Ch 5. Looping

Part 4

CS 1428
Fall 2011

Jill Seaman

Lecture 15

Reading data from a file

- Loops can be used to read a list of data from a file.
- Example file:

```
84
32
99
77
52
```

Reading data from a file

- Problem: when to stop the loop?
- First entry in file could be count of number of items
  - problems: maintenance, large files
- Could use sentinal value
  - problem: may not be one, maintenance
- Want to automatically detect end of file

Using >> to detect end of file

- stream extraction operation produces a value:

```
int number;
ifstream inputFile;
inputFile.open(“numbers.txt”);

bool foundValue = (inputFile >> number);
```

- inputFile >> number:
  - tries to read a value into number
  - if it was successful, value is true
  - if it failed (nothing left to input), value is false (and the value in number does not change!)

Using the result of >>

- Example:

```
int number;
ifstream inputFile;
inputFile.open("numbers.txt");

bool foundValue = (inputFile >> number);

if (foundValue)
   cout << "The data read in was: " << number << endl;
else
   cout << "Could not read data from file." << endl;
```

- Can also use directly as relational expression:

```
if (inputFile >> number)
   ...
```

Sum all the values in the file

-
  ```
  int number;
  ifstream inputFile;
  inputFile.open("numbers.txt");

  int total = 0;

  while (inputFile >> number) {
     total = total + number;
  }

  cout << "The sum of the numbers in the file: " << total
       << endl;
  ```

- Output:

```
The sum of the numbers in the file: 344
```

Loops in C++: a summary

- Any loop can be made to work for a given problem
- while loop:
  - test at start of loop
  - generic
- for loop:
  - initialize/test/update
  - count-controlled loops
- do-while loop
  - always do at least once
  - good for repeating, simple menu processing

Nested Loops

- When one loop appears in the body of another
- For every iteration of the outer loop, we do all the iterations of the inner loop
- Example from "real life":
- A clock. For each hour in a day (24), we iterate over 60 minutes.

```
12:00       1:00       2:00       3:00
12:01       1:01       2:01       .
12:02       1:02       2:02       .
...         ...        ...        .
12:59       1:59       2:59       .
```

Print a bar graph

- Input numbers from a file. For each number, output that many asterisks (*) in a row.

```
int number;
ifstream inputFile;
inputFile.open("numbers.txt");

while (inputFile >> number) {
   for (int i = 1; i <= number; i++)
      cout << '*';
   cout << endl;
}
```

- numbers.txt:

```
8
3
6
10
```

Output:

```
********
***
******
**********
```

Calculate grades for a class

- For each student, input the test scores from the user and output the average.

```
cout << fixed << setprecision(1);

int numStudents, numTests;
cout << "How many students? ";
cin >> numStudents;
cout << "How many test scores? ";
cin >> numTests;

for (int student=1; student <= numStudents; student++) {
   float total = 0, score;
   cout << "Enter the " << numTests
        << " test scores for student " << student << endl;
   for (int test=1; test <= numTests; test++) {
      cin >> score;
      total = total + score;
   }
   float avgScore = total/numTests;
   cout << "Average for student" << student
        << " is: " << avgScore << endl;
}
```

Calculate grades for a class

- Output:

```
How many students? 3
How many test scores? 4
Enter the 4 test scores for student 1
88 90.5 92 77.5
Average for student1 is: 87.0
Enter the 4 test scores for student 2
66.5 70.5 80 86
Average for student2 is: 75.8
Enter the 4 test scores for student 3
99 93.5 80 79
Average for student3 is: 87.9
```

Breaking out of a loop

- Sometimes we want to abort a loop before it has completed.
- The `break` statement can be used to terminate the loop from within.

```
cout << "guess a number between 1 and 10" << endl;
int number;
while (true) {
   cin >> number;
   if (number == 8)
      break;
}
```

- Don't do this. It makes your code hard to read and debug.

Stopping an iteration

- Sometimes want to abort an iteration before it is done.
- The `continue` statement can be used to terminate the current iteration:

```
for (int i=1; i <= 5; i++) {
   if (i == 4)
      continue;
   cout << i << " ";
}
```

- Output:

```
1 2 3 5
```

- Don't do this either. It makes your code hard to read and debug.

Do the DVD demo program

- program 5-18 on page 293.
- What does it do? What is the pricing scheme?