Reading from and Writing to Files

Last Time

- We
  - Covered nested loops
- Today we will
  - Learn how to write C++ programs that can read from and write to files

Data Storage

- Data stored in variables is temporary
- Files are used to permanently store large amounts of data
- We will learn how to write programs that can
  - Create files
  - Write to files
  - Read from files
- This is similar to how we read from the keyboard and wrote to the screen

Libraries

- To access files you will need to include
  - <iostream>
  - <fstream>

File Variables

```
ifstream inputInfo;

ofstream outputInfo;
```

- File variables or pointers are the ways that you refer to the files you are using
  - Can specify which input/output file to use
  - May input from more than one file
  - May output to more than one file

Opening Files

```
fileptr.open("filename")
```

- Same syntax for both input and output files
- Filename is a string literal
- Example:

```
ifstream inputInfo;

inputInfo.open("input.dat", ios::out);
```

Opening Files

- `ios::out`
- Indicates the file opening mode:
  - `out: open a file for output`
  - `in: open a file for input`
  - `app: append all output to end of file`

Check File Opened Correctly

- Before we start using the file for reading or writing, we should make sure that it opened correctly

```
if(!inputInfo == true)
{
   cout << "Error opening input file ";
   exit(1);
}
```

- `Exit(1)` forces the program to exit with an error

== true

- These two statements are equivalent
  - `if(!inputInfo == true)`
  - `if(!inputInfo)`
- Even if you don't have == true in your loop, C++ will put it there by default
- This applies to all conditional statements in repetition and selection structures

Using File Variables

- Use input file variable wherever you use cin
- Examples:
  - `inputInfo >> num;`
- Output output file variable wherever you use cout
- Examples:
  - `outputInfo << num;`

Example: Writing to a File

- The following program asks the user to input numbers and writes these numbers to a file

Example

```
#include<fstream>
#include<iostream>
using namespace std;
int main()
{
   ofstream outputInfo;
   int num;
   outputInfo.open("out.dat", ios::out);
   if (!outputInfo)
   {
       cout << "*** Error opening file" << endl;
        exit (1);
   }
   cout << "Enter a number (9999 to quit): ";
   cin >> num;
   while ( num != 9999 )
   {
        outputInfo << num << " ";
        cin >> num;
   }
   return 0;
}
```

Reading from a File

- Write a program that will read in a sequence of numbers (double) from a file and calculate the sum. Assume that the last number is the trailer (-9999)

Reading Until the EOF

- It is possible to read from a file until the end is reached

```
while ( inInfo >> num )
{
     cout << num << " ";
     sum += num;
}
```


Reading Characters

- Write a program that reads in some text from a file and outputs that text to the screen
- The file contains:

```
Hello Everyone!
I'm a file that
contains some text.
```


Solution

```
ifstream inInfo;
char letter;

inInfo.open("in.dat", ios::in);
if (!inInfo)
{
    cout << "*** Error opening file" << endl;
    exit (1);
}

while ( inInfo >> letter )
{
    cout << letter;
}
cout << endl;
```


The Output

- `HelloEveryone!I'mafilethatcontainssometext.`
- What's happened?!
- All spaces, tabs, and new lines have been ignored.
- This is because >> only reads visible characters
- How can we read all characters so that the output looks exactly like the input

Solution

```
ifstream inInfo;
char letter;

inInfo.open("in.dat", ios::in);
if (!inInfo)
{
    cout << "*** Error opening file" << endl;
    exit (1);
}

while ( inInfo.get( letter ) )
{
    cout << letter;
}
cout << endl;
```

Problem

- Consider the data file below, where - indicate spaces:

```
--12--33.4
-d--12.3
-2--5
```

- What values would be assigned to the variables for each of the statements below where `inInfo` is the file variable?

```
int i,j;
double x,y;
char ch;
```

  - `inInfo >> i >> x >> y;`
  - `inInfo >> i >> j;`
  - `inInfo >> ch >> i;`
  - `inInfo >> x >> y >> ch >> x;`

Summary

- In today's lecture we covered
  - Reading to and writing from files
- Readings
  - P. 809 - 819