CH5

5.1

Increment/ Decrement

* Prefix or postfix
  + Pre
    - Inc dec then return
  + Post
    - Return value then inc or dec
* Can be used in expressions (num++ - num--)
* Can be used with relational operators
* Must be applied to something that has location in memory
  + Not (num – num2) ++

5.2

Loop – control structure that causes statement or block to repeat

While (expression)

{

Statements;

}

While Loop

* Pre test
* Expression checked
* Statement/ block occurs
* Repeat while true
* Pretest loop

Infinite loop

* No false condition to break cycle
  + Not changing control variable
  + Having wrong condition
  + Semicolon at end of loop

5.3

Priming read – takes place just before the loop

While loop validation

* Read value
* Start while
* Is input invalid?
* Yes, get new input
* Break when valid (aka not invalid)

5.4

Counter

* loop control, increment decrement
* initialize before starting loop
* Initialize, update, test

5.5

Do while

* Posttest
* Execute once before test
* Menu driven programs
* Will execute once, even when false

Do

{

Statements;

}

While (expression);

5.6

For

* Initialize, update, test all together
* Useful when we need
  + Initialize
  + False break
  + Update at end of every iteration
* Pretest
* ( x=1 , y=1 ; x <= 5 ; x++, y++)
* Modifications
* Can remove initializations
* Can remove test (automatically T)
* Declare inside initializations (scope of variable is loop ONLY)

5.7

Running total – accumulated sum of numbers from each repetition of loop

Accumulator – variable that holds running total, sum of SERIES

5.8

Sentinels

* special value that marks the end of a list
* used to terminate input

5.9

Deciding which kind

While (input validation)

Do while (menus)

For (initialize, test, update)

5.10

Nested loops

Inner

* + Completes all repetitions

Outer

* + Goes to next loop after inner is complete
  + Total number is product of Inner repetitions and Outer repetitions

5.11

Files for Storage

Steps

* + Open
  + Use file (read, write or both)
  + Close
* Needs
  + Fstream header -> input or output
  + Ifstream -> input from
  + Ofstream -> output to
  + Define
    - File stream object
      * <Data type> <variable name>
      * ifsteam inputFile
      * ofstream outputFile
* Opening
  + Infile.open(“inventory.dat”)
    - Opening non existing INPUT will result if False Boolean
    - infile.open("test.txt");

if (!infile)

{

cout << "File open failure!";

}

* + - infile.open("test.txt");

if (infile.fail())

{

cout << "File open failure!";

}

* + Outfile.open(“report.txt”)
    - Opening a non existing OUTPUT will create the file
  + Names may include drives and path info
  + Doulbe slashes to separate folders \\
  + Object file tested as true false
* Output to file
  + Once we successfully open a file for output, we can use output file object and stream insertion operator
  + Outfile << “stuff” <<
  + Once properly opened for input, we can pull data to variables
  + Infile >> variable
* End of files
  + Steam extraction can be used to determine
  + While ( inputFile >> number)…
  + Will result false when there is nothing less
* Closing
  + Infile.close()
  + Outfile.close()
  + Can use string to store filename input from user then place in () above
  + Prior to C++ 11
    - Had to use null terminating string instead
      * Known as C-string
      * C\_str
      * String literals
      * Not STRING OBJECTS

Know pretest and posttest off the cuff

For loop

What does omitting the test condition do?

File I/O

Text files

Sequential Access

General Operation

Open file

Check it opened correctly

Use file (read in, write to)

Close file (important especially for Output files)

Data might be left in output buffer If not closed correctly

Open function is member of ifstream class

. is membership operator

Default location for I/O is located with .cpp files

C string is a string literal

Double check this ^

// for C string file names

File open failures

File locked

File not there

File open tests

Treat file open object as Boolean variable