Week 7: Advanced Loops

Gaddis: 5.7-12

CS 1428
Fall 2015

Jill Seaman

Loops in C++
(review)

- **while**

```
while (expression)
   statement
```

statement may be a compound statement (a block: {statements})

  - if expression is true, statement is executed, repeat

- **for**

```
for (expr1; expr2; expr3)
   statement
```

  - equivalent to:

```
expr1;
while (expr2) {
   statement
   expr3;
}
```

- **do while**

```
do
   statement
while (expression);
```

statement is executed.
if expression is true, then repeat

Q1

Common tasks solved using loops

- Counting
- Summing
- Calculating an average (the mean value)
- Read input until “sentinel value” is encountered
- Read input from a file until the end of the file is encountered

Counting
(review)

- set a counter variable to 0
- increment it inside the loop (each iteration)
- after each iteration of the loop, it stores the # of loop iterations so far

```
int number;
int count = 0;

cout << “Enter a number between 1 and 10: “;
cin >> number;

while (number < 1 || number > 10) {
   count = count + 1;
   cout << “Please enter a number between 1 and 10: “;
   cin >> number;
}
cout << count << “ invalid numbers entered “ << endl;

// Do something with number here
```

5.7 Keeping a running total (summing)

- After each iteration of the loop, it stores the sum of the numbers added so far (running total)
- set an accumulator variable to 0
- add the next number to it inside the loop

```
int days;           //Count for count-controlled loop
float total = 0.0;  //Accumulator
float miles;        //daily miles ridden

cout << "How many days did you ride your bike? ";
cin >> days;

for (int i = 1; i <= days; i++)  {
   cout << "Enter the miles for day " << i << ": ";
   cin >> miles;
   total = total + miles;
}
cout << "Total miles ridden: " << total << endl;
```

total is 0 first time through

Keeping a running total

- Output:

```
How many days did you ride you bike? 3
Enter the miles for day 1: 14.2
Enter the miles for day 2: 25.4
Enter the miles for day 3: 12.2
Total miles ridden: 51.8
```

- How would you calculate the average mileage?

Q2

5.8 Sentinel controlled loop

- sentinel: special value in a list of values that indicates the end of the data
- sentinel value must **not** be a valid value! -99 for a test score, -1 for miles ridden
- User does not need to count how many values will be entered
- Requires a "priming read" before the loop starts
  - so the sentinel is NOT included in the sum
  - the loop can be skipped (if first value is the sentinel)

Sentinel example

- Example:

```
float total = 0.0;   //Accumulator
float miles;         //daily miles ridden

cout << "Enter the miles you rode on your bike each day, ";
cout << "then enter -1 when finished. " << endl;

cin >> miles;                //priming read
while (miles != -1)  {
   total = total + miles;    //skipped when miles==-1
   cin >> miles;             //get the next one
}

cout << "Total miles ridden: " << total << endl;
```

- Output:

```
Enter the miles you rode on your bike each day,
then enter -1 when finished.
14.2
25.4
12.2
-1
Total miles ridden: 51.8
```

5.9 Which Loop to use?

- Any loop can work for any given problem
- while loop:
  - test at start of loop, good for:
  - validating input, sentinel controlled loops, etc.
- for loop:
  - initialize/test/update, good for:
  - count-controlled loops
- do-while loop
  - always do at least once, good for:
  - repeating on user request, simple menu processing

5.10 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");
inputFile >> number;  //priming read
while (number!=-1) {
   for (int i = 1; i <= number; i++)
      cout << '*';
   cout << endl;
   inputFile >> number;
}
```

- numbers.txt:

```
8
3
6
10
-1
```

Output:

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

Calculate grades for a class

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

```
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;
}
```

Inner loop

Outer loop

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
```

Q3

5.11 More File I/O

- Can test a file stream variable as if it were a boolean variable to check for various errors.
- After opening a file, if the open operation failed, the value of file stream variable is `false`.

```
ifstream infile;
infile.open("test.txt");

if (!infile) {
    cout << "File open failure!";
    return 1;  //abort program!
}
```

- Note: after ANY input operation, if it fails, the value of file stream variable will then be `false`.

Reading data from a file

- Use `fin>>x;` in a loop
- Problem: when to stop the loop?
- First entry in file could be count of number of items
  - problems: maintenance (must update it whenever data is modified), large files (might be hard to count)
- Could use sentinel value
  - problem: may not be one (someone might delete it), maintenance
- Want to <u>automatically</u> detect end of file

Using >> to detect end of file

- stream extraction operation (>>) returns `true` when a value was successfully read, `false` otherwise

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

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

- `inputFile >> num`:
  - tries to read a value into `num`
  - if it was successful, result is true (`foundValue` is true)
  - if it failed (non-number char or no more input), result is false (`foundValue` is false, but the value in `num` 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

without using a count or sentinel value

- Code:

```
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;
```

puts the priming read directly in the test expression

- numbers.txt:

```
84
32
99
77
52
```

Output:

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

5.12 Breaking and Continuing

- Sometimes we want to abort (exit) 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;
}
cout << "You got it." << endl;
```

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

Stopping a single iteration

- Sometimes we want to abort an iteration (skip to the end of loop body) before it is done.
- The `continue` statement can be used to terminate the current iteration:

```
for (int i=1; i <= 6; i++) {
   if (i == 4)
      continue;
   cout << i << " ";
}
```

- Output:

```
1 2 3 5 6
```

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

Q4

Programming Assignment 4.5
Practice only, don't submit

- Rewrite PA3, Prepare a Lab Report, so that it uses a loop to enter the data for any number of rats (ask the user to specify the number of rats before the loop starts).
  - Then rewrite it to take the input from a file (do not input the number of rats, just loop until the end of the file).
- Rewrite PA4, Calculate a Cell Phone Bill, to ask the user if they want to repeat the program after the bill and savings are output. Also put the input validation in a loop.