File I / O in C++

Sukhbir Tatla
sukhbir.tatla@senecacollege.ca

Using Input/Output Files

A computer file

- is stored on a secondary storage device (e.g., disk);
- is permanent;
- can be used to
  - provide input data to a program
  - or receive output data from a program
  - or both;
- should reside in Project directory for easy access;
- must be opened before it is used.

C++ Files and Streams

- C++ views each files as a sequence of bytes.
- Each file ends with an ***end-of-file*** marker.
- When a file is ***opened***, an object is created and a stream is associated with the object.
- To perform file processing in C++, the header files **<iostream>** and **<fstream>** must be included.

General File I/O Steps

1. Include the header file fstream in the program.
2. <fstream> includes <ifstream> and <ofstream>
3. Declare file stream variables.
4. Associate the file stream variables with the input/output sources.
5. Open the file
6. Use the file stream variables with >>, <<, or other input/output functions.
7. Close the file.

Using Input/Output Files

- ***stream*** **-** a sequence of characters
  - interactive (iostream)
    - **cin** - input stream associated with **keyboard.**
    - **cout** - output stream associated with **display**
  - file (fstream)
    - **ifstream** - defines new input stream (normally associated with a file).
    - **ofstream** - defines new output stream (normally associated with a file).

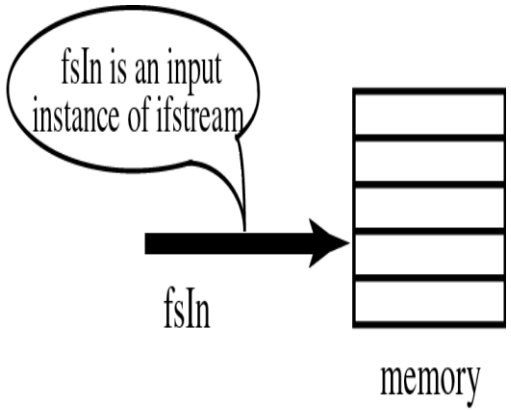
Stream I/O Library Header Files

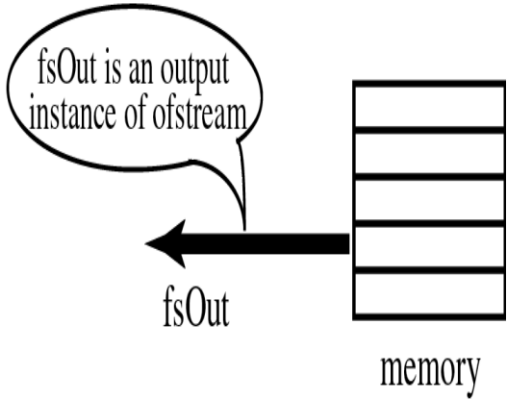
- Note: There is no ".h" on standard header files : <fstream>
- iostream -- contains basic information required for all stream I/O operations
- fstream -- contains information for performing file I/O operations

C++ streams

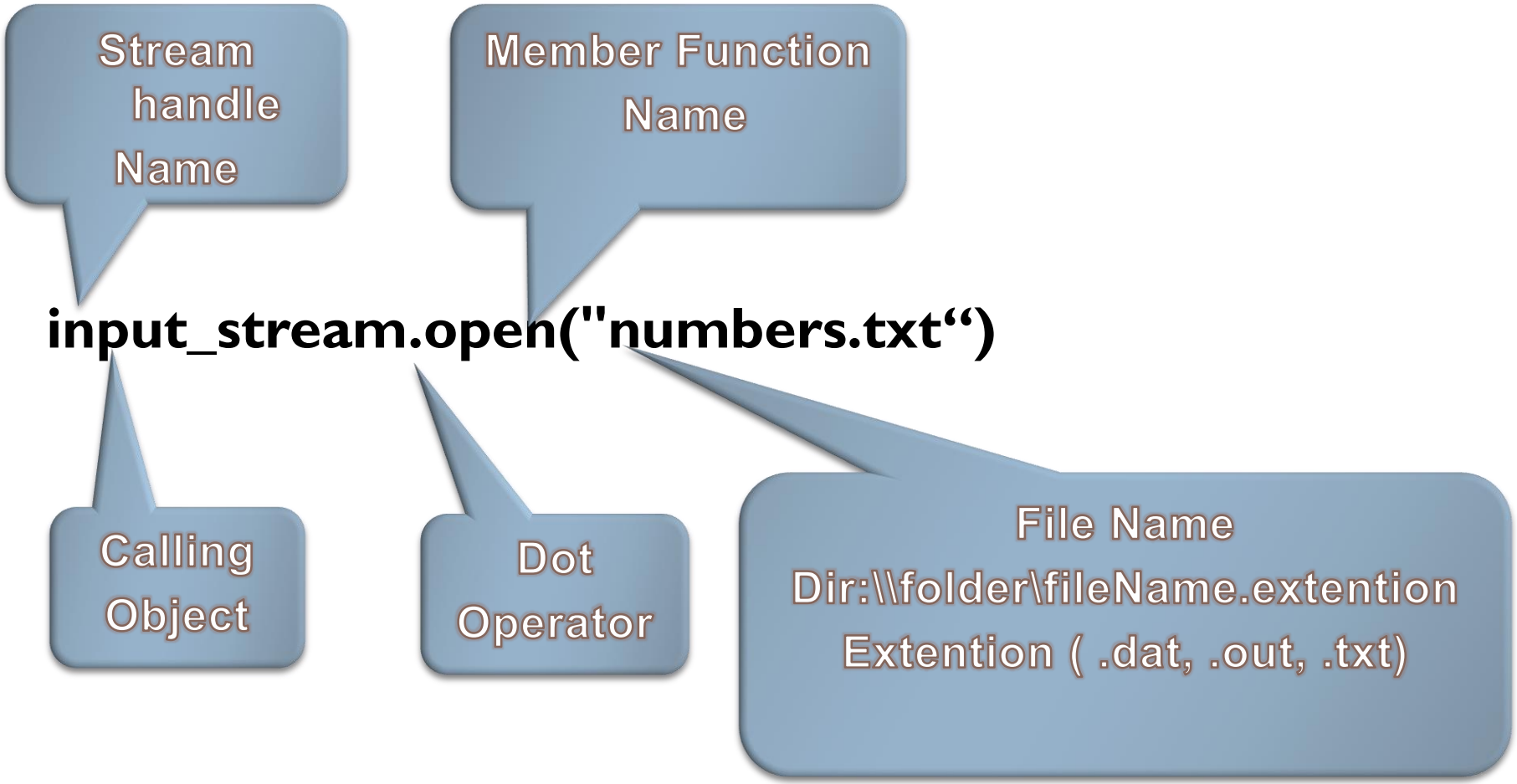
```
#include <iostream>
#include <fstream>
int main ()    {
      ifstream  fsIn;//input
      ofstream fsOut; // output
      stream both //input &
output
      //Open the files
      fsIn.open("prog1.txt");
//open the input file
      fsOut.open("prog2.txt");
//open the output file
//Close files
     fsIn.close();
     fsOut.close();
      return 0;
}
```

```
#include <fstream.h>

int main (void)
{
// Local Declarations
   ifstream     fsIn;

   ofstream     fsOut;
   .
   .
   .

} // main
```





Object and Member Functions



Open()

- Opening a file associates a file stream variable declared in the program with a physical file at the source, such as a disk.
- In the case of an input file:
  - the file must exist before the open statement executes.
  - If the file does not exist, the open statement fails and the input stream enters the fail state
- An output file does not have to exist before it is opened;
  - if the output file does not exist, the computer prepares an empty file for output.
  - If the designated output file already exists, by default, the old contents are erased when the file is opened.

Validate the file before trying to access

First method

By checking the stream variable;

```
If ( ! Mystream)
{
Cout << "Cannot open file.\n ";
}
```

Second method

By using bool is_open() function.

```
If ( ! Mystream.is_open())
{
Cout << "File is not open.\n ";
}
```

Testing for Open Errors

First method

By checking the stream variable;

```
dataFile.open("cust.dat",
   ios::in);
if (!dataFile)
{
   cout << "Error opening
   file.\n";
}
```

Second method

By using fail() function.

```
dataFile.open("cust.dat",
   ios::in);
if (dataFile.fail())
{
   cout << "Error opening
   file.\n";
}
```

Testing for Open Errors

```
dataFile.open("cust.dat", ios::in);
if (!dataFile)
{
   cout << "Error opening file.\n";
}
```

File I/O Example: Open the file with validation

First Method (use the constructor)

```
#include <fstream>
using namespace std;
int main()
{
//declare and automatically
open the file
ofstream outFile(“fout.txt");
// Open validation
if(! outFile) {
Cout << “Cannot open file.\n ”;
return 1;
}
return 0;
}
```

Second Method (use Open function)

```
#include <fstream>
using namespace std;
int main()
{
//declare output file variable
ofstream outFile;
// open an exist file fout.txt
outFile.open(“fout.txt”);
// Open validation
if(! outFile.is_open() ) {
Cout << “Cannot open file.\n ”;
return 1;
}
return 0;
}
```

More Input File-Related Functions

- **ifstream fsin;**
- **fsin.open(const char[] fname)**
  - connects stream **fsin** to the external file ***fname.***
- **fsin.get(char character)**
  - extracts next character from the input stream **fsin** and places it in the character variable ***character.***
- **fsin.eof()**
  - tests for the end-of-file condition.

File I/O Example: Reading

Read char by char

```
#include <iostream>
#include <fstream>
int main()
{       //Declare and open a text file
        ifstream openFile("data.txt");
         char ch;
        //do until the end of file
        while( ! OpenFile.eof() )
        {
                OpenFile.get(ch); // get one character
                cout << ch;   // display the character
        }
        OpenFile.close(); // close the file
        return 0;
}
```

File I/O Example: Reading

Read a line

```
#include <iostream>
#include <fstream>
#include <string>
int main()   {      //Declare and open a text file
      ifstream openFile("data.txt");
      string line;
      while( ! openFile.eof () ){
            //fetch line from data.txt and put it in a
string
            getline(openFile, line);
            cout << line;       }  //End of While Loop
      openFile.close(); // close the file
      return 0;
}
```

More Output File-Related Functions

- **ofstream fsOut;**
- **fsOut.open(const char[] fname)**
  - connects stream **fsOut** to the external file ***fname.***
- **fsOut.put(char character)**
  - inserts character ***character*** to the output stream **fsOut.**
- **fsOut.eof()**
  - tests for the end-of-file condition.

File I/O Example: Writing

First Method (use the constructor)

```
#include <fstream>
using namespace std;
int main()
{     /* declare and automatically open the
file*/
      ofstream outFile("fout.txt");
//behave just like cout, put the word into  the
file
      outFile << "Hello World!";
      outFile.close();
      return 0;
}
```

File I/O Example: Writing

Second Method (use open function)

```
#include <fstream>
using namespace std;
int main()
{     // declare output file variable
      ofstream outFile;
      // open an exist file fout.txt
      outFile.open("fout.txt");

//behave just like cout, put the word into  the file
      outFile << "Hello World!";
      outFile.close();
      return 0;
}
```

Practice

- REALITY CHECK (Week 11)

- Question #1 (Word problem)
- Question #2 (Word problem)

File Open Modes

Name	Description
ios::in	Open file to read
ios::out	Open file to write
ios::app	All the data you write, is put at the end of the file. It calls ios::out
ios::ate	All the data you write, is put at the end of the file. It does not call ios::out
ios::trunc	Deletes all previous content in the file. (empties the file)
ios::nocreate	If the file does not exists, opening it with the open() function gets impossible.
ios::noreplace	If the file exists, trying to open it with the open() function, returns an error.
ios::binary	Opens the file in binary mode.

File Open Modes

```
#include <fstream>
int main(void)
{
      ofstream outFile("file1.txt",
            ios::out);
      outFile << "That's new!\n";
      outFile.close();
      return 0;
}
```

If you want to set more than one open mode, just use the **OR** operator- **|**. This way:

ios::ate | ios::binary

Summary of Input File-Related Functions

```
#include <fstream>
ifstream fsIn;
```

- **`fsIn.open(const char[] fname)`**
  - connects stream **`fsIn`** to the external file ***`fname`***.
- **`fsIn.get(char& c)`**
  - extracts next character from the input stream **`fsIn`** and places it in the character variable ***`c`***.
- **`fsIn.eof()`**
  - tests for the end-of-file condition.
- **`fsIn.close()`**
  - disconnects the stream and associated file.
- **`fsIn >> c;`** //Behaves just like cin

Summary of Output File-Related Functions

```
#include <fstream>
ofstream fsOut;
```

- **`fsOut.open(const char[] fname)`**
  - connects stream **`fsOut`** to the external file ***`fname`***.
- **`fsOut.put(char c)`**
  - inserts character ***`c`*** to the output stream **`fsOut`**.
- **`fsOut.eof()`**
  - tests for the end-of-file condition.
- **`fsOut.close()`**
  - disconnects the stream and associated file.
- **`fsOut << c;`** //Behaves just like cout

File format

- In C++ files we (read from/ write to) them as a stream of characters

- What if I want to write or read numbers ?

Example writing to a file

```
#include <iostream>
#include <fstream>
using namespace std;
void  main() {
  ofstream outFile;
  // open an exist file number.txt
  outFile.open("number.txt",ios::app);
  if ( ! outFile.is_open() )
  {    cout << " problem with opening the file ";    }
  else
  {    outFile << 200 << endl;
       cout << "done writing" <<endl;
  }
  outFile.close();
}
```

Example Reading from a file

```
#include <iostream>
#include <fstream>
#include <string>
#include <sstream>
using namespace std;
void main()    {       //Declare and open a text file
        ifstream INFile("number.txt");
        string line;
        int total=0;
        while(! INFile.eof())         {
                getline(INFile, line);
                //converting line string to int
                stringstream(line) >> total;
                cout << line  <<endl;
                cout <<total +1<<endl;        }  // End of while loop
        INFile.close(); // close the file
 }
```

File position pointer

- <istream> and <ostream> classes provide member functions for repositioning the *file pointer* (the byte number of the next byte in the file to be read or to be written.)
- These member functions are:
- ***seekg*** (seek get) for istream class
- ***seekp*** (seek put) for ostream class

Examples of moving a file pointer

- inClientFile.seekg(0) - repositions the file get pointer to the beginning of the file
- inClientFile.seekg(n, ios:beg) - repositions the file get pointer to the n-th byte of the file
- inClientFile.seekg(n, ios:end) - repositions the file get pointer to the n-th byte from the end of file
- inClientFile.seekg(0, ios:end) - repositions the file get pointer to the end of the file

- **The same operations can be performed with <ostream> function member seekp.**

Member functions tellg() and tellp().

- Member functions **tellg** () and **tellp** () are provided to return the current locations of the get and put pointers, respectively.

```
long location = inClientFile.tellg();
```

- To move the pointer relative to the current location use ios:cur

- inClientFile.seekg (n, ios:cur) - moves the file get pointer n bytes forward.

Detecting the End of a File

- The eof() member function reports when the end of a file has been encountered.

```
if (inFile.eof())
    inFile.close();
```

Detecting the End of a File: Example

```
// This program uses the file stream object's eof() member
// function to detect the end of the file.
#include <iostream>
#include <fstream>
using namespace std;
void main()
{
      fstream dataFile;
      char name[81];
      dataFile.open("demofile.txt", ios::in);
      if (!dataFile)
      {
            cout << "File open error!" << endl;
            return;
      }
```

Detecting the End of a File: Example

```
//Program Continues…
        cout << "File opened successfully.\n";
        cout << "Now reading information from the file.\n\n ";
        dataFile >> name;   // Read first name from the file
        while (!dataFile.eof())
        {
                cout << name << endl;
                dataFile >> name;
        }
        dataFile.close();
        cout << "\nDone.\n";
} //End of main
```

Note on eof()

- In C++, "end of file" doesn't mean the program is at the last piece of information in the file, but beyond it.

- The eof() function returns true when there is no more information to be read.

Practice

- REALITY CHECK (Week 11)
- Question #3 (Word problem)