C++ for Science and Engineering COSC3000/6000

2018 Spring Semester

Part V
I/O Streams

- I/O refers to program input and output
  - Input is delivered to your program via a stream object
  - Input can be from
    - The keyboard
    - A file
  - Output is delivered to the output device via a stream object
  - Output can be to
    - The screen
    - A file

Objects

- Objects are special variables that
  - Have their own special-purpose functions
  - Set C++ apart from earlier programming languages

1 Streams and Basic File I/O

- Files for I/O are the same type of files used to store programs
- A stream is a flow of data.
  - Input stream: Data flows into the program
    - If input stream flows from keyboard, the program will accept data from the keyboard
    - If input stream flows from a file, the program will accept data from the file
  - Output stream: Data flows out of the program
    - To the screen
    - To a file

1.1 cin And cout Streams

- **cin**
  - Input stream connected to the keyboard
- **cout**

    - Output stream connected to the screen

- **cin** and **cout** defined in the **iostream** library
    - Use include directive: #**include** <**iostream**>
- You can declare your own streams to use with files.

1.2 Why Use Files?

- Files allow you to store data permanently!
- Data output to a file lasts after the program ends
- An input file can be used over and over
    - No typing of data again and again for testing
- Create a data file or read an output file at your convenience
- Files allow you to deal with larger data sets

1.3 File I/O

- Reading from a file
    - Taking input from a file
    - Done from beginning to the end (for now)
        - No backing up to read something again (OK to start over)
        - Just as done from the keyboard
- Writing to a file
    - Sending output to a file
    - Done from beginning to end (for now)
        - No backing up to write something again(OK to start over)
        - Just as done to the screen

1.4 Stream Variables

- Like other variables, a stream variable…
    - Must be declared before it can be used
    - Must be initialized before it contains valid data
        - Initializing a stream means connecting it to a file
        - The value of the stream variable can be thought of as the file it is connected to
    - Can have its value changed
        - Changing a stream value means disconnecting from one file and connecting to another

1.5 Streams and Assignment

- A stream is a special kind of variable called an **object**
    - Objects can use special functions to complete tasks
- Streams use special functions instead of the assignment operator to change values

1.6 Declaring An Input-file Stream Variable

http://www.cplusplus.com/reference/fstream/ifstream/

- Input-file streams are of type **ifstream**
- Type **ifstream** is defined in the **fstream** library
  - You must use the include and using directives
    ```
    #include <fstream>
    using namespace std;
    ```
- Declare an input-file stream variable using
  ```
  ifstream in_stream;
  ```

1.7 Declaring An Output-file Stream Variable

http://www.cplusplus.com/reference/fstream/ofstream/

- Ouput-file streams of are type **ofstream**
- Type **ofstream** is defined in the **fstream** library
  - You must use these include and using directives
    ```
    #include <fstream>
    using namespace std;
    ```
- Declare an output-file stream variable using
  ```
  ofstream out_stream;
  ```

1.8 Connecting To A File

- Once a stream variable is declared, connect it to a file
  - Connecting a stream to a file is opening the file
  - Use the open function of the stream object
    ```
    in_stream.open("infile.dat");
    ```

1.9 Using The Input Stream

- Once connected to a file, the input-stream variable can be used to produce input just as you would use **cin** with the extraction operator
  - Example:
    ```
    ifstream in_stream;
    in_stream.open("infile.dat");
    int one_number, another_number;
    in_stream >> one_number >> another_number;
    ```

1.10 Using The Output Stream

- An output-stream works similarly to the input-stream
  ```
  ofstream out_stream;
  out_stream.open("outfile.dat");
  out_stream << "one number = "
                << one_number
                << "another number = "
                << another_number;
  ```

1.10.1 sample13.cpp

```
#include <iostream>
#include <fstream>
#include <iomanip>

using namespace std;

int main (int argc, char *argv[]) {
  ofstream ofile;

  ofile.open("data01.dat");

  for (int i = 1 ; i <= 10 ; i++){
    for (int j = 1 ; j <= 10 ; j++){
      ofile << i << setw(4) << j <<endl;
    }
  }
  ofile.close();

  return 0;
}
```

1.11 External File Names

- An External File Name...
  - is the name for a file that the operating system uses
    - **infile.dat** and **outfile.dat** used in the previous examples
  - is the "real", on-the-disk, name for a file
  - needs to match the naming conventions on your system
  - usually only used in the stream's open statement
  - once open, referred to using the name of the stream connected to it.

1.12 Closing a File

- After using a file, it should be closed
  - This disconnects the stream from the file
  - Close files to reduce the chance of a file being corrupted if the program terminates abnormally
- It is important to close an output file if your program later needs to read input from the output file
- The system will automatically close files if you forget as long as your program ends normally

1.13 Objects

- An object is a variable that has functions and data associated with it
  - **in_stream** and **out_stream** each have a function named open associated with them
  - **in_stream** and **out_stream** use different versions of a function named open
    - One version of open is for input files
    - A different version of open is for output files

1.14 Member Functions

- A **member function** is a function associated with an object
  - The **open** function is a member function of **in_stream** in the previous examples
  - A different **open** function is a member function of **out_stream** in the previous examples

1.15 Objects and Member Function Names

- Objects of different types have different member functions
  - Some of these member functions might have the same name
- Different objects of the same type have the same member functions

1.16 Classes

- A type whose variables are objects, is a class
  - **ifstream** is the type of the in_stream variable (object)
  - **ifstream** is a class
  - The class of an object determines its member functions
  - Example:

    ```
    ifstream in_stream1, in_stream2;
    ```

    * **in_stream1.open** and **in_stream2.open** are the same function but might have different arguments

1.17 Class Member Functions

- Member functions of an object are the member functions of its class
- The class determines the member functions of the object
  - The class **ifstream** has an open function
  - Every variable (object) declared of type **ifstream** has that **open** function

1.18 Calling a Member Function

- Calling a member function requires specifying the object containing the function
- The **calling object** is separated from the member function by the **dot operator**
- Example:

  ```
  in_stream.open("infile.dat");
  ```

1.18.1 Member Function Calling Syntax

- Syntax for calling a member function:
  Calling_object .Member_Function_Name(Argument_list);

1.19 Errors On Opening Files

- Opening a file could fail for several reasons
  - Common reasons for open to fail include
    * The file might not exist
    * The name might be typed incorrectly
- May be no error message if the call to open fails
  - Program execution continues!?

1.19.1 Catching Stream Errors

- Member function **fail**, can be used to test the success of a stream operation
  - **fail** returns a boolean type (true or false)
  - **fail** returns true if the stream operation failed

1.19.2 Halting Execution

- When a stream open function fails, it is generally best to stop the program
- The function **exit**, halts a program
  - **exit** returns its argument to the operating system
  - **exit** causes program execution to stop
  - **exit** is NOT a member function
- Exit requires the include and using directives

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

1.19.3 Using fail and exit

- Immediately following the call to open, check that the operation was successful:

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

1.19.4 sample14.cpp

```
#include <iostream>
#include <fstream>
using namespace std;
int main (int argc, char *argv[]) {
  ifstream ifile;
  ifile.open("data01.dat");
  if (ifile.fail()){
    cout << "file not found" << endl;
    return 0;
  }

  int i,j;
  int ti = 0;
  int tj = 0;
  while(1){
    ifile >> i >> j;
    if (ifile.eof()) break;
    ti += i;
    tj += j;
  }
  ifile.close();
  cout << "totail i =" << ti << endl;
  cout << "totail j =" << tj << endl;

  return 0;
}
```

1.20 Stream object as input argument

- Streams can be arguments to a function
  - The function's formal parameter for the stream must be **call-by-reference**

```
void sum_data(ifstream& ifile){
    int i,j;
    int ti = 0;
    int tj = 0;
    while(1){
        ifile >> i >> j;
        if (ifile.eof()) break;

        ti += i;
        tj += i;
    }

    cout << "totail i =" << ti << endl;
    cout << "totail j =" << tj << endl;
}
```

1.21 Appending Data (optional)

- Output examples so far create new files
  - If the output file already contains data, that data is lost
- To append new output to the end an existing file
  - use the constant **ios::app** defined in the **iostream** library:

    ```
    outStream.open("important.txt", ios::app);
    ```
- If the file does not exist, a new file will be created

1.22 File Names as Input (optional)

- Program users can enter the name of a file to use for input or for output
- Program must use a variable that can hold multiple characters
  - A sequence of characters is called a string
  - Declaring a variable to hold a string of characters:

    ```
    char file_name[16];
    ```

    * file_name is the name of a variable
    * Brackets enclose the maximum number of characters + 1
    * The variable file_name contains up to 15 characters

1.22.1 Using A Character String

```
char file_name[16];
cout << "Enter the file_name ";
cin >> file_name;
ifstream in_stream;
in_stream.open(file_name);
if (in_stream.fail( ) )
{
        cout << "Input file opening failed.\n";
        exit(1);
}
```