Control Statements in C++

if…else and conditional operator (?:)

```
if(condition)
{
    //Body
}


if (condition)
{
    //body
}

else
{
    //Body
}
```

The conditional operator is C++'s only **ternary operator**

```
cout << ( grade >= 60 ? "Passed" : "Failed" );
```

```
grade >= 60 ? cout << "Passed" : cout << "Failed";
```

```
#include <iostream>

using namespace std;

int main()
{
    int a=6;

    cout<< (a>5 ? "a>5" : "a<=5");

    return 0;
}
```

Dangling-else problem

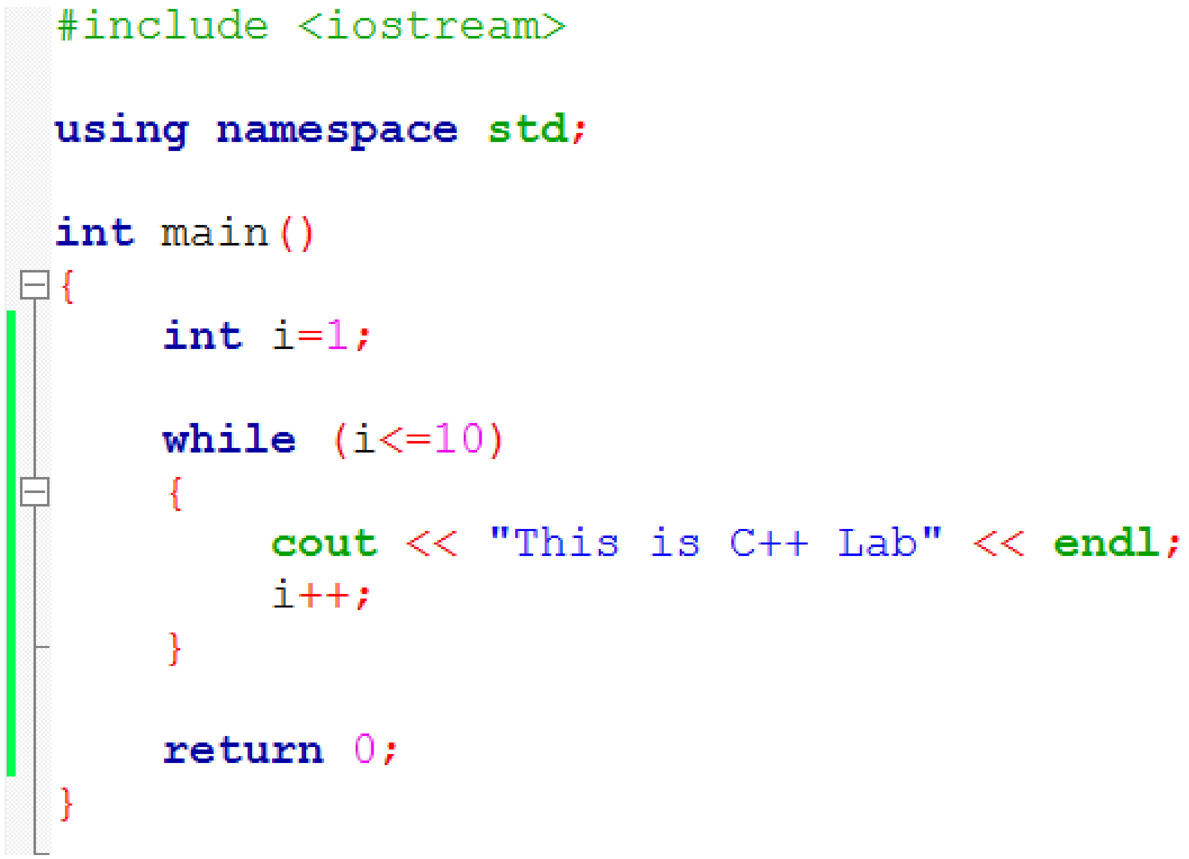
```
if ( x > 5 )
   if ( y > 5 )
      cout << "x and y are > 5";
else
   cout << "x is <= 5";
```

```
if ( x > 5 )
   if ( y > 5 )
      cout << "x and y are > 5";
   else
      cout << "x is <= 5";
```

```
if ( x > 5 )
{
   if ( y > 5 )
      cout << "x and y are > 5";
}
else
   cout << "x is <= 5";
```

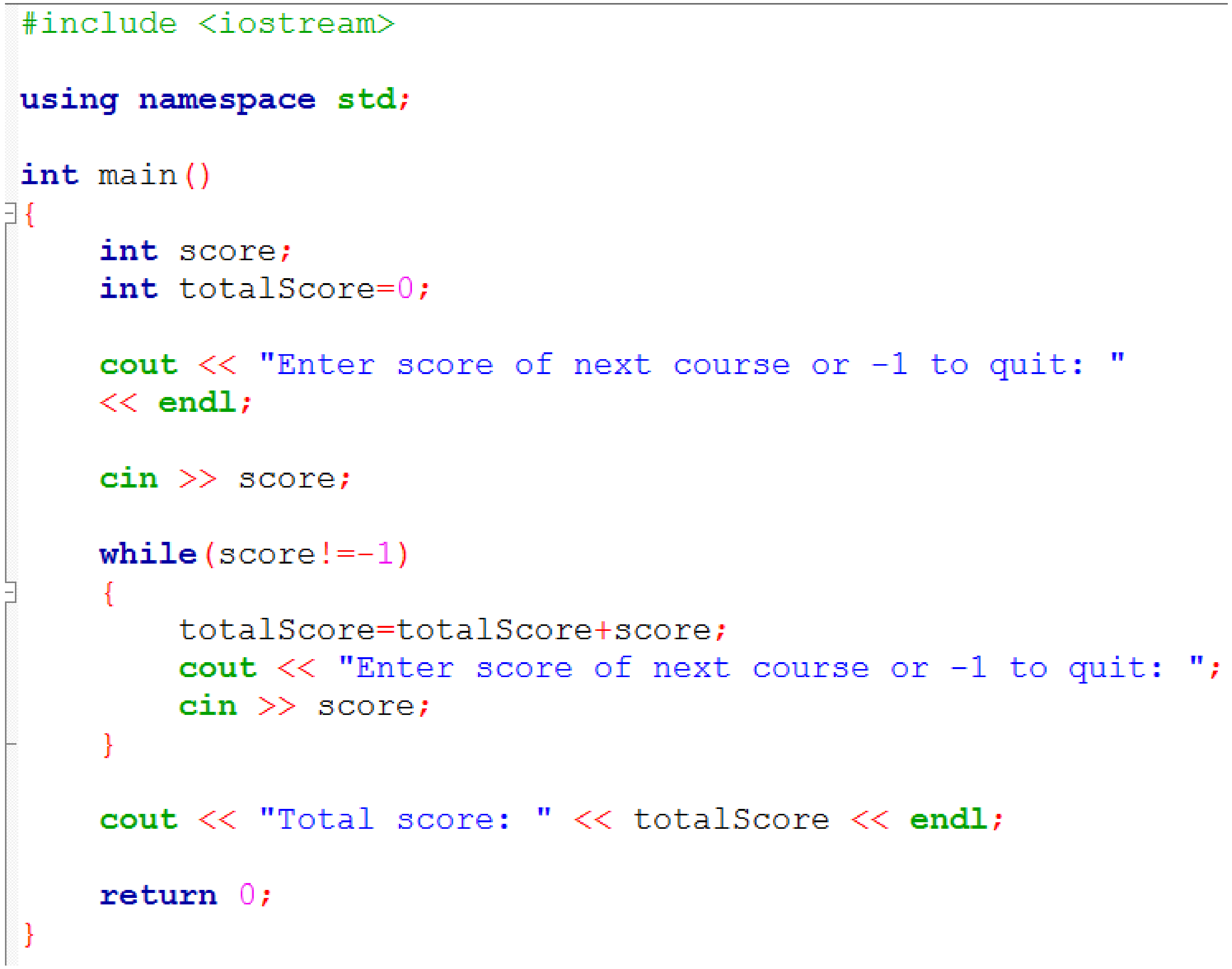
while repetition statement

1) Counter-controlled



```
#include <iostream>

using namespace std;

int main()
{
    int i=1;

    while (i<=10)
    {
        cout << "This is C++ Lab" << endl;
        i++;
    }

    return 0;
}
```

- 2) Sentinel-controlled



```
#include <iostream>

using namespace std;

int main()
{
    int score;
    int totalScore=0;

    cout << "Enter score of next course or -1 to quit: "
    << endl;

    cin >> score;

    while(score!=-1)
    {
        totalScore=totalScore+score;
        cout << "Enter score of next course or -1 to quit: ";
        cin >> score;
    }

    cout << "Total score: " << totalScore << endl;

    return 0;
}
```

Assignment Operators

```
c = c + 3;
```

can be abbreviated with the **addition assignment operator +=** as

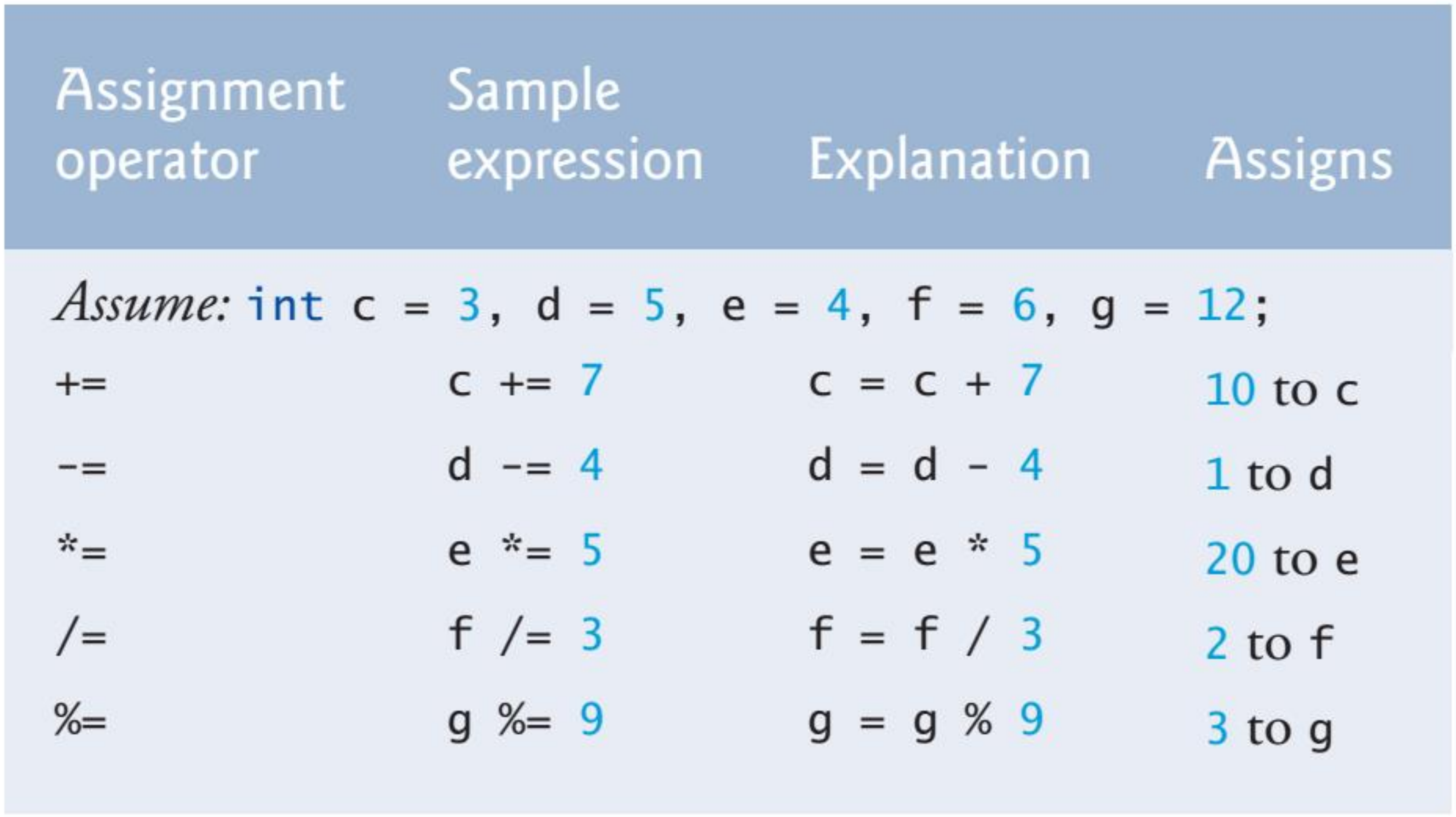
```
c += 3;
```

variable = *variable operator expression* ;

variable operator= *expression* ;

+=	-=	*=	/=	%=

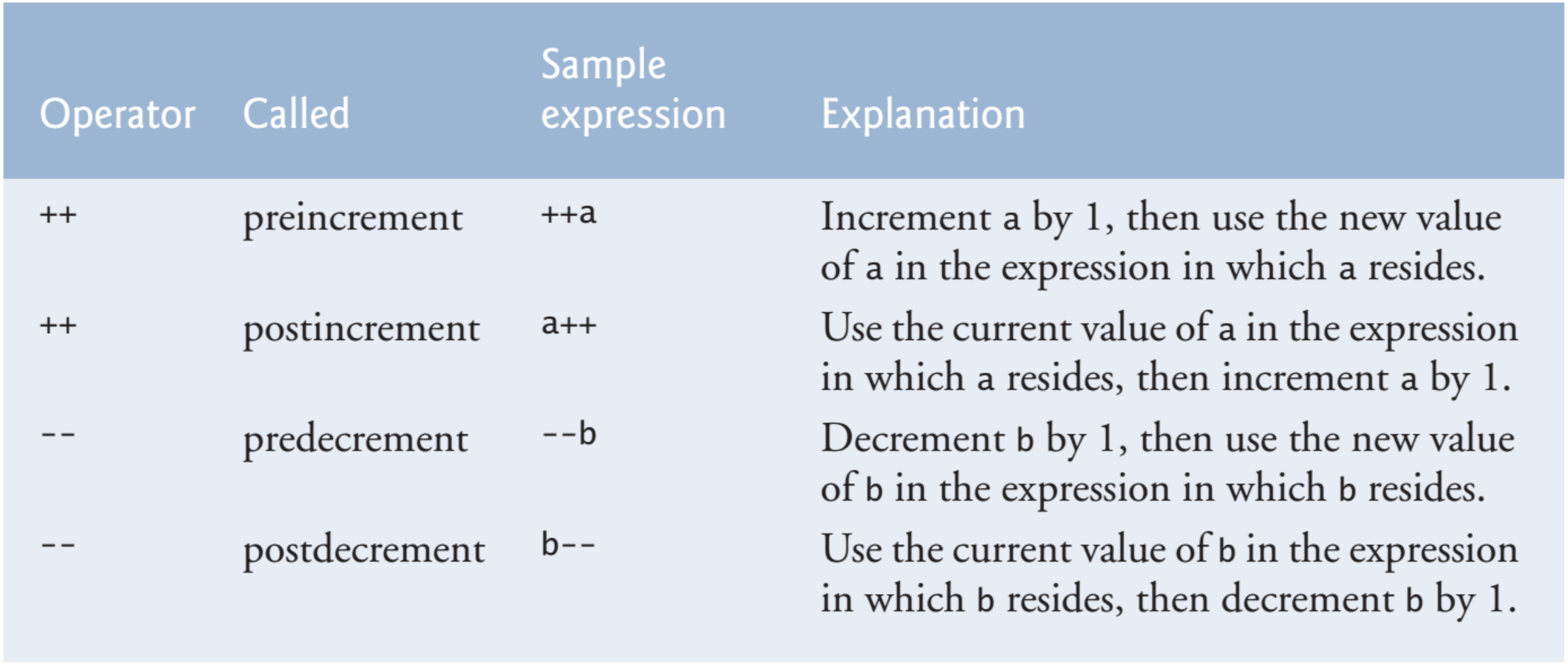
Assignment Operators



Assignment operator	Sample expression	Explanation	Assigns
Assume: `int c = 3, d = 5, e = 4, f = 6, g = 12;`			
`+=`	`c += 7`	`c = c + 7`	10 to c
`-=`	`d -= 4`	`d = d - 4`	1 to d
`*=`	`e *= 5`	`e = e * 5`	20 to e
`/=`	`f /= 3`	`f = f / 3`	2 to f
`%=`	`g %= 9`	`g = g % 9`	3 to g

Arithmetic assignment operators.

Increment and Decrement Operator



Operator	Called	Sample expression	Explanation
`++`	preincrement	`++a`	Increment `a` by 1, then use the new value of `a` in the expression in which `a` resides.
`++`	postincrement	`a++`	Use the current value of `a` in the expression in which `a` resides, then increment `a` by 1.
`--`	predecrement	`--b`	Decrement `b` by 1, then use the new value of `b` in the expression in which `b` resides.
`--`	postdecrement	`b--`	Use the current value of `b` in the expression in which `b` resides, then decrement `b` by 1.

```
// Fig. 4.19: fig04_19.cpp
// Preincrementing and postincrementing.
#include <iostream>
using namespace std;

int main()
{
   int c;

   // demonstrate postincrement
   c = 5; // assign 5 to c
   cout << c << endl; // print 5
   cout << c++ << endl; // print 5 then postincrement
   cout << c << endl; // print 6

   cout << endl; // skip a line

   // demonstrate preincrement
   c = 5; // assign 5 to c
   cout << c << endl; // print 5
   cout << ++c << endl; // preincrement then print 6
   cout << c << endl; // print 6
} // end main
```

A class of ten students took a quiz. The grades (integers in the range 0 to 100) for this quiz are available to you. Calculate and display the total of all student grades and the class average on the quiz.

```cpp
// Fig. 4.8: GradeBook.h
// Definition of class GradeBook that determines a class average.
// Member functions are defined in GradeBook.cpp
#include <string> // program uses C++ standard string class
using namespace std;

// GradeBook class definition
class GradeBook
{
public:
   GradeBook( string ); // constructor initializes course name
   void setCourseName( string ); // function to set the course name
   string getCourseName(); // function to retrieve the course name
   void displayMessage(); // display a welcome message
   void determineClassAverage(); // averages grades entered by the user
private:
   string courseName; // course name for this GradeBook
}; // end class GradeBook
```

```cpp
// Fig. 4.9: GradeBook.cpp
// Member-function definitions for class GradeBook that solves the
// class average program with counter-controlled repetition.
#include <iostream>
#include "GradeBook.h" // include definition of class GradeBook
using namespace std;

// constructor initializes courseName with string supplied as argument
GradeBook::GradeBook( string name )
{
   setCourseName( name ); // validate and store courseName
} // end GradeBook constructor

// function to set the course name;
// ensures that the course name has at most 25 characters
void GradeBook::setCourseName( string name )
{
   if ( name.length() <= 25 ) // if name has 25 or fewer characters
      courseName = name; // store the course name in the object
   else // if name is longer than 25 characters
   { // set courseName to first 25 characters of parameter name
      courseName = name.substr( 0, 25 ); // select first 25 characters
      cout << "Name \"" << name << "\" exceeds maximum length (25).\n"
         << "Limiting courseName to first 25 characters.\n" << endl;
   } // end if...else
} // end function setCourseName

// function to retrieve the course name
string GradeBook::getCourseName()
```

```cpp
{
   return courseName;
} // end function getCourseName

// display a welcome message to the GradeBook user
void GradeBook::displayMessage()
{
   cout << "Welcome to the grade book for\n" << getCourseName() << "!\n"
      << endl;
} // end function displayMessage

// determine class average based on 10 grades entered by user
void GradeBook::determineClassAverage()
{
   int total; // sum of grades entered by user
   int gradeCounter; // number of the grade to be entered next
   int grade; // grade value entered by user
   int average; // average of grades

   // initialization phase
   total = 0; // initialize total
   gradeCounter = 1; // initialize loop counter

   // processing phase
   while ( gradeCounter <= 10 ) // loop 10 times
   {
      cout << "Enter grade: "; // prompt for input
      cin >> grade; // input next grade
      total = total + grade; // add grade to total
      gradeCounter = gradeCounter + 1; // increment counter by 1
   } // end while

   // termination phase
   average = total / 10; // integer division yields integer result

   // display total and average of grades
   cout << "\nTotal of all 10 grades is " << total << endl;
   cout << "Class average is " << average << endl;
} // end function determineClassAverage
```

```cpp
// Fig. 4.10: fig04_10.cpp
// Create GradeBook object and invoke its determineClassAverage function.
#include "GradeBook.h" // include definition of class GradeBook

int main()
{
   // create GradeBook object myGradeBook and
   // pass course name to constructor
   GradeBook myGradeBook( "CS101 C++ Programming" );

   myGradeBook.displayMessage(); // display welcome message
   myGradeBook.determineClassAverage(); // find average of 10 grades
} // end main
```

```
Welcome to the grade book for
CS101 C++ Programming

Enter grade: 67
Enter grade: 78
Enter grade: 89
Enter grade: 67
Enter grade: 87
Enter grade: 98
Enter grade: 93
Enter grade: 85
Enter grade: 82
Enter grade: 100

Total of all 10 grades is 846
Class average is 84
```

Various control structures

- For (…) {…}
- Do {…} while ()
- Switch. switch(…){case : default…}
- break and continue (break;)
- Logical operators (&&, ||, !)