Module 4: I/O

In this module we will cover

- **Keyboard/screen input and output streams**
- **File input/output**

Aims of this module

You do not get far with any program without the need to either inputto and/or output something from the program. This is particularly true in scientific analysis where the very minimum is likely to be a file of data points to analyse.

This module covers the things you need to know to deal with user and file I/O

In several precediing modules we have used "hands on examples" cout<< and cin>> to write things to the screen and read in from the keyboard. In this module we first cover these more formally.

We then show you how to get things from files.

4.1 Writing to the screen with `cout <<`

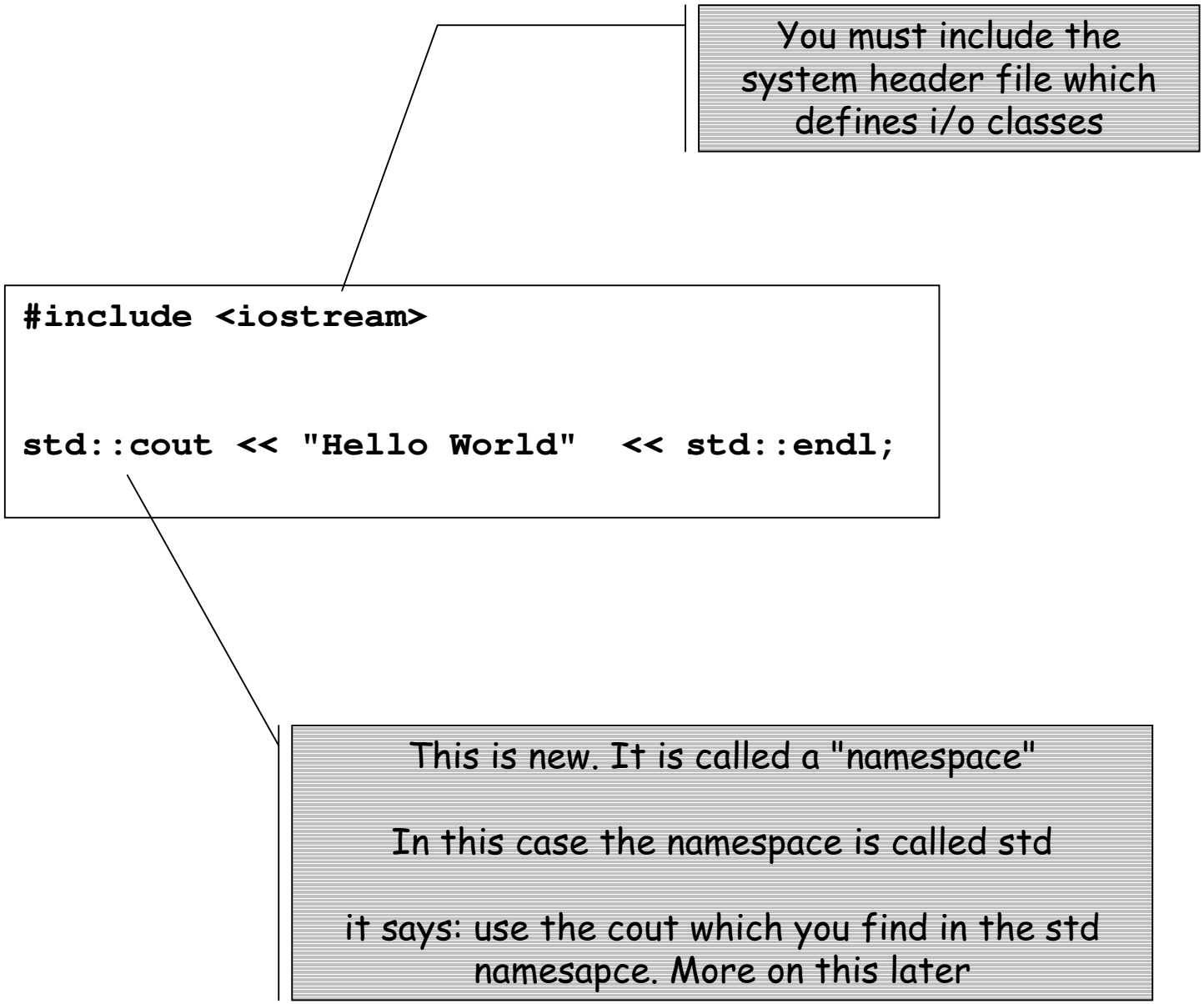
In C++ you write or read to things
called "output streams"

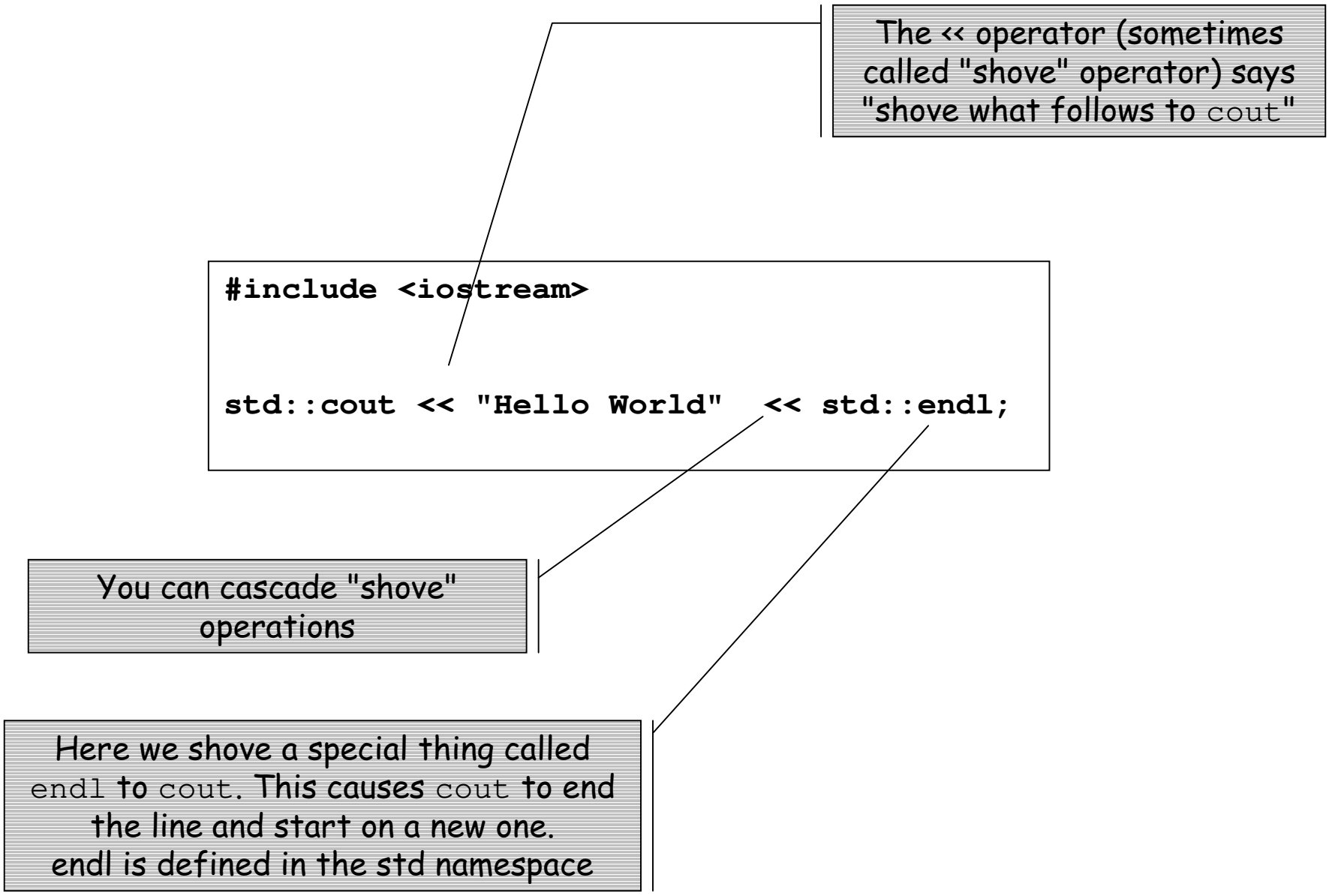
cout is an **output stream** which sends
things to the screen

It is used like this:

```
#include <iostream>


std::cout << "Hello World"  << std::endl;
```


You must include the system header file which defines i/o classes
#include <iostream>
std::cout << "Hello World" << std::endl;
This is new. It is called a "namespace"
In this case the namespace is called std
it says: use the cout which you find in the std namesapce. More on this later


The << operator (sometimes called "shove" operator) says "shove what follows to cout"
#include <iostream>
std::cout << "Hello World" << std::endl;
You can cascade "shove" operations
Here we shove a special thing called endl to cout. This causes cout to end the line and start on a new one. endl is defined in the std namespace

A more complicated example:

You can send all the built-in types to the output streams

You can send more than one thing with the same command

Line breaks are irrelevant. I do it like this for style.

There are more formatting commands like endl - look them up in the reference book if you are interested.

```
#include <iostream>

int a = 2;
int b = 5;

std::cout << "The answer to "
          << a
          << "+"
          << b
          << " is ";

std::cout << (a+b)
          << std::endl;
```

The output on the screen will look like this:

TECHNICAL ASIDE

cout uses "buffered I/O" which means that it doesn't shove stuff out straight away- it puts it in a buffer first and only does output when the buffer fills up

This isn't always what you want- especially if you're trying to track down errors

You might find that the program crashes and doesn't produce the last few cout's of output

To get around this C++ provides the "unbuffered" cerr stream as an alternative

Use this for debugging and error messages

```
#include <iostream.h>
...

if (badError == true) {
  // Oh dear, something bad happened
  std::cerr
        << "WARNING- Serious error!"
        << std::endl;
}
```

cerr is just another OUTPUT STREAM

4.2 Reading from the keyboard with `cin >>`

The equivalent for input is the input stream cin
It uses the "yank" or input operator >>

```
// Here is a 1 function calculator

#include <iostream>

float v1, v2;

std::cout << "Please input two numbers" << std::endl;

std::cin >> v1;
std::cin >> v2;

// or you could just say
// std::cin >> v1 >> v2;
// it does the same thing

std::cout << "The sum of "
          << v1 << " and " << v2
          << " is " << (v1+v2)
          << std::endl;
```

4.3 Input/Output using Files

Files are just ordinary input or output streams

The only complication is "how to attach a file to a stream"

After his you use them just like cin and cout:
you yank or shove things using >> and <<

This is how you attach a file to an output stream

This says "make an object of type `ofstream` and call it `myOutputFile`

```
#include <fstream>

// Here is where the output stream is created:

std::ofstream myOutputFile("filename") ;


// ..and here is how you would write two numbers to it

myOutputFile << 42   << 18 ;
```

This is the actual name of the physical file you want to be used

..in practice you need to also check that the file was opened properly

```
#include <fstream>

// Here is where the output stream is created:

std::ofstream myOutputFile("filename") ;

// Check it opened ok

if ( ! myOutputFile )
  {
    std::cerr << "Unable to open output file!"
              << std::endl;
    return ;
  }

// Carry on and write two numbers to it

myOutputFile << 42   << 18 ;
```

When you use `myOutputFile` like this it automatically returns a bool

`true`: if opened ok
`false`: otherwise

Here is how you open a file as an input stream (in order to receive input from it)

```
#include <fstream>

// Here is where the input stream is created:

std::ifstream myInputFile("filename") ;


// Read two numbers from it

float a,b ;

myInputFile << a  << b ;
```

A complete example:

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

void main() {
  std::ofstream myOutputFile("Degrees.dat");
  std::ifstream myInputFile ("Radians.dat");

  if ( !myInputFile ) {
    cerr << "Error: unable to open input file!" << endl;
  }
  if ( !myOutputFile ) {
    cerr << "Error: unable to open output file!" << endl;
  }
  else {
    float angle;
    while ( myInputFile ) {
    myInputFile >> angle ;
    myOutputFile << ( angle*180.0/3.141592654);
  }
}
```

This example shows a complete program to covert a file of angles in radians to a file of the angles in degrees

We will look at it bit-by-bit in the following slides

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

void main() {
  std::ofstream myOutputFile("Degrees.dat");
  std::ifstream myInputFile ("Radians.dat");

  if ( !myInputFile ) {
    cerr << "Error: unable to open input file!" << endl;
  }
  if ( !myOutputFile ) {
    cerr << "Error: unable to open output file!" << endl;
  }

  float angle;
  while ( myInputFile ) {
    myInputFile >> angle ;
    myOutputFile << ( angle*180.0/3.141592654);
  }
}
```

opens an output file stream and attaches it to file Degrees.dat

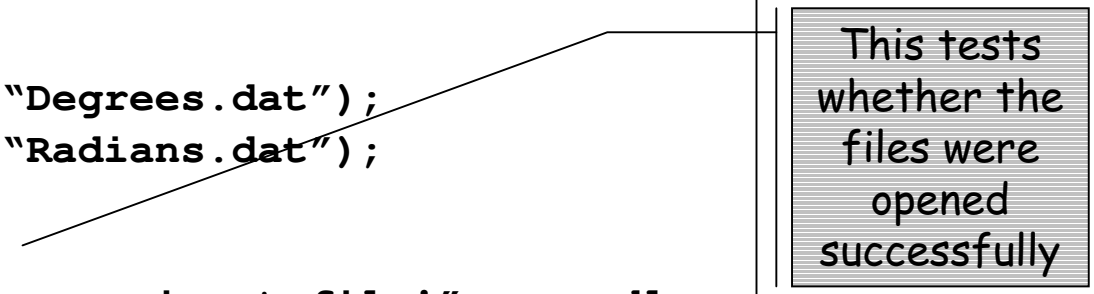
opens an input file stream and attaches it to file Radians.dat

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

void main() {
  std::ofstream myOutputFile("Degrees.dat");
  std::ifstream myInputFile ("Radians.dat");

  if ( !myInputFile ) {
    cerr << "Error: unable to open input file!" << endl;
  }
  if ( !myOutputFile ) {
    cerr << "Error: unable to open output file!" << endl;
  }

  float angle;
  while ( myInputFile ) {
    myInputFile >> angle ;
    myOutputFile << ( angle*180.0/3.141592654);
  }
}
```



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

void main() {
  std::ofstream myOutputFile("Degrees.dat");
  std::ifstream myInputFile ("Radians.dat");

  if ( !myInputFile ) {
    cerr << "Error: unable to open input file!" << endl;
  }
  if ( !myOutputFile ) {
    cerr << "Error: unable to open output file!" << endl;
  }

  float angle;
  while ( myInputFile ) {
    myInputFile >> angle ;
    myOutputFile << ( angle*180.0/3.141592654) << endl;
  }
}
```

Tests whether the end of file has been reached. This will evaluate to false if there is no more data

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

void main() {
  std::ofstream myOutputFile("Degrees.dat");
  std::ifstream myInputFile ("Radians.dat");

  if ( !myInputFile ) {
    cerr << "Error: unable to open input file!" << endl;
  }
  if ( !myOutputFile ) {
    cerr << "Error: unable to open output file!" << endl;
  }

  float angle;
  while ( myInputFile ) {
    myInputFile >> angle ;
    myOutputFile << ( angle*180.0/3.141592654) << endl;
  }
}
```

yanks each angle in turn from the input stream

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

void main() {
  std::ofstream myOutputFile("Degrees.dat");
  std::ifstream myInputFile ("Radians.dat");

  if ( !myInputFile ) {
    cerr << "Error: unable to open input file!" << endl;
  }
  if ( !myOutputFile ) {
    cerr << "Error: unable to open output file!" << endl;
  }

  float angle;
  while ( myInputFile ) {
    myInputFile >> angle ;
    myOutputFile << ( angle*180.0/3.141592654 ) << endl;
  }
}
```

converts the angle

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

void main() {
  std::ofstream myOutputFile("Degrees.dat");
  std::ifstream myInputFile ("Radians.dat");

  if ( !myInputFile ) {
    cerr << "Error: unable to open input file!" << endl;
  }
  if ( !myOutputFile ) {
    cerr << "Error: unable to open output file!" << endl;
  }

  float angle;
  while ( myInputFile ) {
    myInputFile >> angle
    myOutputFile << ( angle*180.0/3.141592654 ) << endl;
  }
}
```

shoves the converted angle to the output file

Note: all these should be std::cerr, std::cout
This is omitted due to lack of space on page

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

void main() {
  std::ofstream myOutputFile("Degrees.dat");
  std::ifstream myInputFile ("Radians.dat");

  if ( !myInputFile ) {
    cerr << "Error: unable to open input file!" << endl;
  }
  if ( !myOutputFile ) {
    cerr << "Error: unable to open output file!" << endl;
  }

  float angle;
  while ( myInputFile ) {
    myInputFile >> angle
    myOutputFile << ( angle*180.0/3.141592654 ) << endl;
  }
}
```

Student exercise

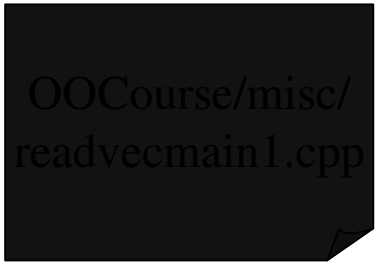
- Create a short data file containing several lines, and on each line put three numbers which are the components of a three vector

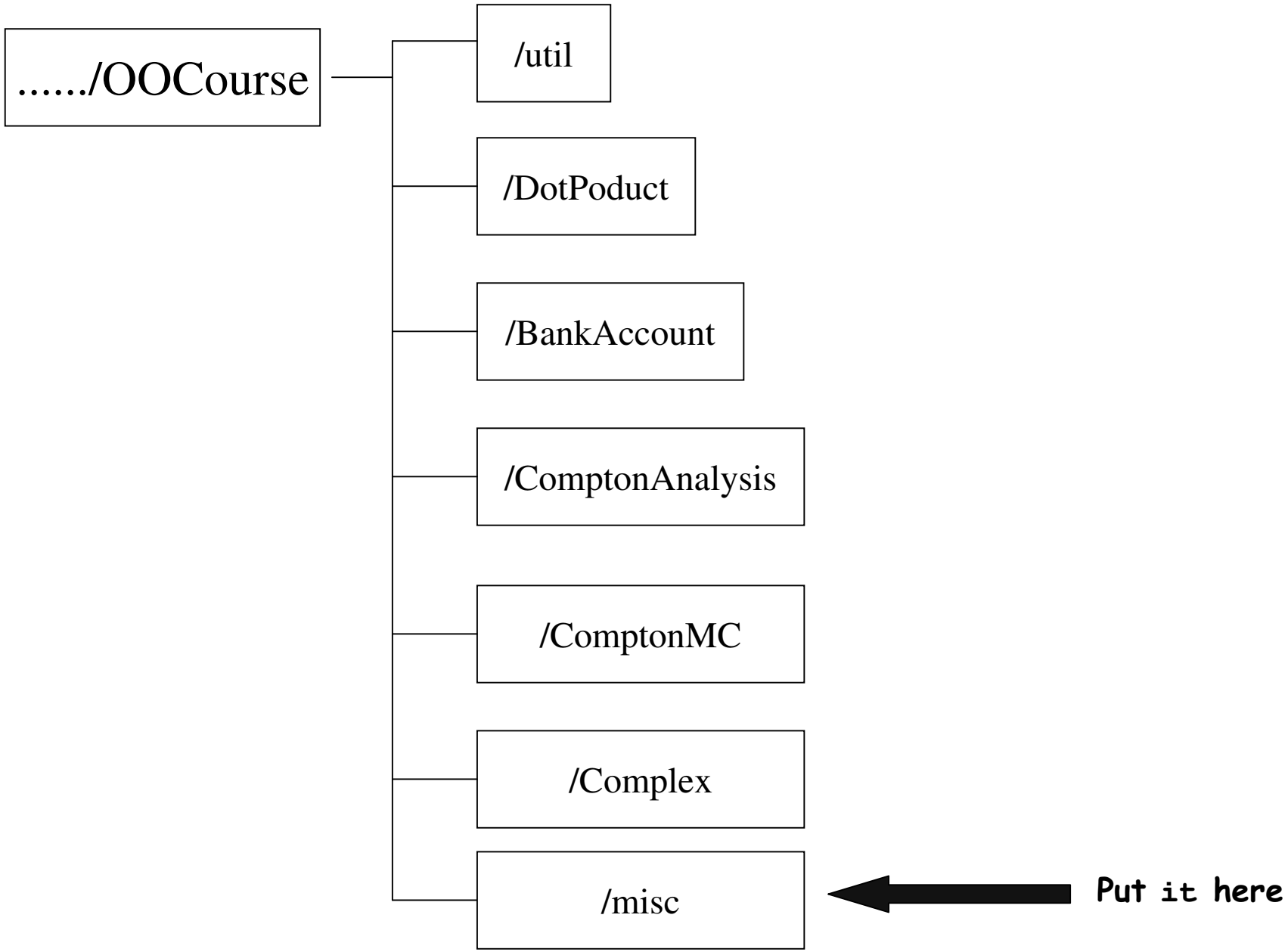
- write a program to read each line in turn from the file

- From each line it should create a ThreeVector

- it should then just dump out the contencts of the vector using the dump( ) method.

(you will need to use your ThreeVector class which you previously put in OOCourse/util/ThreeVector.h)




...../OOCourse
/util
/DotPoduct
/BankAccount
/ComptonAnalysis
/ComptonMC
/Complex
/misc
Put it here

Summary of Module 4: I/O

- **Input and output**
  - **cout << to send to keyboard**
  - **cin << to get from keyboard**
  - **File i/o using streams.**