



I/O Streams and File I/O

- Streams: A **stream** is a flow of characters.
- An input **stream** is a flow of characters into the program.
- An output **stream** is a flow of characters out of the program.
- `cin` is a predefined input stream (defined in `<iostream>`)
- `cout` is a predefined output stream. (defined in `<iostream>`)
- Streams are defined in the streams library header file, `<iostream>` and implemented in the `iostream` library.
- What if you want characters to come from a file?
 - You need an *input file stream*.
- What if you want your output characters to go into a file?
 - You need an *output file stream*.
- Both are called *file streams*.
- File streams are like `cin` and `cout`, except that the flow is to and from a *file*. They share many of the same properties and functions.
- File streams are a special kind of I/O stream. C++ defines file streams in a library called `fstream`, whose header file is `<fstream>`

How to use a file named `infile.txt` as input and a file named `outfile.txt` as output:

1. Include the library header file in your program:
`#include <fstream>`
2. Declare an input file stream with whatever name you choose in your main program (I like `fin`) as follows:

```
int main()  
{  
    ifstream  fin;  
    ofstream  fout;  
    ...  
}
```



3. Before your program attempts to read any input or write any output, open the file streams and associate them with the actual files:

```
fin.open("infile.txt");
fout.open("outfile.txt");
```

Note that this won't work if the files are not in the same directory as the running program. In that case you have to specify the absolute or relative path name.

4. Instead of using statements with `cin` for input and `cout` for output, use `fin` in place of `cin`, and use `fout` in place of `cout`:

```
int first, second;
fin >> first >> second;
fout << "the sum is " << first + second;
```

5. Before your program's closing **return** statement, close all open files:

```
fin.close();
fout.close();
```

```
// s.close() closes a file. This must be done or else the chars that
// were put there will not be saved.
```

Note that these are void functions with no arguments.

Example.

```
#include <fstream.h>

int main()
{
    ifstream fin;    // declares an object of type ifstream
    ofstream fout;  // declares an object of type ofstream

    // ifstream and ofstream are defined in fstream.h

    fin.open("d:\infile.txt");
    fout.open("d:\outfile.txt");

    // for any stream object, s, s.open(<filename>) opens filename
    // and connects the stream s to it so that chars can flow between
    // them.

    int num1, num2, num3;
    fin >> num1 >> num2 >> num3;
    fout << "The sum is " << num1+num2+num3 << endl;
```

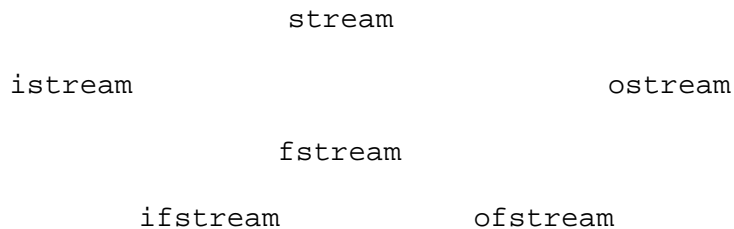


```
    fin.close();
    fout.close();
    return 1;
}
```

istream is a linear connection from a program to a source of characters

ostream is a linear connection from a program to a repository of characters.

fstreams have all of the properties and functions of streams, and extra functions and properties too.



Practical issues with file streams.

1. Testing if a file stream function call was successful:

```
in_stream.open("d:\mydata");

if ( in_stream.fail() )
{
    cout << "Could not open d:\mydata.\n";
    exit(1); // 1 usually means error
}
```

Need to include `<stdlib.h>` to use the exit call.

2. The fail function works on any stream.

3. Do not need prompts when reading from files

4. File names as input

```
char filename[16];
cout << "Enter the absolute path to the file:";
cin >> filename;
fin.open(filename);
```