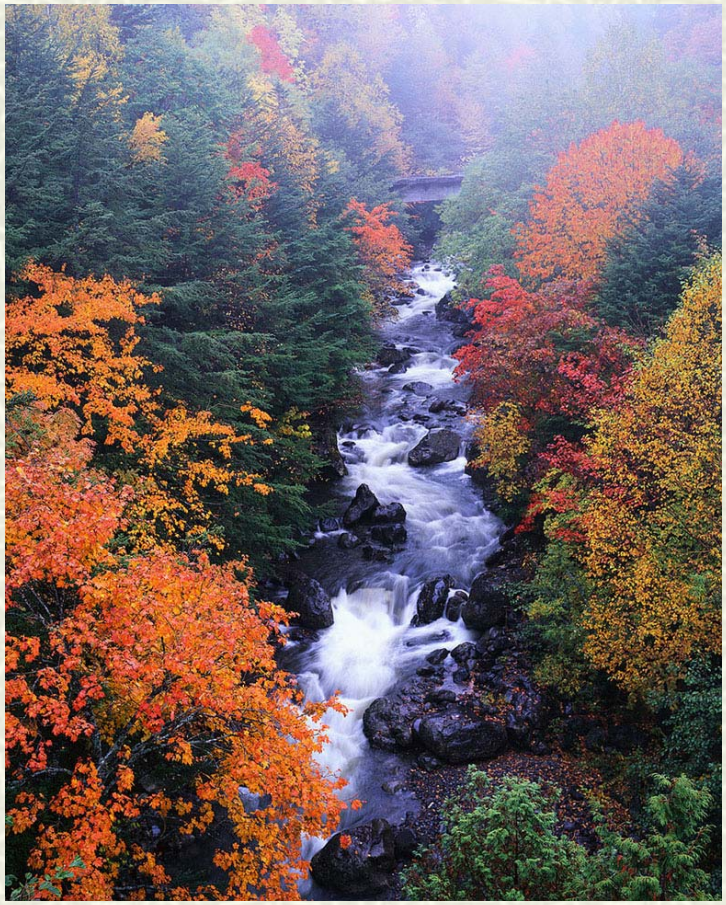
What we will learn about this week:

- Streams
- Basic file I/O
- Tools for Stream I/O
- Manipulators
- Character I/O
- Get and Put
- EOF function
- Pre-defined character functions
- Objects



I/O Streams as an Introduction to Objects and Classes

- Input refers to data flowing INTO your program.
- Output refers to data flowing OUT OF your program.
- Input can be taken from the keyboard or a file.
- Output can be sent to the screen or to a file
- Input is transferred to a C++ program from either a file or the keyboard by means of an input stream.
- Output is transferred from a C++ program to the screen or to a file by means of a output stream.
- Streams are objects defined in the Standard Library's header files <iostream> and <fstream>.

I/O to/from Files

- We will learn to take input from a file and send output to a file.
- Files allow you to store the data permanently.
- Files can be used over and over by many programs.
- Files allow for large amounts of data.

File I/O

- When a program takes data from a file, we say the program is reading from the file.
- When a program sends output to a file, we say the program is writing to the file.
- A stream is a flow of characters or other data. In C++ a stream is a special kind of variable known as an object.
- cin and cout are objects predefined in the iostream library.
- File streams, like other variables, must be declared by the programmer as needed.

4

Defining Streams

Defined in fstream library

- type ifstream – input file stream
- type ofstream – output file stream

examples:

```
ifstream in_stream;     // defines input stream
ofstream out_stream;  // defines output stream
```

Connecting Streams to Files

In the example:

in_stream.open (“infile.dat”); // infile.dat must exist on your system

out_stream.open (“outfile.dat”); // outfile.dat will be created.

This calls the function “open”.

The file stream in_stream is said to be open for reading, and the file stream out_stream is said to be open for writing.

This is new! You didn’t have to do this for cin and cout.

6

File I/O

Once we have declared the file variables, in_stream and out_stream, and connected them to files on our system, we can then take input from in_stream and send output to out_stream in exactly the same manner as we have for cin and cout.

Examples:

```
#include <fstream>
. . .
// appropriate declarations and open statements
int one_number, another_number;
in_stream >> one_number >> another_number;
. . .
out_stream << “one_number: “ << one_number
                << “another_number: << another_number;
```

File I/O

- Every file should be closed when your program is through fetching input or sending output to the file.
- This is done with the close() function.

Example:

```
in_stream.close():
out_stream.close();
```

- Note that the close function takes no arguments.
- If your program terminates normally, the system will close the arguments.
- If the program does not terminate normally, this might not happen.
- File corruption is a real possibility.
- If you want to save output in a file and read it later, the you **must** close the file before opening it the second time for reading.

Display 5.1 page 205

```
//Reads three numbers from the file infile.dat, sums the numbers,
//and writes the sum to the file outfile.dat.
//(A better version of this program will be given in Display 5.2.)
#include <fstream>

int main( )
{
    using namespace std;
    ifstream in_stream;
    ofstream out_stream;

    in_stream.open("infile.dat");
    out_stream.open("outfile.dat");

    int first, second, third;
    in_stream >> first >> second >> third;
    out_stream << "The sum of the first 3\n"
               << "numbers in infile.dat\n"
               << "is " << (first + second + third)
               << endl;

    in_stream.close( );
    out_stream.close( );

    return 0;
}
```

infile.dat

1

2

3

4

outfile.dat

The sum of the first 3 numbers in infile.dat is 6

Introduction to Classes and Objects

- Consider the code fragment:

```
#include <fstream>
. . .
ifstream in_stream;
ostream out_stream;
in_stream.open (“infile.dat”);
out_stream.open (“outfile.dat”);
  . . .
in_stream.close():
out_stream. close();
```

- Here the streams in_stream and out_stream are objects.
- An object is a variable that has functions as well as data associated with it.
- The functions open and close are associated with in_stream and out_stream, as well as the stream data and file data.

10

Member Functions

- Consider the code fragment:

```
#include <fstream>
. . .
ifstream in_stream;
ostream out_stream;
in_stream.open (“infile.dat”);
out_stream.open (“outfile.dat”);
  . . .
in_stream.close();
out_stream. close();
```

- The functions and data associated with an object are refered to as members of the object.
- Functions associated with the object are called member functions.
- Data associated with the object are called data members.

11

Summary of Classes and Objects

- An object is a variable that has functions associated with it.
- Functions associated with an object are called member functions.
- A class is a type whose variables are objects.
- The object's class determines which member functions the object has.
- Specify the function by writing
  the object name, dot operator(.), the function name.

Syntax for calling a member function:

Calling_Object.member_function(Argument_List);

12

Programming Tip

Checking that a file was opened successfully

- A very common error is attempting to open a file for reading where the file does not exist. The member function open fails then. (Except in Microsoft where it **may** create a file.)
- Your program must test for this failure, and in the event of failure, manage the error.
- Use the istream member function fail to test for open failure: in_stream.fail();

This function returns a *bool* value that is *true* if the stream is in a fail state, and *false* otherwise.

Example:

```
#include <cstdlib>  // for the predefined exit(1); library function
. . .
in_stream.open("infile.dat");
if(in_stream.fail())
{
    cout << "Input file opening failed. \n";
    exit(1);  // Predefined function quits the program.
}
```


13

The exit Statement

- The exit statement is written

```
#include <cstdlib>          // exit is defined here
using namespace std;    // to gain access to the names
exit(integer_value);        // to exit the program
```

- When the exit statement is executed, the program ends immediately.
- By convention, 1 is used as an argument to exit to signal an error.
- By convention, 0 is used to signal a normal successful completion of the program. (Use of other values is implementation defined.)
- The exit is defined in the cstdlib library header file, so any use requires
  - #include <cstdlib>
  - a directive to gain access to the names

Display 5.2 File I/O with Checks on open (1 of 2) - Page 210

```
//Reads three numbers from the file infile.dat, sums the numbers,
//and writes the sum to the file outfile.dat.
#include <fstream>
#include <iostream>
#include <cstdlib>

int main( )
{
    using namespace std;
    ifstream in_stream;
    ofstream out_stream;

    in_stream.open("infile.dat");
    if (in_stream.fail( ))
    {
        cout << "Input file opening failed.\n";
        exit(1);
    }

    out_stream.open("outfile.dat");
    if (out_stream.fail( ))
    {
        cout << "Output file opening failed.\n";
        exit(1);
    }
```

Display 5.2 File I/O with Checks on open (2 of 2)

```
    int first, second, third;
    in_stream >> first >> second >> third;
    out_stream << "The sum of the first 3\n"
               << "numbers in infile.dat\n"
               << "is " << (first + second + third)
               << endl;

    in_stream.close( );
    out_stream.close( );

    return 0;
}
```

Summary of File I/O Statements

Input in this code comes from a file with the directory name infile.dat and output goes to a file outfile.dat

- Put the following include directive in your program file:

```
#include <fstream>          // for file i/o
#include <iostream>         // for screen/keyboard i/o
#include <cstdlib>          // for exit
```

- Choose a stream name for the input the stream, such as *in_stream*. Declare it to be a variable of type *ifstream*:

```
using namespace std;
ifstream in_stream;
ofstream out_stream;
```


17

Summary of File I/O Statements

Connect each stream to a file using the *open* member function with external file name as argument. Always use the member function *fail* to test if the call succeeded.

```
in_stream.open("infile.dat");
if (in_streamin.fail());
{
  cout<< "Input file opening failed.\n";
  exit(1);
}
out_stream.open("outfile.dat");
if(out_stream.fail())
{
  cout << "Output file opening failed\n";
  exit(1);
}
```


18

Summary of File I/O Statements

- Use the stream in_stream to get input from the file infile.dat just like you use cin input from the keyboard.

  Example:

  ```
  in_stream >> some_variable >> another_variable;
  ```

- Use the stream out_stream to send output to the file outfile.dat just like you use cout output to the screen.

  Example:

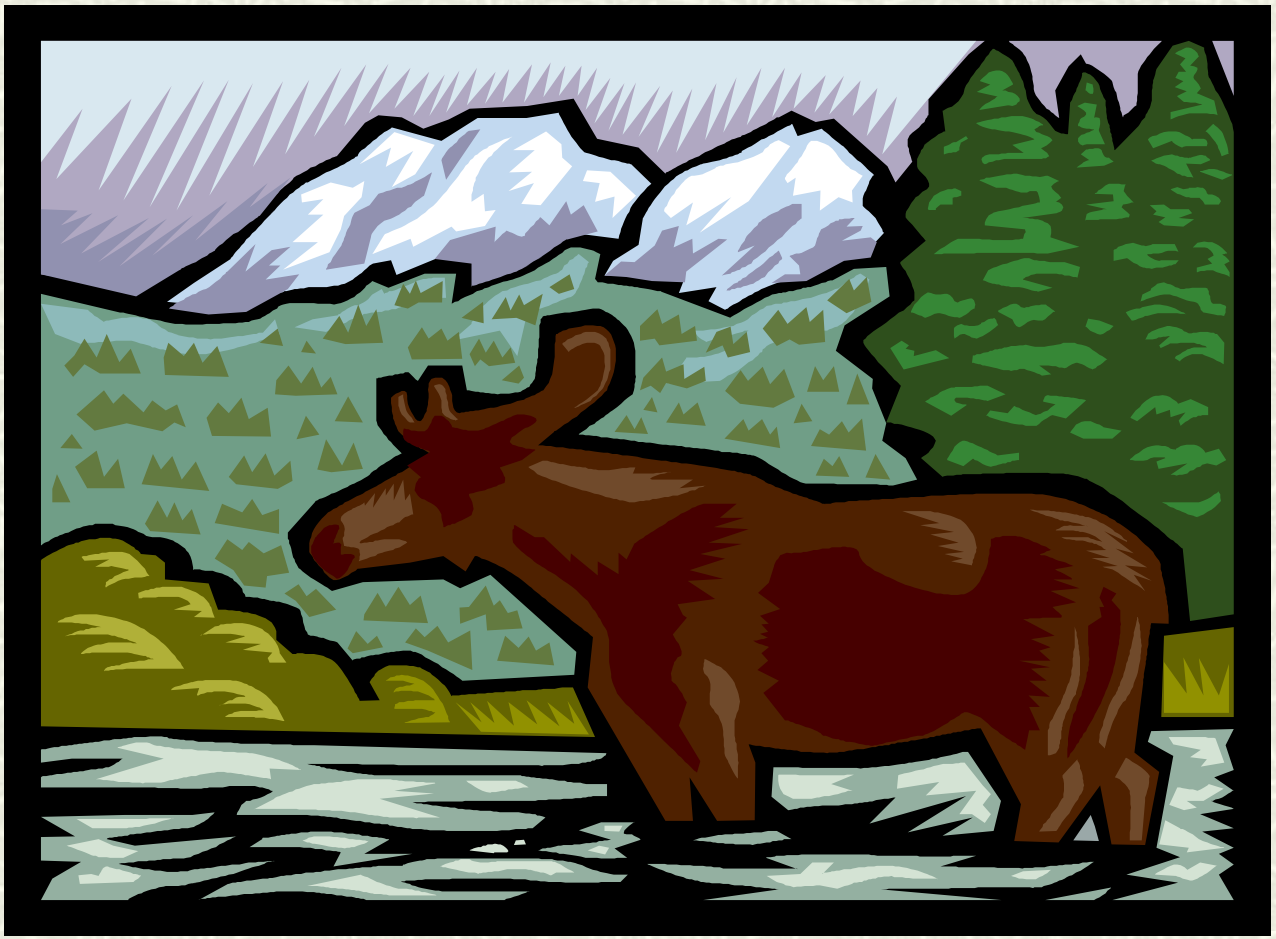
  ```
  out_stream << "some_variable = " << some_variable << endl;
  ```

- Close the streams using the member function close:

  ```
  in_stream.close();
  out_stream.close():
  ```


19

Take a Break



Object notation for streams

(our “magic” formula for decimal point output)

- out_stream.setf(ios::fixed);
- out_stream.setf(ios::showpoint);
- out_stream.precision(2);

member functions for output streams

- precision – number of significant digits (or decimal places, depending on compiler)
- width – how many (minimum) spaces in output (line up output)
- unsetf – unset flags that are set with setf

(see next slide)

set flags (Display 5.5 page 220)

- ios::fixed – decimal point rather than exponential notation
- ios::scientific – floating point numbers written in e-notation (opposite of fixed)
- ios::showpoint – always include the decimal point even if there are no significant digits after the decimal point (2.0 rather than 2)
- ios::showpos – write plus sign in front of positive numbers
- ios::right – right justify value
- ios::left – left justify value (opposite of ios::right)

Manipulators – functions called in a non-traditional way

- setw – same as width
- setprecision – same as precision

Examples of manipulators

```
out_stream << “Start” << setw(4)  << 10
        << setw(4) << 20 << setw(10) << 30;
```

outputs:

```
Start  10  20        30
```

Streams can be arguments to functions

But must be call-by-reference

void make_neat (ifstream& messy, ofstream& neat)

End-of-file

- read returns true if a value is read
- read returns false if no value is read (eof)

```
while (messy_file >> next)                    //check for eof
{
   cout << setw(field_width ) << next << endl;
   neat_file << setw(field_width ) << next << endl;
}

//(messy_file >> next) both action and Boolean
   expression
```

get and put – input and output **characters**

- works sort of like insertion and extraction operators for type char **without automatic skipping of blanks** or understanding of escape sequences.
- takes everything in as a character.
- can detect end of line by checking value of character

Syntax for get and put

input_stream.get(char_variable);

output_stream.put(char_variable);

Example of get

```
char var;                    // character variable
do
{
cin.get(var);                // get a character
cout << var;                 // output the character
}
while (var != '\n')          // check for end of line
```

EOF member function

returns true if end of file

```
instream.get(next);
while (! instream.eof() )          // check for eof
{
   cout << next;
   instream.get(next);
}
```

Predefined Character functions

Pre-defined character functions:

- toupper – converts to upper case
- tolower – converts to lower case
- isupper – returns true if character is uppercase
- islower – returns true if character is lower case
- isalpha – returns true if character is a letter (a – z or A – Z)
- isdigit – returns true if character is a number (0 – 9)
- isspace – returns true id character is a blank or newline

Quirks in Microsoft

If you want to use the input file twice , you must first close it then open it a second time.
If you don't re-open it, the second time you try to use it, it will be at the end of the file.
However, in Microsoft, you won't be able to re-open the same stream. So use two stream variables in_stream and in_stream2 (or fin and fin2 or whatever).
Don't forget to close the first stream before opening the second!!!

Namespace

*Book's note on 'using' directives – just put it after the includes, but realize that when you want to use something other than namespace, you will have to make the using statements local to the functions.

Programming with I/O streams

declare a stream:

```
ifstream numbers_in;
```

connect a stream to a file:

```
numbers_in.open(A://data1.txt);
```

verify that the file opened correctly:

```
if  numbers_in.fail();
{
        cout << “Error message/n”;
        exit (1);
}
```

close the stream:

```
numbers_in.close();
```

Reading Assignment

Arrays

Sections 10.1 - 10.2
pages 493 – 524

Take a Break!