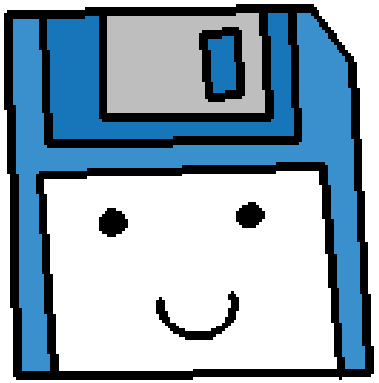
File I/O

Input & Output Streams



cin and cout

We've been using streams to get input and output already, via the **`cin`** and **`cout`** commands.

There are also **`ifstream`** and **`ofstream`** objects that we can use to stream out-of and in-to text files.

If it's a text format file we can write it out easily --

cin and cout

We've been using streams to get input and output already, via the **`cin`** and **`cout`** commands.

There are also **`ifstream`** and **`ofstream`** objects that we can use to stream out-of and in-to text files.

If it's a text format file we can write it out easily --

.csv

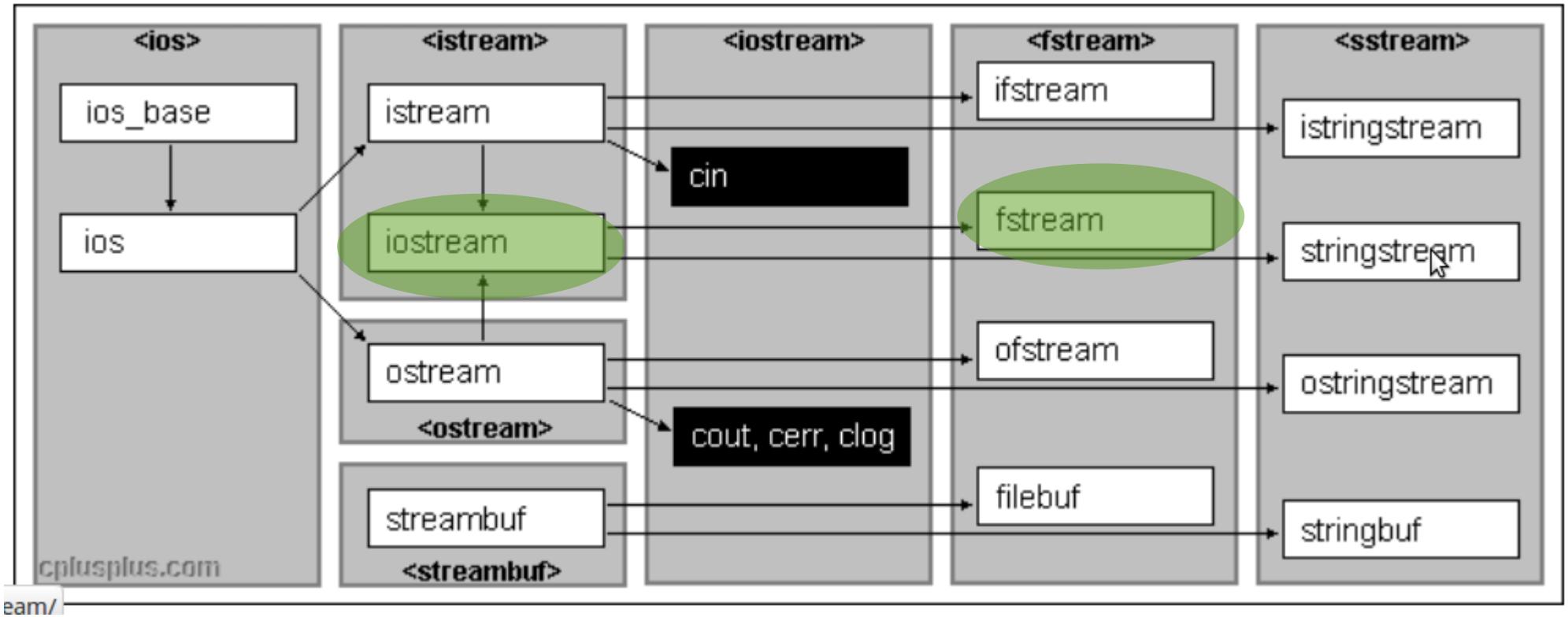
.txt

Even .cpp
If you really wanted to...

.html

cin and cout

cin and **ifstream** are part of the same family, as are **cout** and **ofstream**.



From http://www.cplusplus.com/reference/iolibrary/

ifstream and ofstream

```
#include <iostream>
#include <fstream>
using namespace std;

int main()
{
    string word;
    ofstream output( "Hello.txt" );

    while ( true )
    {
        cout << "Enter a word: ";
        cin >> word;

        cout << word << endl;
        output << word << endl;
    }

    output.close();

    return 0;
}
```

To use **ifstream (input-file-stream)** and **ofstream (output-file-stream)** You need to `#include <fstream>`

Then you can declare an object of type **ofstream** or **ifstream**.

ofstream

```
#include <iostream>
#include <fstream>
using namespace std;

int main()
{
    string word;
    ofstream output( "Hello.txt" );

    while ( true )
    {
        cout << "Enter a word: ";
        cin >> word;

        cout << word << endl;
        output << word << endl;
    }

    output.close();

    return 0;
}
```

Outputting to a text file looks similar to outputting to the screen, except to write to the screen we use

```
cout << "Text";
```

But with our file stream, we use the file object that we've created.

```
outputFile << "Text";
```

We declare a **ofstream** variable, and pass in a file name to open.

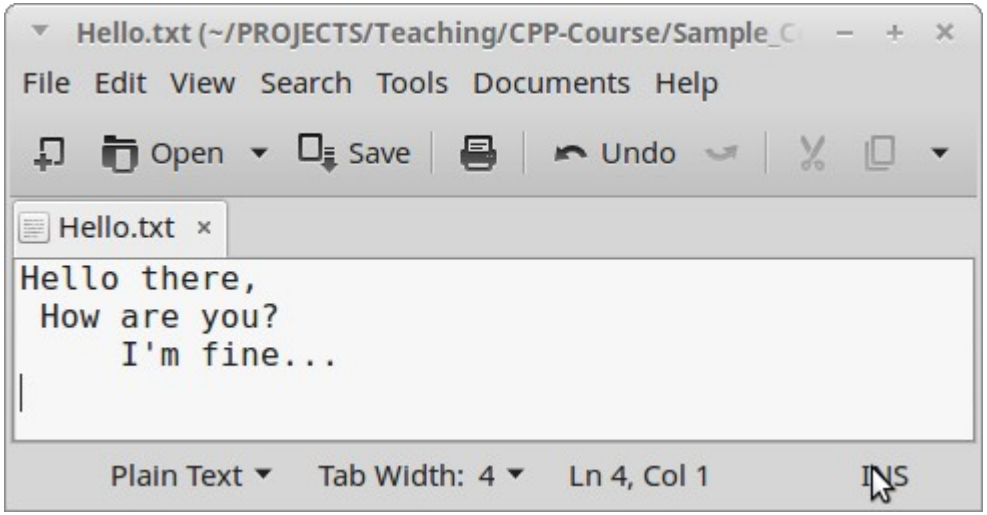
We also need to make sure to close the file once we are done with it.

ofstream

We can also use **`endl`** with our file output stream, as well as **`\t`** and **`\n`**.

```
output << "Hello there,\n How are you?";
output << endl << "\t I'm fine..." << endl;
```

Then we'll just get a simple text file:



ofstream

But we can also output other file formats, if we follow that file's standards:

```
int main()
{
    string word;
    ofstream output( "WordList.csv" );

    // Header
    output << "ENGLISH,ESPERANTO" << endl;

    // Content
    output << "Red,Rugha" << endl;
    output << "Orange,Orangha" << endl;
    output << "Yello,Flava" << endl;
    output << "Green,Verda" << endl;
    output << "Blue,Blua" << endl;
    output << "Purple,Purpura" << endl;

    output.close();

    return 0;
}
```

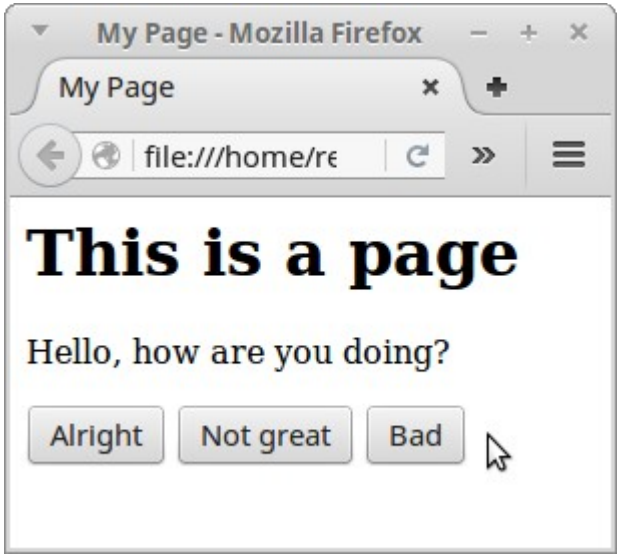
CSV (comma-separated-value) files can be opened in Excel or LibreOffice:

	A	B
1	ENGLISH	ESPERANTO
2	Red	Rugha
3	Orange	Orangha
4	Yello	Flava
5	Green	Verda
6	Blue	Blua
7	Purple	Purpura
8		

ofstream

Many file types are just plaintext if you open them up in **notepad** or something similar...

```
int main()
{
    string word;
    ofstream output( "Webpage.html" );

    output << "<html>" << endl;
    output << "<head><title>My Page</title></head>" << endl;
    output << "<body>" << endl;
    output << "<h1>This is a page</h1>" << endl;
    output << "<p>Hello, how are you doing?</p>" << endl;
    output << "<input type='button' value='Alright'/>" << endl;
    output << "<input type='button' value='Not great'/>" << endl;
    output << "<input type='button' value='Bad'/>" << endl;
    output << "</body>" << endl;
    output << "</html>" << endl;

    output.close();

    return 0;
}
```



ofstream

You could even do this if you really wanted to...

```
int main()
{
    string word = "WHY?!?!";
    ofstream output( "Program.cpp" );

    output << "#include <iostream>" << endl;
    output << "using namespace std;" << endl;
    output << "int main() {" << endl;

    for ( int i = 0; i < 100; i++ )
    {
        output << "cout << \"" << word << "\" << endl;" << endl;
    }

    output << "return 0;" << endl;
    output << "}" << endl;

    output.close();

    return 0;
}
```

Which creates another .cpp file...

```
int main()
{
    string word = "WHY?!?!";
    ofstream output( "Program.cpp" );

    output << "#include <iostream>" << en
    output << "using namespace std;" << e
    output << "int main() {" << endl;

    for ( int i = 0; i < 100; i++ )
    {
        output << "cout << \"" << word <<
    }

    output << "return 0;" << endl;
    output << "}" << endl;

    output.close();

    return 0;
}
```

Terminal

File Edit View Search Terminal Help

```
  GNU nano 2. File: Program.cpp

#include <iostream>
using namespace std;
int main() {
cout << "WHY?!?!" << endl;
cout << "WHY?!?!" << endl;
cout << "WHY?!?!" << endl;
cout << "WHY?!?!" << endl;
cout << "WHY?!?!" << endl;
cout << "WHY?!?!" << endl;
cout << "WHY?!?!" << endl;
cout << "WHY?!?!" << endl;
cout << "WHY?!?!" << endl;
cout << "WHY?!?!" << endl;
cout << "WHY?!?!" << endl;
cout << "WHY?!?!" << endl;
cout << "WHY?!?!" << endl;

^G Get ^O Writ^R Read^Y Prev^K Cut ^C Cur
^X Exit^J Just^W Wher^V Next^U UnCu^T To S
```

ifstream

On the flip side, you can use **ifstream** to read in a text file.

You can load any plaintext file, just like the ones we wrote out, but actually *reading* them is a little more difficult.

Once you load in the file, you have to know how to **parse** it, to get useful information out for your program to use.

(That means, you have to write the logic on how to actually read the file or find a library that does it for you!)

ifstream

Input text file

input.txt (~/PROJECTS/Teaching/CPP-Course/Sample_C

File Edit View Search Tools Documents Help

Open Save Undo

input.txt

```
This is a text file, yep!
```

... Plain Text Tab Width: 4 Ln 1, Col 1 INS

Program code

```
int main()
{
    string word;
    ifstream infile( "input.txt" );

    string words[3];
    infile >> words[0];
    infile >> words[1];
    infile >> words[2];

    infile.close();

    cout << words[0] << endl
         << words[1] << endl
         << words[2] << endl;

    return 0;
}
```

Program output

Simple Inpu

```
This
is
a
```

ifstream

Similar to using **`cin`** to get input from the user to store into a variable, we can use the **ifstream** variable to load text from a text file into variables.

But we need to declare an **ifstream** object, open a text file, and make sure to close it at the end.

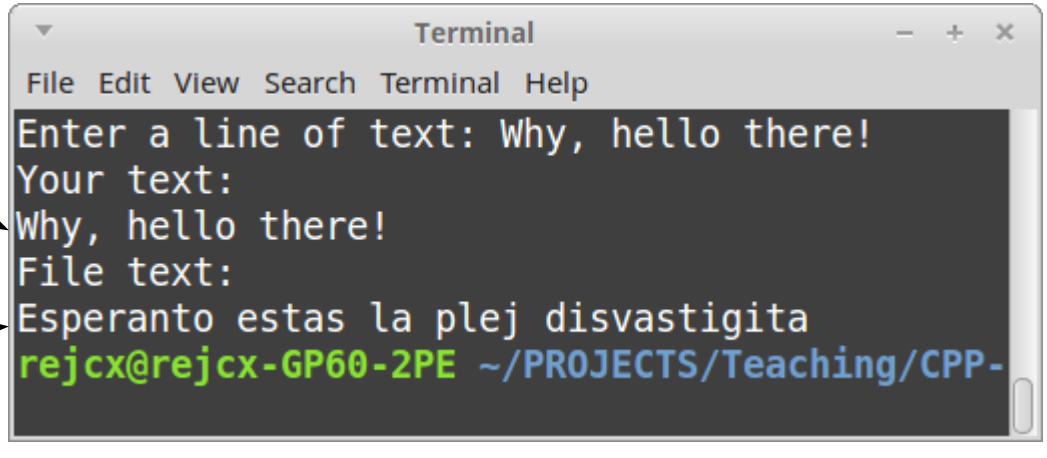
Program code

```
int main()
{
    string word;
    ifstream infile( "input.txt" );

    string words[3];
    infile >> words[0];
    infile >> words[1];
    infile >> words[2];

    infile.close();

    cout << words[0] << endl
         << words[1] << endl
         << words[2] << endl;

    return 0;
}
```

ifstream

Just like we can use the **`getline`** function to get a line of text from the user (keyboard), we can use **`getline`** on an ifstream object to get a line of text from a file.

```
int main()
{
    // Using getline with cin

    string userText;
    cout << "Enter a line of text: ";
    getline( cin, userText );

    // Using getline with an ifstream
    string fileText;

    ifstream infile( "mytext.txt" );
    getline( infile, fileText );
    infile.close();

    cout << "Your text: " << endl << userText << endl;
    cout << "File text: " << endl << fileText << endl;

    return 0;
}
```



ifstream

Why might you want to read a text file?

To store data when your program closes, so you can resume with the same information later!

To analyze text files on a system!

To read in a file, add formatting, and write it back out!

To read in a set of commands to execute!

ifstream

Also...

```
input >> buffer;        // string
input >> usernameCount; // integer
```

You can use the stream operator to store text from a file in data-types like strings, ints, floats, and any others that support input from streams – just line **`cin`**.

```
while ( input >> buffer )
{
    usernames[i] = buffer;
    i++;
}
```

Read every word of the file until there are no more words to read.

You can also keep loading in one word at a time (delimited by whitespace – spaces, tabs, new lines, etc.) until the end of the file with a while loop like this.