**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**ENGE 2314 Engineering Problem Solving With C++**

**Module 4 — Repetition Control Structures, File I/O**

**Summary of Tasks**

1. Read Sections 3.4 – 3.5 in the text ……………………........................................pp. 111 – 132
2. Answer review questions 6,7,8, 9, 14-16……………….…….………………….pp. 133 – 135
3. Watch the two .avi’s in “Loops (Repetition)” …………………………………………………

…………………………………….in the Classroom AVI Files folder in Scholar’s Resources

1. Read Sections 4.1 – 4.2 in the text……………………………………………….pp. 139 – 153
2. Answer review questions 1- 5……………………………………………………pp. 176 – 177
3. Watch:

“HW4 Intro”

“File I/O”

“Rounding”……………………….in the Classroom AVI Files folder in Scholar’s Resources

1. Read “Reading Assignment Amplifications and Clarifications”…...provided in this document
2. Contribute to the forum by answering some of the “Questions about Week 4 Material,” or starting a thread on anything you’re confused about.
3. **Complete and submit homework 4……………..…………………individual graded effort**

**Reading assignment amplifications and clarifications**

Note that both the while loop and the for loop can accomplish the same thing so it is a programmer’s preference issue unless otherwise directed. In this course, as far as the homework is concerned, I do not care which you use as long as it is programmed and working correctly. Both of these repetition structures are pre-test loops – in order to enter the loop at all, you must satisfy the condition statement at the beginning. The danger, especially with the **while** loop, occurs when the programmer does not take steps to ensure that the **condition statement** has a clear answer before it is “asked”. For example – look at the following code:

int number;

while(number < 15)

{

*C++ statements*

}

Since the identifier number was not initialized prior to the opening while() statement, there is no clear answer and different compilers handle that situation differently. Some might give you a warning (but not an error) during compile time, others might not compile (that would be best). Others might compile just fine without warnings or errors and the program will run, giving wrong results (that is the absolute worst case scenario). The code below corrects the problem.

int number = 0;

while(number < 15)

{

*C++ statements*

}

The syntax of the for() loop structure actually helps prevent this kind of problem (although the programmer can certainly choose to circumvent that structure). The textual presentation of that loop on pages 116 and 117 is very clear. For example, the code for the while() loop above would simply become:

for(int number = 0; number < 15; number++)

{

*C++ statements*

}

The important thing for the programmer to be aware of is that, in the case above, the identifier number is local to the for() loop structure and attempting to output the value of that identifier outside the for() loop will cause errors. If the programmer needs to have that value available after leaving the loop, the identifier needs to be declared before starting the loop. For example:

int number;

for(number = 0; number < 15; number++)

{

*C++ statements*

}

The authors present the for() loop syntax as follows:

for(statement\_1; expression\_2; statement\_3)

{

*C++ statements*

}

and don’t mention the fact that expression\_1 and expression\_3 can actually include multiple, comma-separated, C++ statements as shown below:

for(statement\_1a, statement\_1b, statement\_1c;

expression\_2;

statement\_3a, statement\_3b, statement\_3c)

{

*C++ statements*

}

Multiple lines can be used as shown above. All of the statements beginning with statement\_1 are only executed from left to right, one time – at the beginning of the loop – so that means that multiple values can be initialized. Similarly, all of the statements beginning with statement\_3 are executed from left to right, from 0 to n times where n is the number of times the loop is executed. It should be obvious to the programmer that this practice can cause confusion if carried to the extreme so, if used at all, it is the programmer’s responsibility to ensure readability.

The do/while() loop structure is a post-test loop structure that is always executed at least once because the condition statement is not evaluated until the end of the loop.

Finally, make sure that you understand the appropriate use of the break and continue statements as discussed on page 120 before continuing on to the section dealing with loops for reading data.

The most important thing to remember when using a loop to read data is to make sure an attempt to read is performed before the condition statement for the same reason that was given in the first paragraph of this document – that is – the condition statement must have a clear answer before it is encountered.

As you read through the example provided in section 3.5 on pages 125 – 129, pay attention to the use of the **stream manipulators** used by including the header **iomanip** and using the built-in functionality provided in that header such as setf() and precision().

As you begin reading in chapter 4, notice that the authors have to introduce the object-oriented concepts of classes and inheritance to explain the syntax used to read and write files. That is precisely why I choose to have you start the use of user-defined classes early in this course.

The best way to solidify your understanding of this material is to simply follow the examples, line-by-line, and then try some code on your own.

**Complete and submit homework 4**

**PROBLEM STATEMENT:** New York City desperately needs to conserve water to support its continued population and industrial growth. The New York City Water Authority is therefore extremely interested in evaluating the impact of new water-efficient toilets that are under development by an undergraduate student team at the Virginia Tech.

The design team has placed the critical data for each model of the new toilets in a specific data file for that model.  These model constant data files are in simple ASCII text files and contain the following five floating point numbers in the order given: installation cost, new toilet liters per flush, old toilet liters per flush, typical usage per day, and number of people per toilet.  Your program must read in these constants from a file called [constants.dat](https://grader.enge.vt.edu/students/ENGE2314/constants.dat) which has been provided for your convenience. The sample ASCII text file contains five (5) floating point numbers: installation cost (147.0), new toilet liters per flush (2.0), old toilet liters per flush (14.0), typical usage per day (16.0), and number of people per toilet (3.0).

Write a program that, given a population size and the cost of water, estimates (A) the cost of replacing all existing toilets, (B) the potential water and cost savings per day, and (C) the number of days required for the new toilets to recoup their installation cost.

You must *round* the number of toilets to the closest integer, and use a class data type to hold the above constants. You may choose the names of this class and its members.

Once you have it compiling and generating the correct output as shown below, submit the source code (\*.cpp file) to the Grader.

**FORMATTING:** Declare all floating point variables as double precision floating point numbers. Define floating point constants with a period. For instance, use "**1.0**" instead of only "**1**". Make certain that there is one space on each side of the ">>>" prompts.

**SAMPLE HOMEWORK OUTPUT:** Your program will be tested with several input sets, including the one shown here in RED. The results shown here use the constants that were given in the file [constants.dat](https://grader.enge.vt.edu/students/ENGE2314/constants.dat). Other input sets may use a file with the same name but with different values for these same constants.

Please enter the population of the community >>> 5000

Please enter the estimated cost of water ($/liter) >>> 0.016

A total of 1667 new toilets would be needed.

Their total replacement cost would be $245049.

The potential flushed water savings is 320064 liters per day.

The potential cost savings approaches $5121.02 per day.

The breakeven point will be reached after 47.8516 days of usage.

You are required to read the data in from the constants.dat file and use that data. This will be worth 50% of the grade when it is checked by hand the next week. Also, be careful to write your code to work with the constants.dat file given. Your code will be tested with those exact inputs in that exact order, as well as with other data.

Note that when you’re asked to create and use a class data type to store the constants, we’re referring to a class that you create yourself as you did in HW2 and HW3; we’re not just asking you to use an ifstream object. Note also that you may include other members in your class, but your class must store the constants from the data file.

While it is easier to write this program using simple variables, you are required to use a class object (compound variable). 50% of your grade will be based on whether you define a class and instantiate, initialize and use a class object. Although your initial grade from the Grader may return 100, when we check the code by hand the next week, it can be reduced by 50 points for not following these instructions.

**IT IS NOT SUFFICIENT TO GET WORKING CODE AND THE ANSWER.**

Up to 15 points may be deducted for not using proper style. Besides those style issues discussed before, include a check to see whether the file opened properly (see the information on the fail() function and the cerr object in Chapter 4.1).

Note that this is a graded assignment and should be completed on an individual basis. Again, refer to the course policy document for further details on how this restricts how you may work with others on this and other graded assignments.

Make certain that your program matches the sample output shown online with the assignment instructions. Perfect character-by-character matching, including the same case, is required for a passing grade. The course policy document explains why this is so important.

Make certain that you “sign” the program you submit online and include the honor statement on Line 2. Programs submitted without this statement will receive a 0 at the time the homework is reviewed by hand for style and completeness. Please, be sure to log in to the Grader and sign the program using all lower-case letters. Programs submitted without “signatures” will be automatically rejected by the grading system. The course policy document describes how you will need to go about “signing” your program file.

Use the Grader to submit your assignment. Make certain that you select the correct file to submit (it should be the source code, which is the file with the lowercase .cpp extension). Once it has been submitted, you should immediately see your grade along with any errors that were found.

You will need to resubmit your assignment if there are problems with it (see the course policy document for details).

**Think About It**

As I begin another round of Honor Court hearings this semester, let me remind you what you can and cannot do with regard to working together in this course.

 We encourage you to work together via the discussion forum and via study groups on learning the material and on all the Summary of Tasks except for the coding of the homework assignment at the end.  When starting the homework, we encourage you to work through the problem, NOT THE PROGRAM, together -- ON PAPER – IN ENGLISH – so that you can solve the engineering problem step by step from beginning to end and get the correct answer as given in the assignment.

 Beyond that, there can be no discussion of code, sharing of code or copying of code from another person or from any source other than that provided to you in the assignment itself and by the instructors via the .avi’s and modules.  You may, of course, re-use your own code from previous assignments.  It is an honor violation to show someone your code, even if you worked it out completely by yourself.

 The assignments are an assessment to see if you have learned enough that you can actually program.  Just understanding someone else’s code is not enough.  If you cannot program that week’s assignment, it will be impossible for you to program any of the following assignments.  The skills in this course build.  If you cannot complete the assignment by yourself by the deadline, do not be tempted to hand in someone else’s.  Dedicate more time and continue to study the material and get help (see the policy for information on the help available in this course) until you are able to successfully finish the code and catch up with the following week’s work.