Streams and Files in C++ Chapter 14

CS 2212 INTERMEDIATE PROG/C++

LPAGE

Streams and Files

CHAPTER 14

Files

We will use input text files and output text files.

- A file is data that is stored on your hard disk or some other device.
- Files have many properties that the programmer would rather not deal with:
  - Its physical location.
  - Its actual size
  - Decoding the access permissions.
  - How much has been processed so far, and how much remains?
  - Streams were invented to take care of all those troublesome properties of a file.

Streams in C

A **stream** is an **abstraction** that makes files or system input/output less complex to use.

- An **input stream** is a data structure that permits you to bring data from a file or the system into your program .
- An **output stream** is a data structure that permits you to write your data out to permanent storage or the console.
- It is implemented as a **structure that contains** buffers, error flags, pointers, and other parts. So all the needed information about the stream is packaged together.
- In **standard C**, a stream has type FILE* (a pointer at the file stream)

Predeclared Streams in C and C++

Predeclared streams:

C: stdin, stdout, stderr

C++: cin, cout, cerr, clog

The C++ streams cin and cout **share buffers with the corresponding C streams** stdin and stdout.

The streams **stderr and cerr** are **unbuffered**.

The new C++ stream, clog is used for **logging transactions**.

Flushing the Output Stream in C

The output stream is buffered, so when you switch from output (such as displaying a menu) to input (such as reading user input), make sure you flush (print to screen, file, etc), the contents of the stream’s buffer or the sequence will get out of order.

In C:

```
printf(" Show prompt with instructions:");
fflush(stdout);
//makes sure that you see the prompt before input.
//Note: DO NOT USE WITH stdin!
scanf("%i%i",&num1,&num2);
```

Libraries to include for File I/O in C

When using text files in C the tools you need are in the standard libraries:

```
#include <stdio.h>
#include <stdlib.h>
```

Working with Files in C

OPENING FILES

FILE IO

Opening a File in C

To read from or write to a file, you must first open the file and check to be sure the file was properly opened. Here are two correct approaches. The second one is better, clearer, and shorter.

```
FILE* dataFile = fopen("mydata.txt", "r"); // input.
if (! dataFile) {
puts ("Cannot open myfile.txt for input.");
exit(1);
}
//********************  This is better, clearer and shorter  **********************
#define OUTF "myout.txt"
FILE* outFile = fopen( OUTF, "w"); // output.
if (! outFile)
  fprintf (stderr, "\nCannot open %s for output.", OUTF);
```

Making code more maintainable

```
#include <stdio.h>

#include <stdlib.h>

Usomg namespace std;

#define OUTF "myout.txt"    //make the file name a constant, edit once.
int main(){

FILE* outFile = fopen( OUTF, "w"); // output.

if (! outFile)

  fprintf (stderr, "\nCannot open %s for output.", OUTF);

…

}
```

File I/O in C

The functions for file-input, fscanf(), and file-output, fprintf(), are like scanf and printf.

Here, we read a series of pairs of numbers from one file and write them in reverse order on the other.

```
double x, y;
int status; // To store the result of fscanf.
for(;;) { //infinite loop, must break inside the look to end it
  if( feof( dataFile ) ) break; //ends loop when EOF is reached
  status = fscanf( dataFile, "%lg %lg", &x, &y );
  if( status == 0 ) fprintf(stderr, "\nFile format error" );
  // Otherwise, process the data you just read.
  fprintf( outFile, "%g %g", y, x );
}
```

The program will terminate with a “File format error" comment if alphabetic data is read when numeric data was expected. Mac users I think it goes in the debug folder in the project. PC users put it in the project folder (not a subfolder).

Reading a String in C

Reading a string is more complex than reading a number because a string can be any length. Following is a simple way to read a string safely into an array and echo-print it.

```
char wordlist[10][16]; //row then column
for( int row=0; row<10; ++row) {
if( feof( dataFile ) ) break; //ends loop when EOF is reached
fgets( wordlist[row], 16, dataFile );
fprintf( outFile, "%s", wordlist[row] );
}
```

The fgets() function reads characters from the file into the char array **until it reaches the given limit minus 1.** Reading **also stops when a newline character is found, or end-of-file occurs.** A null terminator is always stored after the last input character.

Manipulating Input and Output

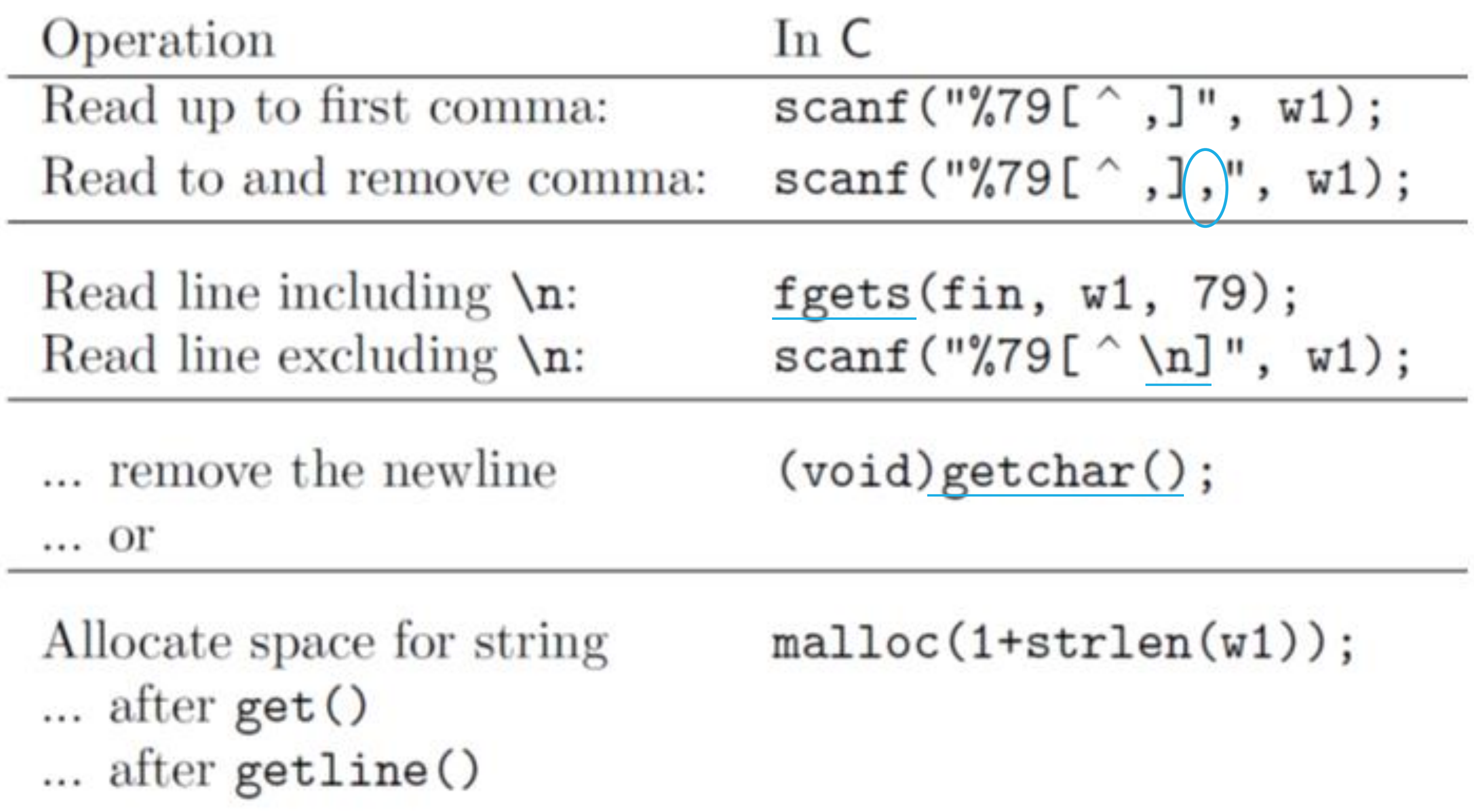
IN C

Reading String and Security Vulnerability in C

String input. The simplest form for a string input command is something that should **NEVER be used**. It skips leading whitespace, reads characters starting with the first non-whitespace character, and stops reading at next whitespace. There is no limit on length of the string that is read. DO NOT DO THIS!

If you try to read an indefinite-length input into a finite array, and you do not specify a length limit, the input can overflow the boundaries of the array and overlay nearby variables. A program that makes this error is **open to exploitation by hackers.**

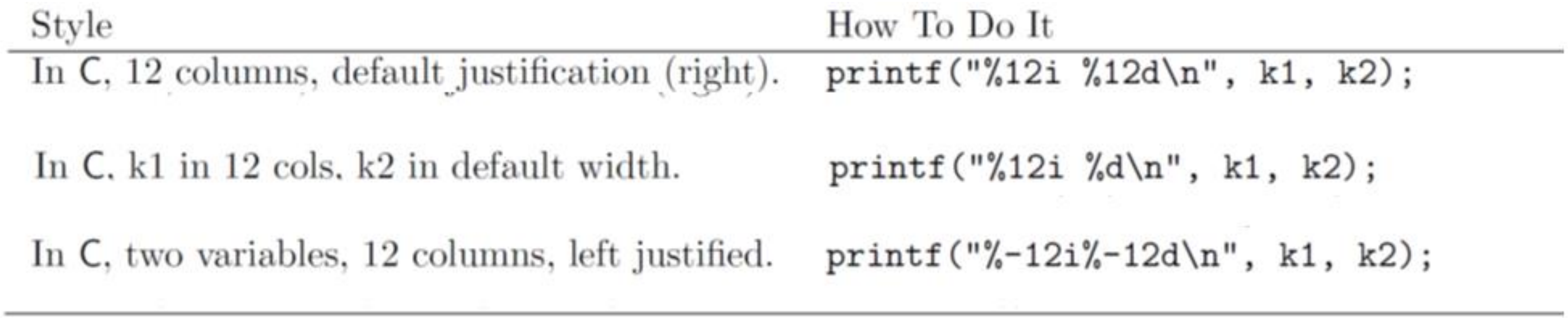
In C: `scanf("%s", w1);`



Operation	In C
Read up to first comma:	`scanf("%79[^,]", w1);`
Read to and remove comma:	`scanf("%79[^,],", w1);`
Read line including `\n`:	`fgets(fin, w1, 79);`
Read line excluding `\n`:	`scanf("%79[^\n]", w1);`
... remove the newline	`(void)getchar();`
... or	
Allocate space for string	`malloc(1+strlen(w1));`
... after `get()`	
... after `getline()`	

Input functions & modifiers in C

Output Functions and modifiers C



Style	How To Do It
In C, 12 columns, default justification (right).	`printf("%12i %12d\n", k1, k2);`
In C, k1 in 12 cols, k2 in default width.	`printf("%12i %d\n", k1, k2);`
In C, two variables, 12 columns, left justified.	`printf("%-12i%-12d\n", k1, k2);`

Working with Files in C++

OPENING FILES

FILE I/O

Predeclared Streams in C and C++

Predeclared streams:

C: stdin, stdout, stderr

C++: cin, cout, cerr, clog

The C++ streams cin and cout **share buffers with the corresponding C streams** stdin and stdout.

The streams **stderr and cerr** are **unbuffered**.

The new C++ stream, clog is used for **logging transactions**.

Flushing the Output Stream in C++

The output stream is buffered, so when you switch from output (such as displaying a menu) to input (such as reading user input), make sure you flush (print to screen, file, etc), the contents of the stream's buffer or the sequence will get out of order.

In C++:

Add flush or endl (newline then flush) to the output statement.

```
cout << "This is output " << x << endl;
//Use flush instead of endl if you don't want the newline
```

Libraries to include for File I/O in C++

When using text files in C++ you might find the following libraries useful:

- `#include <iostream> // For the standard streams.`
- `#include <fstream> // For file streams.`
- `#include <iomanip>`
- `// A set of stream management and formatting tools.`
- `#include <sstream> // For string-streams.`

Look up documentation for details on functions related to each.

`using namespace std;`

//Used to bring the names of included library facilities into your working namespace.

Open a file for input in C++

```
# define INF "mydata.txt"
int main() {
    double x, y; //variables
    ifstream dataFile (INF);    // create an input stream for the input file
    if (!dataFile) {    //test if the input file failed to open
        cerr << "Can't open " << INF << " for input";
        //Print an error message to error stream, in console
        exit(1);
    }
```

Open a file for output in C++

```
# define OUTF "myout.txt"
int main() {
    double x, y; //variables
    ofstream outFile( OUTF ); //Create a stream for the output file
    if (!outFile.is_open())    //test if the output file failed to open
        cerr << "Can't open " << OUTF << " for output";
        //Error message to error stream, in console
```

End of File detection in C++

The **`fileObject.eof()`** function checks to see if the end of the file has been reached. It can be checked at the start of each time through the loop or somewhere inside the loop. You decide what you need to for your logic.

To check at the start of each loop, you could test it in the middle section of a for loop like this:

```
for(int i=1;!dataFile.eof( ); ++i) {
```

If your logic calls for checking in the middle of the body of your loop because you have to do something else first, you can use:

```
if (dataFile.eof( )) break;
```

If this is the first or last statement in your for loop, put it in the middle section of the for(;<here>;).

Common error: If there is **a \n \t blank space or other whitespace character** at the end of your file, then the *end of file mark*er has not been reached. However, the **end of data** has been reached. If this case, you will run into trouble **because the attempt to read the next piece of data will fail**. Don’t end your file with a return or other whitespace. If you **know** it’s there, you can read it to remove it, it’s extra work.

Close your files

Practice good housekeeping, or “RAM keeping” to be more accurate, open files waste RAM.

```
    dataFile.close();
    outFile.close();
```

In Class Practice File I/O

- Based on the the code from the previous slides, add the necessary code to make it a C++ program, include iostream and fstream. You might need iomanip.
- Your program should read the numbers from the file into an array and print the array to the console.
- Challenge: Also print to your file.
- You will need to use the file called mydata.txt, it has a column of integer values in it. Pay attention to type.
- NOTE: Use your IDE to create text files so they are put in the right place in your system.

Manipulating Input and Output

IN C++

Reading data from a file in C++

Reading data from a file is very similar to reading data from the keyboard, except instead of **`cin`**, you use the **file stream name**.

```
ifstream dataFile (“myinput.txt”);
```

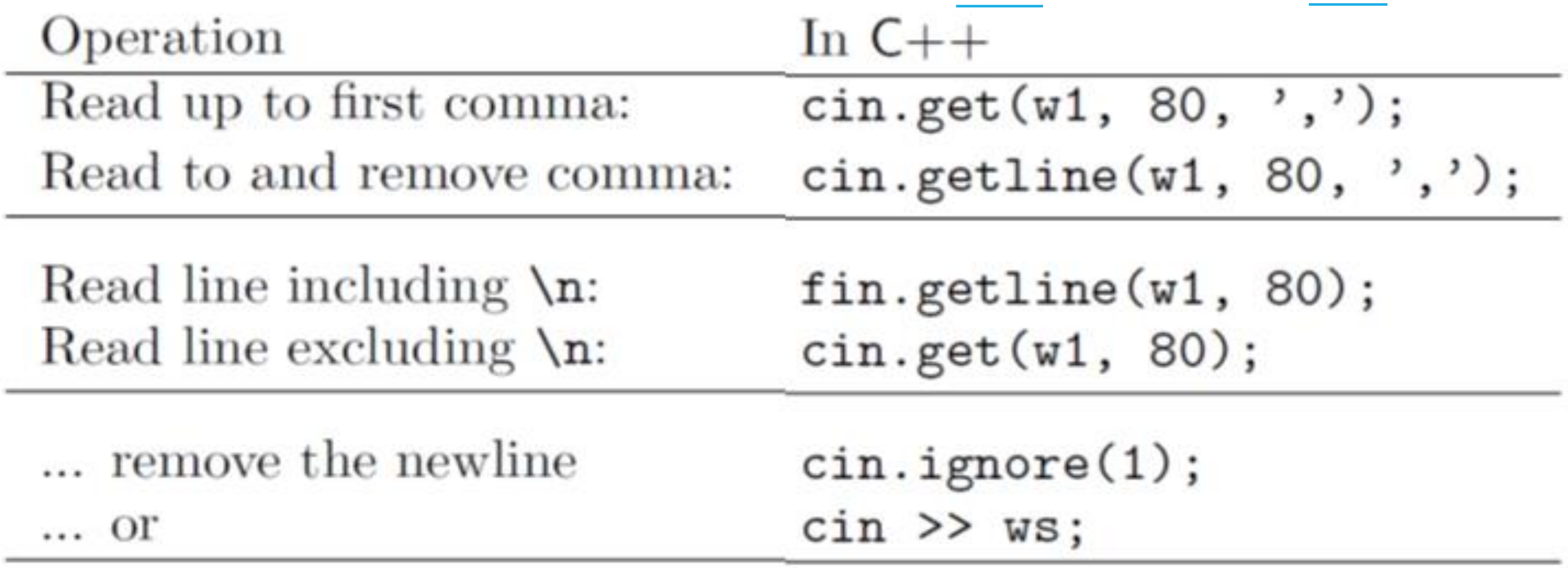
// create an input stream for a file

```
dataFile >>x >>y; //Read from a file
```

There are **many modifiers for input**.

Ex: `ws, skipws`

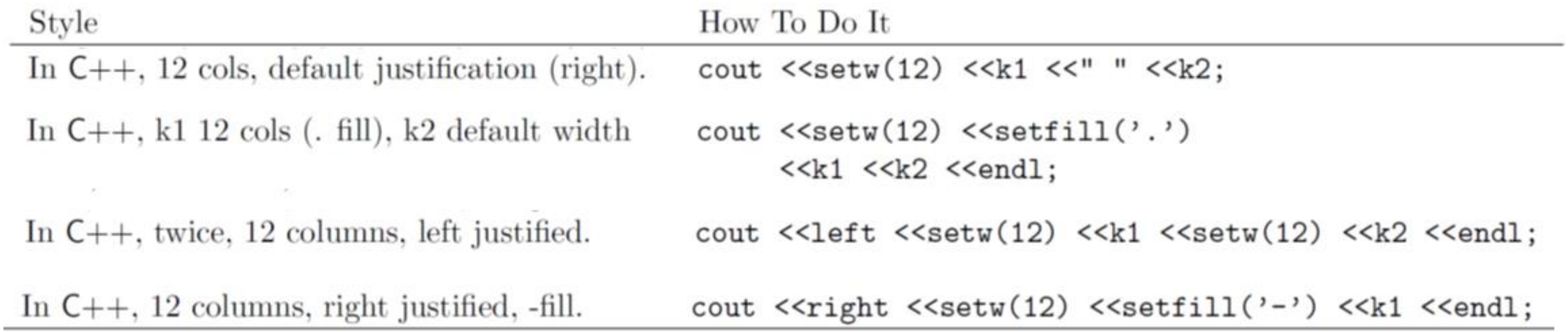
For the whole list, go to http://www.cplusplus.com/reference/library/manipulators/



Operation	In C++
Read up to first comma:	`cin.get(w1, 80, ’,’);`
Read to and remove comma:	`cin.getline(w1, 80, ’,’);`
Read line including \n:	`fin.getline(w1, 80);`
Read line excluding \n:	`cin.get(w1, 80);`
... remove the newline	`cin.ignore(1);`
... or	`cin >> ws;`

Input functions & modifiers in C++

Output Functions and modifiers C vs C++



Style	How To Do It
In C++, 12 cols, default justification (right).	`cout <<setw(12) <<k1 <<" " <<k2;`

In C++, k1 12 cols (. fill), k2 default width	`cout <<setw(12) <<setfill('.')`
	`<<k1 <<k2 <<endl;`

In C++, twice, 12 columns, left justified.	`cout <<left <<setw(12) <<k1 <<setw(12) <<k2 <<endl;`
In C++, 12 columns, right justified, -fill.	`cout <<right <<setw(12) <<setfill('-') <<k1 <<endl;`

`endl` is an output modifier that **inserts a newline character and flushes the output stream**.

Summary

- Defined Streams and Files
- Built in Streams in C and C++
- How to flush the output stream in C and C++.
- File IO in C
- File IO in C++

That’s it for now!