CISC181 Introduction to Computer Science

Dr. McCoy

Lecture 4
September 10, 2009

2.11 Assignment Operators

- Assignment expression abbreviations
  - Addition assignment operator
    `c = c + 3;` abbreviated to `c += 3;`
- Statements of the form
  `variable = variable operator expression;`
  can be rewritten as
  `variable operator= expression;`
- Other assignment operators

```
d -= 4      (d = d - 4)
e *= 5      (e = e * 5)
f /= 3      (f = f / 3)
g %= 9      (g = g % 9)
```


2.7 while Repetition Structure

- Repetition structure
  - Action repeated while some condition remains true
  - Psuedocode
    while there are more items on my shopping list
    Purchase next item and cross it off my list
  - `while` loop repeated until condition becomes false
- Example

```
int product = 2;
while ( product <= 1000 )
   product = 2 * product;
```


2.7 The while Repetition Structure

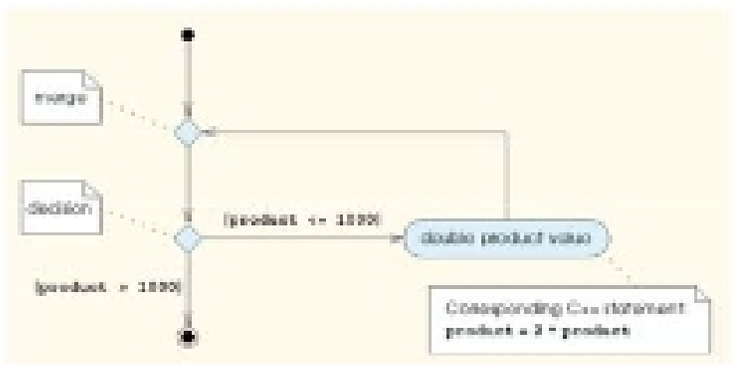

Fig. 2.8 while repetition structure activity diagram.

2.8 Formulating Algorithms (Counter-Controlled Repetition)

- Counter-controlled repetition
  - Loop repeated until counter reaches certain value
- Definite repetition
  - Number of repetitions known
- Example

  A class of ten students took a quiz. The grades (integers in the range 0 to 100) for this quiz are available to you. Determine the class average on the quiz.

2.8 Formulating Algorithms (Counter-Controlled Repetition)

- Pseudocode for example:

  Set total to zero
  Set grade counter to one
  While grade counter is less than or equal to ten
  Input the next grade
  Add the grade into the total
  Add one to the grade counter
  Set the class average to the total divided by ten
  Print the class average

- Next: C++ code for this example

```cpp
// Fig. 2.7: fig02_07.cpp
// Class average program with sentinel-controlled repetition.
#include <iostream>

using std::cout;
using std::cin;
using std::endl;

// function main begins program execution
int main()
{
   int total;        // sum of grades input by user
   int gradeCounter; // number of grade to be entered next
   int grade;        // grade value
   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;                // read grade from user
      total = total + grade;       // add grade to total
      gradeCounter = gradeCounter + 1;  // increment counter
   }

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

   // display result
   cout << "Class average is " << average << endl;

   return 0;   // indicate
}
```

Outline

fig02_07.cpp
(1 of 2)


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

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

   // display result
   cout << "Class average is " << average << endl;

   return 0;   // indicate
}
```

The counter gets incremented each time the loop executes. Eventually, the counter causes the loop to end.

```
Enter grade: 88
Enter grade: 79
Enter grade: 95
Enter grade: 87
Enter grade: 48
Enter grade: 58
Enter grade: 87
Enter grade: 78
Enter grade: 83
Enter grade: 94
Class average is 81
```

Outline

fig02_07.cpp
(2 of 2)

fig02_07.cpp
output (1 of 1)

2.9 Formulating Algorithms (Sentinel-Controlled Repetition)

- Suppose problem becomes:
  Develop a class-averaging program that will process an arbitrary number of grades each time the program is run
  - Unknown number of students
  - How will program know when to end?
- Sentinel value
  - Indicates "end of data entry"
  - Loop ends when sentinel input
  - Sentinel chosen so it cannot be confused with regular input
    - -1 in this case

2.9 Formulating Algorithms (Sentinel-Controlled Repetition)

- Top-down, stepwise refinement
  - Begin with pseudocode representation of top
    Determine the class average for the quiz
  - Divide top into smaller tasks, list in order
    Initialize variables
    Input, sum and count the quiz grades
    Calculate and print the class average

2.9 Formulating Algorithms (Sentinel-Controlled Repetition)

- Many programs have three phases
  - Initialization
    - Initializes the program variables
  - Processing
    - Input data, adjusts program variables
  - Termination
    - Calculate and print the final results
  - Helps break up programs for top-down refinement

2.9 Formulating Algorithms (Sentinel-Controlled Repetition)

- Refine the initialization phase
  Initialize variables
  goes to
  Initialize total to zero
  Initialize counter to zero
- Processing
  Input, sum and count the quiz grades
  goes to
  Input the first grade (possibly the sentinel)
  While the user has not as yet entered the sentinel
  Add this grade into the running total
  Add one to the grade counter
  Input the next grade (possibly the sentinel)

2.9 Formulating Algorithms (Sentinel-Controlled Repetition)

- Termination

Calculate and print the class average

goes to

If the counter is not equal to zero

Set the average to the total divided by the counter

Print the average

Else

Print "No grades were entered"

- Next: C++ program


```
// Fig. 2.9: fig02_09.cpp
// Class average program with sentinel-controlled repetition.
#include <iostream>

using std::cout;
using std::cin;
using std::endl;
using std::fixed;

#include <iomanip>          // parameterized stream manipulators

using std::setprecision;    // sets numeric output precision

// function main begins program execution
int main()
{
   int total;              // sum of grades
   int gradeCounter;       // number of grades entered
   int grade;              // grade value

   double average;         // number with decimal point for average

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

Data type double used to represent decimal numbers.

Outline

fig02_09.cpp
(1 of 2)


```
   // processing phase
   // get first grade from user
   cout << "Enter grade, -1 to end: ";  // prompt for input
   cin >> grade;                        // read grade from user

   // loop until sentinel value read from user
   while ( grade != -1 ) {
      total = total + grade;
      gradeCounter = gradeCounter + 1;

      cout << "Enter grade, -1 to end: ";
      cin >> grade;

   } // end while

   // termination phase
   // if user entered at least one grade ...
   if ( gradeCounter != 0 ) {

      // calculate average of all grades entered
      average = static_cast< double >( total ) / gradeCounter;
```

static_cast<double>() treats total as a double temporarily (casting).

Required because dividing two integers truncates the remainder.

gradeCounter is an int, but it gets promoted to double.

Outline

fig02_09.cpp
(2 of 2)


```
      // display average with two digits of precision
      cout << "Class average is " << setprecision( 2 )
           << fixed << average << endl;

   } // end if part of if...else

   else // if no grades were entered, output appropriate message
      cout << "No grades were entered" << endl;

   return 0;   // indicate program ended successfully

} // end function main
```

fixed forces output to print in fixed point format (not scientific notation). Also, forces trailing zeros and decimal point to print.

Include <iostream>

setprecision (2) prints two digits past decimal point (rounded to fit precision).

Programs that use this must include <iomanip>

Outline

fig02_09.cpp
(2 of 2)

fig02_09.cpp
output (1 of 1)

```
Enter grade, -1 to end: 75
Enter grade, -1 to end: 94
Enter grade, -1 to end: 97
Enter grade, -1 to end: 88
Enter grade, -1 to end: 70
Enter grade, -1 to end: 64
Enter grade, -1 to end: 83
Enter grade, -1 to end: 89
Enter grade, -1 to end: -1
Class average is 82.50
```


Example Program

- Write a program that uses a sentinel-controlled while loop to read in a number of (positive) integers from the terminal and computes (and outputs) both the smallest and largest.
- User should type -1 as input to indicate there are no more numbers.

Exercise 2.16

- Drivers are concerned with the mileage obtained by their automobiles. One driver has kept track of several tankfuls of gasoline by recording miles driven and gallons used for each tankful. Develop a C++ program that uses a while structure to input the miles driven and gallons used for each tankful. The program should calculate and display the miles per gallon obtained for each tankful. After processing all input information, the program should