Chapter 5

Tuesday, October 3, 2017 12:38 PM

Chapter 5: Loops and Files

5.1 The Increment and Decrement Operators

The increment and decrement operators are ++ and --, respectively

Incrementation:	Decrementation:
num = num + 1;	num = num – 1;
num += 1;	num -= 1;
num++;	num--;

<u>Difference Between Postfix and Prefix Modes</u>

Previous example of num++ and num-- can be written ++num and --num;

postfix - operators come after the variable
prefix - operators come before the variable

Example of postfix:
```
num = 4;
cout << num++;
```

Outputs value of num, 4, then increments to 5

…however, writing in prefix results in the following
```
num = 4;
cout << ++num;
```

Increments to 5 then outputs value of num, 5

<u>Using ++ and -- in Mathmatical Expressions</u>

Example:
```
a = 2;
b = 5;
c = a * b++;
cout << a << " " << b << " " << c << endl;
```

Output: 2 6 10

Example:
```
a = 2;
b = 5;
c = a * ++b;
cout << a << " " << b << " " << c << endl;
```

Output: 2 6 12

Example: a = 2;
b = 5;
c = ++(a * b);
cout << a << " " << b << " " << c << endl;

Output: *Compiling error

Using ++ and -- in Relational Expressions

Example: x = 10;
if (x++ > 10)
cout << "x is greater than 10." << endl;

Output: *No output*

* change expression to (++x > 10) and output will occur

5.2 Introduction to Loops: The while Loop

loop - a control structure that causes a statement or group of statements to repeat

There are 3 loops covered:

1. The while loop
2. The do-while loop
3. The for loop

The while loop

Two important parts:

1. An expression that is tested for a true or false value
2. A statement or block of code that is repeated so long as the expression is true

```
#include <iostream>
using namespace std;


int main()
{
    // Outputs "Hello" five times
    int num = 0;

    while (num < 5)
    {
        cout << "Hello" << endl;
        num++;
    }

    system("pause");
    return 0;
}
```

```
Hello
Hello
Hello
Hello
Hello
Press any key to continue . . .
```

The while Loop is a Pretest Loop

Pretest loop - a loop where the expression is tested before each iteration

- Important as it prevents a loop from executing if the condition were to be false

Common Errors to Avoid

- Semicolon after the loop header (*null statement*)
- Lack of brackets resulting in an infinite loop

5.3 Using the while Loop for input Validation

Example provided:

```
#include <iostream>
using namespace std;


int main()
{
    // Program asks you to enter a 1 through 100
    int value;

    cout << "Please input a value between 1 and 100: ";
    cin >> value;

    while (value > 100 || value < 0)
    {
        cout << "The value was not within range.\n"
            << "Please input another value: ";
        cin >> value;
    }

    system("pause");
    return 0;
}
```

```
Please input a value between 1 and 100: 900
The value was not within range.
Please input another value: 200
The value was not within range.
Please input another value: 101
The value was not within range.
Please input another value: 100
Press any key to continue . . .
```

5.4 Counters

These are variables regularly incremented or decremented each time a loop iterates

- As previously displayed with num++

From this, you can set how much you'd want a loop to iterate

Can be concactenated with strings to prevent editing of strings over and over

```
#include <iostream>
using namespace std;


int main()
{
    int num = 1;

    while (num <= 10)
    {
        cout << "Employee #" << num << endl;
        num++;  // counter variable used here to stop loop
    }


    system("pause");
    return 0;
}
```

```
Employee #1
Employee #2
Employee #3
Employee #4
Employee #5
Employee #6
Employee #7
Employee #8
Employee #9
Employee #10
Press any key to continue . . .
```

5.5 The do-while Loop

General format: *do*
statement;
while (expression);

Alternatively...

do
{
statement;
statement;
<other statements>
} while (expression);

This is a *posttest loop* - expressions not tested until after the first iteration

```
#include <iostream>
using namespace std;


int main()
{
    int x = 1;

    do
    {
        cout << x << endl;
    } while (x < 0);        // Expression tested after block of code runs
                            // Even though x is greater than 0, the statement prior
                            // to the test expression executes.
    system("pause");
    return 0;
}
```

5.6 The for Loop

Two categories of loops:

Conditional loops - executes based on expression being true
Count-controlled loop - user determines how many times a loop iterates

Three elements to a count-controlled loop:

1. Initialization of a counter variable to a starting value
2. It must test the counter variable by comparing it to a maximum value.

   When the counter variable reaches its maximum value, the loop terminates
3. It must update the counter variable during each iteration; usually done via incrementation

General format: *for (initialization; test; update)*
statement;

With block of code: *for (initialization; test; update)*

{

block of statements;

}

for (initialization; test; update)

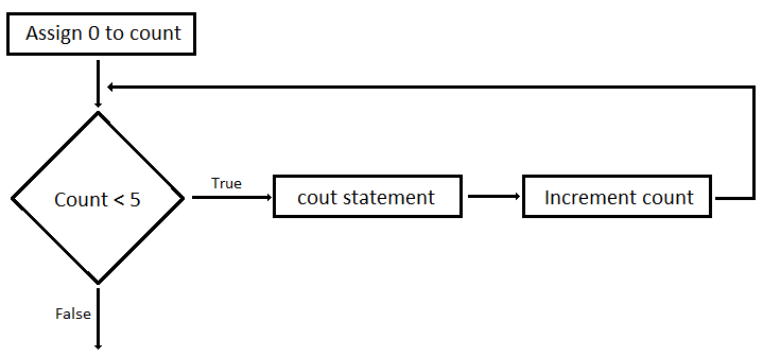
- first expression is the *initialization expression* - used to initialize a counter variable to a starting value
  - first action undertaken by for loop; initialization done once

- second expression is the *test expression* - expression that controls the execution of the loop; loop reiterates so long as it is true
  - this makes the for loop a pretest loop

- third expression is the *update expression* - it executes at the end of each iteration, it typically increments the loop's counter variable

Example program:

```
int main()
{
    for (int count = 0; count < 5; count++) {
        cout << "Hello.";
        cout << endl;
    }


    system("pause");
    return 0;
}
```



5.7 Keeping a Running Total

Running total – sum of numbers that accumulate with each iteration of a loop; initialize with 0
Accumulator – variable used to keep a running total

In the following program *total* is the accumulator and *sales* holds the running total:

```
int days;
double total = 0.0;

cout << "For how many days do you have sales figures: ";
cin >> days;

for (int count = 1; count <= days; count++)
{
    double sales;
    cout << "Enter the sales for day " << count << ": ";
    cin >> sales;
    total += sales;
}

cout << endl;
cout << fixed << showpoint << setprecision(2);
cout << "The total sales over " << days << " days are $" << total << endl;
```

```
For how many days do you have sales figures: 5
Enter the sales for day 1: 1200
Enter the sales for day 2: 3000
Enter the sales for day 3: 1000
Enter the sales for day 4: 900
Enter the sales for day 5: 800

The total sales over 5 days are $6900.00
Press any key to continue . . .
```

5.8 Sentinels

Sentinel – a special value that marks the end of a list of values
* a condition is set within a loop expression to terminate once this value is entered
* should be a value that would not be expected to be input
    - ex: inputting values for teams, inputting sales (neg. Is unexpected)

In a situation where you don't know how many values need to be entered, you can use these to end a loop

Focus on Software Engineering: Deciding Which Loop to Use

The while loop	conditional loop, pretest loop (ideal when you do not want the loop to iterate when a condition is false from the beginning), useful with unknown amount of iterations Example: input validation
The do-while loop	conditional loop, similar to while loop but is in *posttest* (ideal when you want the loop to iterate at least once) Example: menus
The for loop	pretest loop; built in expressions for initialization, testing, and updating Example: inputting several values, data transfer

5.10 Nested Loops

Points:
- Inner loop goes through all of its iterations for each iteration of an outer loop
- Inner loops complete their iterations faster than outer loops
- To get the total number of iterations of a nested loop, multiple the number of iterations of all the loops

```
// Clock program
#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
    int hours, minutes, seconds;

    // Manipulate output so that it resembles a clock (i.e. 00:00:00)
    cout << fixed << right;
    cout.fill('0');                                         // Replaces spaces with 0's

    for (int hours = 0; hours < 24; hours++)                // Iterates once every 60 iterations of middle for loop for a total of 24 times
    {
        for (int minutes = 0; minutes < 60; minutes++)      // Iterates once every 60 iterations of innermost for loop for a total of 60 times
        {
            for (int seconds = 0; seconds < 60; seconds ++) // Iterates 60 times
            {
                cout << setw(2) << hours << ":";
                cout << setw(2) << minutes << ":";
                cout << setw(2) << seconds << endl;
                if (hours > 0 && minutes == 0 && seconds == 0) {    // if statement used to pause program every hour
                    cout << hours << " hours have passed.\n";       // Displays how many hours have passed
                    cin.get();
                }
            }
        }
    }                                                       // Loop will iterate a total of 86,400 times (24 * 60 * 60)

    system("pause");
    return 0;
}
```

```
00:59:35
00:59:36
00:59:37
00:59:38
00:59:39
00:59:40
00:59:41
00:59:42
00:59:43
00:59:44
00:59:45
00:59:46
00:59:47
00:59:48
00:59:49
00:59:50
00:59:51
00:59:52
00:59:53
00:59:54
00:59:55
00:59:56
00:59:57
00:59:58
00:59:59
01:00:00
1 hours have passed.
```

5.11 Using Files for Data Storage

population.txt

population.txt

```
17700
74683
203341
573224
875538
1227000
```

population

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;

int main() {
	string filename; // for user to specify file name with data
	int population; //to hold the population
	ifstream inputFile; // the input file
	ofstream outputFile;
	outputFile.open("results.txt");
	cout << "Enter File Name: " << endl;
	getline(cin, filename);
	inputFile.open(filename);
	//Display the table headings
	outputFile << "SAN DIEGO POPULATION GROWTH\n";
	outputFile << "each * represents 10000 people\n\n";
	//Display the table data
	while (inputFile >> population) {
		population = population / 10000;
		while (population != 0) {
			outputFile << "*";
			population--;
		}
		outputFile << "\n";
	}

	inputFile.close();
	outputFile.close();
	system("pause");
	return 0;
}
```

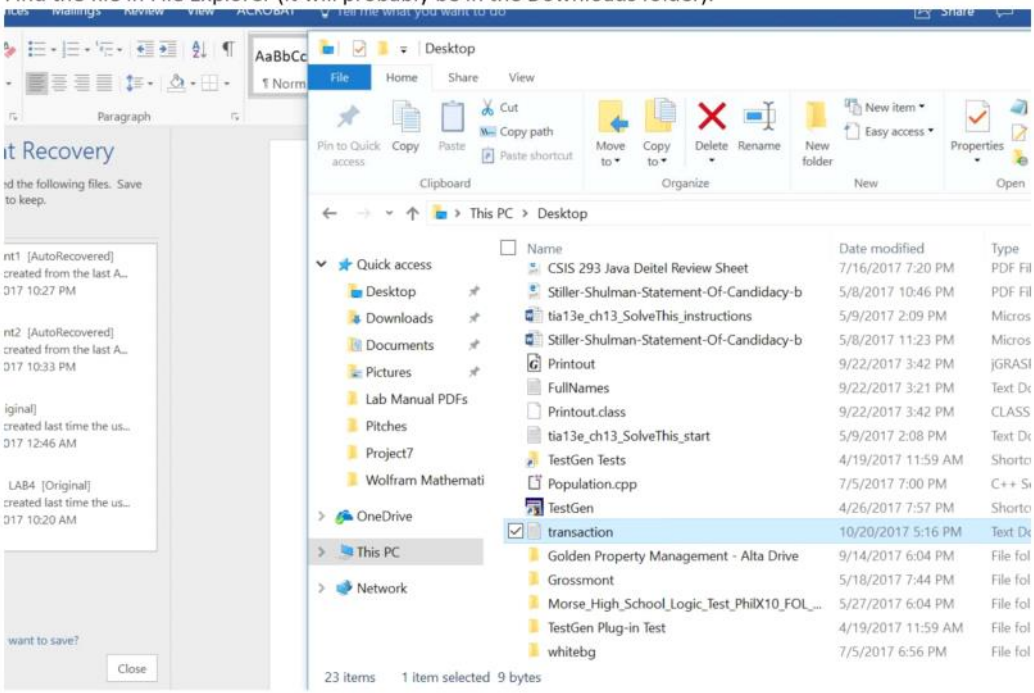
Page 1

results.txt

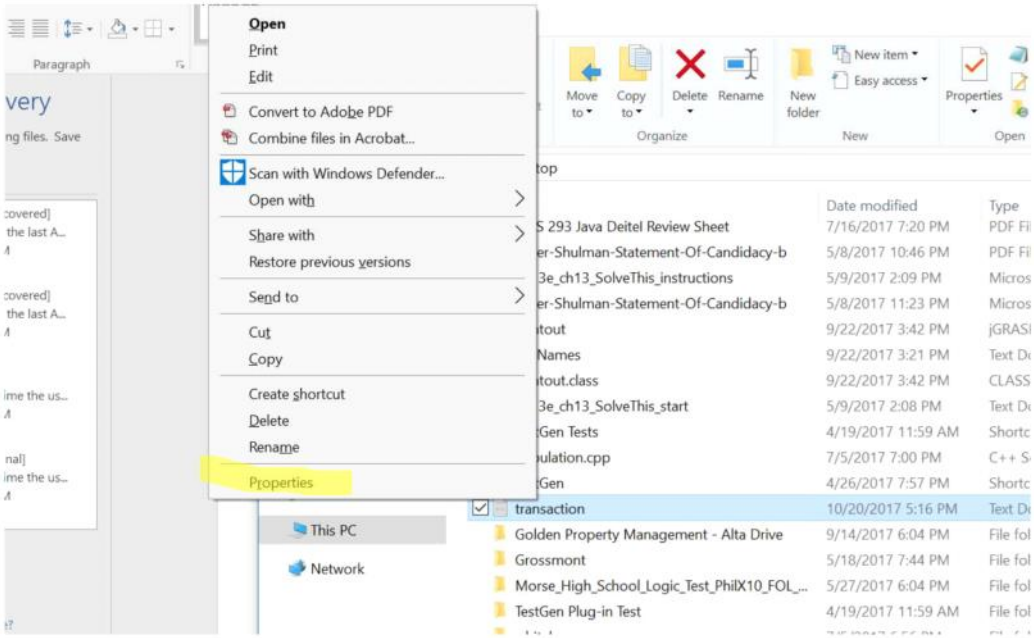
```
SAN DIEGO POPULATION GROWTH
each * represents 10000 people

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

LocatingTheInputFile

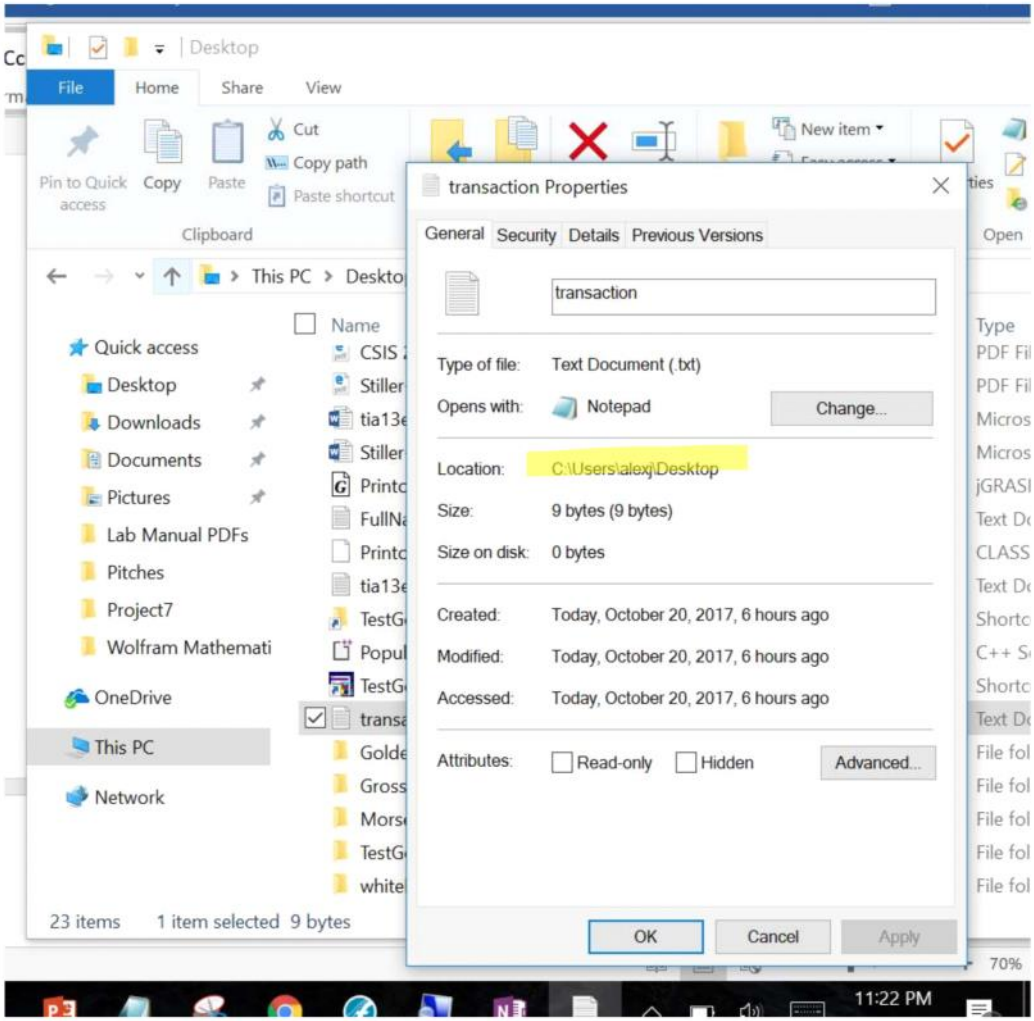
Find the file in File Explorer (it will probably be in the Downloads folder).



Right click the file and select properties



Look at the **Location** (in the example below, it would be C:\Users\alexj\Desktop. For your computer, it might be something different, like C:\Users\jessica\Downloads or something similar).



The file location will be the Location followed by the file name, such as C:\Users\alexj\Desktop\transaction.txt.

Then in your program, you will have the following lines (just make sure to replace the location below with the location of the file on your computer).

```
ifstream dataIn;     // defines an input stream for a data file
ofstream dataOut;    // defines an output stream for an output file
int quantity;        // contains the amount of items purchased
float itemPrice;     // contains the price of each item
float  totalBill;    // contains the total bill.  The price of all items

dataIn.open("C:\Users\alexj\Desktop\transaction.txt");  // This opens the file.
dataOut.open("bill.out");

dataOut<< setprecision(2) << fixed << showpoint;  // formatted output

dataIn >> quantity >> itemPrice;
```

5.12 Breaking and Continuing a Loop

Break statement - causes a loop to terminate early
* useful for terminating loops or exiting them

Continue statement - causes a loop to stop its current iteration and begin a next one

In a while loop – program jumps to expression at top of the loop
In a do-while loop – program jumps to the bottom of the loop
In a for loop – the update expression is executed and then the test expression is evaluation

Useful for programs such as ones that calculate the charges of items brought and have an item free after every nth purpose.

Optional programs to make

Student Test Average Calculator

You have been tasked with creating a program for your teacher. They want the program to ask for the number of students and how many tests were taken overall. It will then prompt the user to input a grade for each test until each test score has been recorded. Afterwards, the average will be found. This will occur for each student.

Arithmetic Operations Menu

You have been tasked with creating a simple program for a group of young students. A menu similar to the following must be included as a display:

1. Addition
2. Subtraction
3 .Multiplication
4. Division
5. [EXIT PROGRAM]

A user is to choose an option from the menu and it will perform the type of arithmetic operation on two values input by the user.

Pyramid of Stars

Create a pyramid of stars similar to the following:

```
*
**
***
****
 *****
******
*******
********
*********
********** ← Maximum of 10 asterisks
*********
```

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