File I/O in C++

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.

General File I/O Steps

- Include the header file fstream in the program.
- Declare file stream variables.
- Associate the file stream variables with the input/output sources.
- Open the file
- Use the file stream variables with >>, <<, or other input/output functions.
- 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).

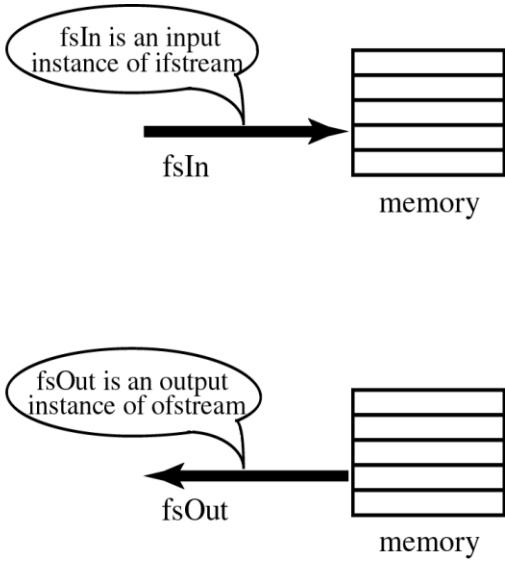
Stream I/O Library Header Files

- 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

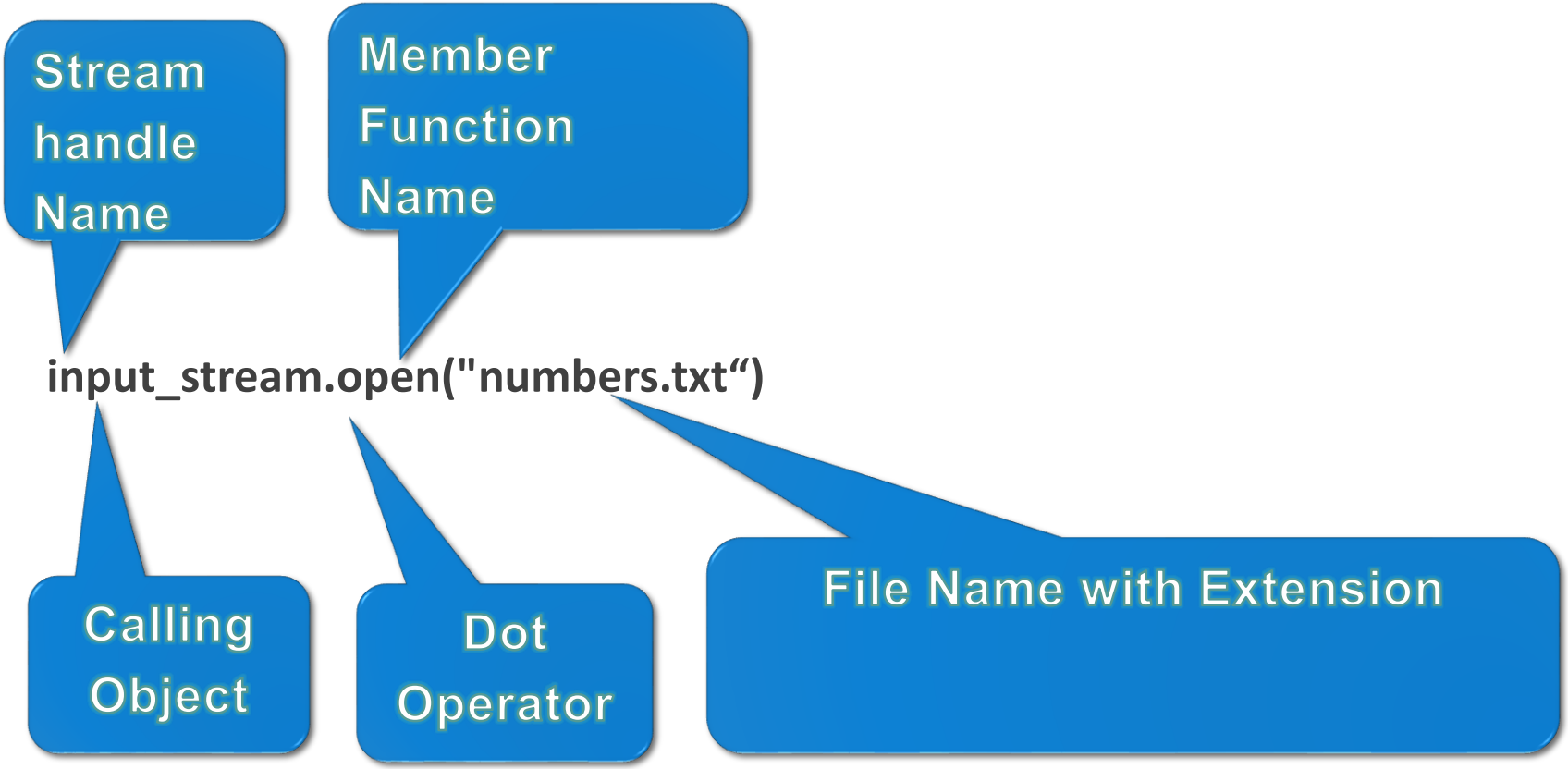
C++ streams

```
//Add additional header files you use

#include <fstream>
using namespace std;
int main ()
{ /* Declare file stream variables such as
 the following  */
ifstream  fsIn;//input
ofstream fsOut; // output
//Open the files
fsIn.open("prog1.txt"); //open the input file
fsOut.open("prog2.txt"); //open the output
file
//Code for data manipulation

//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 ";
}
```

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;
}
```

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;
}
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;
}
```

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;
}
```

File Open Mode

Mode	Description
`ios::app`	*Append* all output to the end of the file.
`ios::ate`	Open a file for output and move to the end of the file (normally used to append data to a file). Data can be written *anywhere* in the file.
`ios::in`	Open a file for *input*.
`ios::out`	Open a file for *output*.
`ios::trunc`	*Discard* the file's contents (this also is the default action for `ios::out`).
`ios::binary`	Open a file for binary, i.e., *nontext*, input or output.

File Open Mode

```
#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

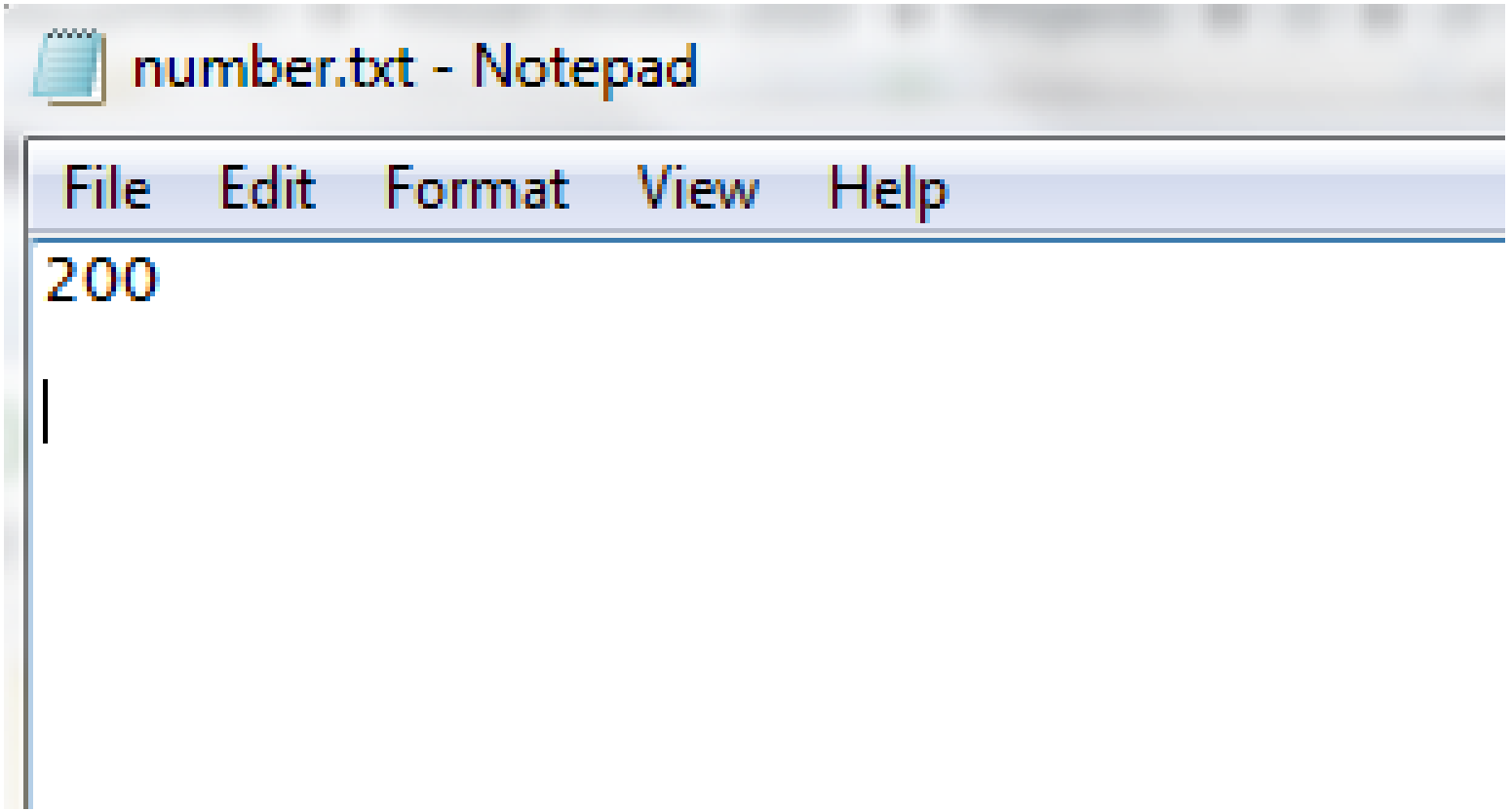
- `#include <fstream>`
- `ofstream fsOut;`
- `fsOut.open(const char[] fname)`
  - connects stream `fsOut` to the external file ***`fname`***.
- `fsOut.put(char c)`
  - inserts character ***`c`*** to the output stream `fsOut`.
- `fsOut.eof()`
  - tests for the end-of-file condition.
- `fsOut.close()`
  - 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 ?

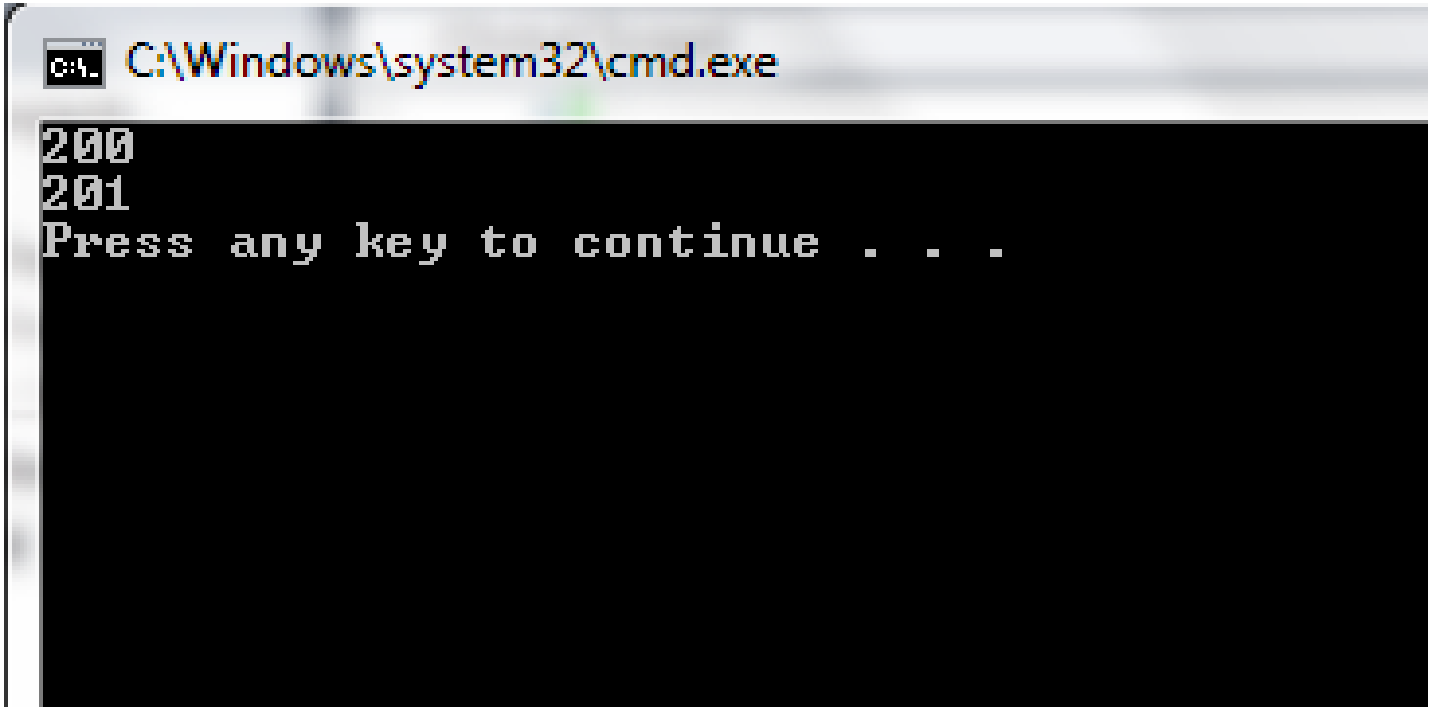
Example writing to file

```
#include <iostream>
#include <fstream>
using namespace std;
void  main()
{
ofstream outFile;
// open an exist file numbers.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();
}
```


number.txt - Notepad
File Edit Format View Help
200

Example Reading from 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;
INFile.ignore();}
INFile.close(); // close the file
 }
```


C:\Windows\system32\cmd.exe
200
201
Press any key to continue . . .