C++:
Namespaces

Namespaces - motivation

- What if you have 2 functions (or objects or variables) with the same signature?

```cpp
//main.cpp
#include <iostream>
using namespace std;

#include "ms1.h"
#include "ms2.h"

int main() {
    //which myFunc will print?
    //will it compile?
    cout<<"myFunc returns "<<myFunc()<<endl;
    return 0;
}
```

```cpp
/*
 * ms2.h
 */

#ifndef MS2_H_
#define MS2_H_

int myFunc();

#endif /* MS2_H_ */
```

```cpp
/*
 * ms1.h
 */

#ifndef MS1_H_
#define MS1_H_
    int myFunc();

#endif /* MS1_H_ */
```

Namespaces - motivation

- The compiler cannot distinguish which myFunc() to call, so the project will not compile.

```cpp
//main.cpp
#include <iostream>
using namespace std;

#include "ms1.h"
#include "ms2.h"

int main() {
    //which myFunc will print?
    //will it compile?
    cout<<"myFunc returns "<<myFunc()<<endl;
    return 0;
}
```

```cpp
/*
 * ms2.h
 */

#ifndef MS2_H_
#define MS2_H_

int myFunc();

#endif /* MS2_H_ */
```

```cpp
/*
 * ms1.h
 */

#ifndef MS1_H_
#define MS1_H_
    int myFunc();

#endif /* MS1_H_ */
```

Namespaces – box analogy

- How can a compiler distinguish between the following 2 functions?

```
//ms1.h
int myFunc()
```

```
//ms2.h
int myFunc()
```

Namespaces – box analogy

- How can a compiler distinguish between the following 2 functions?
- It can't

```
//ms1.h
int myFunc()
```

```
//ms2.h
int myFunc()
```

Namespaces – box analogy

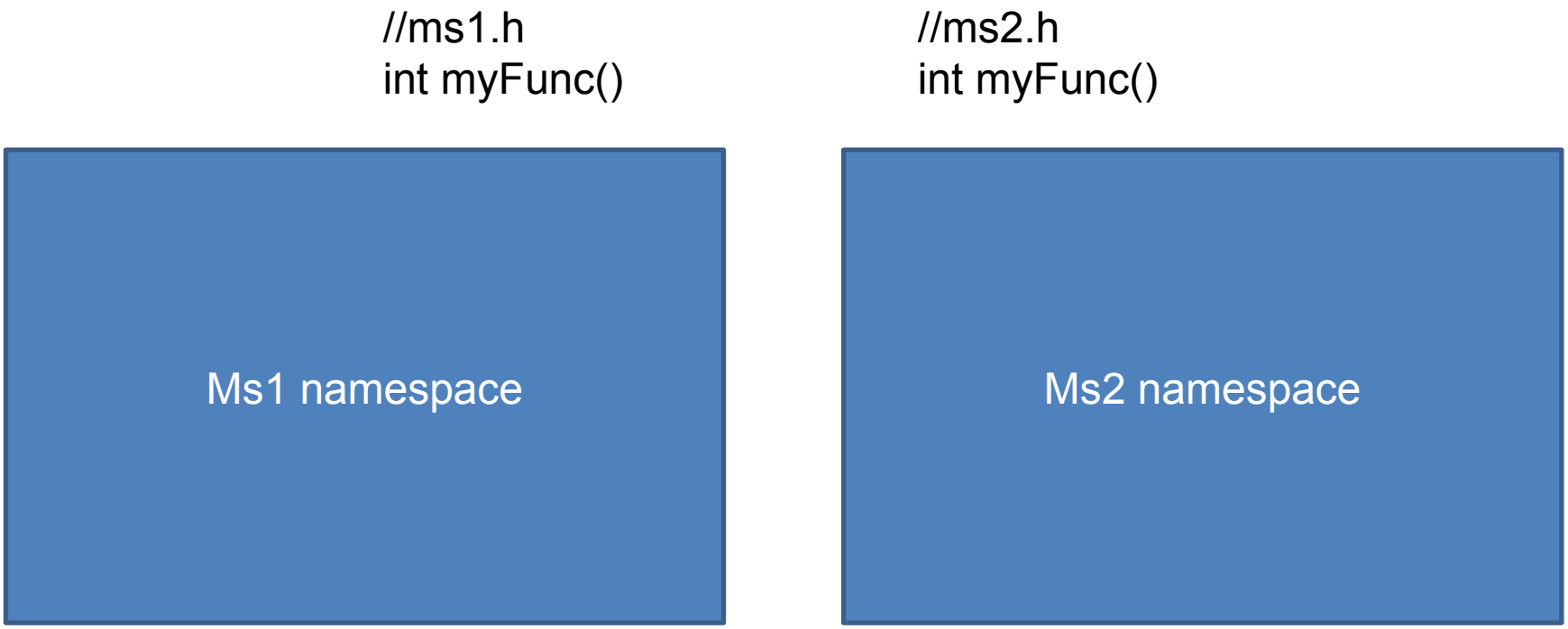
- How about if each is in a seperate scope? (think of scope as a box)

```
//ms1.h
int myFunc()
```

```
//ms2.h
int myFunc()
```

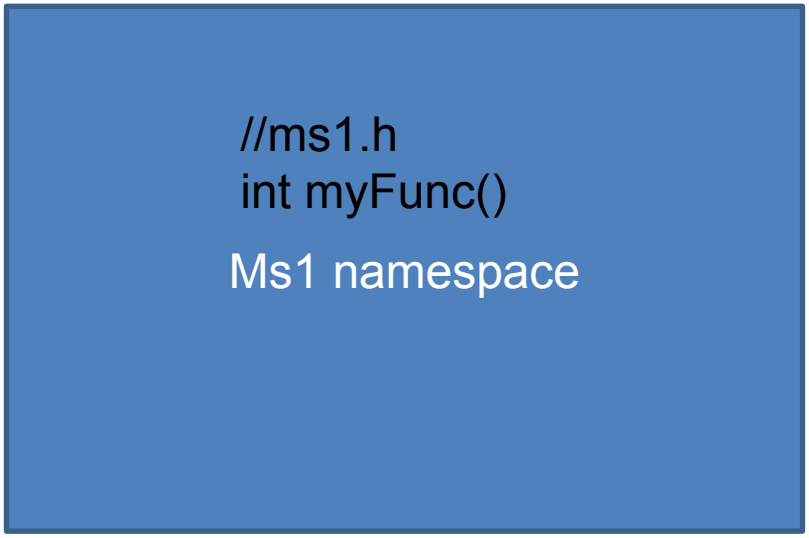
Namespaces – box analogy

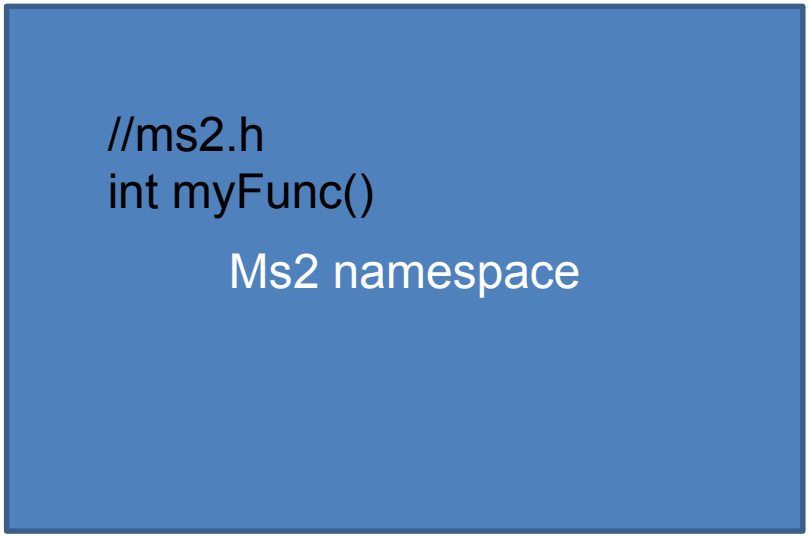
- Create 2 scopes (boxes). Give them distinct names.



Namespaces – box analogy

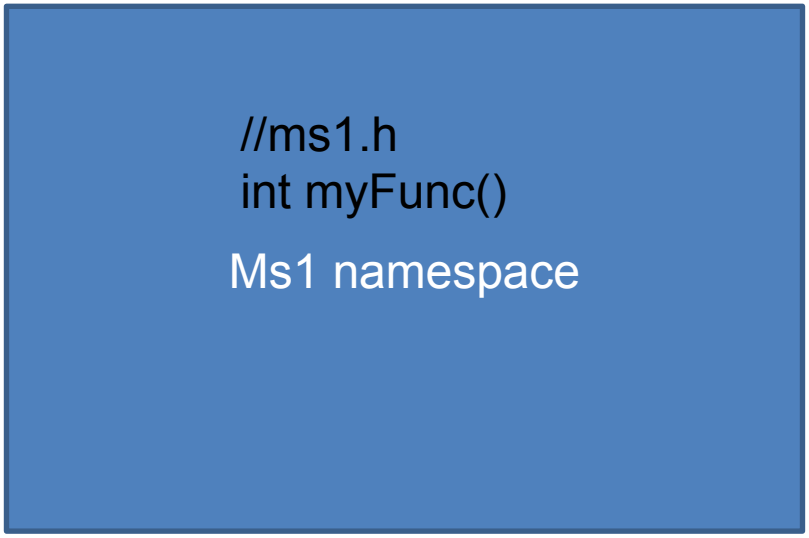
- Put each function in its own scope.

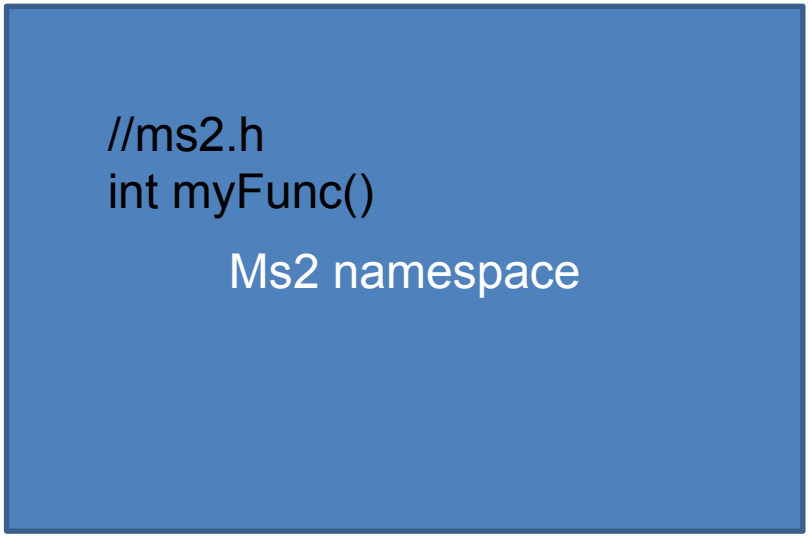




Namespaces – box analogy

- To get to either function, tell compiler which scope (box) to use
  - Ms1::myFunc()
  - Ms2::myFunc()





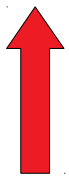
Namespaces - solution

- First, put the functions in seperate namespaces

```cpp
//main.cpp
#include <iostream>
using namespace std;

#include "ms1.h"
#include "ms2.h"

int main() {
    //which myFunc will print?
    //will it compile?
    cout<<"myFunc returns "<<myFunc()<<endl;
    return 0;
}
```

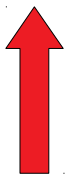
```cpp
/*
 * ms2.h
 */

#ifndef MS2_H_
#define MS2_H_

namespace ms2{
    int myFunc();
}

#endif /* MS2_H_ */
```

```cpp
/*
 * ms2.cpp
 */

#include "ms2.h"

namespace ms2{
    int myFunc(){
        return 2;
    }
}
```



```cpp
/*
 * ms1.h
 */

#ifndef MS1_H_
#define MS1_H_
namespace ms1{
    int myFunc();
}

#endif /* MS1_H_ */
```

```cpp
/*
 * ms1.cpp
 */

#include "ms1.h"

int ms1::myFunc(){
    return 1;
}
```



Namespaces - solution

- Then, select which myFunc() by namespace

```cpp
//main.cpp
#include <iostream>
using namespace std;

#include "ms1.h"
#include "ms2.h"

int main() {
    //first have none of the myfuncs in a namespace
    //next have just 1 of the myFuncs in a namespace and see what happens
    //finally put both in a namespace, dont forget to modify the code below
    cout << "ms1's myFunc returns " << ms1::myFunc() << endl;
    cout << "ms2's myFunc returns " << ms2::myFunc() << endl;
    return 0;
}
```

```cpp
/*
 * ms2.h
 */

#ifndef MS2_H_
#define MS2_H_

namespace ms2{
    int myFunc();
}

#endif /* MS2_H_ */
```

```cpp
/*
 * ms2.cpp
 */

#include "ms2.h"

namespace ms2{
    int myFunc(){
        return 2;
    }
}
```

```cpp
/*
 * ms1.h
 */

#ifndef MS1_H_
#define MS1_H_
namespace ms1{
    int myFunc();
}

#endif /* MS1_H_ */
```

```cpp
/*
 * ms1.cpp
 */

#include "ms1.h"

int ms1::myFunc(){
    return 1;
}
```

Namespaces

- Use ‘using’ construct – tells compiler to look in a particular namespace.

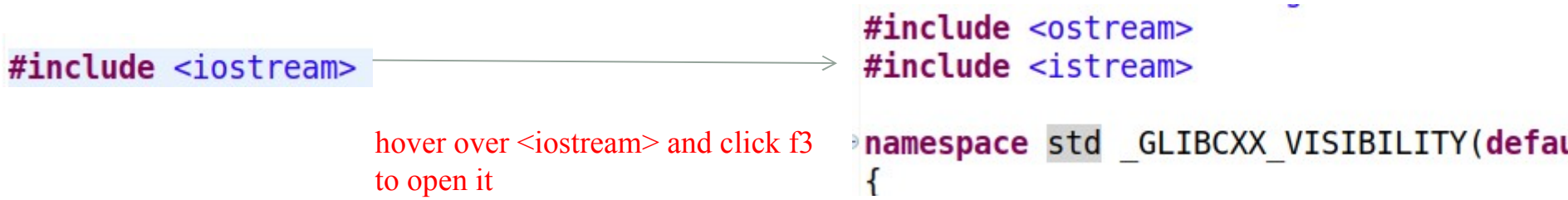
```
using namespace std;
```

- Allows cout instead of std::cout
- There are many namespaces. Wrap your code in namespaces if there is a chance that your functions have the same name as others (encrypt, decrypt, open, close etc…)

Namespaces - std

All C++ standard library constructs live in the std:: namespace

You can open any system include and see that it lives in the std namespace.



So always scope these constructs to the std:: namespace

Namespaces - Summary

Allow grouping code so there are no name conflicts

NOTE:must wrap both declaration (.h) and definition (.cpp) with namespace declaration!

Exercise - Part 2

- 3_refactor_monolithic_file.cpp project

1. Refactor to use namespaces (both constants.h and utilities.cpp and .h)