

Lecture 5: Control Statements

– Part III

Dr. Tsung-Wei Huang

Department of Electrical and Computer Engineering
University of Utah, Salt Lake City, UT

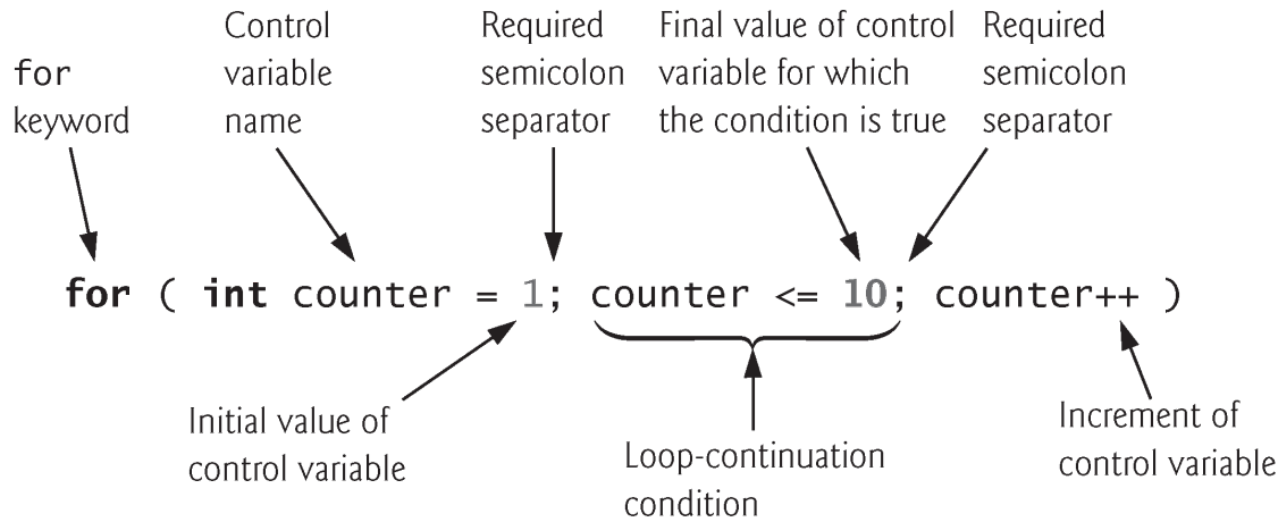


for Repetition Statement Example

```
1
2 // Counter-controlled repetition with the for statement.
3 #include <iostream>
4 using namespace std;
5
6 int main()
7 {
8     // for statement header includes initialization,
9     // loop-continuation condition and increment.
10    for ( int counter = 1; counter <= 10; counter++ )
11        cout << counter << " ";
12
13    cout << endl; // output a newline
14 } // end main
```

```
1 2 3 4 5 6 7 8 9 10
```

for Repetition Statement



do...while Repetition Statement

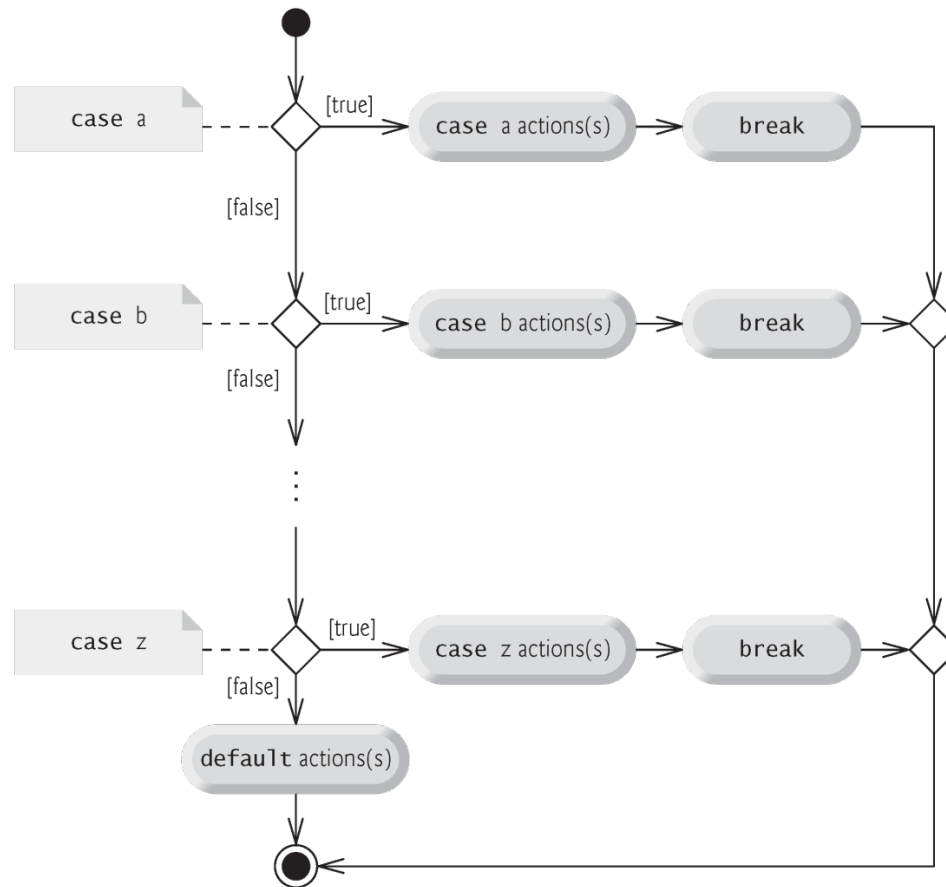
```
1
2 // do...while repetition statement.
3 #include <iostream>
4 using namespace std;
5
6 int main()
7 {
8     int counter = 1; // initialize counter
9
10    do
11    {
12        cout << counter << " "; // display counter
13        counter++; // increment counter
14    } while ( counter <= 10 ); // end do...while
15
16    cout << endl; // output a newline
17 }
```

```
1 2 3 4 5 6 7 8 9 10
```

switch Multiple-Selection Statement

- ❑ The **switch multiple-selection** statement performs many different actions based on the possible values of a variable or expression.
- ❑ Each action is associated with the value of a **constant integral expression**
 - ❑ i.e., any combination of character and integer constants that evaluates to a constant integer value.

switch Multiple-Selection Statement



switch Multiple-Selection Statement

```
1
2 // Using a switch statement to count A, B, C, D and F grades.
3 #include <iostream>
4 using namespace std;
5
6 int main()
7 {
8     int grade; // letter grade entered by user
9     int aCount; // count of A grades
10    int bCount; // count of B grades
11    int cCount; // count of C grades
12    int dCount; // count of D grades
13    int fCount; // count of F grades
14
15    cout << "Enter the letter grades." << endl
16         << "Enter the EOF character to end input." << endl;
17
```

switch Multiple-Selection Statement

```
18 // loop until user types end-of-file key sequence
19 while ( ( grade = cin.get() ) != EOF )
20 {
21     // determine which grade was entered
22     switch ( grade ) // switch statement nested in while
23     {
24         case 'A': // grade was uppercase A
25         case 'a': // or lowercase a
26             aCount++; // increment aCount
27             break; // necessary to exit switch
28
29         case 'B': // grade was uppercase B
30         case 'b': // or lowercase b
31             bCount++; // increment bCount
32             break; // exit switch
33
34         case 'C': // grade was uppercase C
35         case 'c': // or lowercase c
36             cCount++; // increment cCount
37             break; // exit switch
38     }
```

switch Multiple-Selection Statement

```
39     case 'D': // grade was uppercase D
40     case 'd': // or lowercase d
41         dCount++; // increment dCount
42         break; // exit switch
43
44     case 'F': // grade was uppercase F
45     case 'f': // or lowercase f
46         fCount++; // increment fCount
47         break; // exit switch
48
49     case '\n': // ignore newlines,
50     case '\t': // tabs,
51     case ' ': // and spaces in input
52         break; // exit switch
53
54     default: // catch all other characters
55         cout << "Incorrect letter grade entered."
56             << " Enter a new grade." << endl;
57         break; // optional; will exit switch anyway
58 } // end switch
59 } // end while
60
```

switch Multiple-Selection Statement

```
61 // output summary of results
62 cout << "\n\nNumber of students who received each letter grade:"
63     << "\nA: " << aCount // display number of A grades
64     << "\nB: " << bCount // display number of B grades
65     << "\nC: " << cCount // display number of C grades
66     << "\nD: " << dCount // display number of D grades
67     << "\nF: " << fCount // display number of F grades
68     << endl;
69 } // end function main
```

switch Multiple-Selection Statement

- ❑ The `switch` statement consists of a series of **case labels** and an optional **default case**.
- ❑ The `switch` statement compares the value of the **controlling expression** with each **CASE** label.
- ❑ If a match occurs, the program executes the statements for that **CASE**.
- ❑ Listing **CASEs** consecutively with no statements between them enables the **CASEs** to perform the same set of statements.
- ❑ The `break` statement causes program control to proceed with the first statement after the `switch`.

```
switch ( grade ) // switch statement nested in while
{
    case 'A': // grade was uppercase A
    case 'a': // or lowercase a
        aCount++; // increment aCount
        break; // necessary to exit switch
```

switch Multiple-Selection Statement

- ❑ Each **case** can have multiple statements.
 - ❑ The **switch** selection statement does not require braces around multiple statements in each **case**.
- ❑ Without **break** statements, the statements for that **case** and subsequent **cases** **are all executed** when a match occurs
 - ❑ Until a **break** statement or the end of the **switch** is encountered.
 - ❑ Referred to as “falling through” to the subsequent **cases**.
- ❑ If no match occurs between the controlling expression’s value and a **case** label, the **default** case executes.
- ❑ If a **switch** statement does not contain a **default** case, program control continues with the first statement after the **switch** when no match occurs

```
case '\n': // ignore newlines,
case '\t': // tabs,
case ' ': // and spaces in input
    break; // exit switch

default: // catch all other characters
    cout << "Incorrect letter grade entered."
        << " Enter a new grade." << endl;
    break; // optional; will exit switch anyway
} // end switch
```

switch Multiple-Selection Statement

- ❑ The `cin.get()` function reads one character from the keyboard.
- ❑ A character is stored as a “number” in the computer according to its **ASCII code**.
- ❑ Normally, characters are stored in variables of type **char**; however, characters can be stored in any integer data type.
- ❑ Can treat a character either as an integer or as a character, depending on its use. For example:

- ```
cout << "The character (" << 'a' << ") has the
value "
 << static_cast< int > ('a') << endl;
```

**prints the character a and its integer value as follows:**

- The character (a) has the value 97

# ASCII Code

## ASCII Table

| Dec | Hex | Oct | Char | Dec | Hex | Oct | Char    | Dec | Hex | Oct | Char | Dec | Hex | Oct | Char |
|-----|-----|-----|------|-----|-----|-----|---------|-----|-----|-----|------|-----|-----|-----|------|
| 0   | 0   | 0   |      | 32  | 20  | 40  | [space] | 64  | 40  | 100 | @    | 96  | 60  | 140 | `    |
| 1   | 1   | 1   |      | 33  | 21  | 41  | !       | 65  | 41  | 101 | A    | 97  | 61  | 141 | a    |
| 2   | 2   | 2   |      | 34  | 22  | 42  | "       | 66  | 42  | 102 | B    | 98  | 62  | 142 | b    |
| 3   | 3   | 3   |      | 35  | 23  | 43  | #       | 67  | 43  | 103 | C    | 99  | 63  | 143 | c    |
| 4   | 4   | 4   |      | 36  | 24  | 44  | \$      | 68  | 44  | 104 | D    | 100 | 64  | 144 | d    |
| 5   | 5   | 5   |      | 37  | 25  | 45  | %       | 69  | 45  | 105 | E    | 101 | 65  | 145 | e    |
| 6   | 6   | 6   |      | 38  | 26  | 46  | &       | 70  | 46  | 106 | F    | 102 | 66  | 146 | f    |
| 7   | 7   | 7   |      | 39  | 27  | 47  | '       | 71  | 47  | 107 | G    | 103 | 67  | 147 | g    |
| 8   | 8   | 10  |      | 40  | 28  | 50  | (       | 72  | 48  | 110 | H    | 104 | 68  | 150 | h    |
| 9   | 9   | 11  |      | 41  | 29  | 51  | )       | 73  | 49  | 111 | I    | 105 | 69  | 151 | i    |
| 10  | A   | 12  |      | 42  | 2A  | 52  | *       | 74  | 4A  | 112 | J    | 106 | 6A  | 152 | j    |
| 11  | B   | 13  |      | 43  | 2B  | 53  | +       | 75  | 4B  | 113 | K    | 107 | 6B  | 153 | k    |
| 12  | C   | 14  |      | 44  | 2C  | 54  | ,       | 76  | 4C  | 114 | L    | 108 | 6C  | 154 | l    |
| 13  | D   | 15  |      | 45  | 2D  | 55  | -       | 77  | 4D  | 115 | M    | 109 | 6D  | 155 | m    |
| 14  | E   | 16  |      | 46  | 2E  | 56  | .       | 78  | 4E  | 116 | N    | 110 | 6E  | 156 | n    |
| 15  | F   | 17  |      | 47  | 2F  | 57  | /       | 79  | 4F  | 117 | O    | 111 | 6F  | 157 | o    |
| 16  | 10  | 20  |      | 48  | 30  | 60  | 0       | 80  | 50  | 120 | P    | 112 | 70  | 160 | p    |
| 17  | 11  | 21  |      | 49  | 31  | 61  | 1       | 81  | 51  | 121 | Q    | 113 | 71  | 161 | q    |
| 18  | 12  | 22  |      | 50  | 32  | 62  | 2       | 82  | 52  | 122 | R    | 114 | 72  | 162 | r    |
| 19  | 13  | 23  |      | 51  | 33  | 63  | 3       | 83  | 53  | 123 | S    | 115 | 73  | 163 | s    |
| 20  | 14  | 24  |      | 52  | 34  | 64  | 4       | 84  | 54  | 124 | T    | 116 | 74  | 164 | t    |
| 21  | 15  | 25  |      | 53  | 35  | 65  | 5       | 85  | 55  | 125 | U    | 117 | 75  | 165 | u    |
| 22  | 16  | 26  |      | 54  | 36  | 66  | 6       | 86  | 56  | 126 | V    | 118 | 76  | 166 | v    |
| 23  | 17  | 27  |      | 55  | 37  | 67  | 7       | 87  | 57  | 127 | W    | 119 | 77  | 167 | w    |
| 24  | 18  | 30  |      | 56  | 38  | 70  | 8       | 88  | 58  | 130 | X    | 120 | 78  | 170 | x    |
| 25  | 19  | 31  |      | 57  | 39  | 71  | 9       | 89  | 59  | 131 | Y    | 121 | 79  | 171 | y    |
| 26  | 1A  | 32  |      | 58  | 3A  | 72  | :       | 90  | 5A  | 132 | Z    | 122 | 7A  | 172 | z    |
| 27  | 1B  | 33  |      | 59  | 3B  | 73  | ;       | 91  | 5B  | 133 | [    | 123 | 7B  | 173 | {    |
| 28  | 1C  | 34  |      | 60  | 3C  | 74  | <       | 92  | 5C  | 134 | \    | 124 | 7C  | 174 |      |
| 29  | 1D  | 35  |      | 61  | 3D  | 75  | =       | 93  | 5D  | 135 | ]    | 125 | 7D  | 175 | }    |
| 30  | 1E  | 36  |      | 62  | 3E  | 76  | >       | 94  | 5E  | 136 | ^    | 126 | 7E  | 176 | ~    |
| 31  | 1F  | 37  |      | 63  | 3F  | 77  | ?       | 95  | 5F  | 137 | _    | 127 | 7F  | 177 |      |

# ASCII Table

---

- ❑ *You cannot type the value  $-1$  as the sentinel value.*  
(ASCII code is 0~255)
- ❑ EOF stands for “end-of-file”. Commonly used as a sentinel value for characters.
- ❑ EOF is a symbolic integer constant defined in the `<iostream>` header file.
  - ❑ The EOF has type `int`
- ❑ The keystroke combinations for entering *end-of-file* are system dependent.
  - ❑ Windows: Ctrl-Z ; UNIX: Ctrl-D

# ASCII Code

---

- ❑ To have the program read the characters, we must send them to the computer by pressing the *Enter key*.
- ❑ This places a newline character in the input after the character we wish to process.
  - ❑ Often, this newline character must be specially processed.
- ❑ The `cin.get()` function can ignore the newline character automatically
  - ❑ Some functions can do this, but some functions cannot.



# break and continue statements

---

- ❑ The **break statement**, when executed in a `while`, `for`, `do...while` or `switch` statement, causes immediate exit from that statement.
- ❑ Program execution continues with the next statement.
- ❑ Common uses of the **break** statement are to escape early from a loop or to skip the remainder of a `switch` statement.

# break and continue statements

```
1
2 // break statement exiting a for statement.
3 #include <iostream>
4 using namespace std;
5
6 int main()
7 {
8 int count; // control variable also used after loop terminates
9
10 for (count = 1; count <= 10; count++) // loop 10 times
11 {
12 if (count == 5)
13 break; // break loop only if x is 5
14
15 cout << count << " ";
16 } // end for
17
18 cout << "\nBroke out of loop at count = " << count << endl;
19 } // end main
```

```
1 2 3 4
Broke out of loop at count = 5
```

# break and continue statements

---

- ❑ The **continue statement** skips the remaining statements in its body and proceeds with the **next iteration** of the loop.
  - ❑ When executed in a `while`, `for` or `do...while` statement
- ❑ In `while` and `do...while` statements, the loop-continuation test evaluates immediately after the **continue** statement executes.
- ❑ In the `for` statement, the increment expression executes, then the loop-continuation test evaluates.

# break and continue statements

```
1
2 // continue statement terminating an iteration of a for statement.
3 #include <iostream>
4 using namespace std;
5
6 int main()
7 {
8 for (int count = 1; count <= 10; count++) // loop 10 times
9 {
10 if (count == 5) // if count is 5,
11 continue; // skip remaining code in loop
12
13 cout << count << " ";
14 } // end for
15
16 cout << "\nUsed continue to skip printing 5" << endl;
17 } // end main
```

```
1 2 3 4 6 7 8 9 10
Used continue to skip printing 5
```

# Break & Continue Scope of Effect

- ❑ In nested loops, *break/continue* can only affect the most inner loop where the *break/continue* stands

```
for (i=1; i<=5; i++)
{ for (j=1; j<=3; j++)
 {
 printf("(d,d) ", i, j);
 if (i==3) break;
 }
 printf("\n");
}
```

break inner  
loop j only

Jump out the inner loop  
and start from here

|       |       |       |
|-------|-------|-------|
| (1,1) | (1,2) | (1,3) |
| (2,1) | (2,2) | (2,3) |
| (3,1) |       |       |
| (4,1) | (4,2) | (4,3) |
| (5,1) | (5,2) | (5,3) |

- ❑ If *break* is used to skip the following *switch* statements, it has no effects on the outside loop

- ❑ One-time use only

# Usage in Infinite Loop


---

- ❑ Infinite loops are helpful when the termination condition is generated inside the loop

```
while (1) {

 ans = a * b;
 if (ans == 0) break;

}
```

 1 (non-zero value)  
means always TRUE

- ❑ Should be used with **break** to terminate the loop
  - ❑ Make sure the condition will eventually become TRUE
- ❑ If sentinel-controlled loop can be used instead, use it !!
  - ❑ Infinite loops are not easy to debug

# Logical Operators

---

- ❑ C++ provides logical operators that are used to form more complex conditions by combining simple conditions.
  - ❑ && (logical AND)
  - ❑ || (logical OR)
  - ❑ ! (logical NOT, also called logical negation).

# Logical Operators

---

## AND Operator &&

| expression1 | expression2 | expression1 && expression2 |
|-------------|-------------|----------------------------|
| false       | false       | false                      |
| false       | true        | false                      |
| true        | false       | false                      |
| true        | true        | true                       |

## OR Operator ||

| expression1 | expression2 | expression1    expression2 |
|-------------|-------------|----------------------------|
| false       | false       | false                      |
| false       | true        | true                       |
| true        | false       | true                       |
| true        | true        | true                       |



# Summary

---

- ☐ while loop statement
- ☐ do...while loop statement
- ☐ for loop statement
- ☐ switch statement
- ☐ continue and break

# LAB 4

---

- ☐ Sign up your attendance here:

[https://docs.google.com/spreadsheets/d/1-UcFXgP9A3SDcwU\\_f5XxV68YFjVbHQeYIMI3m8mGtKI/edit#gid=0](https://docs.google.com/spreadsheets/d/1-UcFXgP9A3SDcwU_f5XxV68YFjVbHQeYIMI3m8mGtKI/edit#gid=0) Mark 'Y'

- ☐ Work in a group to solve three programs

- ☐ 100 minutes to work

- ☐ Select one person to present

- ☐ Instructor will randomly decide which problem to present your solution

# Practice 1

---

☐ **Write a program that prints all multiples of 3**

☐ Version 1: for loop + continue

☐ Version 2: while loop + break

☐ **Input: N**

☐ **Output: multiples of 3 in the range [1, n]**

☐ **Example**

☐ N=10 => output: 3, 6, 9

☐ N=17 => output: 3, 6, 9, 12, 15

# Practice 2: $3n+1$

---

Consider the following two operations on an arbitrary positive integer:

- If the number is even, divide it by two.
- If the number is odd, triple it and add one.

The Collatz conjecture is: *This process will eventually reach the number 1, regardless of which positive integer is chosen initially.*

Write a program that takes an input  $N$  and then prints your progress of  $3n+1$ . For example, when  $N=12$ , your program prints 12, 6, 3, 10, 5, 16, 8, 4, 2, 1.

# Practice 3

---

☐ **Input: N**

☐ **Output**

- ☐ If N is larger than 10 and N is an even number
  - Print "N is an even number larger than 10"
- ☐ If N is larger than 10 and N is an odd number
  - Print "N is an odd number larger than 10"
- ☐ If N is smaller than or equal to 10, and is even
  - Print "N is an even number  $\leq 10$ "
- ☐ If N is smaller than or equal to 10, and is odd
  - Print "N is an odd number  $\leq 10$ "