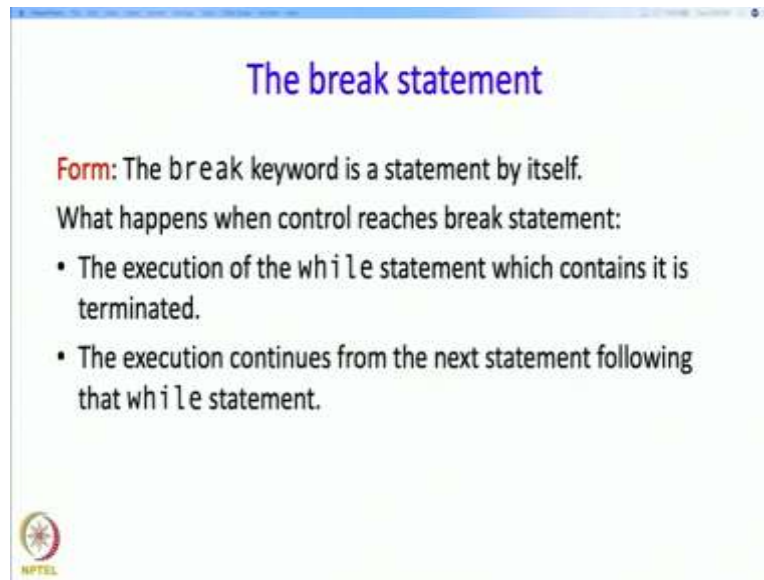


Introduction to Programming through C++
Professor Abhiram G. Ranade
Department of Computer Science and Engineering
Indian Institute of Technology Bombay
Lecture No. 7 Part - 3
Looping Statements
The break and continue statement

(Refer Slide Time: 0:17)



In the previous segment, we discussed how we could modify the arrangement of code in our program or in our flowchart so that it matches the structure of the while loop, so that we can use the while loop in our program. Next, we will discuss a different way of doing the same thing. But without too much rearrangement of any code. For this we need the so called break statement, the break statement is perhaps the simplest statement. The word, break is a keyword, which by itself is a statement okay, so you will see an example soon. What happens when control reaches a break statement? Well, the break statement had better be inside a while and if you reach the break statement the execution of the while statement, which contains it is terminated. So the control then goes to the statement following the containing while statement, that is about it

(Refer Slide Time: 1:21)

Example of break

```
main_program{
float nextmark, sum = 0;
int count = 0;
while(true){
cin >> nextmark;
if(nextmark < 0) break;
sum += nextmark;
count++;
}
cout << sum/count << endl;
}
```

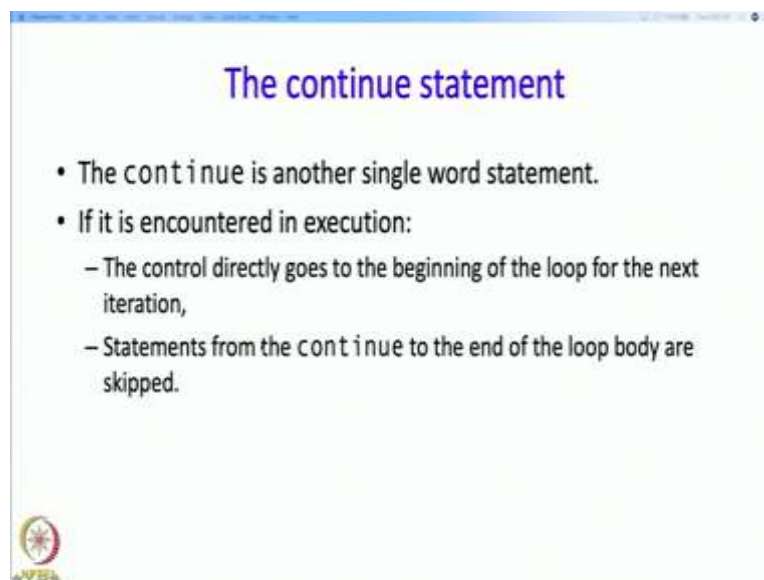
- The condition of the while statement is given as true – body will always be entered.
- If nextmark < 0:
 - the while loop execution will terminate
 - Execution continues from the statement after while, i.e. cout ...
- Exactly what we wanted!
 - No need to copy code.
- Some programmers do not like break statements because continuation condition gets hidden inside body, instead of being at the top.
- Condition for breaking = compliment of condition for continuing loop

So here is an example of a break. Okay, so main program, float nextmark, sum as before, count is 0 and now we are going to run the loop, but we are going to run it with a condition true, so I will explain this in a minute, so we are going to read in the mark and if the nextmark is less than 0, then we are going to break okay. So the consequent over here is the single word statement break, otherwise we are going to add nextmark to sum, we are going to increment count, and that is it as far as loop is concerned and that we are going to print. So I will claim that is program actually does what we want. So why is that? So first of all, let us go to the new part, the first new part is the while which contains true, so the condition of the while statement is given as true, which means that this condition will always evaluate to true, so which means whenever you try and check the condition, it will come out to be true and you will always enter the body, so you will always enter the body, so you will always read in the nextmark. The next element is, if nextmark less than 0. So if nextmark less than 0, then there is the break statement and which means that the loop will terminate, the execution of the while statement will terminate and you will execute, you will come to this point and execute the statement. So remember that was exactly what you wanted, if nextmark was less than 0. nextmark was signalling the condition that no more input will be given and therefore, you want that is to be calculate it and printed, otherwise you want the currently read mark to be treated as a real mark and you wanted added to sum, so that is what is going to happen over here any also want count to be incremented.

So the nice part over here is that we did not need to copy this code, remember in the earlier case we have to pull it out over here, we had to move it to the end, here we do not need to do

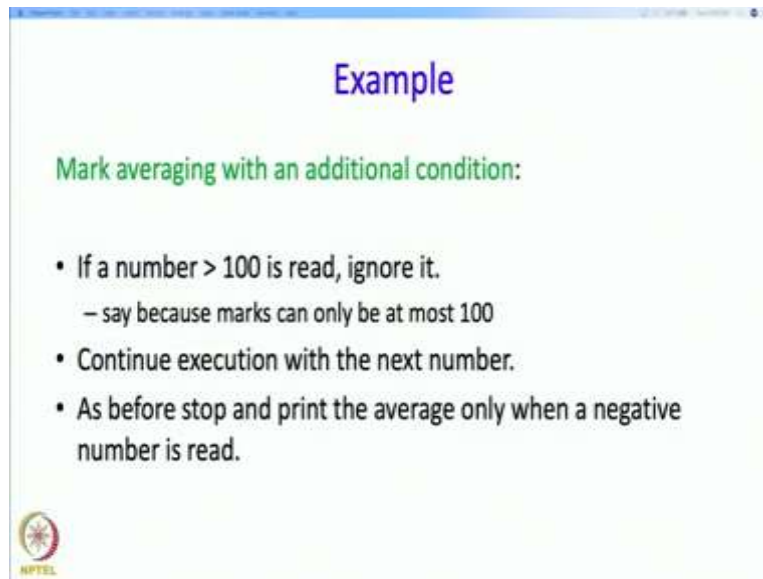
anything like that and this pretty much resembles the way we think about this logic. So when we wrote down, then we thought about it manually, you would see okay, read the next mark if it is less than 0, then go off and terminate, otherwise I read to sum and so on, so this is exactly the way we think about it. However, some programmers do not like break statements because the continuation condition. So this condition, this is really the condition, which tells whether or not you want to continue the loop or break the loop, this continuous condition now is hidden inside the body, whereas earlier, it was at a nice obvious place. So therefore, some programmers have preference for not having a break statement, but using a proper continuation condition over here and at the cost of maybe moving some code outside and moving it end of the loop body. So I think programmers do both of these things, so you should pick the style that you are most comfortable with and then I should note that the condition for breaking, is `nextmark < 0`, whereas the condition for continuing however, is `nextmark >= 0`, so while `nextmark >= 0`, I should execute this. So they of course complimentary conditions, but clearly one is the condition for continuing and the other is the condition for breaking out, so we should expect them to be exactly the opposite of each other.

(Refer Slide Time: 5:33)



So there is another single word statement that I should tell you about and this is the continue statement. Okay, so if it is encountered in any execution, then the control directly goes to the beginning of the loop for the next iteration, so basically the statement from the continue to the end of the loop body are skipped.

(Refer Slide Time: 6:00)



Example

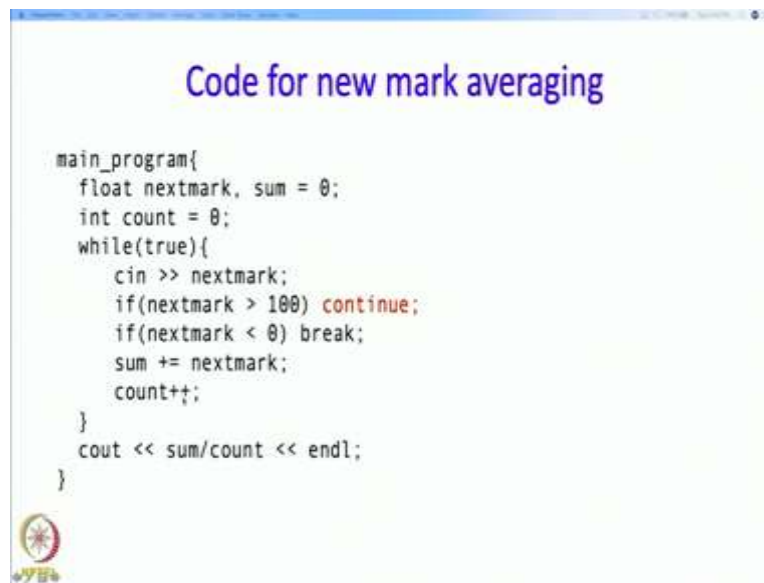
Mark averaging with an additional condition:

- If a number > 100 is read, ignore it.
 - say because marks can only be at most 100
- Continue execution with the next number.
- As before stop and print the average only when a negative number is read.

NPTEL

Okay, now you might see why you might need such a statement, so here is an example. So we look at the same mark averaging problem, but now we have an additional condition, so what is this condition? So suppose, somebody types a number bigger than 100, than we are going to see that look that must have been a mistake, so if it is a mistake what should we do? We should just ignore it or in other words, we should just not execute the statement that comes after this reading statement. And therefore, the continue statement will come in handy. Okay and as before we want to stop and print average only when a negative number is read and so the other parts will remain the same.

(Refer Slide Time: 6:42)

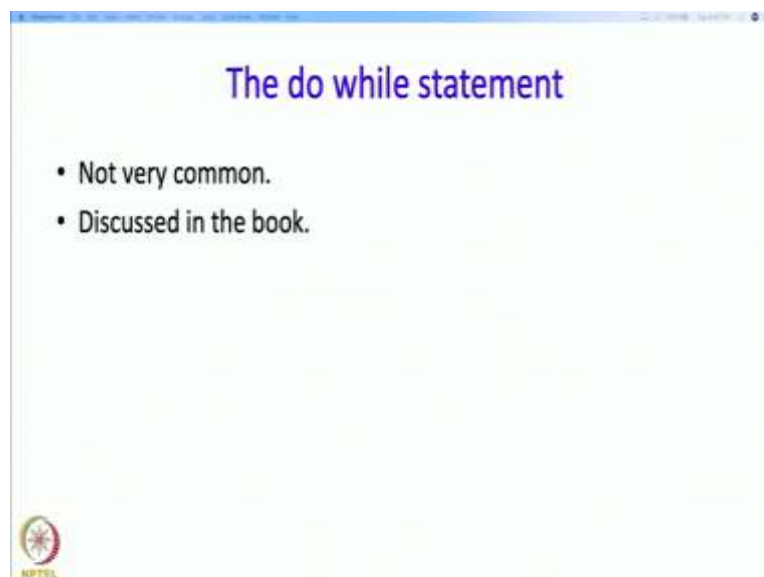


```
Code for new mark averaging

main_program{
  float nextmark, sum = 0;
  int count = 0;
  while(true){
    cin >> nextmark;
    if(nextmark > 100) continue;
    if(nextmark < 0) break;
    sum += nextmark;
    count++;
  }
  cout << sum/count << endl;
}
```

So here is the new program with this new additional requirement, so the main difference is this statement if nextmark is bigger than 100 we continue which means we will skip this entire part and we will start again. Okay, so essentially, this last mark that you read is going to be ignored, otherwise if it is less than or equal to 100, then it is a valid mark and so we do everything that we were doing earlier.

(Refer Slide Time: 7:13)



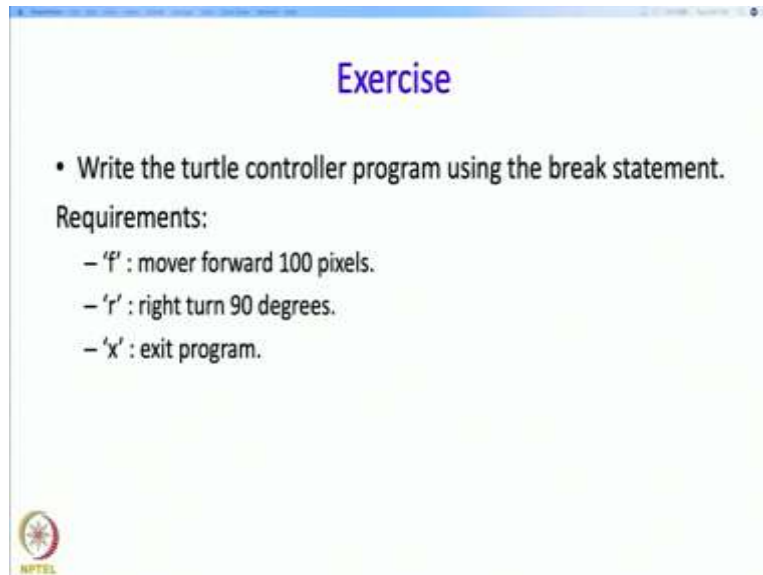
The do while statement

- Not very common.
- Discussed in the book.

Now, I should probably talk about one more statement, the do while statement, but I am not going to do so because it is not very common. And it has been discussed in the book, so I will leave it for you to read, but I am not really going to ask any problems regarding this

statement, and you do not really need to know it, to solve any of the problems that we are going to discuss in this class, in this course.

(Refer Slide Time: 7:46)




Exercise

- Write the turtle controller program using the break statement.

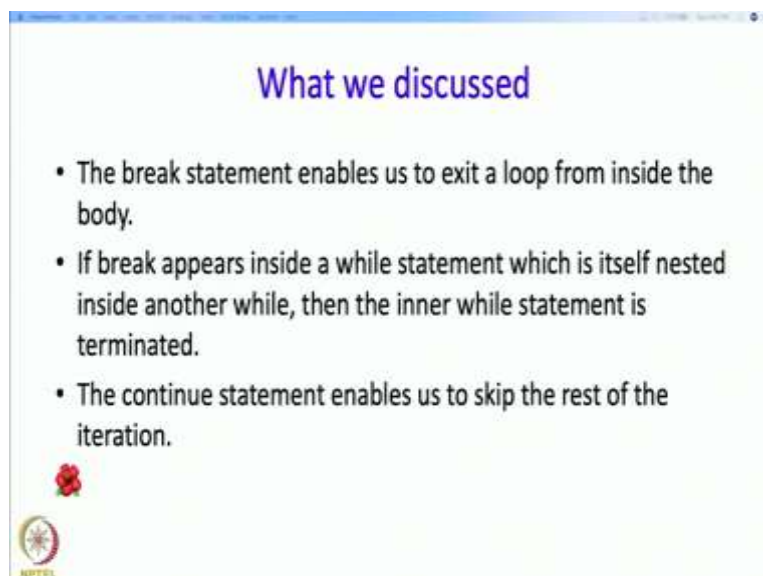
Requirements:

- 'f' : mover forward 100 pixels.
- 'r' : right turn 90 degrees.
- 'x' : exit program.





So you can use the break statement to advantage in writing that turtle controller program as well, so instead of replicating you can use the break statement and so that is a good exercise just to check whether you have understood the break statement.

(Refer Slide Time: 8:01)



What we discussed

- The break statement enables us to exit a loop from inside the body.
- If break appears inside a while statement which is itself nested inside another while, then the inner while statement is terminated.
- The continue statement enables us to skip the rest of the iteration.

Alright, so what have we discussed? So we have discussed the break statement which enables us to break or exit from a loop inside the body. Now a point to be noted is that if break appears inside a while statement, which itself nested inside another while, then the break only

breaks out of the inner while, not out of the outer while. And then you also discuss the continue statement which enables us to skip the rest of the iteration, so will take a break here and return later.