

Chapter 3, File Input and Output

Programming Concepts in Scientific
Programming

EPFL, Master class

October 2, 2017

Redirecting command output

```
$ ./exec > output.txt
```

Try it with:

```
$ ls > output.txt
```

Writing to standard output/error

```
int x = 1.;  
int y = 0.;  
  
if (y == 0) {  
    std::cerr << "Error - division by zero\n";  
  
} else {  
    std::cout << x / y << "\n";  
    std::cout.flush();  
}
```

What is the result of doing this ?

```
$ ./exec > output.txt
```

Writing to standard output/error

```
int x = 1.;  
int y = 0.;  
  
if (y == 0) {  
    std::cerr << "Error - division by zero\n";  
  
} else {  
    std::cout << x / y << "\n";  
    std::cout.flush();  
}
```

Why flush() ?

Writing to file

```
#include <cassert>
#include <fstream>
#include <iostream>

int main() {
    std::ofstream write_output("Output.dat");
    assert(write_output.is_open());

    write_output << "Hello world !" << std::endl;

    write_output.close();
}
```

Writing to file

```
#include <cassert>
#include <fstream>
#include <iostream>

int main() {
    std::ofstream write_output("Output.dat");
    assert(write_output.is_open());

    write_output << "Hello world !" << std::endl;

    write_output.close();
}
```

Writing to file

```
#include <cassert>
#include <fstream>
#include <iostream>

int main() {
    std::ofstream write_output("Output.dat");
    assert(write_output.is_open());

    write_output << "Hello world !" << std::endl;

    write_output.close();
}
```

Writing to file

```
#include <cassert>
#include <fstream>
#include <iostream>

int main() {
    std::ofstream write_output("Output.dat");
    assert(write_output.is_open());

    write_output << "Hello world !" << std::endl;

    write_output.close();
}
```


Writing to file

```
#include <cassert>
#include <fstream>
#include <iostream>

int main() {
    std::ofstream write_output("Output.dat");
    assert(write_output.is_open());

    write_output << "Hello world !" << std::endl;

    write_output.close();
}
```

Writing to file

```
#include <cassert>
#include <fstream>
#include <iostream>

int main() {
    std::ofstream write_output("Output.dat");
    assert(write_output.is_open());

    write_output << "Hello world !" << std::endl;

    write_output.close();
}
```

Writing to file

```
#include <cassert>
#include <fstream>
#include <iostream>

int main() {
    std::ofstream write_output("Output.dat");
    assert(write_output.is_open());

    write_output << "Hello world !" << std::endl;

    write_output.close();
}
```

Writing to file

```
#include <cassert>
#include <fstream>
#include <iostream>

int main() {
    std::ofstream write_output("Output.dat");
    assert(write_output.is_open());

    write_output << "Hello world !" << std::endl;

    write_output.close();
}
```

Writing a vector to file

```
double x[3] = {0.0, 1.0, 0.0};  
double y[3] = {0.0, 0.0, 1.0};  
  
std::ofstream write_output("Output.dat");  
assert(write_output.is_open());  
  
for (int i = 0; i < 3; i++) {  
    write_output << x[i] << " " << y[i] << "\n";  
}  
  
write_output.close();
```

Setting the precision of the output

```
write_output.precision(10); // 10 sig figs  
write_output << x << "\n";  
write_output.close();
```

Reading from File

```
std::ifstream read_file("Output.dat");  
assert(read_file.is_open());  
  
for (int i = 0; i < 6; i++) {  
    read_file >> x[i] >> y[i];  
}  
  
read_file.close();
```

Reading from File

```
std::ifstream read_file("Output.dat");  
assert(read_file.is_open());  
  
for (int i = 0; i < 6; i++) {  
    read_file >> x[i] >> y[i];  
}  
  
read_file.close();
```


Reading from File

```
std::ifstream read_file("Output.dat");  
assert(read_file.is_open());  
  
for (int i = 0; i < 6; i++) {  
    read_file >> x[i] >> y[i];  
}  
  
read_file.close();
```

Reading from File

```
std::ifstream read_file("Output.dat");  
assert(read_file.is_open());  
  
for (int i = 0; i < 6; i++) {  
    read_file >> X[i] >> y[i];  
}  
  
read_file.close();
```

What can be the type of x and y ?

Reading from File

Unknown number of entries

```
double x[100], y[100];  
while (!read_file.eof()) {  
    read_file >> x[i] >> y[i];  
    i++;  
}
```

Reading from File

Unknown number of entries

```
double x[100], y[100];  
  
while (!read_file.eof()) {  
    read_file >> x[i] >> y[i];  
    i++;  
}
```

Reading from the command line

```
int main(int argc, char *argv[]) {
```

Reading from the Command Line

```
std::cout << "Number of command line arguments = ";  
std::cout << argc << "\n";  
for (int i = 0; i < argc; i++) {  
    std::cout << "Argument " << i << " = " << argv[i];  
    std::cout << "\n";  
}
```

Reading from the Command Line

```
std::cout << "Number of command line arguments = ";  
std::cout << argc << "\n";  
for (int i = 0; i < argc; i++) {  
    std::cout << "Argument " << i << " = " << argv[i];  
    std::cout << "\n";  
}
```

What is the memory
representation of `argv[i]` ?

Reading from the Command Line

```
std::string program_name = argv[0];  
int number_of_nodes = atoi(argv[1]);  
double conductivity = atof(argv[2]);
```


Tips: Controlling Output Format

- ▶ Output in scientific format. 4.6578e2 is achieved by the use of the flag: `std::ios::scientific`
- ▶ Always showing a + or - sign: `std::ios::showpos`
- ▶ Precision of scientific output: `precision`

```
write_file.setf(std::ios::scientific);  
write_file.setf(std::ios::showpos);  
write_file.precision(13);
```

New assignment

- ▶ Chapter 4, reading
- ▶ Exercises of Chapter 3.