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

```
3     int x = 1.;
4     int y = 0.;
5
6     if (y == 0) {
7
8       std::cerr << "Error - division by zero\n";
9
10    } else {
11
12      std::cout << x / y << "\n";
13      std::cout.flush();
```

What is the result of doing this ?

$ ./exec > output.txt

Writing to standard output/error

```
3     int x = 1.;
4     int y = 0.;
5
6     if (y == 0) {
7
8       std::cerr << "Error - division by zero\n";
9
10    } else {
11
12      std::cout << x / y << "\n";
13      std::cout.flush();
```

Why flush() ?

Writing to file

```
1  #include <cassert>
2  #include <fstream>
3  #include <iostream>
4
5  int main() {
6    std::ofstream write_output("Output.dat");
7    assert(write_output.is_open());
8
9    write_output << "Hello world !" << std::endl;
10
11   write_output.close();
12 }
```

Writing to file

```
1  #include <cassert>
2  #include <fstream>
3  #include <iostream>
4
5  int main() {
6    std::ofstream write_output("Output.dat");
7    assert(write_output.is_open());
8
9    write_output << "Hello world !" << std::endl;
10
11   write_output.close();
12 }
13
```

Writing to file

```
1  #include <cassert>
2  #include <fstream>
3  #include <iostream>
4
5  int main() {
6    std::ofstream write_output("Output.dat");
7    assert(write_output.is_open());
8
9    write_output << "Hello world !" << std::endl;
10
11   write_output.close();
12 }
```

Writing to file

```
1  #include <cassert>
2  #include <fstream>
3  #include <iostream>
4
5  int main() {
6    std::ofstream write_output("Output.dat");
7    assert(write_output.is_open());
8
9    write_output << "Hello world !" << std::endl;
10
11   write_output.close();
12 }
```

Writing to file

```
1  #include <cassert>
2  #include <fstream>
3  #include <iostream>
4
5  int main() {
6    std::ofstream write_output("Output.dat");
7    assert(write_output.is_open());
8
9    write_output << "Hello world !" << std::endl;
10
11   write_output.close();
12 }
```

Writing to file

```
1  #include <cassert>
2  #include <fstream>
3  #include <iostream>
4
5  int main() {
6    std::ofstream write_output("Output.dat");
7    assert(write_output.is_open());
8
9    write_output << "Hello world !" << std::endl;
10
11   write_output.close();
12 }
```

Writing to file

```
1  #include <cassert>
2  #include <fstream>
3  #include <iostream>
4
5  int main() {
6    std::ofstream write_output("Output.dat");
7    assert(write_output.is_open());
8
9    write_output << "Hello world !" << std::endl;
10
11   write_output.close();
12 }
```

Writing to file

```cpp
#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

```
6     double x[3] = {0.0, 1.0, 0.0};
7     double y[3] = {0.0, 0.0, 1.0};
8
9     std::ofstream write_output("Output.dat");
10    assert(write_output.is_open());
11
12    for (int i = 0; i < 3; i++) {
13      write_output << x[i] << " " << y[i] << "\n";
14    }
15
16    write_output.close();
```

Setting the precision of the output

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

Reading from File

```
8     std::ifstream read_file("Output.dat");
9     assert(read_file.is_open());
10
11    for (int i = 0; i < 6; i++) {
12      read_file >> x[i] >> y[i];
13    }
14
15    read_file.close();
```

Reading from File

```
8     std::ifstream read_file("Output.dat");
9     assert(read_file.is_open());
10
11    for (int i = 0; i < 6; i++) {
12      read_file >> x[i] >> y[i];
13    }
14
15    read_file.close();
```

Reading from File

```
8     std::ifstream read_file("Output.dat");
9     assert(read_file.is_open());
10
11    for (int i = 0; i < 6; i++) {
12      read_file >> x[i] >> y[i];
13    }
14
15    read_file.close();
```

Reading from File

```
8     std::ifstream read_file("Output.dat");
9     assert(read_file.is_open());
10
11    for (int i = 0; i < 6; i++) {
12      read_file >> x[i] >> y[i];
13    }
14
15    read_file.close();
```

What can be the type of x and y ?

Reading from File

Unknown number of entries

```
6     double x[100], y[100];

12    while (!read_file.eof()) {
13      read_file >> x[i] >> y[i];
14      i++;
15    }
```

Reading from File

Unknown number of entries

```
6     double x[100], y[100];

12    while (!read_file.eof()) {
13      read_file >> x[i] >> y[i];
14      i++;
15    }
```

Reading from the command line

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

Reading from the Command Line

```
5     std::cout << "Number of command line arguments = ";
6     std::cout << argc << "\n";
7     for (int i = 0; i < argc; i++) {
8       std::cout << "Argument " << i << " = " << argv[i];
9       std::cout << "\n";
10    }
```

Reading from the Command Line

```
5     std::cout << "Number of command line arguments = ";
6     std::cout << argc << "\n";
7     for (int i = 0; i < argc; i++) {
8       std::cout << "Argument " << i << " = " << argv[i];
9       std::cout << "\n";
10    }
```

What is the memory representation of argv[i] ?

Reading from the Command Line

```
12      std::string program_name = argv[0];
13      int number_of_nodes = atoi(argv[1]);
14      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

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

New assignement

- Chapter 4, reading
- Exercises of Chapter 3.