Lecture 03 Notes

Derek Harter

CSci 515 Spring 2014 *<2015-01-28 Wed>*

1 First Session (11 - 11:40)

1.1 Streams

The IOStream library is a new (object oriented) library, added with the C++ language, to support Input and Output to source and destination devices.

The source of input can be a keyboard, a file, or some other device. Likewise the destination of output can be to a file, to a terminal screen, or to some other device (for example you can send output into another C variable, like a string in memory).

A stream is a way of visualizing how data is transferred from the source to destination. A stream is inherently serial, the order in which you put things into the stream, is the order they will be received when they come out of the stream.

1.2 iostream header

You've already seen many examples of specifying the iostream header using

```
#include <iostream>
```

Iostream operators and objects are defined in the std namespace, thus you explicitly have to specify "std::" before using them, or include the

```
using namespace std;
```

directive.

In addition to iostream, if you want to do I/O to files, you need to include the fstream header. If you want to manipulate and format the data in/out of the stream, you need to include the iomanip header.

2 Second Session (11:45 - 12:30)

2.1 Algorithm

An algorithm is a series of unambiguous explicit steps that can be taken to solve a problem. An algorithm is a procedure for solving a problem in terms of:

1. the actions to execute and
2. the order in which the actions execute

2.2 Control Structures

Normally statements in a program execute one after the other in the order in which they are written. This is sequential execution.

- The sequence structure is built into the C language normal operation. Question: what/where is the first instruction executed in a C program?
- Selection statements: single if, double if / else, multiple switch
- Repetition statements: for, while (0 or more) and do..while (1 or more)

2.3 Multiple Selection with if..else

- can make 1 of multiple possible decisions using nested if statements.

2.4 Basic loop

- while loop repeats a set of statements 0 or more times, checking for a condition before beginning and before each repetition of the statements in the body of the loop.
- An infinite loop occurs when you don't properly change values so that the condition of the loop is eventually false.

2.5 Different types of repetition

- Our first while loop is a kind of purely logical controlled repetition. We keep repeating as long as the condition

is true, and once it becomes false we stop.

- A common variation is, Counter-Controlled repetition, when you know you need to process X specific items (we will see a better way of doing

this with a for loop next.)

- Discuss an off-by-one error in a counter-controlled loop.
- Sentenial-controlled repetition means we keep repeating until we see some condition. Our first example is really a sentenial-controlled loop, though a more typical example is to have some explicit sentenial/flag.

2.6 Increment, decrement and assignment

- C provides several assignment operators for abbreviating assignment since it is very common to need to manipulate a value and assign it back to the variable. (+=, -=, *=, etc.)
- increment and decrement are uniary operators. We can increment and decrement using pre or post increment, which means either we first incrmeent the value then use that in the expression, or we first use current value in expression then increment after.

3 Third Session (12:40 - 1:40)

3.1 for Loop for Counter-Controlled Repetition

- Counter-controlled loops are so commonly needed (especially when processing arrays of elements later), that C provides a special construct for implementing them.
- Parts of a for loop, initialization, test, manipulation.
- Give examples of counting up and counting down.
- Give example of counting with a step size.

3.2 do..while Repetition

- If we need to ensure some statements are always executed at least once, use do..while construct. Useful so we can avoid duplicating some code.

3.3 switch statement

- You may run across the switch statement. Switch provides multiple-selection (like nested if).
- It is limited, can only compare the value of a variable to a constant integral expression (e.g. our example of grades in ranges can't be accomplished using switch, we still need to use nested if for them.)
- switch statement requires use of break statements.
- break and continue statements can be useful sometimes in loops, to avoid some unnecessary repetitions of the loop.
- For example, in the prime number algorithm, we don't need to perform any more loops after we find first negative example.
- Can use a break statement. An equivalent effect can be achieved by using a flag.
- continue can be used to skip any remaining statemens in current iteration, and then start immediately with the next iteration of the loop.

3.4 Visual Studio Tips

- Turn on expert settings
- Format selection Ctrl-k Ctrl-f