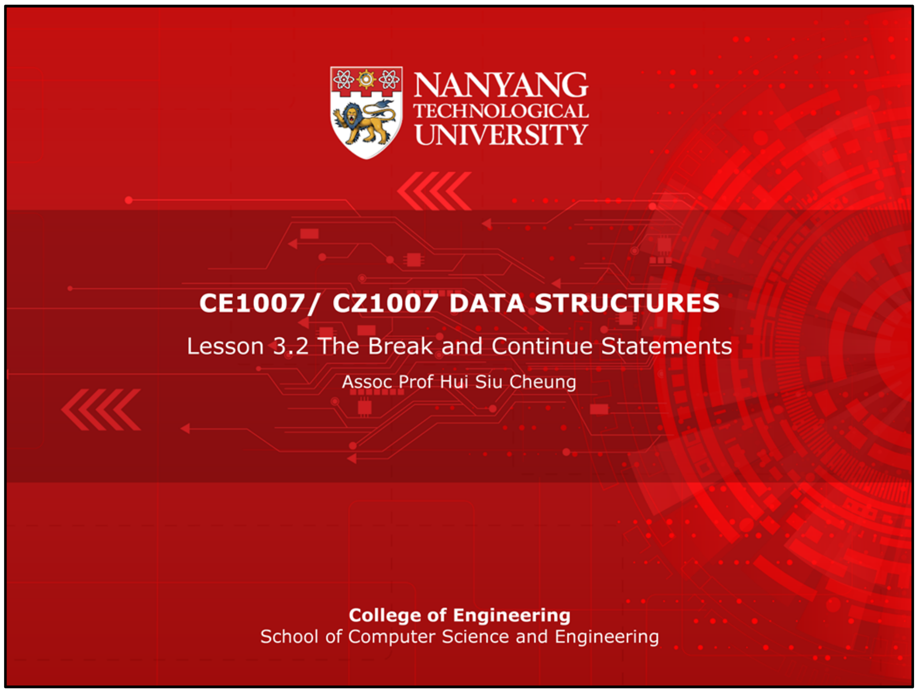


NANYANG TECHNOLOGICAL UNIVERSITY

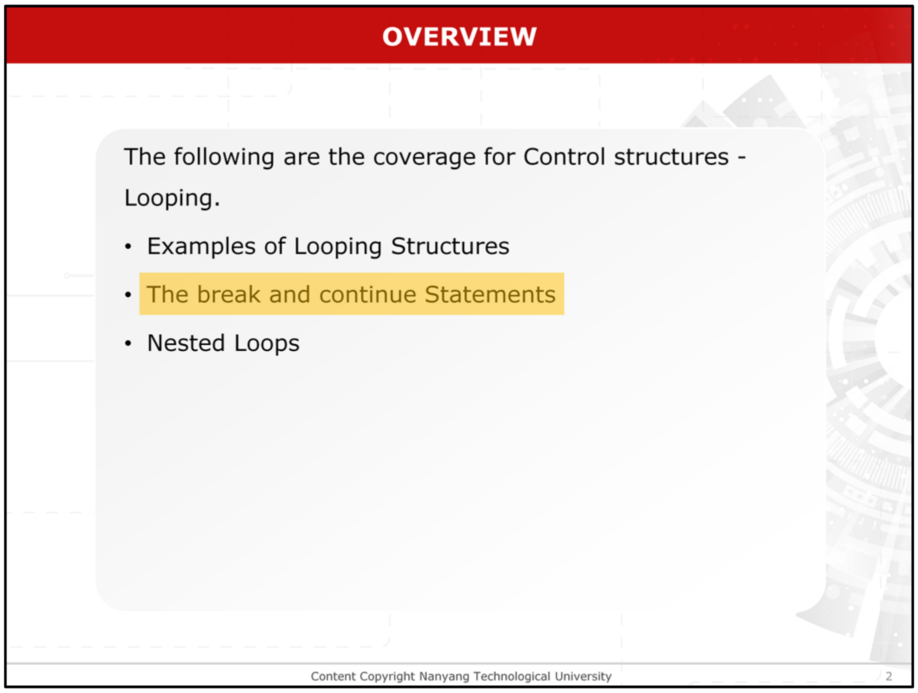
CE1007/ CZ1007 DATA STRUCTURES

Lesson 3.2 The Break and Continue Statements

Assoc Prof Hui Siu Cheung

College of Engineering
School of Computer Science and Engineering

This lesson is on Control structures (looping)



OVERVIEW

The following are the coverage for Control structures - Looping.

- Examples of Looping Structures
- The break and continue Statements
- Nested Loops

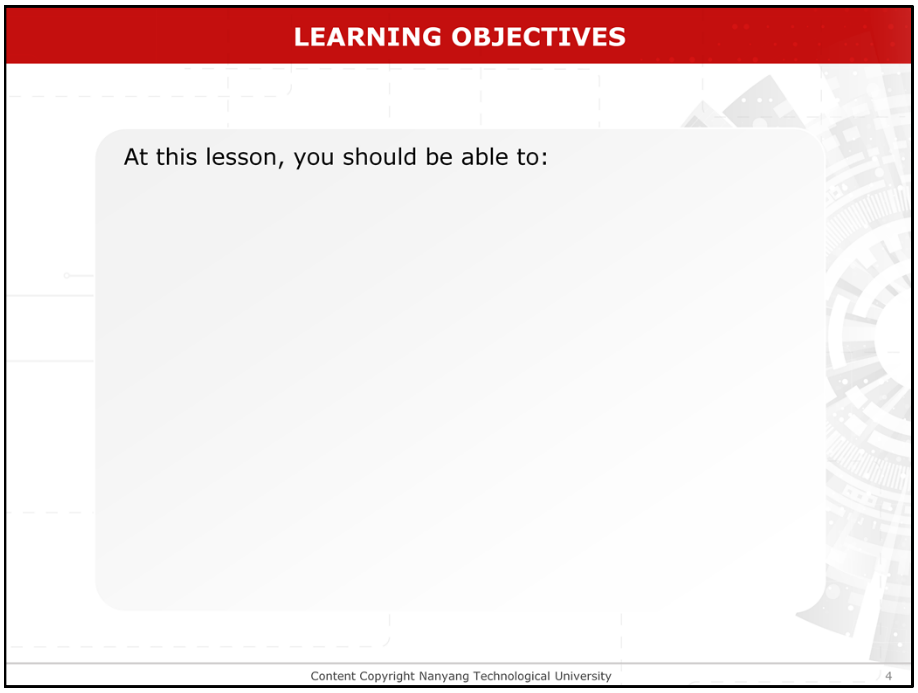
Basic C Programming

There are 3 main sections to cover for Control structures (looping).

LEARNING OBJECTIVES

3

Learning objectives

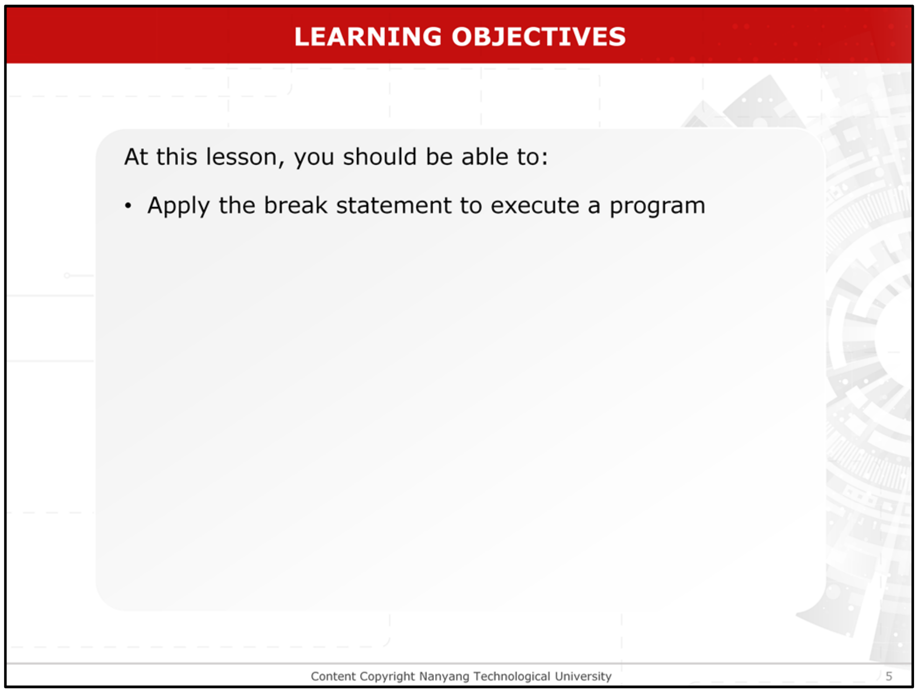


LEARNING OBJECTIVES

At this lesson, you should be able to:

4

At this lesson, you should be able to:



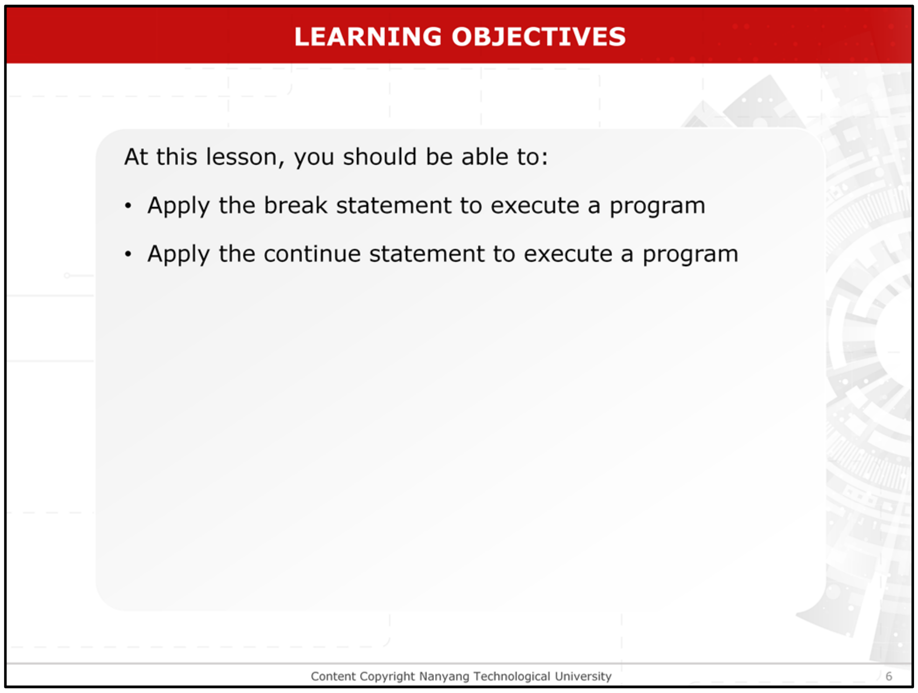
LEARNING OBJECTIVES

At this lesson, you should be able to:

- Apply the break statement to execute a program

5

Apply break statement to execute a program



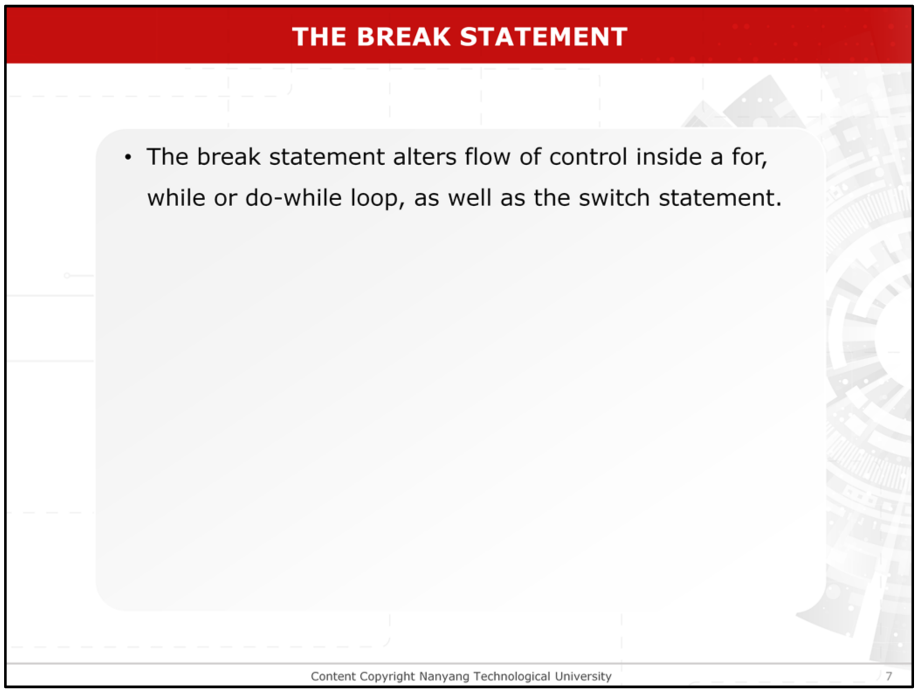
LEARNING OBJECTIVES

At this lesson, you should be able to:

- Apply the break statement to execute a program
- Apply the continue statement to execute a program

6

Apply the continue statement to execute a program

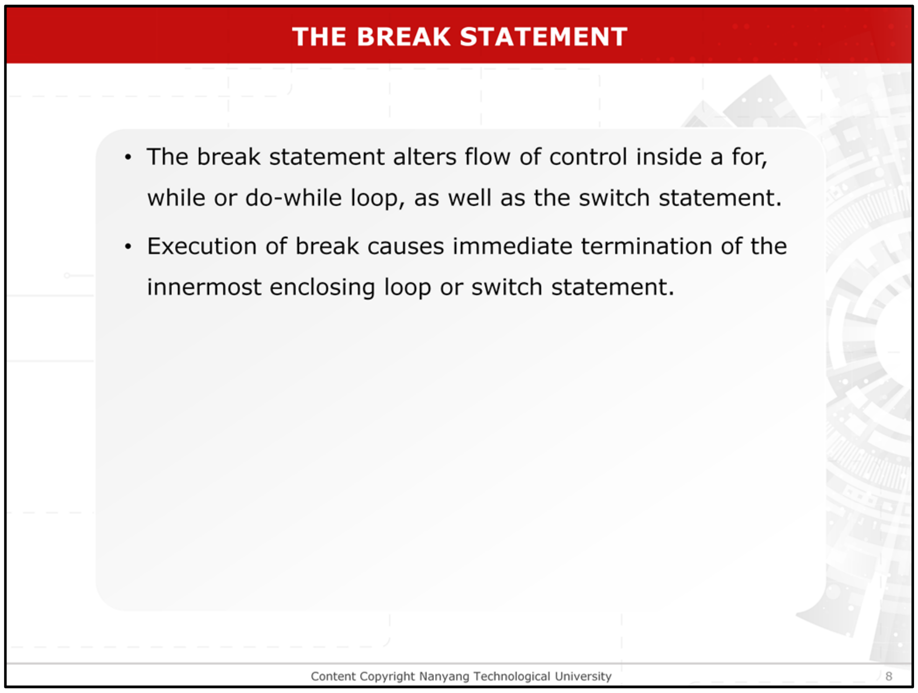


THE BREAK STATEMENT

- The break statement alters flow of control inside a for, while or do-while loop, as well as the switch statement.

7

The break statement alters flow of control inside a for, while or do-while loop, as well as the switch statement.

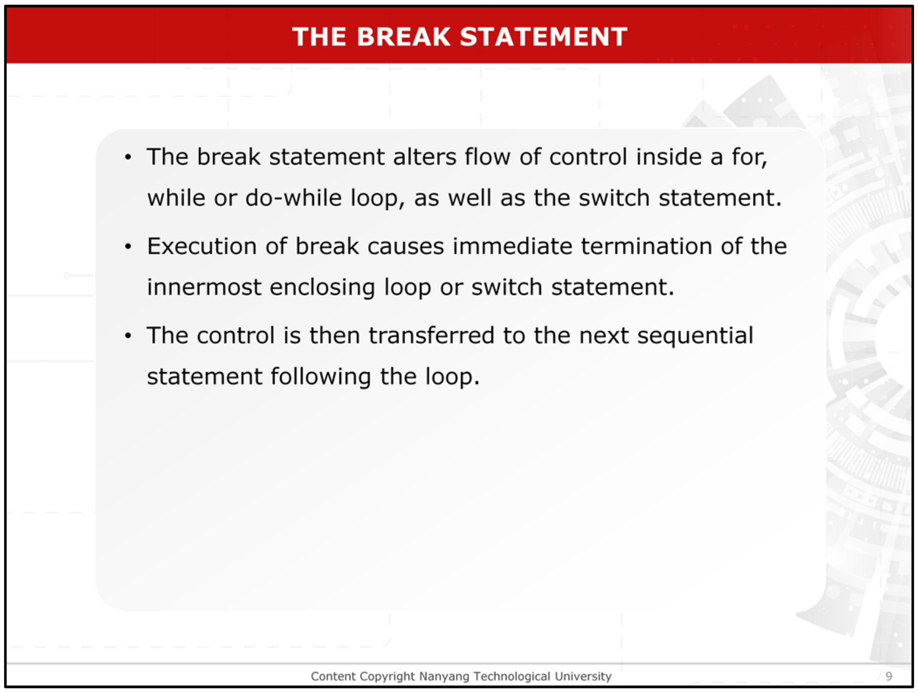


THE BREAK STATEMENT

- The break statement alters flow of control inside a for, while or do-while loop, as well as the switch statement.
- Execution of break causes immediate termination of the innermost enclosing loop or switch statement.

8

The execution of **break** causes immediate termination of the innermost enclosing loop or the **switch** statement.

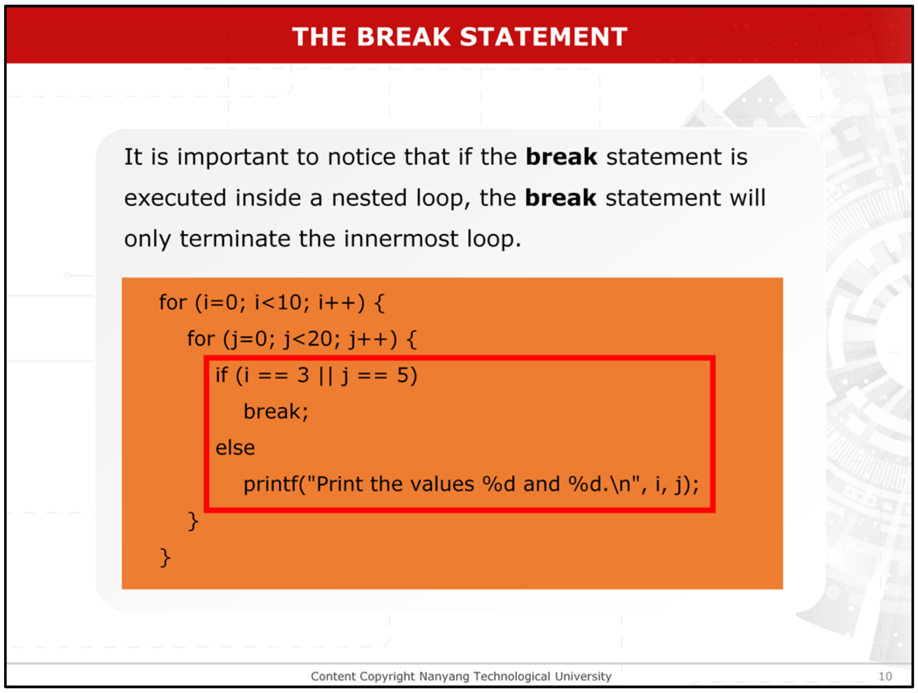


THE BREAK STATEMENT

- The break statement alters flow of control inside a for, while or do-while loop, as well as the switch statement.
- Execution of break causes immediate termination of the innermost enclosing loop or switch statement.
- The control is then transferred to the next sequential statement following the loop.

9

The control is then transferred to the next sequential statement following the loop.



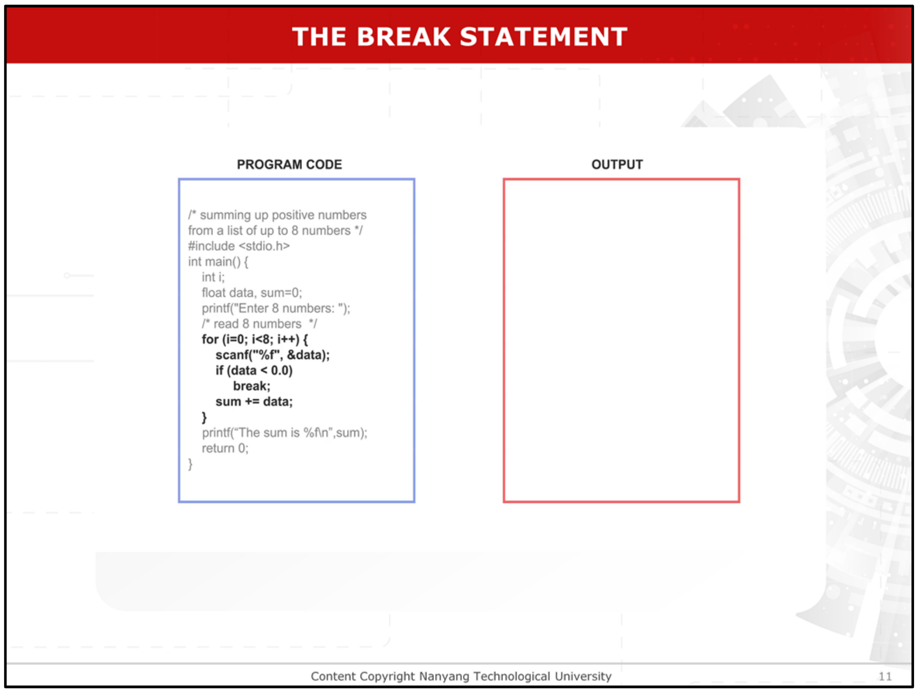
THE BREAK STATEMENT

It is important to notice that if the **break** statement is executed inside a nested loop, the **break** statement will only terminate the innermost loop.

```
for (i=0; i<10; i++) {
    for (j=0; j<20; j++) {
        if (i == 3 || j == 5)
            break;
        else
            printf("Print the values %d and %d.\n", i, j);
    }
}
```

10

It is important to notice that if the **break** statement is executed inside a nested loop, the **break** statement will only terminate the innermost loop.



THE BREAK STATEMENT

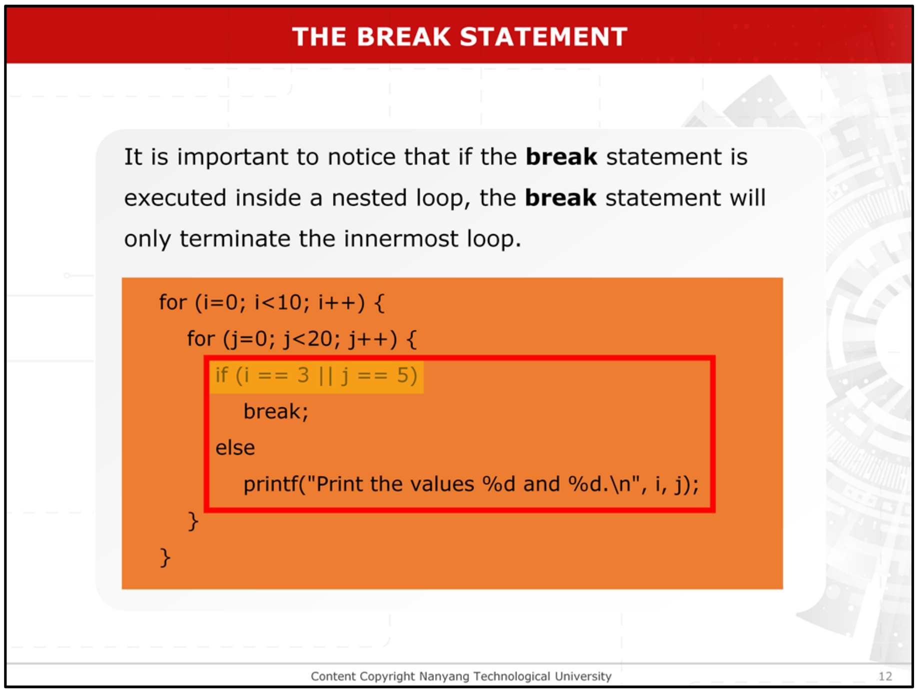
PROGRAM CODE

```
/* summing up positive numbers
from a list of up to 8 numbers */
#include <stdio.h>
int main() {
   int i;
   float data, sum=0;
   printf("Enter 8 numbers: ");
   /* read 8 numbers  */
   for (i=0; i<8; i++) {
      scanf("%f", &data);
      if (data < 0.0)
         break;
      sum += data;
   }
   printf("The sum is %f\n",sum);
   return 0;
}
```

OUTPUT

11

The **break** statement is usually used in a special situation where immediate termination of the loop is required. The program gives an example on using the **break** statement to sum up positive numbers from a list of numbers until a negative number is encountered. The program reads the input numbers of data type **float** for at most 8 numbers. A **for** loop is used to process the input number one by one. If the number is not less than zero, then the value is added to **sum.** Otherwise the **break** statement is used to terminate the loop. The control is then transferred to the next statement following the loop construct.



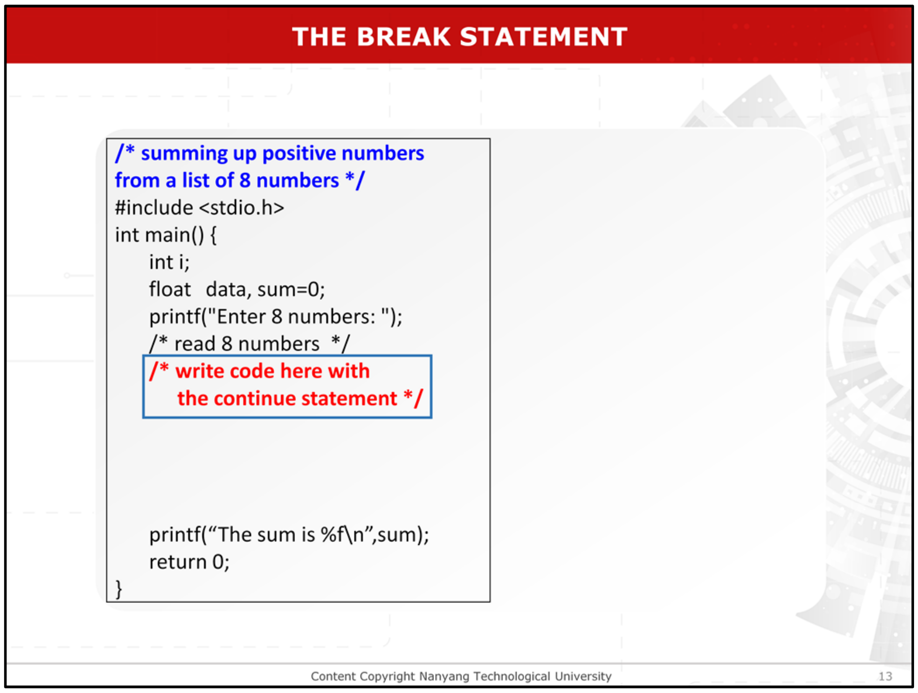
THE BREAK STATEMENT

It is important to notice that if the **break** statement is executed inside a nested loop, the **break** statement will only terminate the innermost loop.

```
for (i=0; i<10; i++) {
   for (j=0; j<20; j++) {
      if (i == 3 || j == 5)
         break;
      else
         printf("Print the values %d and %d.\n", i, j);
   }
}
```

12

In the statement shown, the **break** statement will only terminate the innermost loop of the **for** statement when **i** equals to 3 or **j** equals to 5. When this happens, the outer loop will carry on execution with **i** equals to 4, and the inner loop starts execution again.



THE BREAK STATEMENT

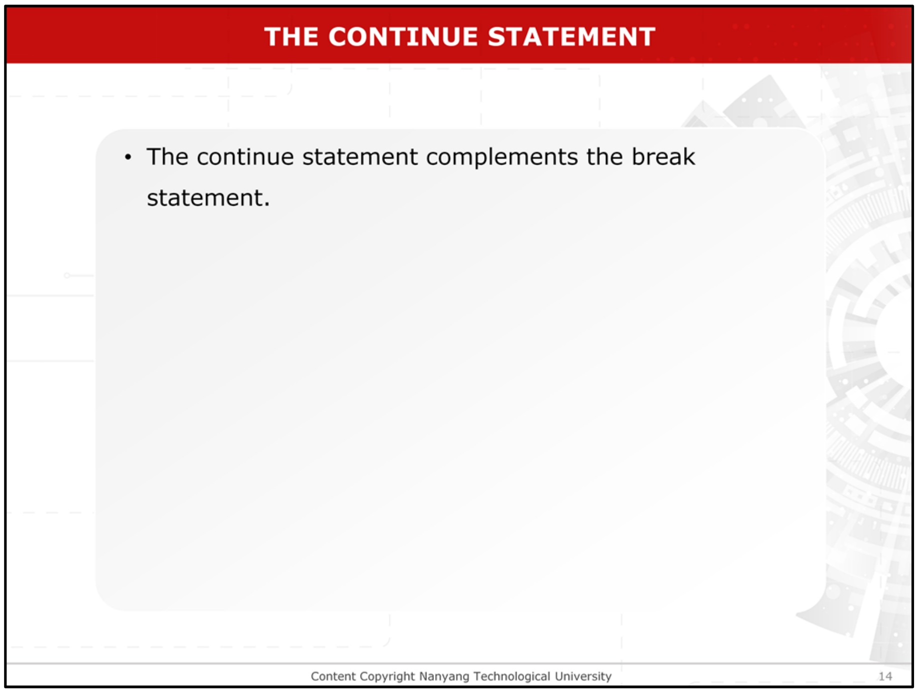
```
/* summing up positive numbers
from a list of 8 numbers */
#include <stdio.h>
int main() {
     int i;
     float   data, sum=0;
     printf("Enter 8 numbers: ");
     /* read 8 numbers  */
     /* write code here with
         the continue statement */




     printf("The sum is %f\n",sum);
     return 0;
}
```

13

In the example program, it aims to sum up only positive numbers from a list of 8 numbers, but terminate when a negative number is encountered.



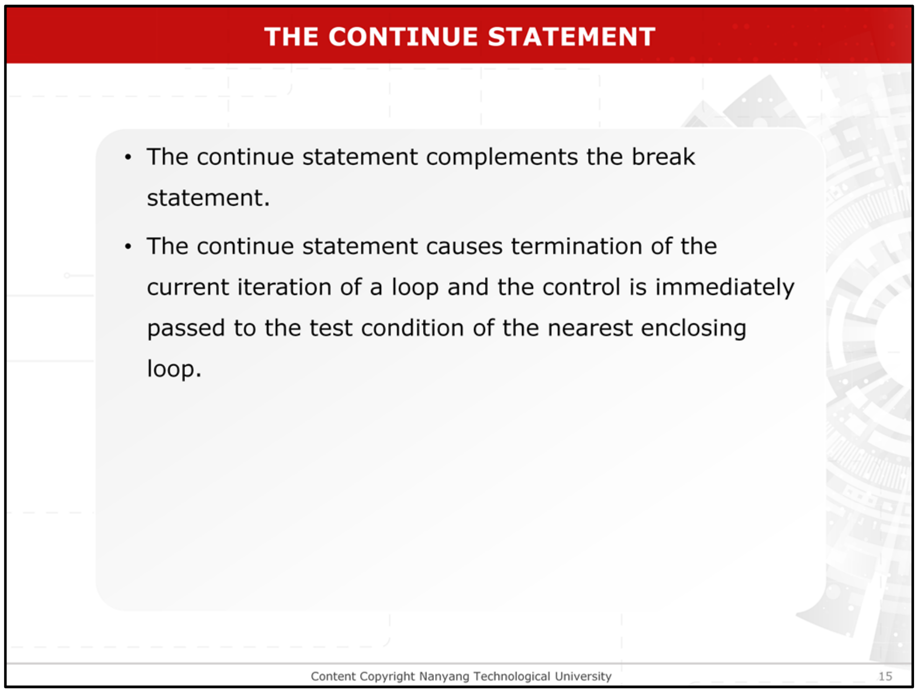
THE CONTINUE STATEMENT

- The continue statement complements the break statement.

14

The continue Statement

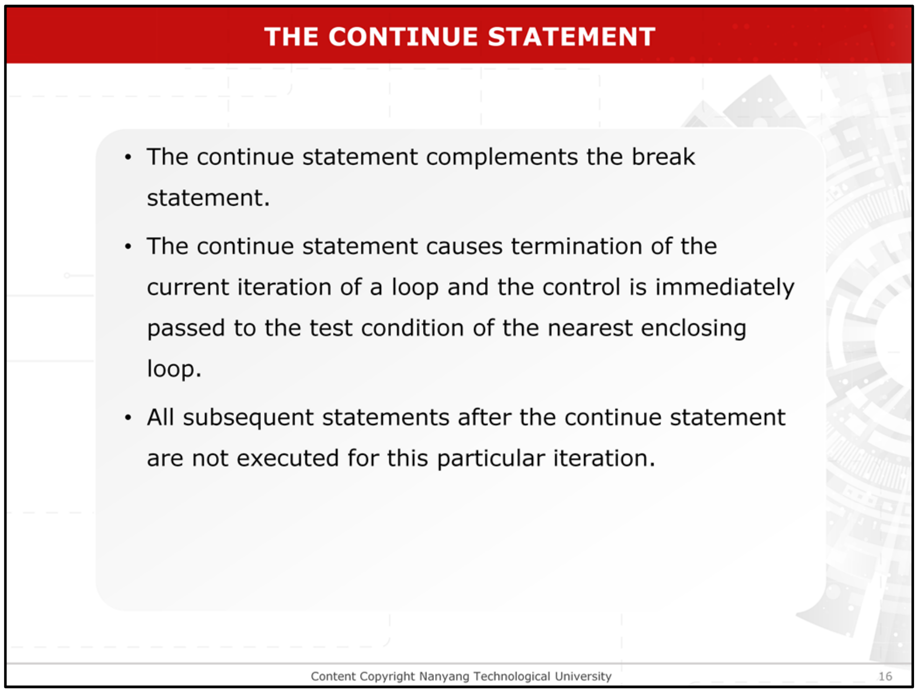
The **continue** statement complements the **break** statement



THE CONTINUE STATEMENT

- The continue statement complements the break statement.
- The continue statement causes termination of the current iteration of a loop and the control is immediately passed to the test condition of the nearest enclosing loop.

The **continue** statement causes termination of the current iteration of a loop and the control is immediately passed to the **test** condition of the nearest enclosing loop.

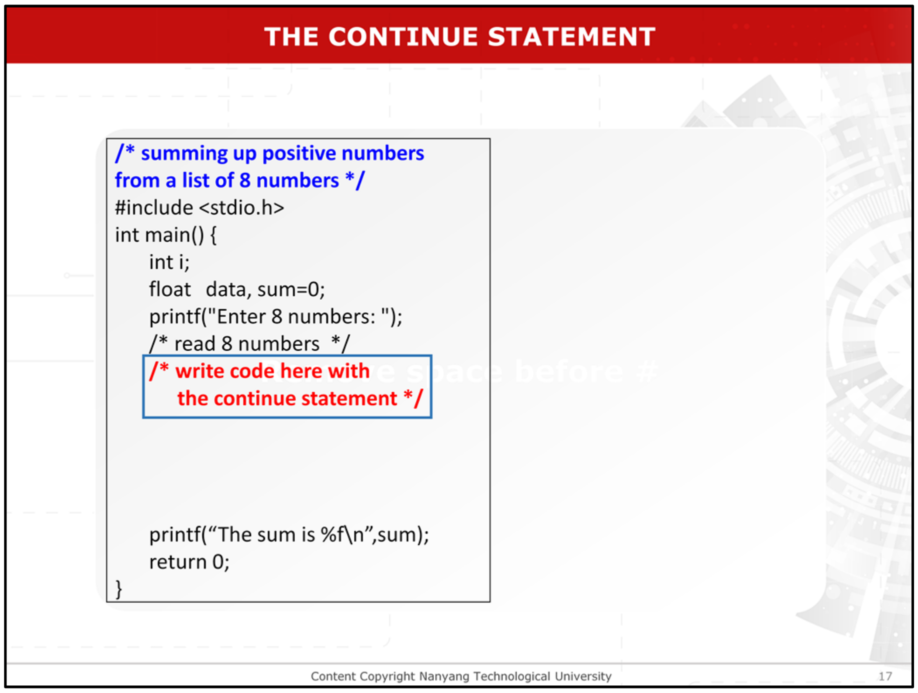


THE CONTINUE STATEMENT

- The continue statement complements the break statement.
- The continue statement causes termination of the current iteration of a loop and the control is immediately passed to the test condition of the nearest enclosing loop.
- All subsequent statements after the continue statement are not executed for this particular iteration.

16

All subsequent statements after the **continue** statement are not executed for this particular iteration.



THE CONTINUE STATEMENT

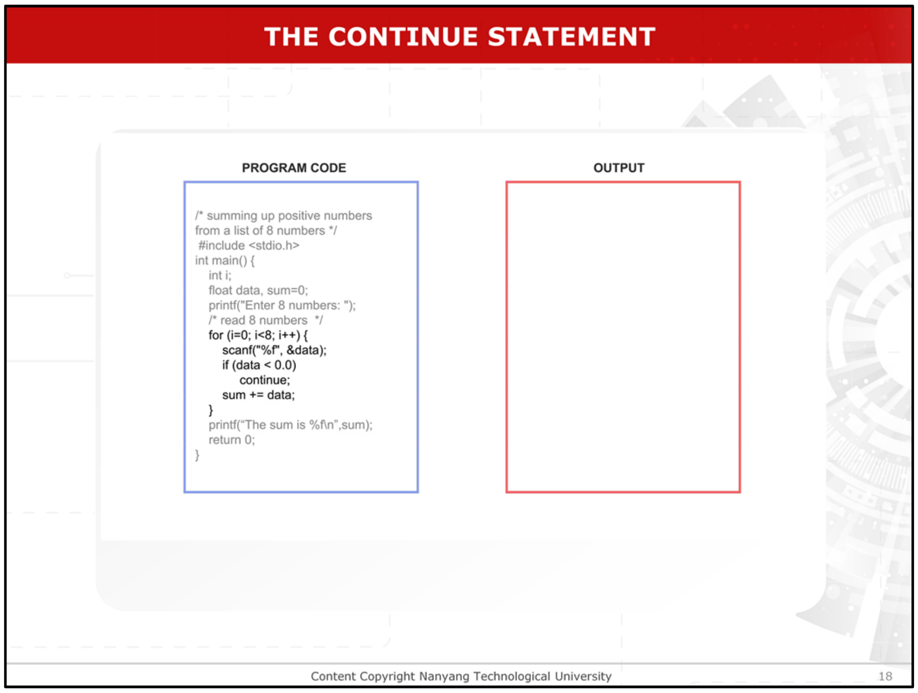
```
/* summing up positive numbers
from a list of 8 numbers */
#include <stdio.h>
int main() {
     int i;
     float   data, sum=0;
     printf("Enter 8 numbers: ");
     /* read 8 numbers  */
     /* write code here with
         the continue statement */



     printf("The sum is %f\n",sum);
     return 0;
}
```

17

The **continue** statement differs from the **break** statement in that the **continue** statement terminates the execution of the current iteration, and the loop still carries on with the next iteration if the **test** condition is fulfilled, while the **break** statement terminates the execution of the loop and passes the control to the next statement immediately after the loop. However, the use of the **break** statement and **continue** statement are generally not recommended for good programming practice, as both statements interrupt normal sequential execution of the program.



THE CONTINUE STATEMENT

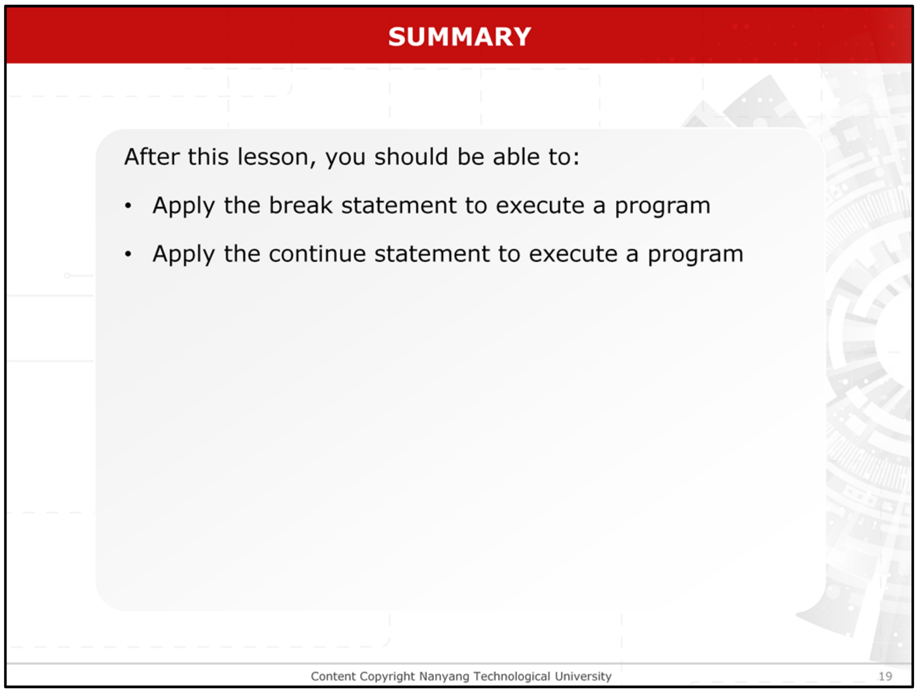
PROGRAM CODE

```
/* summing up positive numbers
from a list of 8 numbers */
 #include <stdio.h>
int main() {
   int i;
   float data, sum=0;
   printf("Enter 8 numbers: ");
   /* read 8 numbers  */
   for (i=0; i<8; i++) {
      scanf("%f", &data);
      if (data < 0.0)
          continue;
      sum += data;
   }
   printf("The sum is %f\n",sum);
   return 0;
}
```

OUTPUT

18

In the program, it uses the **continue** statement to help sum up a list of positive numbers. The program reads eight numbers of data type **float**. A **for** loop is used to process the input number one by one. If the number is not less than zero, then the value is added to **sum.** Otherwise the **continue** statement is used to terminate the current iteration of the loop, and the control is transferred to the next iteration of the loop. Notice that the **for** loop will process all the eight numbers when they are read in.



SUMMARY

After this lesson, you should be able to:

- Apply the break statement to execute a program
- Apply the continue statement to execute a program

19

In summary, after viewing this video lesson, your should be able to do the listed.