CS 11: Introduction to Computer Science

Separate Compilation
Organizing Programs into Multiple Source Files

Mark Sheldon
Tufts University
Email: msheldon@cs.tufts.edu
Web: http://www.cs.tufts.edu/~msheldon

Noah Mendelsohn
Tufts University
Email: noah@cs.tufts.edu
Web: http://www.cs.tufts.edu/~noah

Goals for this session

- **Understand why we'd want to break a program into multiple source files**
- **Understand how to do it!**

The largest C++ programs are *millions of lines long…*

Some problems with that:

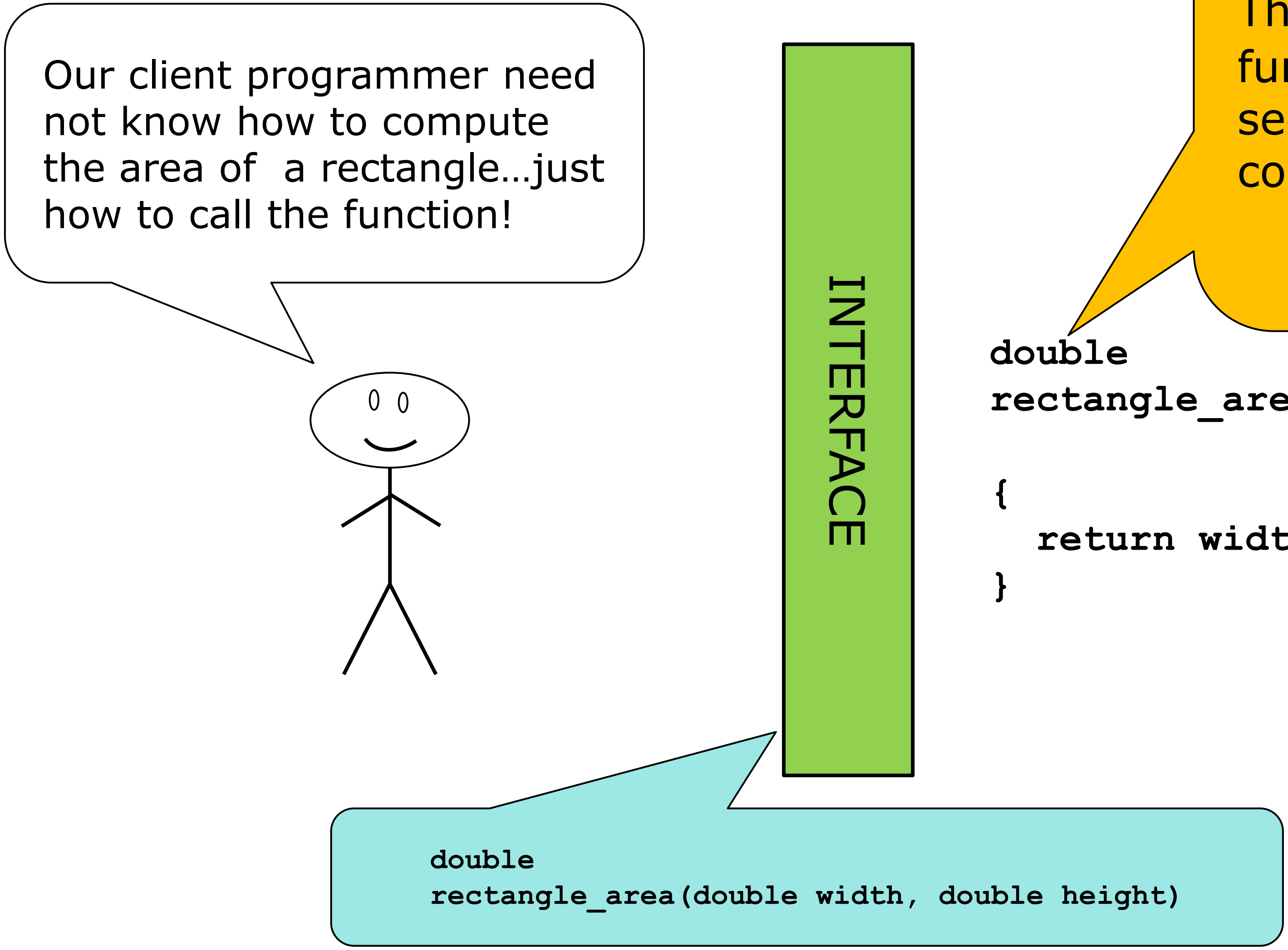
- *…*a million line .cpp file would be somewhat inconvenient
- …it's hard for 50 programmers to edit the same file at once
- …recompiling a 10 million line .cpp file might take awhile
- …*where's the modularity?*
- *…etc, etc.*

Reasons to break up even small programs

- **Modularity**
  - Keeping separate things that don't need to be tangled together
  - Easier to reason about
  - Easier to test and debug
  - I don't want to include many thousands of lines of C++ in my program just to call sqrt
- **Sharing**
  - I want to reuse the same little pieces in many different progams!

Interfaces

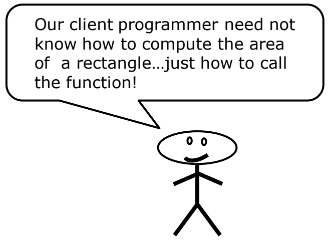
Client / Interface / Implementation


Our client programmer need not know how to compute the area of a rectangle…just how to call the function!
INTERFACE
The rectangle_area function "keeps the secret" of how to compute areas.

```
double
rectangle_area(double width,
                  double height)
{
  return width * height;
}
```

```
double
rectangle_area(double width, double height)
```

Client / Interface / Implementation


Our client programmer need not know how to compute the area of a rectangle…just how to call the function!

```
double
rectangle_area(double width,
                 double height);
main()
{
  double w=3;
  double h=4;
  cout << rectangle_area(w, h);
}
```

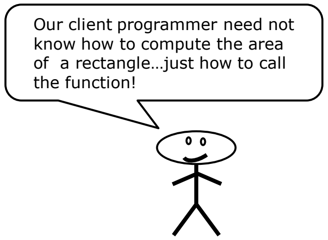
INTERFACE

The `rectangle_area` function "keeps the secret" of how to compute areas.

```
double
rectangle_area(double width,
                 double height)
{
  return width * height;
}
```

```
double
rectangle_area(double width, double height)
```

Client / Interface / Implementation


Our client programmer need not know how to compute the area of a rectangle…just how to call the function!

We declare the function interface in rectangle.h, which *both files include!*

```
#include "rectangle.h"

main()
{
  double w=3;
  double h=4;
  cout << rectangle_area(w, h);
}
```

INTE

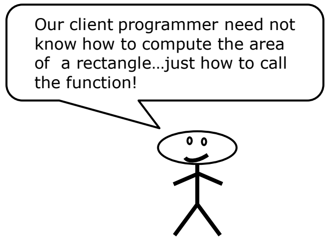
```
#include "rectangle.h"

double
rectangle_area(double width,
                  double height)
{
  return width * height;
}
```

```
double
rectangle_area(double width, double height);
```

rectangle.h

Client / Interface / Impleme

As we compile, the C++ compiler *includes the contents of rectangle.h*, as if it were in the .cpp being compiled.


Our client programmer need not know how to compute the area of a rectangle…just how to call the function!

```
#include "rectangle.h"

main()
{
  double w=3;
  double h=4;
  cout << rectangle_area(w, h);
}
```

INTERFACE

```
#include "rectangle.h"

double
rectangle_area(double width,
                  double height)
{
  return width * height;
}
```

```
double
rectangle_area(double width, double height);
```

rectangle.h

Client / Interface / Implementation

We could use `rectangle_area` again in a totally different application.

INTERFACE

```
#include "rectangle.h"

double
rectangle_area(double width,
                  double height)
{
  return width * height;
}
```

rectangle.cpp

```
double
rectangle_area(double width, double height)
```

rectangle.h

How To Compile
With Multiple Files

Compiling

```
g++  -Wall –Wextra –Werror  -o test_rectangle testrectangle.cpp rectangle.cpp
```

Compiling

Build executable program named:

test_rectangle

```
g++  -Wall –Wextra –Werror  -o test_rectangle testrectangle.cpp rectangle.cpp
```

Compiling

Using these *two .cpp files*

```
g++  -Wall –Wextra –Werror  -o test_rectangle testrectangle.cpp rectangle.cpp
```

Later we will learn some fancier ways to avoid recompiling all the pieces all the time, but for now this is simple and it works.

Summary

Building Programs from Multiple Source Files

- **Modularity**
- **Ease of maintenance**
- **Sharing**
- **Etc.**
- **Common interface goes into .h file #included by both implementation and user**
- **User does not see implementation details, which can be replaced**