

Chapter 6

Looping

Dale/Weems/Headington

1

Chapter 6 Topics

- **While Statement Syntax**
- **Count-Controlled Loops**
- **Event-Controlled Loops**
- **Using the End-of-File Condition to Control Input Data**
- **Using a While Statement for Summing and Counting**
- **Nested While Loops**
- **Loop Testing and Debugging**

2

What is a loop?

- **A loop is a repetition control structure.**
- **it causes a single statement or block to be executed repeatedly**

3

Two Types of Loops

count controlled loops

repeat a specified number of times

event-controlled loops

some condition within the loop body changes and this causes the repeating to stop

4

Iteration
6. Iteration 5

Iteration causing a set of statements (the body) to be executed repeatedly.

C++ provides three control structures to support iteration (or looping).

Before considering specifics we define some general terms that apply to any iteration construct:

pass (or iteration)	- one complete execution of the body
loop entry	- the point where flow of control passes to the body
loop test	- the point at which the decision is made to (re)enter the body, or not
loop exit	- the point at which the iteration ends and control passes to the next statement after the loop
termination condition	- the condition that causes the iteration to stop

When designing a loop, it is important to clearly identify each of these.

Understanding the termination condition, and what guarantees it will eventually occur, are particularly vital.

Computer Science Dept Va Tech August, 2002
Intro Programming in C++
©1995-2002 Barnette ND & McQuain WD

while Statement

SYNTAX

```

while ( Expression )
{
    .
    .
    .
}
// loop body
```

NOTE: Loop body can be a single statement, a null statement, or a block.

while Loop
6. Iteration 7

The most versatile loop construct in C++ is the `while` statement. Syntactically:

```

  graph LR
    Start(( )) --> While(while)
    While --> BoolExpr[<bool expr>]
    BoolExpr --> LoopBody[<loop body>]
    LoopBody --> End(( ))
  
```

The Boolean expression must be enclosed in parentheses, and **<loop body>** can be either a single statement or a compound statement.

Semantically:

```

  graph TD
    Start(( )) --> While(while)
    While --> BoolExpr[<bool expr>]
    BoolExpr -- true --> LoopBody[<loop body>]
    LoopBody --> BoolExpr
    BoolExpr -- false --> Exit(( ))
  
```

The Boolean expression is examined before each pass through the body of the loop. If the Boolean expression is true, the loop body is executed; if it is false, execution proceeds to the first statement after the loop body..

Computer Science Dept Va Tech August, 2002
Intro Programming in C++
©1995-2002 Barnette ND & McQuain WD

while Loop Example
6. Iteration 8

Example:

```

#include <iostream>
using namespace std;

int main() {

    const char ASTERISK = '*';           // 1
    int numAst;                          // 2
    int numPrinted = 0;                  // 3

    cout << "How many asterisks do you want? "; // 4
    cin >> numAst;                        // 5

    while (numPrinted < numAst) {        // 6

        cout << ASTERISK;                 // 7
        numPrinted++;                     // 8
    }
    cout << endl;                          // 9

    return 0;                             // 10
}
  
```

This is a count-controlled loop. The loop control variable (LCV) is a variable whose value determines whether the loop body is executed.

Computer Science Dept Va Tech August, 2002
Intro Programming in C++
©1995-2002 Barnette ND & McQuain WD

while Loop Characteristics
6. Iteration 9

Some observations:

- It is possible that the body of the loop will never be executed.
- It is possible that the loop test condition will become false in the middle of an iteration. Even so, the remainder of the loop body will be executed before the loop test is performed and iteration stops.
- The loop test must involve at least one variable whose value is (potentially) changed by at least one statement within the loop body. Why?

```

. . .
while (numPrinted < numAst) { // 6
    cout << ASTERISK;         // 7
    numPrinted++;             // 8
}
cout << endl;                 // 9
. . .

```

Computer Science Dept Va Tech August, 2002
Intro Programming in C++
©1995-2002 Barnette ND & McQuain WD

Count-controlled Loop

```

int count ;

count = 4;           // initialize loop variable

while (count > 0)   // test expression
{
    cout << count << endl ; // repeated action

    count -- ;      // update loop variable
}
cout << "Done" << endl ;

```

Count-controlled Loop

```
int count ;  
  
count = 4;  
  
while (count > 0)  
{  
    cout << count << endl ;  
  
    count -- ;  
}  
cout << "Done" << endl ;
```

count

OUTPUT

11

Count-Controlled Loop Example

myInfile contains 100 blood pressures

Use a while loop to read the 100 blood pressures and find their total

12

```
ifstream  myInfile ;
int       thisBP ;
int       total ;
int       count ;

count = 0 ;                               // initialize

while ( count < 100 )                     // test expression
{
    myInfile >> thisBP ;
    total = total + thisBP ;
    count++ ;                             // update
}

cout << "The total = " << total << endl ;
```

13

Event-controlled Loops

- **Sentinel controlled**
keep processing data until a special value which is not a possible data value is entered to indicate that processing should stop
- **End-of-file controlled**
keep processing data as long as there is more data in the file
- **Flag controlled**
keep processing data until the value of a flag changes in the loop body

14

Examples of Kinds of Loops

Count controlled loop	Read exactly 100 blood pressures from a file.
End-of-file controlled loop	Read all the blood pressures from a file no matter how many are there.

15

Examples of Kinds of Loops

Sentinel controlled loop	Read blood pressures until a special value (like -1) selected by you is read.
Flag controlled loop	Read blood pressures until a dangerously high BP (200 or more) is read.

16

Count Controlled Loop
6. Iteration 17

Count Controlled Loop a loop that terminates when a counter reaches some limiting value

The LCV (loop control variable) will be:

- an integer variable used in the Boolean expression
- initialized before the loop,
- incremented or decremented within the loop body

```

. . .
cout << "How many asterisks do you want? "; // 4
cin  >> numAst;                               // 5

// modified implementation
while ( numAst > 0 ) {                          // 6

    cout << ASTERISK;                            // 7
    numAst--;                                    // 8
}
cout << endl;                                    // 9
. . .

```

Computer Science Dept Va Tech August, 2002
Intro Programming in C++
©1995-2002 Barnette ND & McQuain WD

Event Controlled Loop
6. Iteration 18

Event Controlled Loop: loop that executes until a specified situation arises to signal the end of the iteration.

```

. . .
const int CENTSPERDOLLAR = 100; // 1
string Description; // 2
int Dollars, // 3
    Cents; // 4
int totalCents = 0; // 5

getline(In, Description, '\t'); // 6
In >> Dollars; // 7
In.ignore(1, '.'); // 8
In >> Cents; // 8
In.ignore(INT_MAX, '\n'); // 10

while ( In ) { // 11

    totalCents = totalCents +
        + CENTSPERDOLLAR*Dollars // 12
        + Cents;

    // repeat lines 6 through 10
}
. . .

```

Visual C++	46.50
Excedrin	2.99
Excedrin Migraine	3.99
How to Solve It	33.95
The Art of War	17.50

This loop is terminated by the event of an input failure.

Computer Science Dept Va Tech August, 2002
Intro Programming in C++
©1995-2002 Barnette ND & McQuain WD

■ Sentinel Controlled Loop
6. Iteration 19

Sentinel loop: loop that terminates when a specified marker (dummy data value) is read, signaling the end of the iteration.

Here we modify the previous input file to include a special marker line to signify the end of the price data in the file:

```

const int CENTSPERDOLLAR = 100; // 1
const string SENTINEL = "xxxNoItemxxx"; // 2
. . .
getline(In, Description, '\t'); // 7
In >> Dollars; // 8
In.ignore(1, '.'); // 9
In >> Cents; // 10
In.ignore(INT_MAX, '\n'); // 11

while ( Description != SENTINEL ) { // 12

    totalCents = totalCents +
        + CENTSPERDOLLAR*Dollars
        + Cents; // 13

    // repeat lines 7 through 11
}
. . .

```

Visual C++	46.50
Excedrin	2.99
Excedrin Migraine	3.99
How to Solve It	33.95
The Art of War	17.50
xxxNoItemxxx	0.00
. . .	
Deliver to:	
Joe Bob Hokie	
666 Pritchard Hall	
. . .	

This loop is terminated by the event of reading the special sentinel value.

Computer Science Dept Va Tech August, 2002
Intro Programming in C++
©1995-2002 Barnette ND & McQuain WD

■ An Improvement Using Hybrid Control
6. Iteration 20

The pure sentinel control on the previous slide is risky. What would happen if the sentinel line were omitted?

We can eliminate this problem by adding a check for input failure to the loop test:

```

const int CENTSPERDOLLAR = 100; // 1
const string SENTINEL = " xxxNoItemxxx "; // 2
. . .
getline(In, Description, '\t'); // 7
In >> Dollars; // 8
In.ignore(1, '.'); // 9
In >> Cents; // 10
In.ignore(INT_MAX, '\n'); // 11

while ( In && Description != SENTINEL ) { // 12

    totalCents = totalCents +
        + CENTSPERDOLLAR*Dollars
        + Cents; // 13

    // repeat lines 7 through 11
}
. . .

```

Visual C++	46.50
Excedrin	2.99
Excedrin Migraine	3.99
How to Solve It	33.95
The Art of War	17.50
xxxNoItemxxx	0.00
. . .	

This loop is terminated by either the event of an input failure, or by the event of reading the special sentinel value.

Computer Science Dept Va Tech August, 2002
Intro Programming in C++
©1995-2002 Barnette ND & McQuain WD

Boolean Loop Control Variable 6. Iteration 21

It's fairly common to use a Boolean variable to record the results of a more complicated test. Here we assume that there is a sentinel line in which the price is invalid:

```

. . .
getline(In, Description, '\t');           // 6
In >> Dollars;                            // 7
In.ignore(1, '.');                        // 8
In >> Cents;                               // 9
In.ignore(INT_MAX, '\n');                 // 10

bool priceOK = (Dollars > 0) ||
               (Dollars == 0 && Cents > 0); // 11

while ( In && priceOK ) {                  // 12

    totalCents = totalCents +
                + CENTS PER DOLLAR * Dollars
                + Cents;                  // 13

    // repeat lines 6 through 10
    . . .
    priceOK = (Dollars > 0) ||
              (Dollars == 0 && Cents > 0); // 19
}
. . .
    
```

Visual C++	46.50
Excedrin	2.99
Excedrin Migraine	3.99
How to Solve It	33.95
The Art of War	17.50
xxxNoItemxxx	0.00
. . .	

Note that the Boolean loop control variable `priceOK` could be replaced by the condition used to set it, but the resulting code would perhaps not be as readable.

Computer Science Dept Va Tech August, 2002
Intro Programming in C++
©1995-2002 Barnette ND & McQuain WD

Count Controlled Input 6. Iteration 22

In some cases, the number of input records to be read will be specified in the input file itself. This is sometimes called the data header approach.

Here, we have a file of temperature data, with the first line specifying how many temperature values are given.

We might process this input file with the following loop:

```

. . .
inData >> tempsPromised; // get # temps expected
inData >> Temperature;   // read first one

while ( inData && tempsRead < tempsPromised ) {

    tempsRead++;           // count temps read
    // processing code goes here
    . . .

    inData >> Temperature; // read next one
}
. . .
    
```

7
79
65
78
62
73
81
89

Note that we do NOT place absolute trust in the data header value.

Computer Science Dept Va Tech August, 2002
Intro Programming in C++
©1995-2002 Barnette ND & McQuain WD

Complete Example
6. Iteration 23

Let's extend the last code fragment to a complete program. Suppose that we are required to determine the highest temperature and the overall average temperature.

```

#include <iostream>
#include <fstream>
#include <iomanip>
using namespace std;

int main() {

    ifstream inData("Temperature.data");

    int    tempsPromised;        // # temp values expected
    int    tempsRead = 0;        // # temps read so far
    int    hiTemp = INT_MIN;     // max temp so far
    double sumOfTemps = 0.0;     // sum for averaging
    int    Temperature;         // temperature just read

    inData >> tempsPromised;     // get # temps expected
    inData >> Temperature;       // read first one
    . . .

```

7
70
65
73
62
73
81
89

Note that we do NOT place absolute trust in the data header value.

Computer Science Dept Va Tech August, 2002
Intro Programming in C++
©1995-2002 Barnette ND & McQuain WD

Complete Example
6. Iteration 24

```

. . .
while ( inData && tempsRead < tempsPromised ) {

    tempsRead++;                // count temps read
                                // and keep sum
    sumOfTemps = sumOfTemps + Temperature;

    if ( Temperature > hiTemp ) // check for new max
        hiTemp = Temperature; // update if so

    inData >> Temperature;      // read next one
}

// Report results:
if ( tempsRead > 0 ) {
    cout << "Highest temperature was: "
         << hiTemp << '.' << endl;

    double avgTemp = sumOfTemps / tempsRead;
    cout << fixed << showpoint;
    cout << "Average temperature was: "
         << setprecision(1) << avgTemp << '.' << endl;
}

return 0;
}

```

7
70
65
73
62
73
81
89

Computer Science Dept Va Tech August, 2002
Intro Programming in C++
©1995-2002 Barnette ND & McQuain WD

End-of-line Controlled Loop
6. Iteration 25

```

. . .
In.get(Next);    // Read 1st char

while ( In ) {
  lineLength = 0;           // Restart line length count.
  while (Next != '\n') {   // Process next line:
    lineLength++;         //   count char
    cout << Next;         //   echo char
    In.get(Next);        //   Read next char.
  }
  numChars = numChars + lineLength;
  numLines++;
  cout << setw(50 - lineLength) << lineLength << endl;
  In.get(Next);
}

cout << "Number of characters: " << numChars << endl;
cout << "Number of lines:      " << numLines << endl;
. . .

```

Things should be made
as simple as possible,
but no simpler.

¶ represents the newline char
\$ represents the end of file char

Computer Science Dept Va Tech August, 2002
Intro Programming in C++
©1995-2002 Barnette ND & McQuain WD

Loop Design Considerations
6. Iteration 26

Questions that one should consider carefully when coding a loop:

- What is the condition that terminates the loop?
- How should the condition be initialized?
- How should the condition be updated?
- What guarantees this condition will eventually occur?
- What is the process to be repeated?
- How should the process be initialized?
- How should the process be updated?
- What is the state of the program on exiting the loop?
- Are "boundary conditions" handled correctly?

Computer Science Dept Va Tech August, 2002
Intro Programming in C++
©1995-2002 Barnette ND & McQuain WD

A Count-Controlled Loop

SYNTAX

```
for ( initialization ; test expression ; update )  
{  
    0 or more statements to repeat  
}
```

27

The for loop contains

an initialization

an expression to test for
continuing

an update to execute after each
iteration of the body

28

Example of Repetition

```
int num;

for ( num = 1 ; num <= 3 ; num++ )
{
    cout << num << "Potato" << endl;
}
```

29

num

?

Example of Repetition

```
int num;

for ( num = 1 ; num <= 3 ; num++ ) {
    cout << num << "Potato" << endl;
}
```

OUTPUT

30

for Loop
6. Iteration 31

For loops are used whenever the number of times a loop needs to execute is known or can be calculated beforehand.

```
for (initial expression; test expression; update expression)
<statement>
```

The initial expression is performed just once, prior to loop execution. The initial expression may declare variables as well as specify initializations for them.

The test expression is evaluated and if false then the next statement after the for loop is executed.

If the test expression evaluates to true, then the <statement> of the for loop is executed, the update expression is performed, and test expression is evaluated again.

For loops are generally count-controlled, but hybrid control is also fairly common, especially when input failure control is involved.

Computer Science Dept Va Tech August, 2002
Intro Programming in C++
©1995-2002 Barnette ND & McQuain WD

for Loop Example
6. Iteration 32

Example:

```
#include <iostream>
using namespace std;

int main() {

    const char ASTERISK = '*';           // 1
    int numAst;                          // 2
    int numPrinted;                      // 3

    cout << "How many asterisks do you want? "; // 4
    cin >> numAst;                        // 5

    for (numPrinted =0; numPrinted < numAst; numPrinted++) { // 6

        cout << ASTERISK;                 // 7
    }
    cout << endl;                          // 8

    return 0;                              // 9
}
```

Compare this with the implementation on slide 3 using a while loop.

The for loop is simply an alternative control structure (to the while); any for loop can be expressed as a while loop, and vice versa.

Computer Science Dept Va Tech August, 2002
Intro Programming in C++
©1995-2002 Barnette ND & McQuain WD

More for Loop Examples
6. Iteration 33

```
const char Separator = '-';
const int Length = 80;
int toWrite;

for (toWrite = Length; toWrite > 0; toWrite--) { // count down
    cout << Separator;
}
```

```
const int LIMIT = 100;
int sumOfSquares,
    Curr;

for (Curr = 1, sumOfSquares = 0; Curr <= LIMIT; Curr++)
    sumOfSquares = sumOfSquares + Curr * Curr;

cout << "Sum of squares = " << sumOfSquares;
```

```
. . .
for (Curr = 1, sumOfSquares = 0;
     Curr <= LIMIT;
     sumOfSquares = sumOfSquares + Curr * Curr, Curr++)
    ; // empty body, all the "action" is in the update section
. . .
```

Computer Science Dept Va Tech August, 2002
Intro Programming in C++
©1995-2002 Barnette ND & McQuain WD

Do-While Statement

Is a looping control structure in which the loop condition is tested after each iteration of the loop.

SYNTAX

```
do
{
    Statement
} while ( Expression );
```

Loop body statement can be a single statement or a block.

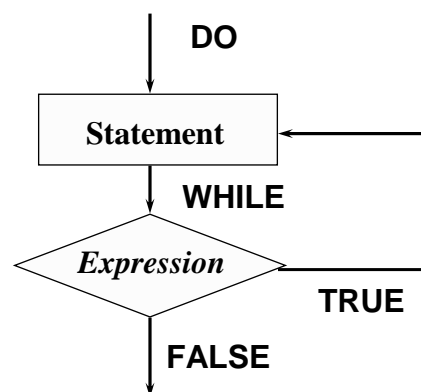
Do-While Loop vs. While Loop

- POST-TEST loop (exit-condition)
- The looping condition is tested after executing the loop body.
- Loop body is always executed at least once.

- PRE-TEST loop (entry-condition)
- The looping condition is tested before executing the loop body.
- Loop body may not be executed at all.

35

Do-While Loop



When the expression is tested and found to be false, the loop is exited and control passes to the statement that follows the do-while statement.

36

do-while Loop
6. Iteration 37

The third loop construct in C++ is the do-while statement. Syntactically:

```

  graph LR
    A(do {) --> B(<loop body>)
    B --> C(} while)
    C --> D(<bool expr>)
    D --> E(;)
  
```

The Boolean expression must be enclosed in parentheses, and **<loop body>** can be either a single statement or a list of statements. A peculiarity is that the loop body must be enclosed in braces, even if it is a single statement.

Semantically:

```

  graph TD
    A(do {) --> B(<loop body>)
    B --> C(} while)
    C --> D(<bool expr>)
    D -- true --> B
    D -- false --> E[ ]
  
```

Computer Science Dept Va Tech August, 2002 Intro Programming in C++ ©1995-2002 Barnette ND & McQuain WD

do-while Example
6. Iteration 38

Example: find first occurrence, if any, of a specific character in an input stream:

```

const char TOFIND = 'X';
char Next;
int Skipped = -1;

do {
    In.get(Next);
    Skipped++;
} while (In && Next != TOFIND);

if ( In ) {
    cout << TOFIND " found after " << Skipped
        << " characters were skipped." << endl;
}
else {
    cout << TOFIND " was not found." << endl;
}
  
```

Computer Science Dept Va Tech August, 2002 Intro Programming in C++ ©1995-2002 Barnette ND & McQuain WD