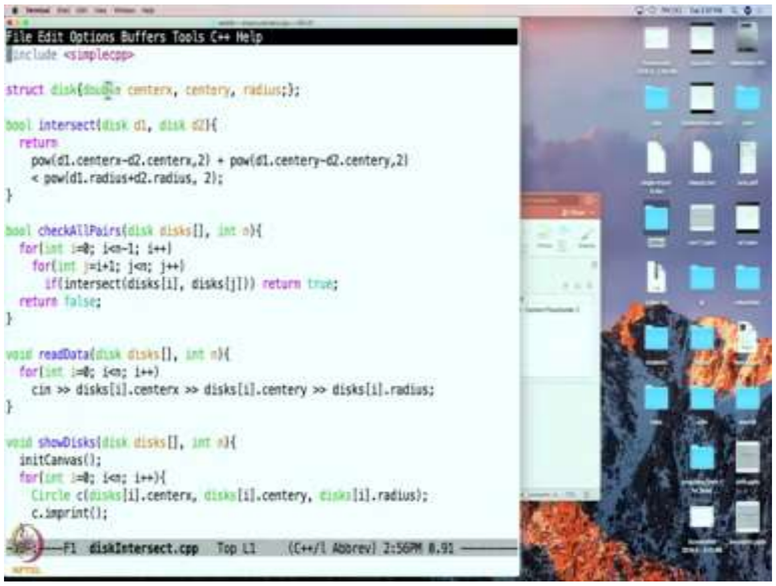
(Refer Slide Time: 7:05)

Demo

- diskIntersect.cpp

So, let us do a quick demo of this.

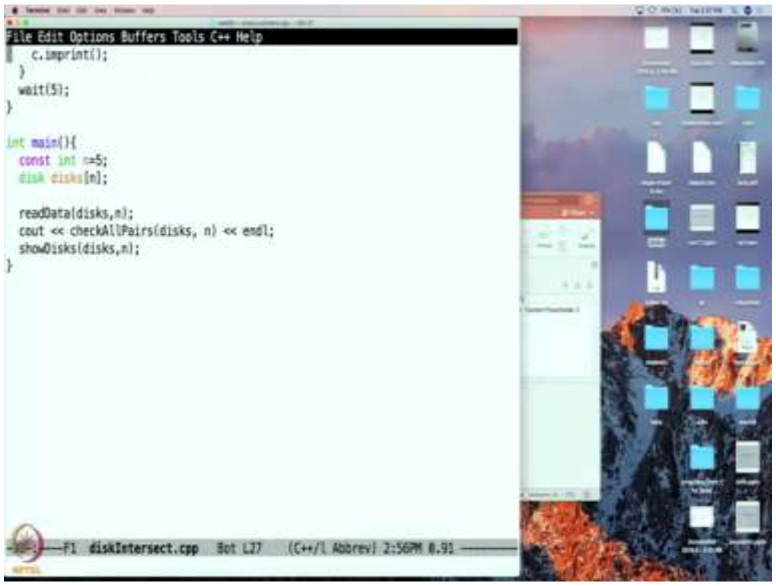
(Refer Slide Time: 7:12)



```
#include <simplecpp>

struct disk{double centerx, centery, radius;};

bool intersect(disk d1, disk d2){
  return
    pow(d1.centerx-d2.centerx,2) + pow(d1.centery-d2.centery,2)
    < pow(d1.radius+d2.radius, 2);
}

bool checkAllPairs(disk disks[], int n){
  for(int i=0; i<n-1; i++)
    for(int j=i+1; j<n; j++)
      if(intersect(disks[i], disks[j])) return true;
  return false;
}

void readData(disk disks[], int n){
  for(int i=0; i<n; i++)
    cin >> disks[i].centerx >> disks[i].centery >> disks[i].radius;
}

void showDisks(disk disks[], int n){
  initCanvas();
  for(int i=0; i<n; i++){
    Circle c(disks[i].centerx, disks[i].centery, disks[i].radius);
    c.imprint();
-:---F1  diskIntersect.cpp   Top L1    (C++/l Abbrev) 2:56PM 0.91 ------
```

So, here is my here is my program: the disk structure type, the intersect function, check all pairs function, read data. Now, I have also added over here a show discs functions. So, at the end it is going to show us the disc on our cameras. So, this place we will know whether our intersection answers is correct or not.

(Refer Slide Time: 7:36)

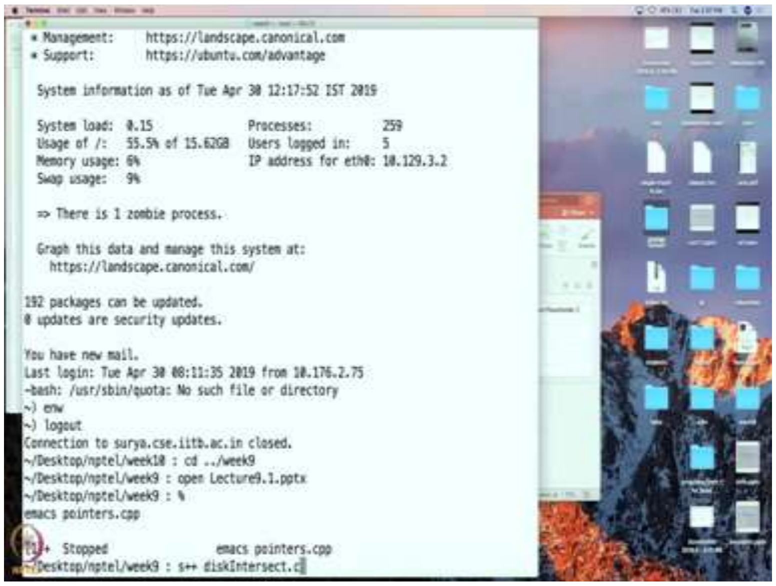


```
    c.imprint();
  }
  wait(5);
}

int main(){
  const int n=5;
  disk disks[n];

  readData(disks,n);
  cout << checkAllPairs(disks, n) << endl;
  showDisks(disks,n);
}
```

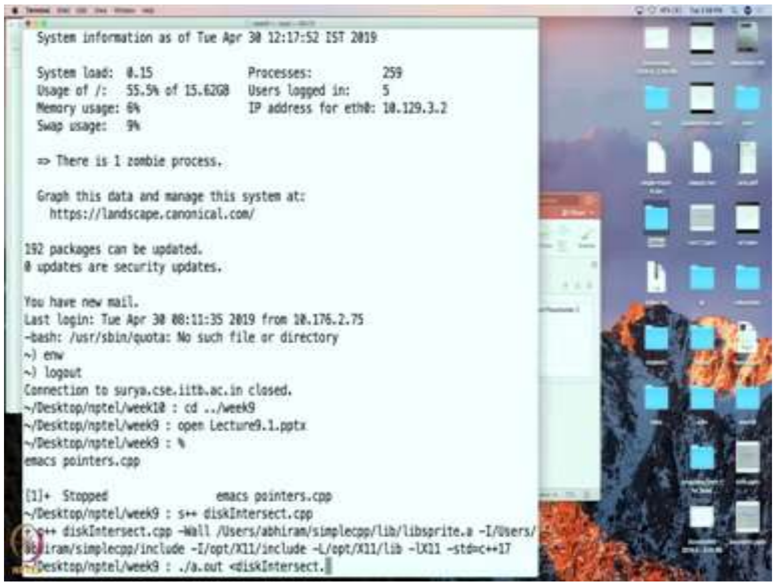
```
-F1  diskIntersect.cpp   Bot L27    (C++/l Abbrev) 2:56PM 0.91
```

Following this we have the main program. So, really what we had on the slide plus this additional show function, so in the main program as well we are going to print the answer whether there is an intersection or not and then we are going to show whether the disc are intersecting or not. Let me just shift a little bit so that you can see this character.

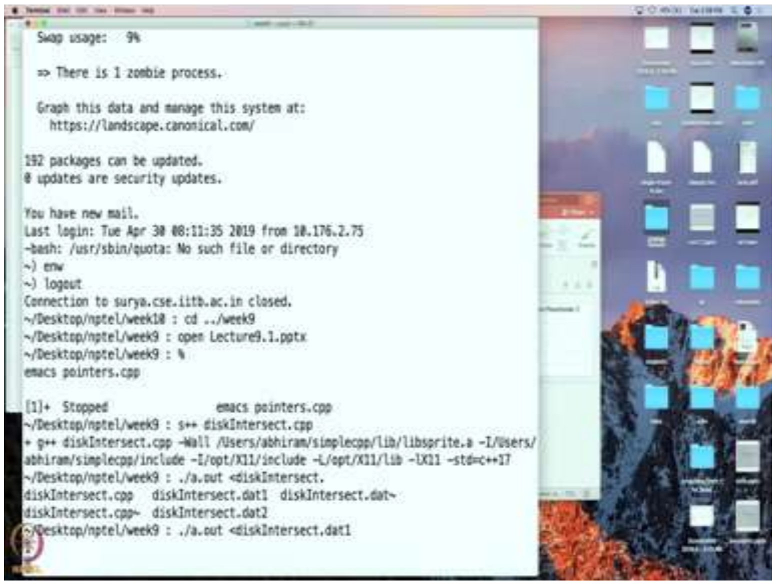
(Refer Slide Time: 8:02)



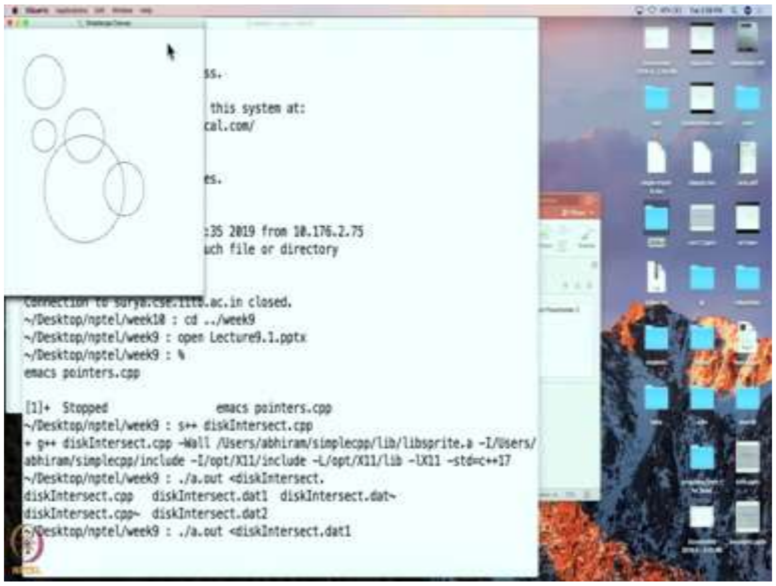
```
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Tue Apr 30 12:17:52 IST 2019

  System load:  0.15              Processes:           259
  Usage of /:   55.5% of 15.62GB  Users logged in:     5
  Memory usage: 6%                IP address for eth0: 10.129.3.2
  Swap usage:   9%

  => There is 1 zombie process.

  Graph this data and manage this system at:
    https://landscape.canonical.com/

192 packages can be updated.
0 updates are security updates.

You have new mail.
Last login: Tue Apr 30 08:11:35 2019 from 10.176.2.75
-bash: /usr/sbin/quota: No such file or directory
~) enw
~) logout
Connection to surya.cse.iitb.ac.in closed.
~/Desktop/nptel/week10 : cd ../week9
~/Desktop/nptel/week9 : open Lecture9.1.pptx
~/Desktop/nptel/week9 : %
emacs pointers.cpp

[1]+  Stopped                 emacs pointers.cpp
~/Desktop/nptel/week9 : s++ diskIntersect.c
```



```
  System information as of Tue Apr 30 12:17:52 IST 2019

  System load:  0.15              Processes:           259
  Usage of /:   55.5% of 15.62GB  Users logged in:     5
  Memory usage: 6%                IP address for eth0: 10.129.3.2
  Swap usage:   9%

  => There is 1 zombie process.

  Graph this data and manage this system at:
    https://landscape.canonical.com/

192 packages can be updated.
0 updates are security updates.

You have new mail.
Last login: Tue Apr 30 08:11:35 2019 from 10.176.2.75
-bash: /usr/sbin/quota: No such file or directory
~) enw
~) logout
Connection to surya.cse.iitb.ac.in closed.
~/Desktop/nptel/week10 : cd ../week9
~/Desktop/nptel/week9 : open Lecture9.1.pptx
~/Desktop/nptel/week9 : %
emacs pointers.cpp

[1]+  Stopped                 emacs pointers.cpp
~/Desktop/nptel/week9 : s++ diskIntersect.cpp
+ g++ diskIntersect.cpp -Wall /Users/abhiram/simplecpp/lib/libsprite.a -I/Users/
abhiram/simplecpp/include -I/opt/X11/include -L/opt/X11/lib -lX11 -std=c++17
~/Desktop/nptel/week9 : ./a.out <diskIntersect.
```



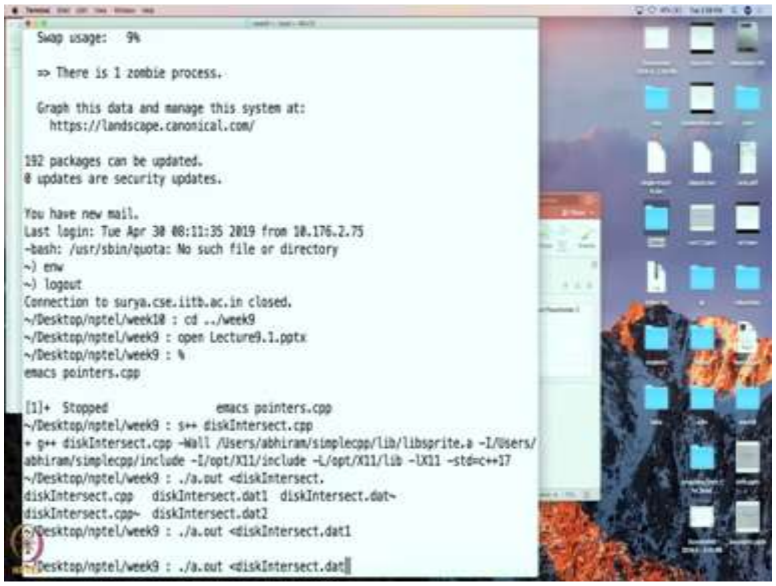
```
  Swap usage:   9%

  => There is 1 zombie process.

  Graph this data and manage this system at:
    https://landscape.canonical.com/

192 packages can be updated.
0 updates are security updates.

You have new mail.
Last login: Tue Apr 30 08:11:35 2019 from 10.176.2.75
-bash: /usr/sbin/quota: No such file or directory
~) emw
~) logout
Connection to surya.cse.iitb.ac.in closed.
~/Desktop/nptel/week10 : cd ../week9
~/Desktop/nptel/week9 : open Lecture9.1.pptx
~/Desktop/nptel/week9 : %
emacs pointers.cpp

[1]+  Stopped                 emacs pointers.cpp
~/Desktop/nptel/week9 : s++ diskIntersect.cpp
+ g++ diskIntersect.cpp -Wall /Users/abhiram/simplecpp/lib/libsprite.a -I/Users/
abhiram/simplecpp/include -I/opt/X11/include -L/opt/X11/lib -lX11 -std=c++17
~/Desktop/nptel/week9 : ./a.out <diskIntersect.
diskIntersect.cpp   diskIntersect.dat1  diskIntersect.dat~
diskIntersect.cpp~  diskIntersect.dat2
~/Desktop/nptel/week9 : ./a.out <diskIntersect.dat1
```



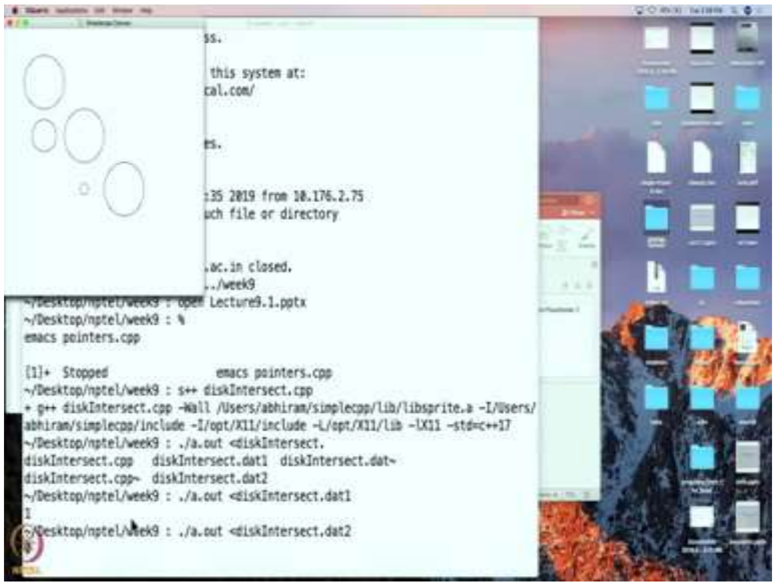
```
Connection to surya.cse.iitb.ac.in closed.
~/Desktop/nptel/week10 : cd ../week9
~/Desktop/nptel/week9 : open Lecture9.1.pptx
~/Desktop/nptel/week9 : %
emacs pointers.cpp

[1]+  Stopped                 emacs pointers.cpp
~/Desktop/nptel/week9 : s++ diskIntersect.cpp
+ g++ diskIntersect.cpp -Wall /Users/abhiram/simplecpp/lib/libsprite.a -I/Users/
abhiram/simplecpp/include -I/opt/X11/include -L/opt/X11/lib -lX11 -std=c++17
~/Desktop/nptel/week9 : ./a.out <diskIntersect.
diskIntersect.cpp   diskIntersect.dat1  diskIntersect.dat~
diskIntersect.cpp~  diskIntersect.dat2
~/Desktop/nptel/week9 : ./a.out <diskIntersect.dat1
```

So let us compile this and we're going to run it and we're going to run it by redirecting the file. So, .dat1 is one file. So, the answer is 1 and here is the plot. So, here are the circles so clearly there is intersection and therefore the answer was 1.

(Refer Slide Time: 8:46)



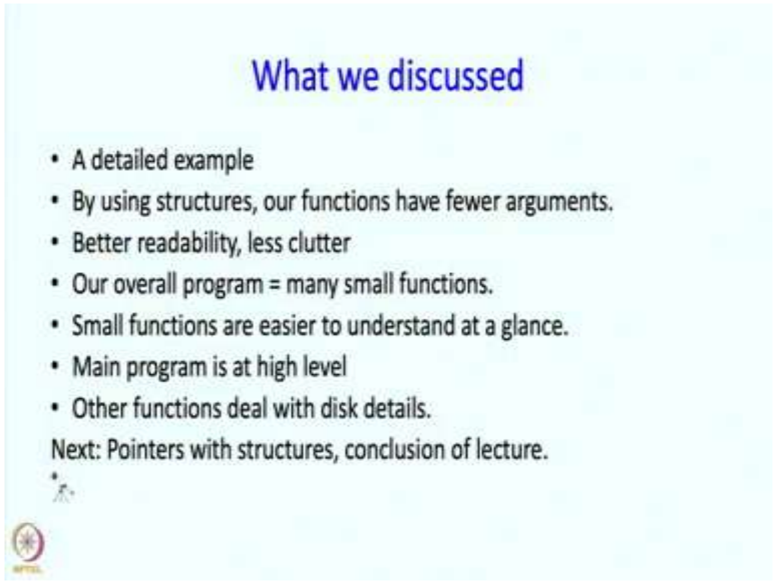
```
  Swap usage:   9%

  => There is 1 zombie process.

  Graph this data and manage this system at:
    https://landscape.canonical.com/

192 packages can be updated.
0 updates are security updates.

You have new mail.
Last login: Tue Apr 30 08:11:35 2019 from 10.176.2.75
-bash: /usr/sbin/quota: No such file or directory
~) emw
~) logout
Connection to surya.cse.iitb.ac.in closed.
~/Desktop/nptel/week10 : cd ../week9
~/Desktop/nptel/week9 : open Lecture9.1.pptx
~/Desktop/nptel/week9 : %
emacs pointers.cpp

[1]+  Stopped                 emacs pointers.cpp
~/Desktop/nptel/week9 : s++ diskIntersect.cpp
+ g++ diskIntersect.cpp -Wall /Users/abhiram/simplecpp/lib/libsprite.a -I/Users/
abhiram/simplecpp/include -I/opt/X11/include -L/opt/X11/lib -lX11 -std=c++17
~/Desktop/nptel/week9 : ./a.out <diskIntersect.
diskIntersect.cpp   diskIntersect.dat1  diskIntersect.dat~
diskIntersect.cpp~  diskIntersect.dat2
~/Desktop/nptel/week9 : ./a.out <diskIntersect.dat1

~/Desktop/nptel/week9 : ./a.out <diskIntersect.dat
```



```
ss.

 this system at:
cal.com/


es.


:35 2019 from 10.176.2.75
uch file or directory


.ac.in closed.
../week9
 Lecture9.1.pptx
~/Desktop/nptel/week9 : %
emacs pointers.cpp

[1]+  Stopped                 emacs pointers.cpp
~/Desktop/nptel/week9 : s++ diskIntersect.cpp
+ g++ diskIntersect.cpp -Wall /Users/abhiram/simplecpp/lib/libsprite.a -I/Users/
abhiram/simplecpp/include -I/opt/X11/include -L/opt/X11/lib -lX11 -std=c++17
~/Desktop/nptel/week9 : ./a.out <diskIntersect.
diskIntersect.cpp   diskIntersect.dat1  diskIntersect.dat~
diskIntersect.cpp~  diskIntersect.dat2
~/Desktop/nptel/week9 : ./a.out <diskIntersect.dat1
1
~/Desktop/nptel/week9 : ./a.out <diskIntersect.dat2
```

So, let me do one more. So we do it for 2 and this time there is no intersection and in fact the answer is 0.

(Refer Slide Time: 8:58)



What we discussed

- A detailed example
- By using structures, our functions have fewer arguments.
- Better readability, less clutter
- Our overall program = many small functions.
- Small functions are easier to understand at a glance.
- Main program is at high level
- Other functions deal with disk details.

Next: Pointers with structures, conclusion of lecture.

So what have we discussed in this segment? So we discussed a detailed example and here are some important points that you should observe. So, by using structures, our functions could be written with fewer arguments and this makes for better readability, less clutter and, of course, less possibility of errors as well. We will not forget giving a function or we will not exchange two arguments. So, such errors are sort of very-very easily prevented.

Our overall program can be now more easily written as a bunch of small functions. If we had lots of arguments to be supplied, we sort have said, “Oh my God, I am tired of suppling all these arguments”, and so you might have tried not to write small functions. But writing small functions is a good idea because in a small function you can just one idea, and therefore, you can understand that function and make sure that it is correct a lot more easily.

Then we also said that because we break things up into functions, each function has its own limited use. So, for example, the main program has a very high-level view, the other functions may have views at their own levels. So, in the next segment we are going to talk about pointers with structures and we will also conclude this entire lecture series.