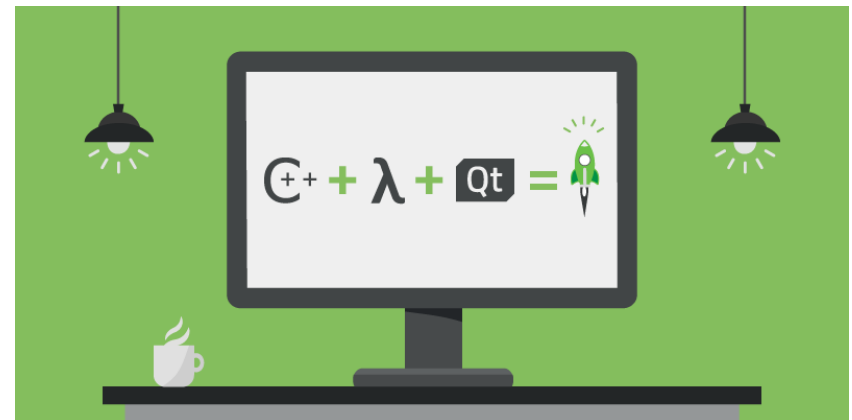
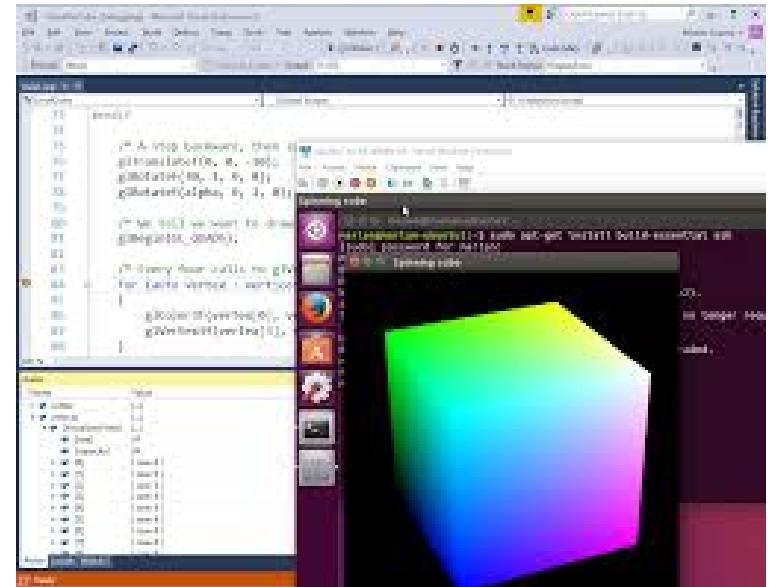


C/C++ Programming: Intro to C++ (1/3)



Historical Timeline

Origins:

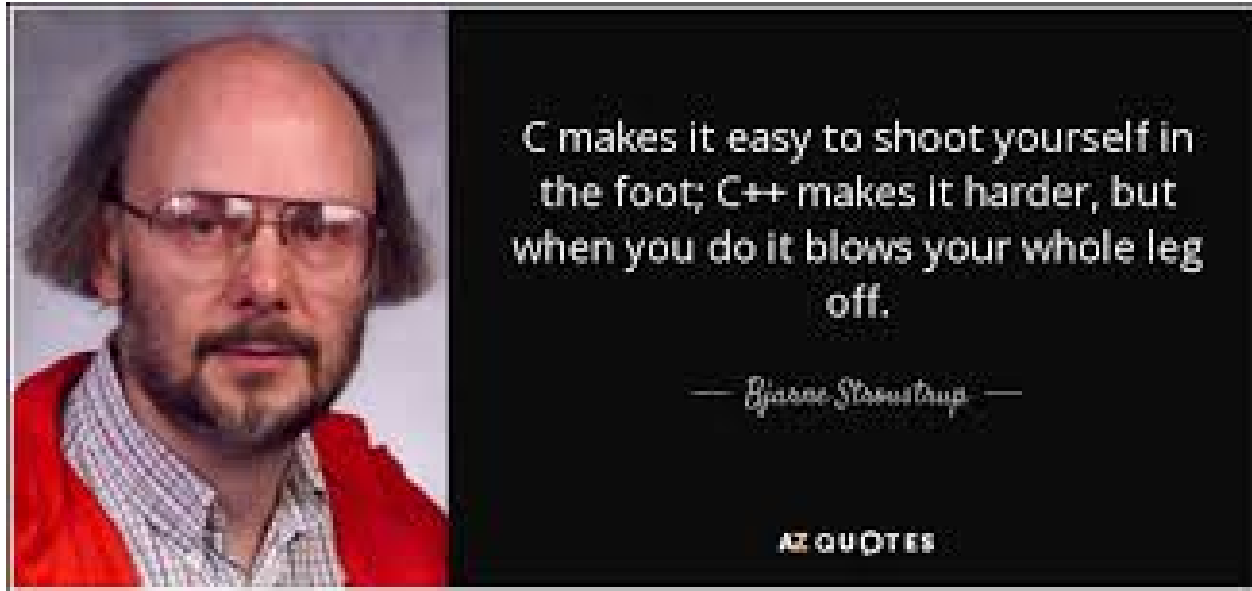
Danish computer scientist Bjarne Stroustrup began work on “C with Classes” in 1979. Working in AT&T Bell Labs, Stroustrup set out to enhance the C language, including **classes**, **derived classes**, strong typing, inlining and default arguments.

In 1983, “C with Classes” was renamed to “C++”, adding new features including **virtual functions**, function/operator **overloading**, type-safe free-store memory allocation (new/delete). Furthermore, it included the development of a standalone compiler for C++, Cfront.

In 1985, the first edition of The C++ Programming Language was released, which became the definitive reference for the language, as there was not yet an official standard. The first commercial implementation of C++ was released in October of the same year.

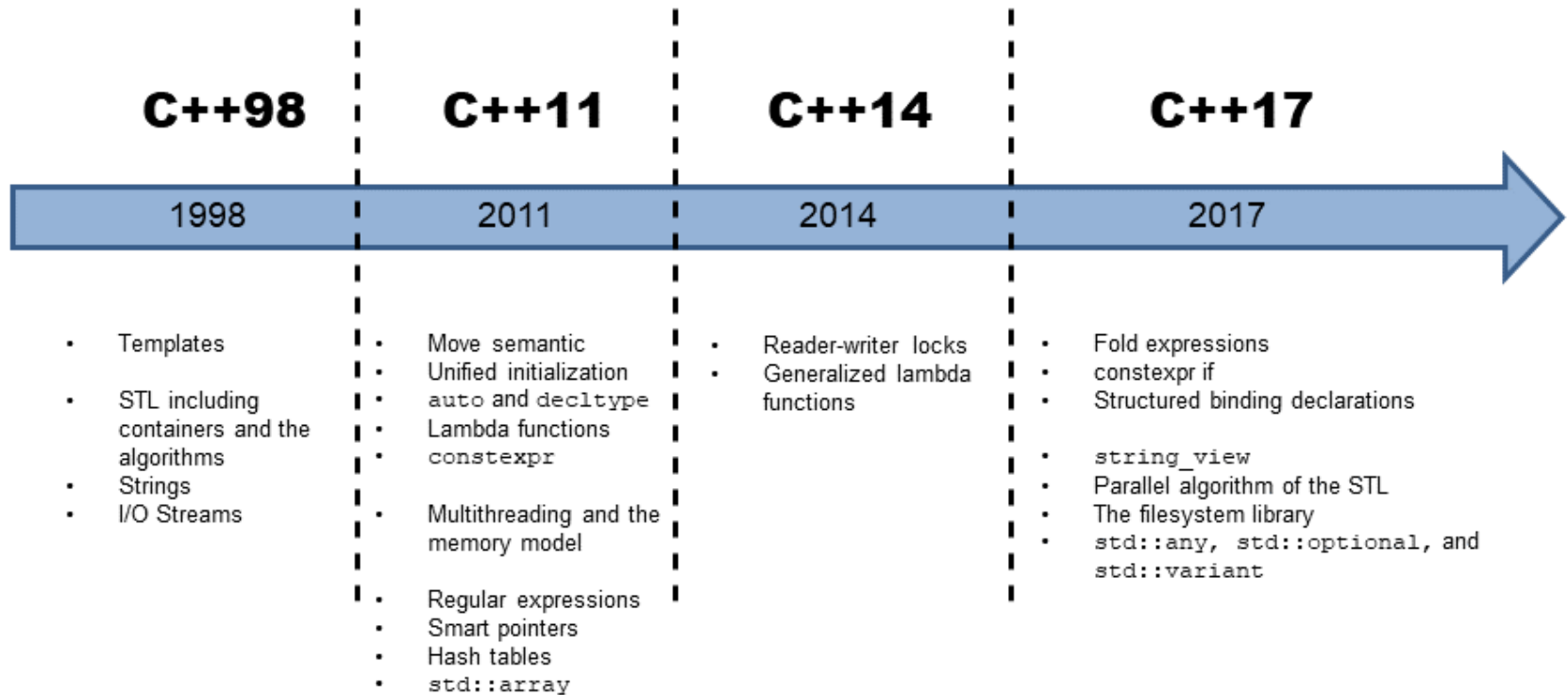
Historical Timeline

Bjarne Stroustrup:



Historical Timeline

Modern developments:



Where do we start?

C++ is an **object-oriented** programming language; C++ support for **classes** enables creation of “objects”.

A **class** is the **blueprint** to create a **software object**; it includes:

- Variables (data members);
- Functions (member function).

An **Object** is an **instance** of a **Class**, i.e. an actual variable that belongs to that class.

When a class is defined, no memory is allocated but when it is instantiated (i.e. an object is created) memory is allocated for that object.

Program with some Class

A **C++ class** has the features of a **C struct**, a collection of different variables (fields), and many additional ones, including:

- Member functions
- Public/private/protected members
- Constructor/destructor
- More advanced features: Inheritance, Virtualization, Friend classes

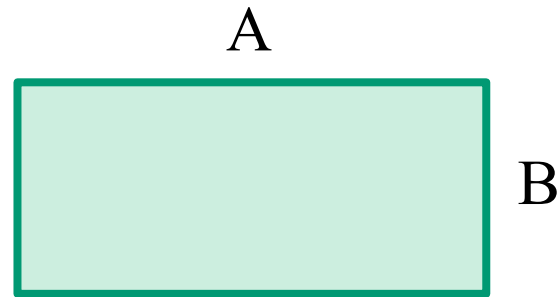
Example: “Rectangle”

Geometric object: “rectangle” is specified by defining:

- Length of two sides (A & B).

This (automatically) gives:

- Area ($\text{area} = A * B$);
- Perimeter ($\text{perimeter} = 2 * (A + B)$).



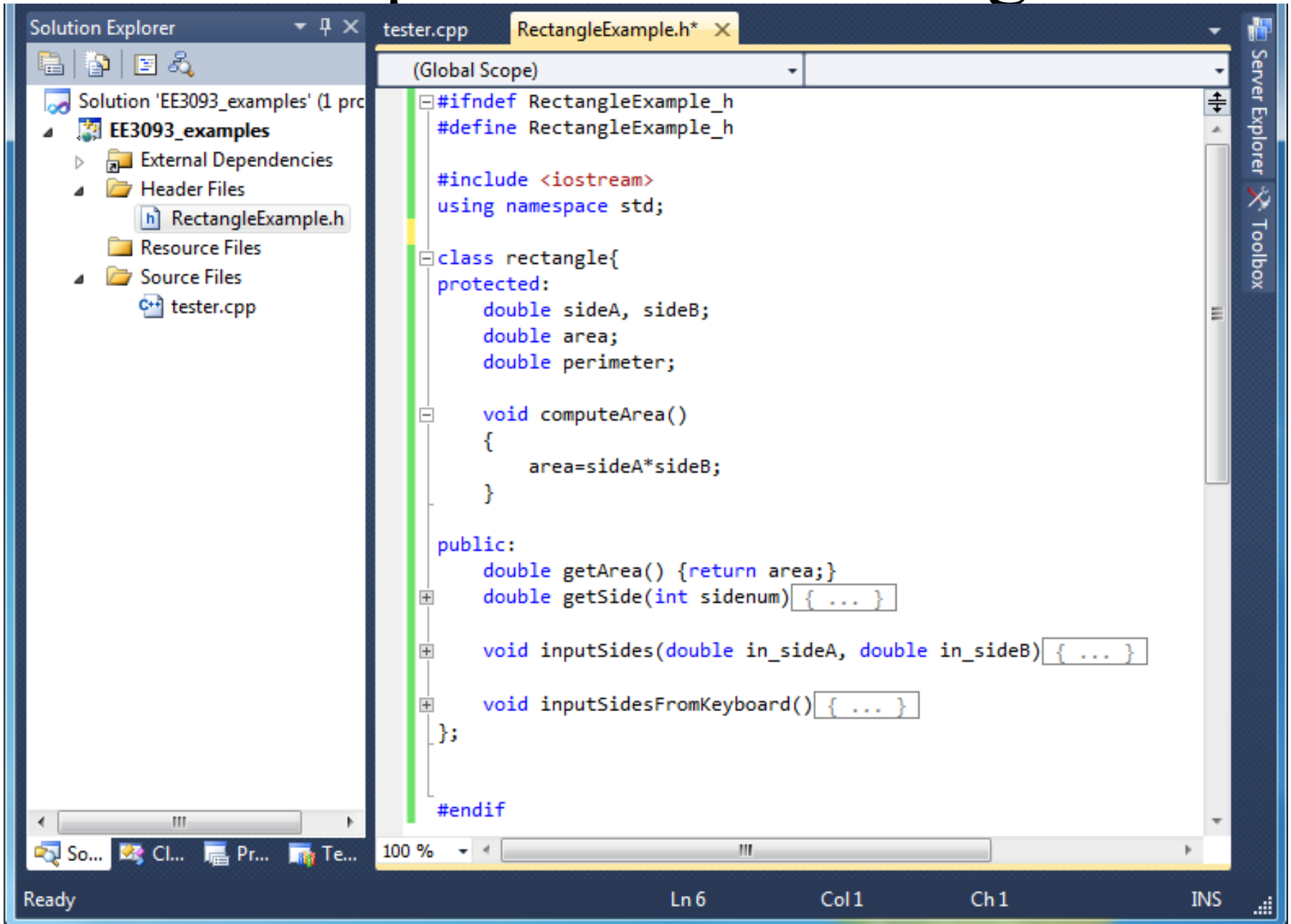
SW object: “rectangle” is specified by defining:

- Length of two sides (type *double*).

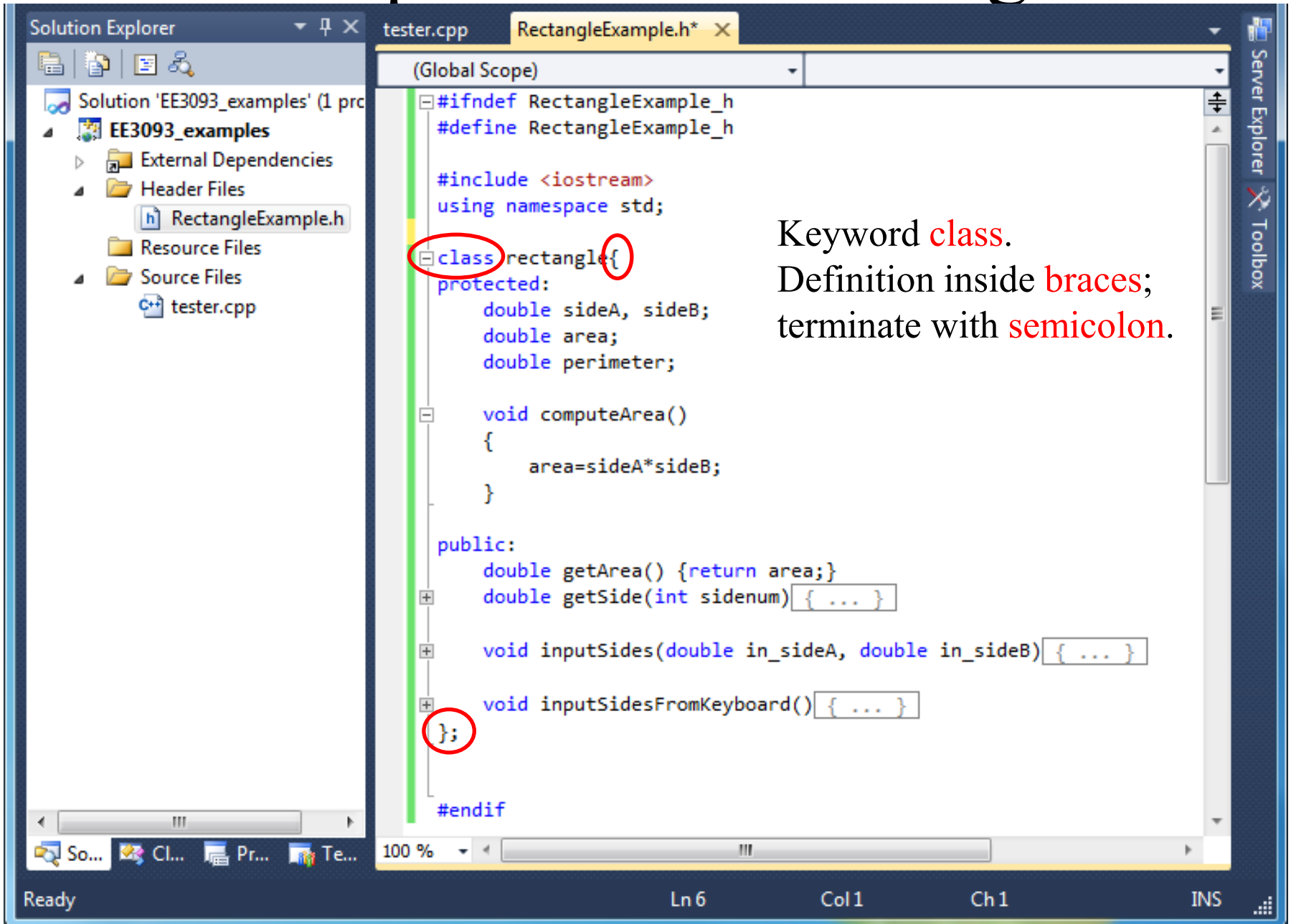
The object should then be able to (autonomously) compute:

- Area;
- Perimeter.

Example: class “Rectangle”



Example: class “Rectangle”



The screenshot shows a C++ IDE with a Solution Explorer on the left and a code editor on the right. The Solution Explorer displays a project named 'EE3093_examples' with folders for 'External Dependencies', 'Header Files', 'Resource Files', and 'Source Files'. The 'Header Files' folder contains 'RectangleExample.h', which is the active file in the code editor. The code editor shows the following C++ code:

```
(Global Scope)
#ifndef RectangleExample_h
#define RectangleExample_h

#include <iostream>
using namespace std;

class rectangle{
protected:
    double sideA, sideB;
    double area;
    double perimeter;

    void computeArea()
    {
        area=sideA*sideB;
    }

public:
    double getArea() {return area;}
    double getSide(int sidenum){ ... }

    void inputSides(double in_sideA, double in_sideB){ ... }

    void inputSidesFromKeyboard(){ ... }
};

#endif
```

Key features of the code are highlighted with red circles and text:

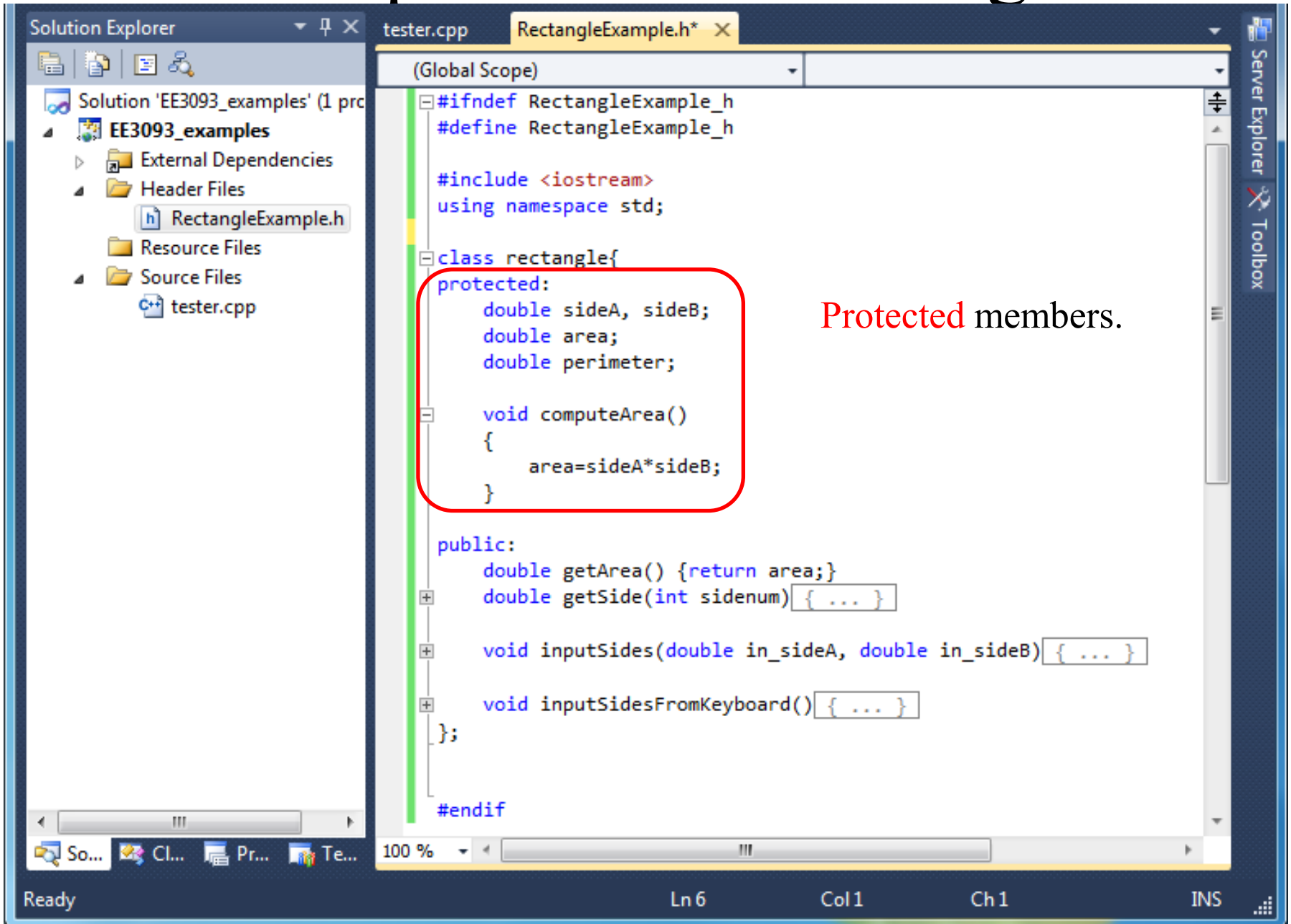
- The keyword **class** is circled in red.
- The opening curly brace **{** of the class definition is circled in red.
- The closing curly brace **}** and semicolon **;** of the class definition are circled in red.

Text annotations on the right side of the code editor state:

- Keyword **class**.
- Definition inside **braces**;
- terminate with **semicolon**.

The status bar at the bottom of the IDE shows 'Ready', '100 %', 'Ln 6', 'Col 1', 'Ch 1', and 'INS'.

Example: class “Rectangle”



The screenshot shows a C++ IDE with a Solution Explorer on the left and a code editor on the right. The Solution Explorer displays a project named 'EE3093_examples' with folders for 'External Dependencies', 'Header Files' (containing 'RectangleExample.h'), 'Resource Files', and 'Source Files' (containing 'tester.cpp'). The code editor shows the contents of 'RectangleExample.h' with the following code:

```
(Global Scope)
#ifndef RectangleExample_h
#define RectangleExample_h

#include <iostream>
using namespace std;

class rectangle{
protected:
    double sideA, sideB;
    double area;
    double perimeter;

    void computeArea()
    {
        area=sideA*sideB;
    }

public:
    double getArea() {return area;}
    double getSide(int sidenum){ ... }

    void inputSides(double in_sideA, double in_sideB){ ... }

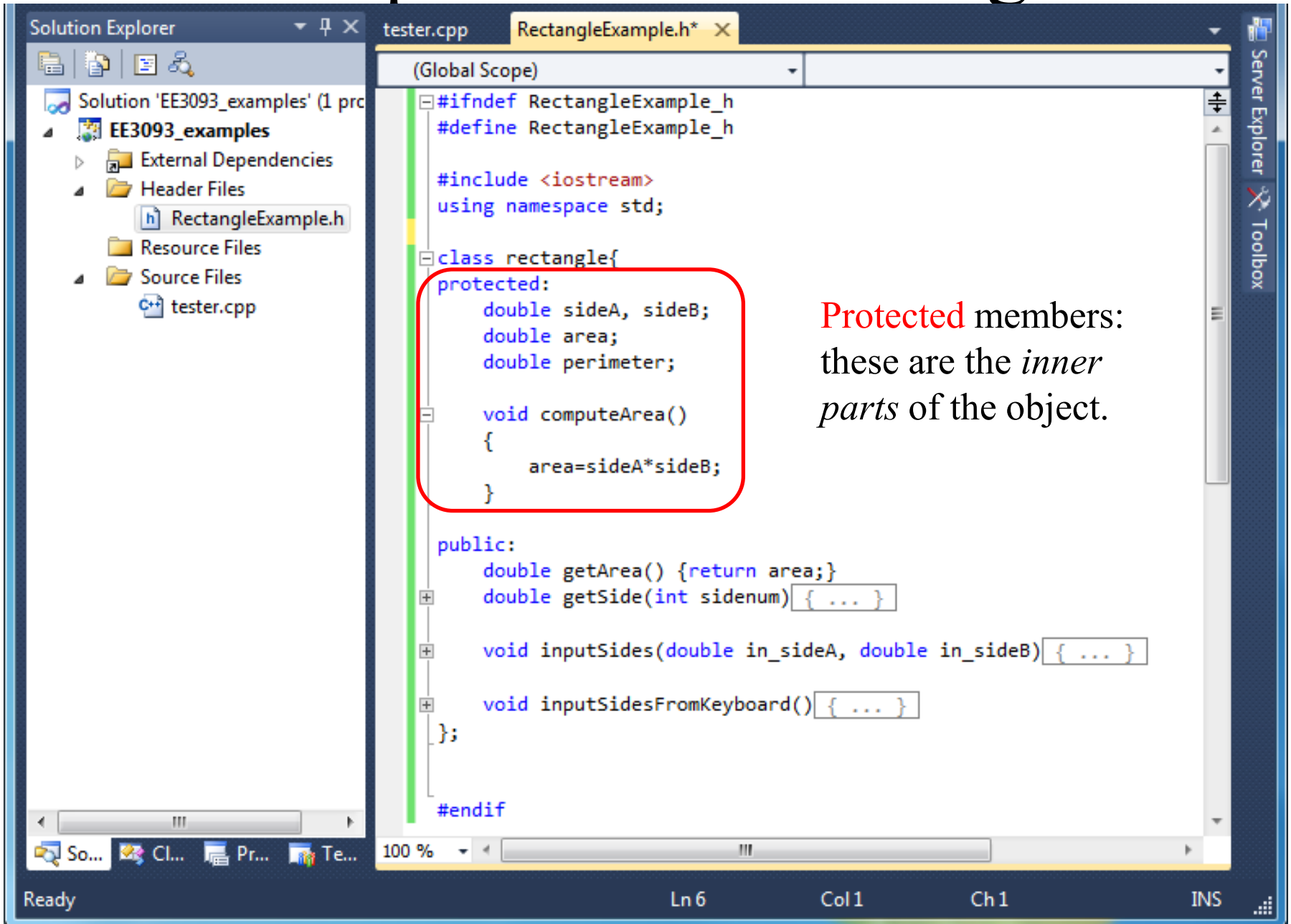
    void inputSidesFromKeyboard(){ ... }
};

#endif
```

A red rounded rectangle highlights the `protected:` section of the `rectangle` class, which contains the member variables `sideA`, `sideB`, `area`, and `perimeter`, and the `computeArea()` method. To the right of this highlighted section, the text "Protected members." is written in red.

The status bar at the bottom of the IDE shows "Ready", "100 %", and line/col/chapter indicators: "Ln 6", "Col 1", "Ch 1", and "INS".

Example: class “Rectangle”



The screenshot shows a C++ IDE with a Solution Explorer on the left and a code editor on the right. The Solution Explorer displays a project named 'EE3093_examples' with folders for 'External Dependencies', 'Header Files' (containing 'RectangleExample.h'), 'Resource Files', and 'Source Files' (containing 'tester.cpp'). The code editor shows the contents of 'RectangleExample.h'.

```
#ifndef RectangleExample_h
#define RectangleExample_h

#include <iostream>
using namespace std;

class rectangle{
protected:
    double sideA, sideB;
    double area;
    double perimeter;

    void computeArea()
    {
        area=sideA*sideB;
    }

public:
    double getArea() {return area;}
    double getSide(int sidenum){ ... }

    void inputSides(double in_sideA, double in_sideB){ ... }

    void inputSidesFromKeyboard(){ ... }
};

#endif
```

A red rounded rectangle highlights the `protected:` section of the class, which contains the `double` member variables (`sideA`, `sideB`, `area`, `perimeter`) and the `void computeArea()` method. To the right of the code editor, a text annotation reads: "Protected members: these are the *inner parts* of the object."

At the bottom of the IDE, the status bar shows 'Ready', '100 %', and line/column indicators 'Ln 6', 'Col 1', 'Ch 1', and 'INS'.

Example: class “Rectangle”

Solution Explorer

tester.cpp RectangleExample.h*

Solution 'EE3093_examples' (1 project)

- EE3093_examples
 - External Dependencies
 - Header Files
 - RectangleExample.h
 - Resource Files
 - Source Files
 - tester.cpp

(Global Scope)

```
#ifndef RectangleExample_h
#define RectangleExample_h

#include <iostream>
using namespace std;

class rectangle{
protected:
    double sideA, sideB;
    double area;
    double perimeter;

    void computeArea()
    {
        area=sideA*sideB;
    }


public:
    double getArea() {return area;}
    double getSide(int sidenum) { ... }

    void inputSides(double in_sideA, double in_sideB) { ... }

    void inputSidesFromKeyboard() { ... }
};

#endif
```

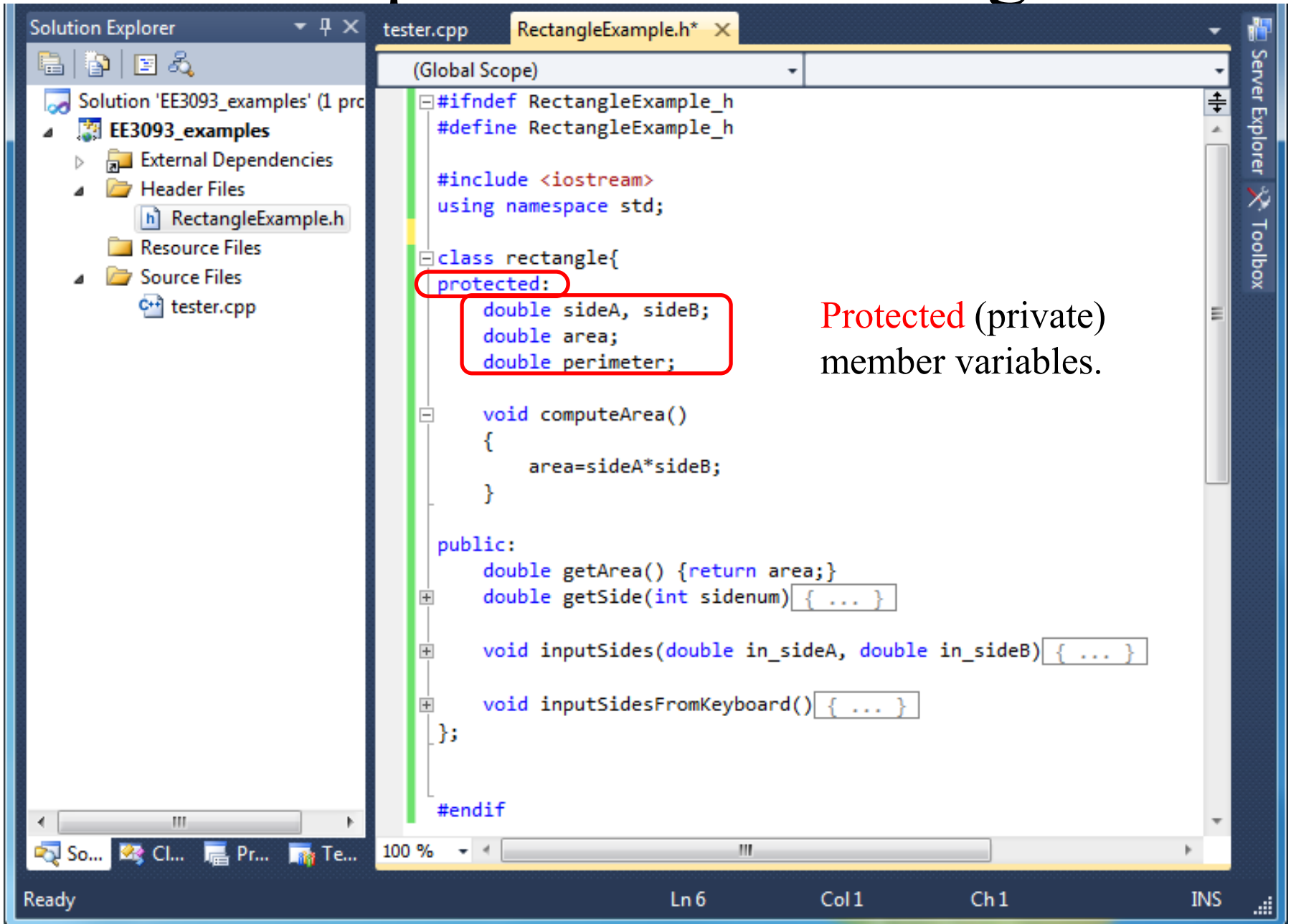
Protected members:



Ready

Ln 6 Col 1 Ch 1 INS

Example: class “Rectangle”



The screenshot shows a C++ IDE with a Solution Explorer on the left and a code editor on the right. The Solution Explorer displays a project named 'EE3093_examples' with folders for 'External Dependencies', 'Header Files', 'Resource Files', and 'Source Files'. The 'Header Files' folder contains 'RectangleExample.h', and the 'Source Files' folder contains 'tester.cpp'. The code editor shows the contents of 'RectangleExample.h' with the following code:

```
(Global Scope)
#ifndef RectangleExample_h
#define RectangleExample_h

#include <iostream>
using namespace std;

class rectangle{
protected:
    double sideA, sideB;
    double area;
    double perimeter;

    void computeArea()
    {
        area=sideA*sideB;
    }

public:
    double getArea() {return area;}
    double getSide(int sidenum){ ... }

    void inputSides(double in_sideA, double in_sideB){ ... }

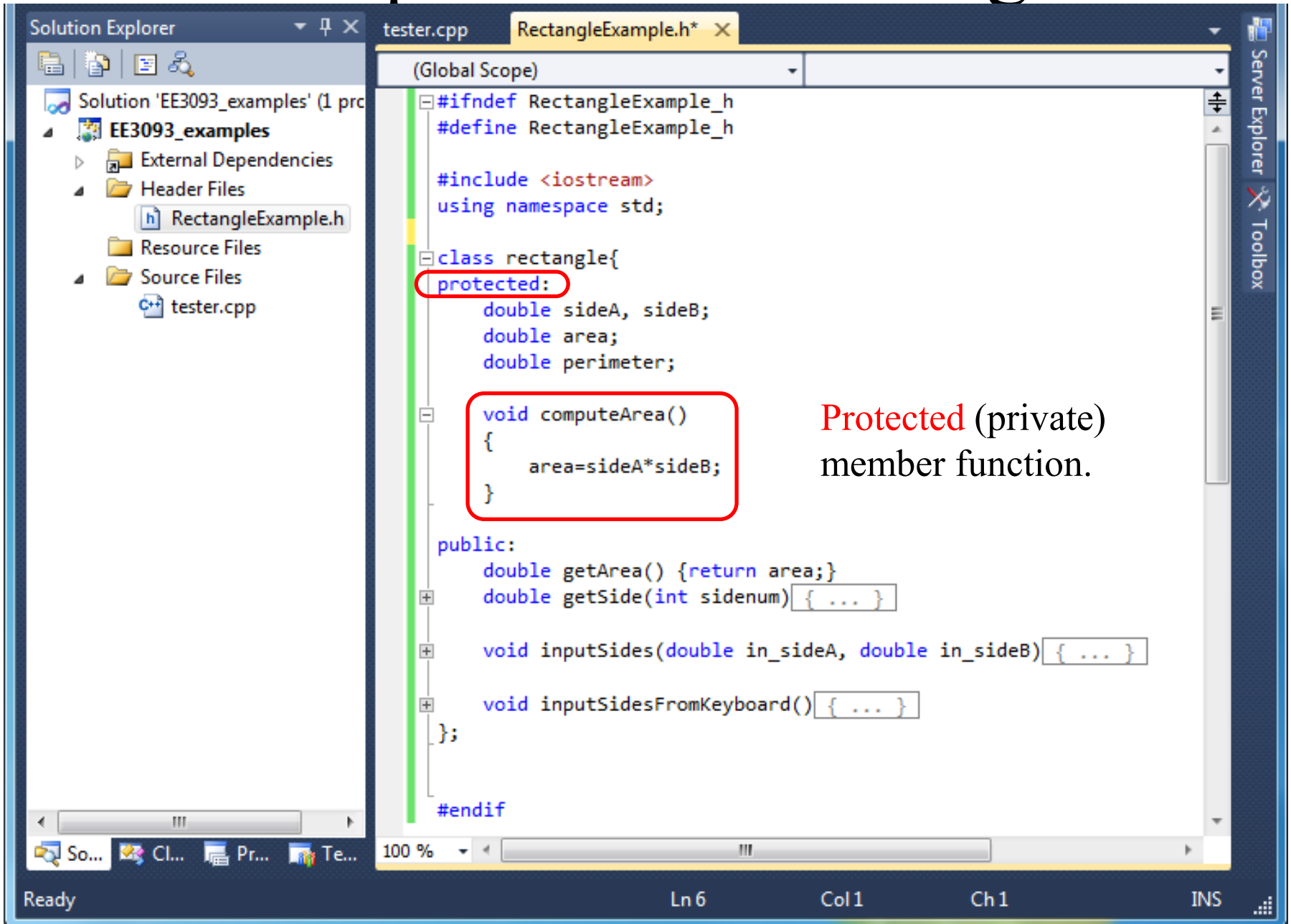
    void inputSidesFromKeyboard(){ ... }
};

#endif
```

The word **protected:** is highlighted with a red circle. To the right of the code, a text annotation reads: **Protected** (private) member variables.

The status bar at the bottom indicates 'Ready', '100 %', 'Ln 6', 'Col 1', 'Ch 1', and 'INS'.

Example: class “Rectangle”



Solution Explorer

tester.cpp RectangleExample.h*

(Global Scope)

```
#ifndef RectangleExample_h
#define RectangleExample_h

#include <iostream>
using namespace std;

class rectangle{
protected:
    double sideA, sideB;
    double area;
    double perimeter;

    void computeArea()
    {
        area=sideA*sideB;
    }

public:
    double getArea() {return area;}
    double getSide(int sidenum){ ... }

    void inputSides(double in_sideA, double in_sideB){ ... }

    void inputSidesFromKeyboard(){ ... }
};

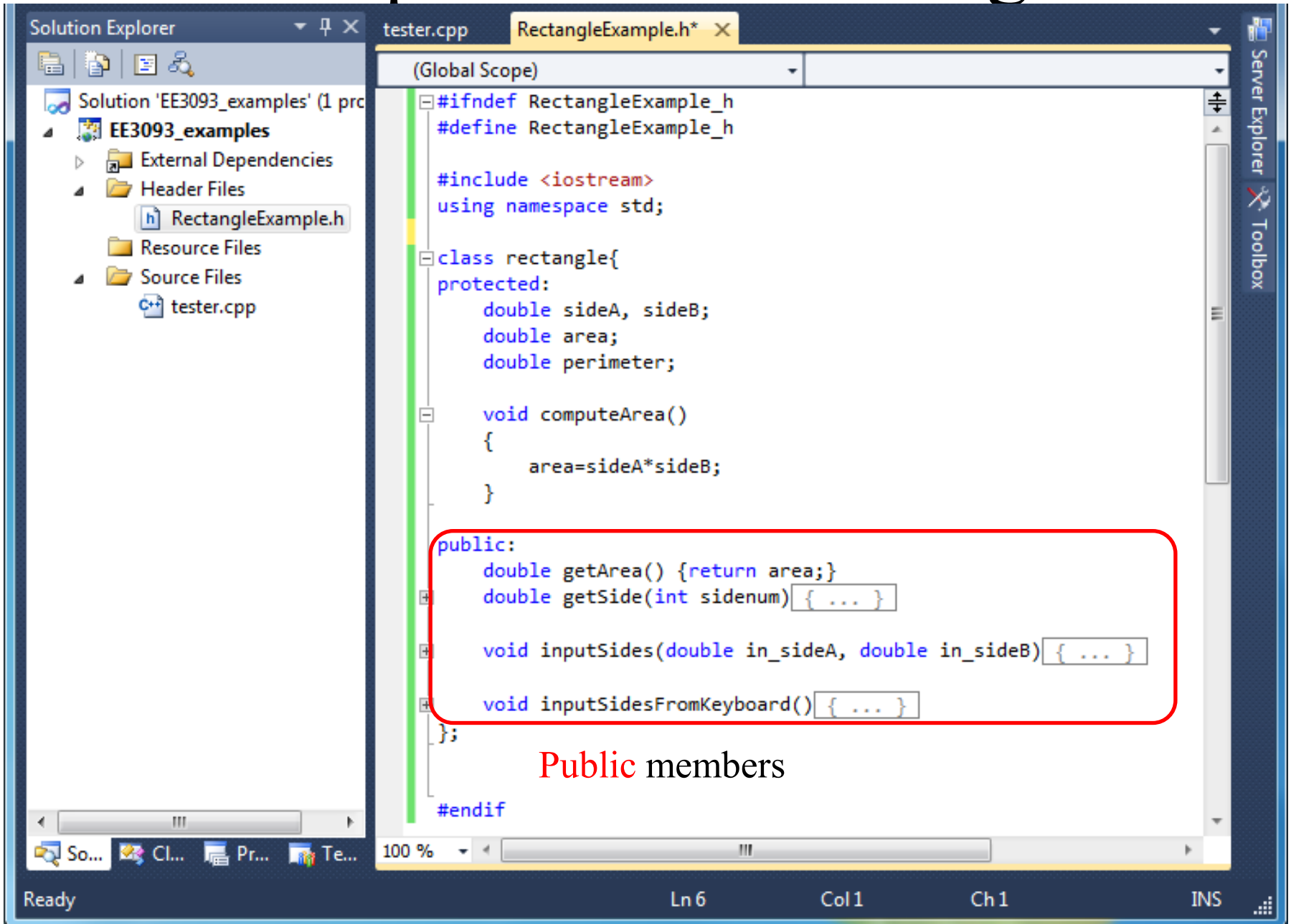
#endif
```

Protected (private) member function.

Ready

Ln 6 Col 1 Ch 1 INS

Example: class “Rectangle”



The screenshot shows a C++ IDE with a Solution Explorer on the left and a code editor on the right. The Solution Explorer displays a project named 'EE3093_examples' with folders for 'External Dependencies', 'Header Files', 'Resource Files', and 'Source Files'. The 'Header Files' folder contains 'RectangleExample.h', and the 'Source Files' folder contains 'tester.cpp'. The code editor shows the contents of 'RectangleExample.h' in the '(Global Scope)'. The code defines the 'Rectangle' class with private attributes and public methods. A red rectangle highlights the public methods: 'getArea()', 'getSide()', 'inputSides()', and 'inputSidesFromKeyboard()'. Below the code, the text 'Public members' is written in red.

```
#ifndef RectangleExample_h
#define RectangleExample_h

#include <iostream>
using namespace std;

class rectangle{
protected:
    double sideA, sideB;
    double area;
    double perimeter;

    void computeArea()
    {
        area=sideA*sideB;
    }

public:
    double getArea() {return area;}
    double getSide(int sidenum){ ... }

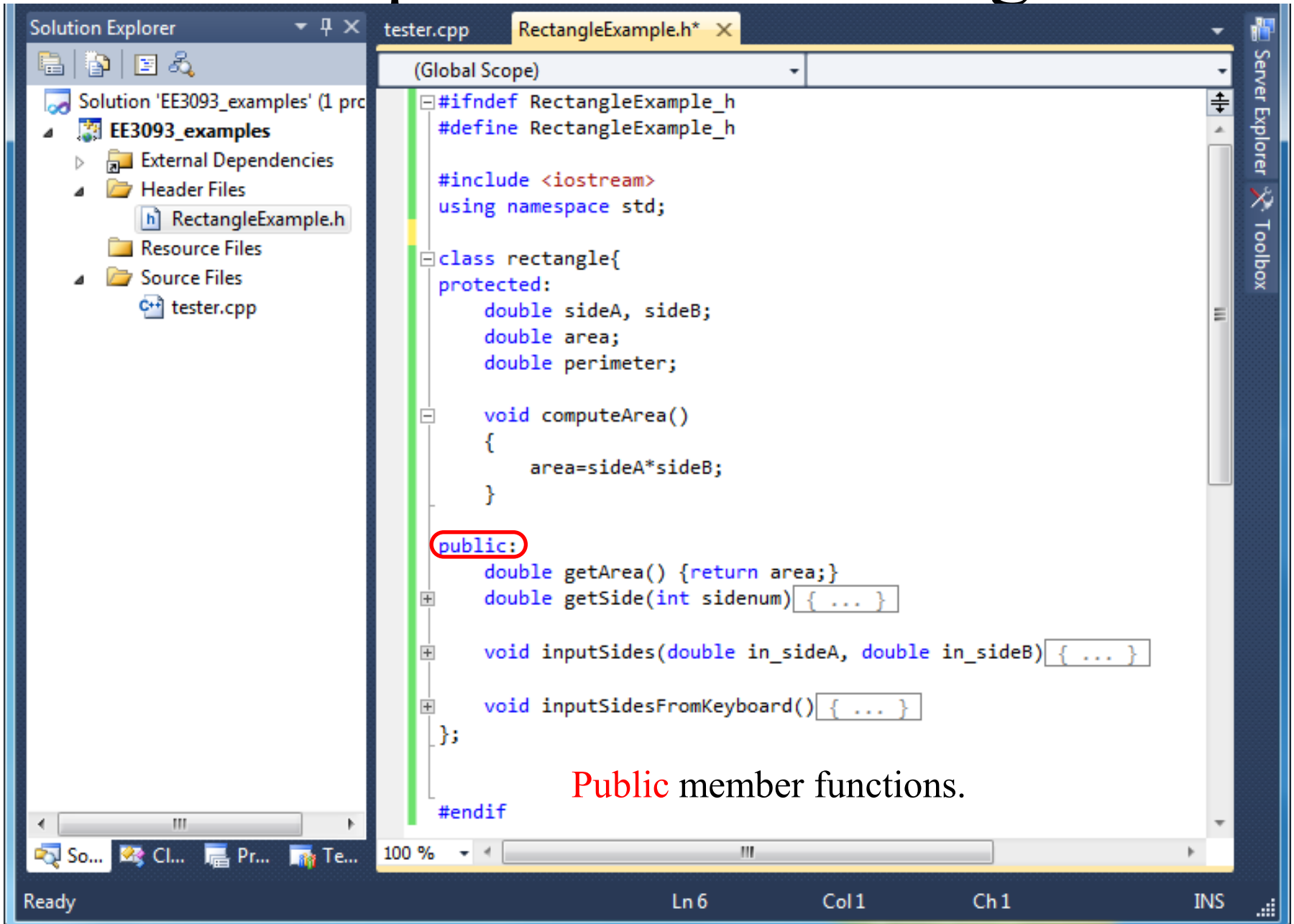
    void inputSides(double in_sideA, double in_sideB){ ... }

    void inputSidesFromKeyboard(){ ... }
};

#endif
```

Public members

Example: class “Rectangle”



The screenshot shows a C++ IDE with a Solution Explorer on the left and a code editor on the right. The Solution Explorer displays a project named 'EE3093_examples' with folders for 'External Dependencies', 'Header Files', 'Resource Files', and 'Source Files'. The 'Header Files' folder contains 'RectangleExample.h', and the 'Source Files' folder contains 'tester.cpp'. The code editor shows the contents of 'RectangleExample.h' in the '(Global Scope)'. The code defines the 'Rectangle' class with private attributes and public member functions. The 'public:' access specifier is highlighted with a red circle. The status bar at the bottom indicates 'Ready', '100 %', 'Ln 6', 'Col 1', 'Ch 1', and 'INS'.

```
#ifndef RectangleExample_h
#define RectangleExample_h

#include <iostream>
using namespace std;

class rectangle{
protected:
    double sideA, sideB;
    double area;
    double perimeter;

    void computeArea()
    {
        area=sideA*sideB;
    }

public:
    double getArea() {return area;}
    double getSide(int sidenum){ ... }

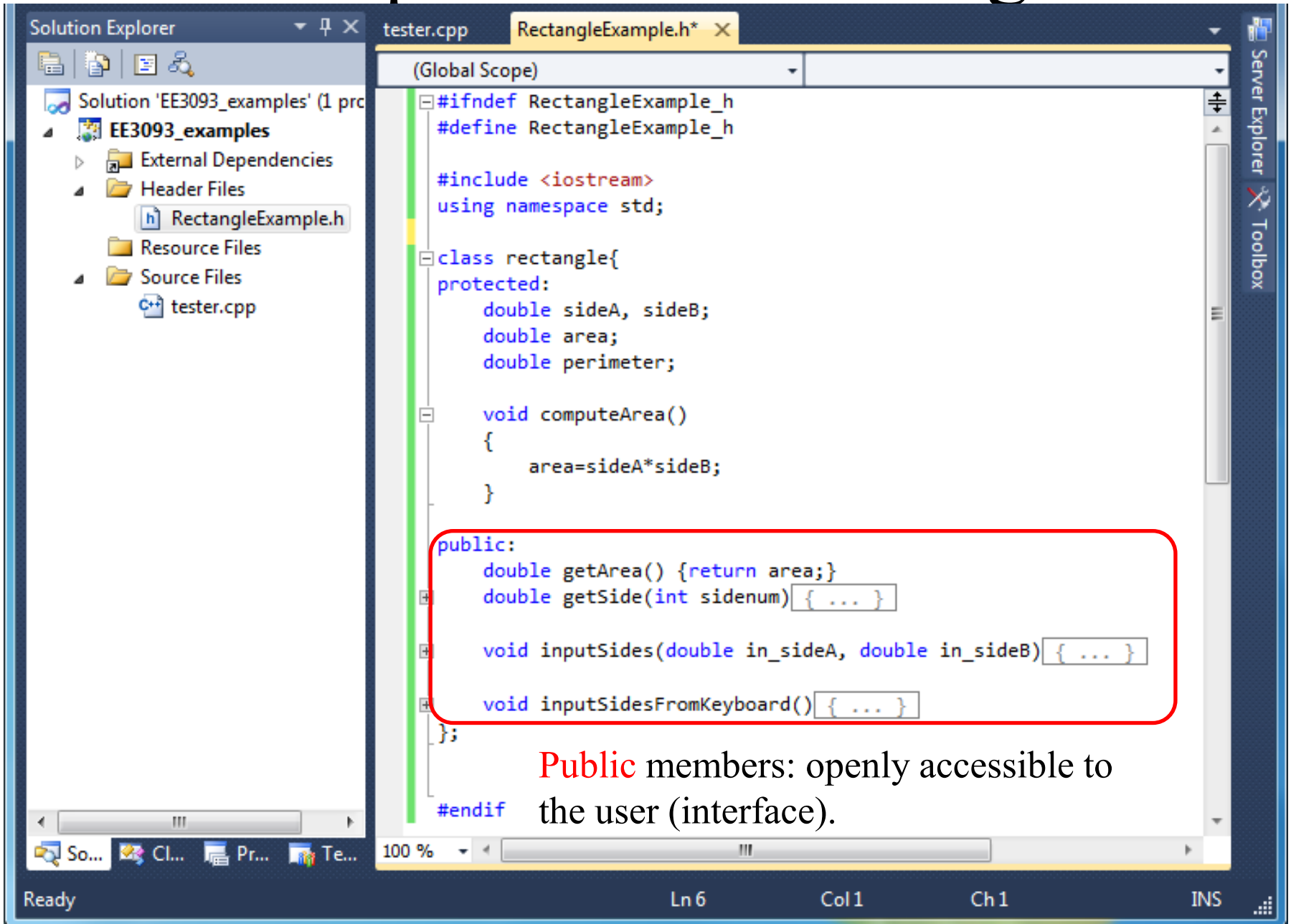
    void inputSides(double in_sideA, double in_sideB){ ... }

    void inputSidesFromKeyboard(){ ... }
};

#endif
```

Public member functions.

Example: class “Rectangle”



The screenshot shows a C++ IDE with a Solution Explorer on the left and a code editor on the right. The Solution Explorer displays a project named 'EE3093_examples' with folders for 'External Dependencies', 'Header Files', 'Resource Files', and 'Source Files'. The 'Header Files' folder contains 'RectangleExample.h', and the 'Source Files' folder contains 'tester.cpp'. The code editor shows the contents of 'RectangleExample.h' with the following code:

```
(Global Scope)
#ifndef RectangleExample_h
#define RectangleExample_h

#include <iostream>
using namespace std;

class rectangle{
protected:
    double sideA, sideB;
    double area;
    double perimeter;

    void computeArea()
    {
        area=sideA*sideB;
    }

public:
    double getArea() {return area;}
    double getSide(int sidenum){ ... }

    void inputSides(double in_sideA, double in_sideB){ ... }
    void inputSidesFromKeyboard(){ ... }
};

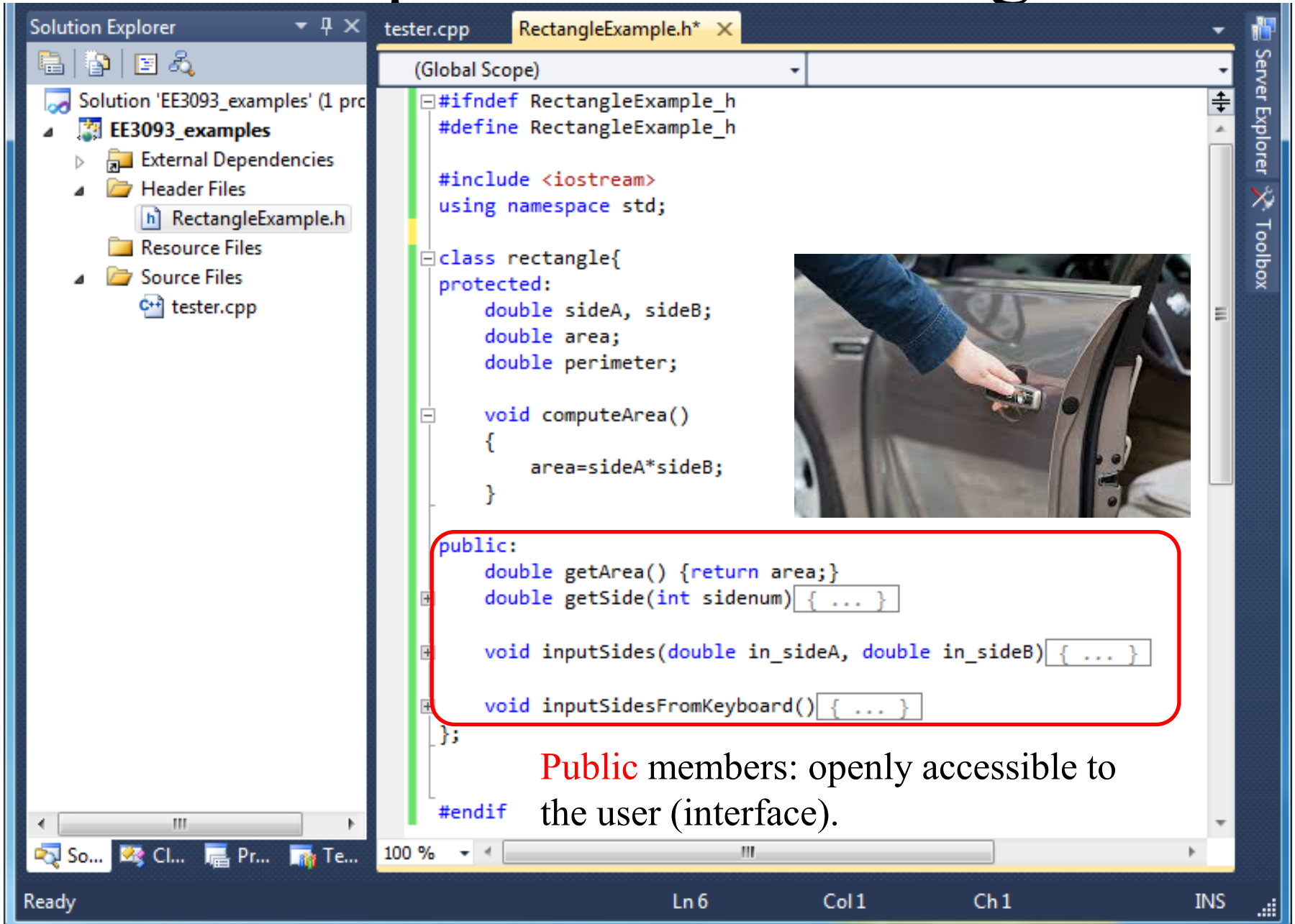
#endif
```

The 'public:' section of the class is highlighted with a red rectangle. Below the code editor, a text box explains the significance of the public members:

Public members: openly accessible to the user (interface).

The status bar at the bottom of the IDE shows 'Ready', '100 %', 'Ln 6', 'Col 1', 'Ch 1', and 'INS'.

Example: class “Rectangle”



The screenshot displays a C++ IDE with a solution named 'EE3093_examples'. The project structure in the Solution Explorer includes 'External Dependencies', 'Header Files' (containing 'RectangleExample.h'), 'Resource Files', and 'Source Files' (containing 'tester.cpp'). The main editor window shows the 'RectangleExample.h' file with the following code:

```
(Global Scope)
#ifndef RectangleExample_h
#define RectangleExample_h

#include <iostream>
using namespace std;

class rectangle{
protected:
    double sideA, sideB;
    double area;
    double perimeter;

    void computeArea()
    {
        area=sideA*sideB;
    }

public:
    double getArea() {return area;}
    double getSide(int sidenum){ ... }

    void inputSides(double in_sideA, double in_sideB){ ... }
    void inputSidesFromKeyboard(){ ... }
};

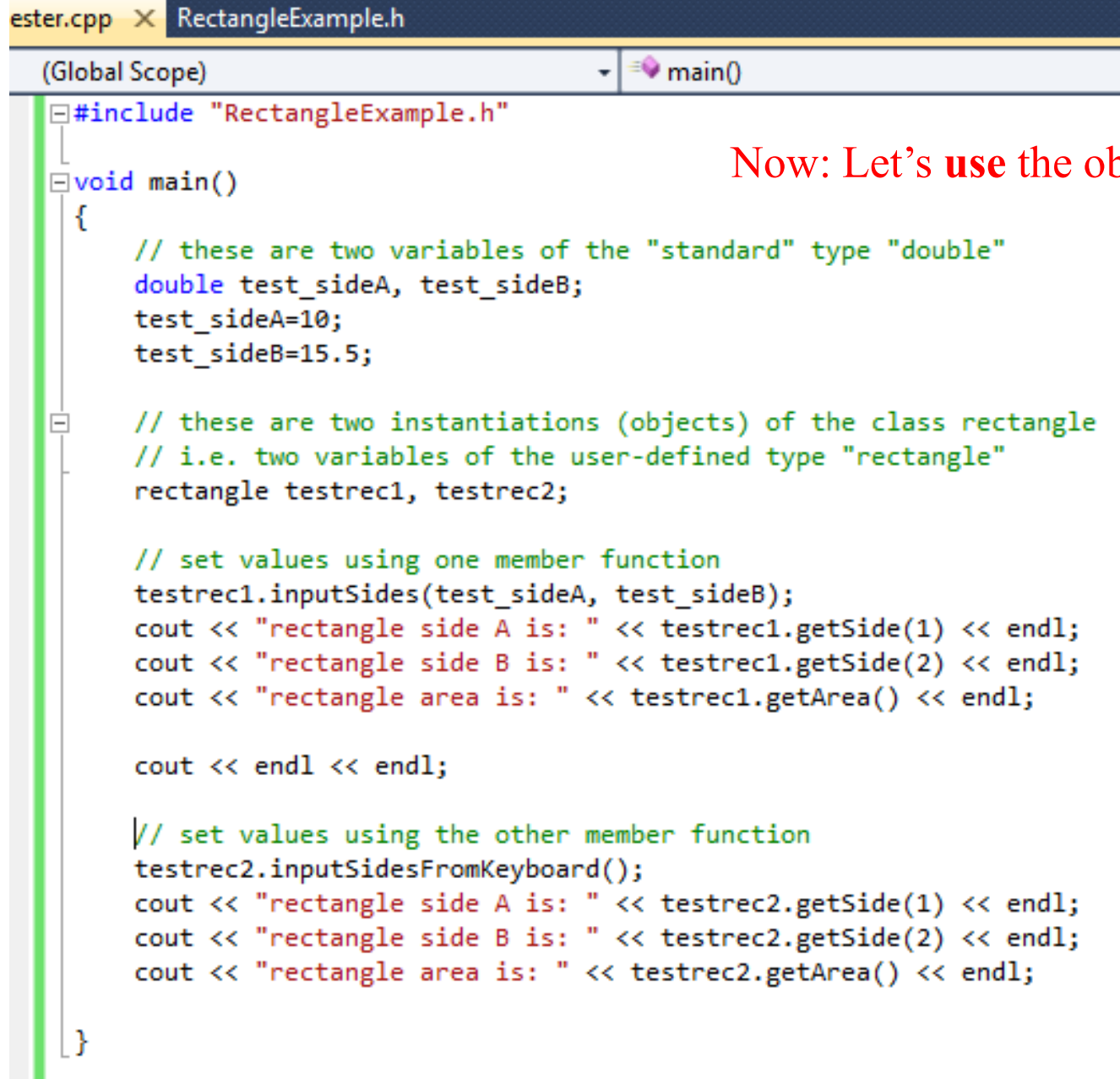
#endif
```

The public methods of the class are highlighted with a red box. An inset image shows a hand opening a car door, symbolizing access to the public interface.

Public members: openly accessible to the user (interface).

Ln 6 Col 1 Ch 1 INS

Example: class “Rectangle”



```
ester.cpp X RectangleExample.h
(Global Scope) main()
#include "RectangleExample.h"
void main()
{
    // these are two variables of the "standard" type "double"
    double test_sideA, test_sideB;
    test_sideA=10;
    test_sideB=15.5;

    // these are two instantiations (objects) of the class rectangle
    // i.e. two variables of the user-defined type "rectangle"
    rectangle testrec1, testrec2;

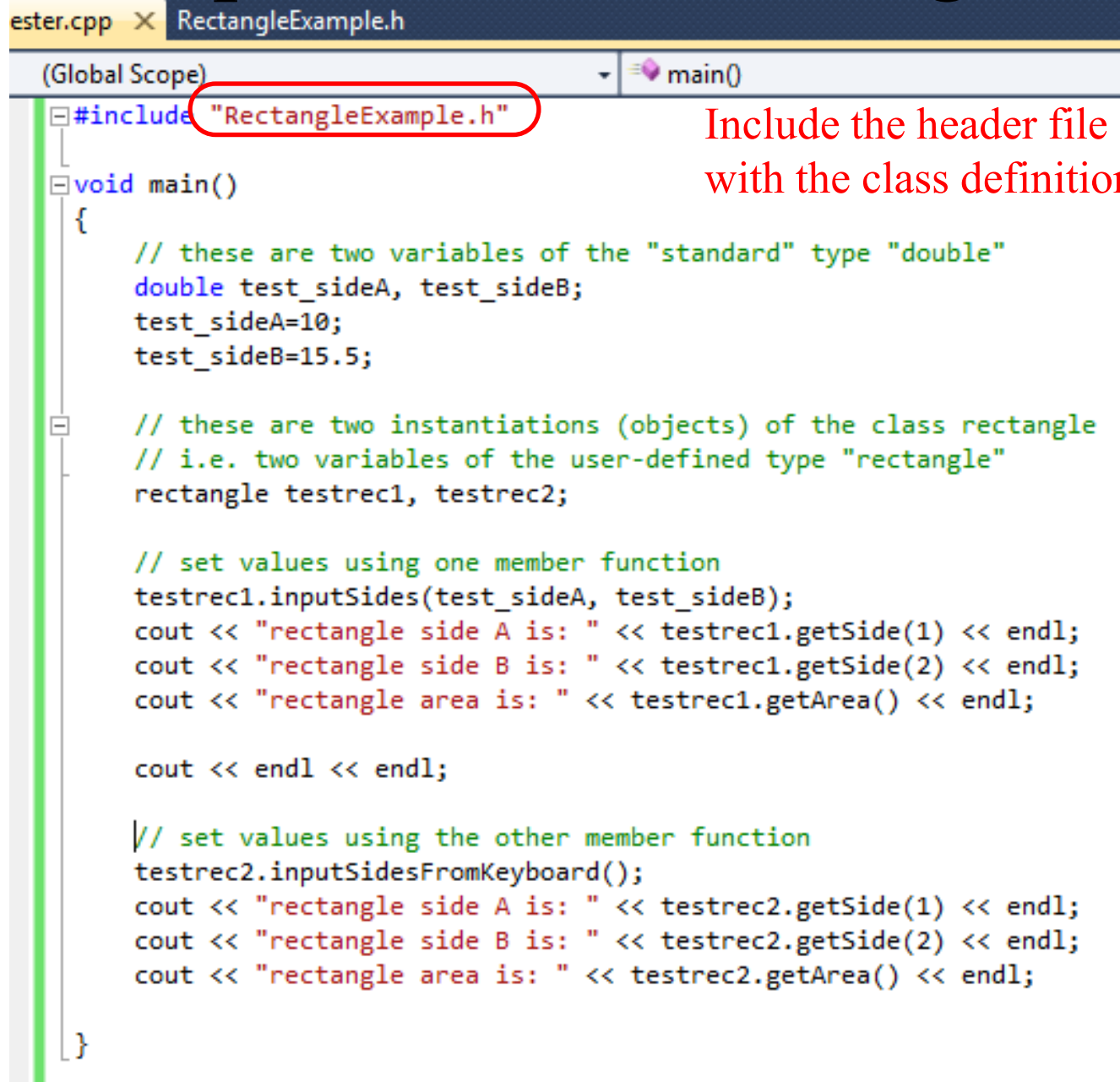
    // set values using one member function
    testrec1.inputSides(test_sideA, test_sideB);
    cout << "rectangle side A is: " << testrec1.getSide(1) << endl;
    cout << "rectangle side B is: " << testrec1.getSide(2) << endl;
    cout << "rectangle area is: " << testrec1.getArea() << endl;

    cout << endl << endl;

    // set values using the other member function
    testrec2.inputSidesFromKeyboard();
    cout << "rectangle side A is: " << testrec2.getSide(1) << endl;
    cout << "rectangle side B is: " << testrec2.getSide(2) << endl;
    cout << "rectangle area is: " << testrec2.getArea() << endl;
}
```

Now: Let's use the object!

Example: class “Rectangle”



```
ester.cpp x RectangleExample.h
(Global Scope) main()
#include "RectangleExample.h"
void main()
{
    // these are two variables of the "standard" type "double"
    double test_sideA, test_sideB;
    test_sideA=10;
    test_sideB=15.5;

    // these are two instantiations (objects) of the class rectangle
    // i.e. two variables of the user-defined type "rectangle"
    rectangle testrec1, testrec2;

    // set values using one member function
    testrec1.inputSides(test_sideA, test_sideB);
    cout << "rectangle side A is: " << testrec1.getSide(1) << endl;
    cout << "rectangle side B is: " << testrec1.getSide(2) << endl;
    cout << "rectangle area is: " << testrec1.getArea() << endl;

    cout << endl << endl;

    // set values using the other member function
    testrec2.inputSidesFromKeyboard();
    cout << "rectangle side A is: " << testrec2.getSide(1) << endl;
    cout << "rectangle side B is: " << testrec2.getSide(2) << endl;
    cout << "rectangle area is: " << testrec2.getArea() << endl;
}
```

Include the header file with the class definition.

Example: class “Rectangle”

```
ester.cpp X RectangleExample.h
(Global Scope) main()
#include "RectangleExample.h"
void main()
{
    // these are two variables of the "standard" type "double"
    double test_sideA, test_sideB;
    test_sideA=10;
    test_sideB=15.5;

    // these are two instantiations (objects) of the class rectangle
    // i.e. two variables of the user-defined type "rectangle"
    rectangle testrec1, testrec2;

    // set values using one member function
    testrec1.inputSides(test_sideA, test_sideB);
    cout << "rectangle side A is: " << testrec1.getSide(1) << endl;
    cout << "rectangle side B is: " << testrec1.getSide(2) << endl;
    cout << "rectangle area is: " << testrec1.getArea() << endl;

    cout << endl << endl;

    // set values using the other member function
    testrec2.inputSidesFromKeyboard();
    cout << "rectangle side A is: " << testrec2.getSide(1) << endl;
    cout << "rectangle side B is: " << testrec2.getSide(2) << endl;
    cout << "rectangle area is: " << testrec2.getArea() << endl;
}
```

Language-defined variable type **double**

User-defined variable type **rectangle**

Example: class “Rectangle”

```
ester.cpp X RectangleExample.h
(Global Scope) main()
#include "RectangleExample.h"
void main()
{
    // these are two variables of the "standard" type "double"
    double test_sideA, test_sideB;
    test_sideA=10;
    test_sideB=15.5;

    // these are two instantiations (objects) of the class rectangle
    // i.e. two variables of the user-defined type "rectangle"
    rectangle testrec1, testrec2;
    // set values using one member function
    testrec1.inputSides(test_sideA, test_sideB);
    cout << "rectangle side A is: " << testrec1.getSide(1) << endl;
    cout << "rectangle side B is: " << testrec1.getSide(2) << endl;
    cout << "rectangle area is: " << testrec1.getArea() << endl;

    cout << endl << endl;

    // set values using the other member function
    testrec2.inputSidesFromKeyboard();
    cout << "rectangle side A is: " << testrec2.getSide(1) << endl;
    cout << "rectangle side B is: " << testrec2.getSide(2) << endl;
    cout << "rectangle area is: " << testrec2.getArea() << endl;
}
```

“input” functions allow the user *supervised input access* to member variables

// set values using one member function

testrec1.inputSides(test_sideA, test_sideB);

cout << "rectangle side A is: " << testrec1.getSide(1) << endl;

cout << "rectangle side B is: " << testrec1.getSide(2) << endl;

cout << "rectangle area is: " << testrec1.getArea() << endl;

cout << endl << endl;

// set values using the other member function

testrec2.inputSidesFromKeyboard();

cout << "rectangle side A is: " << testrec2.getSide(1) << endl;

cout << "rectangle side B is: " << testrec2.getSide(2) << endl;

cout << "rectangle area is: " << testrec2.getArea() << endl;

Example: class “Rectangle”

```
ester.cpp X RectangleExample.h
(Global Scope) main()
#include "RectangleExample.h"
void main()
{
    // these are two variables of the "standard" type "double"
    double test_sideA, test_sideB;
    test_sideA=10;
    test_sideB=15.5;

    // these are two instantiations (objects) of the class rectangle
    // i.e. two variables of the user-defined type "rectangle"
    rectangle testrec1, testrec2;

    // set values using one member function
    testrec1.inputSides(test_sideA, test_sideB);
    cout << "rectangle side A is: " << testrec1.getSide(1) << endl;
    cout << "rectangle side B is: " << testrec1.getSide(2) << endl;
    cout << "rectangle area is: " << testrec1.getArea() << endl;

    cout << endl << endl;

    // set values using the other member function
    testrec2.inputSidesFromKeyboard();
    cout << "rectangle side A is: " << testrec2.getSide(1) << endl;
    cout << "rectangle side B is: " << testrec2.getSide(2) << endl;
    cout << "rectangle area is: " << testrec2.getArea() << endl;
}
```

“get” functions allow the user *read access* to member variables (not allowed to change)

Example: class “Rectangle”

```
void main()
{
    // these are two variables of the "standard" type "double"
    double test_sideA, test_sideB;
    test_sideA=10;
    test_sideB=15.5;

    // these are two instantiations (objects) of the class rectangle
    // i.e. two variables of the user-defined type "rectangle"
    rectangle testrec1, testrec2;

    // this is not allowed: variable member "sideA" is NOT public; the user cannot access it directly
    testrec1.sideA=10;

    // set values using one member function
    testrec1.inputSides(test_sideA, test_sideB);
    cout << "rectangle 1: " << endl;
    cout << "rectangle side A is: " << testrec1.getSide(1) << endl;
    cout << "rectangle side B is: " << testrec1.getSide(2) << endl;
    cout << "rectangle area is: " << testrec1.getArea() << endl;
    //cout << "rectangle perimeter is: " << testrec1.getPerimeter() << endl;
    cout << " Count of rectangle objects is: " << testrec1.getObjectCount() << endl;

    cout << endl << endl;
}
```

Variables that are **NOT public** cannot be accessed directly (e.g. in a line of code inside main): these are **only accessible** from **within a member function**

Example: class “Rectangle”

```
void main()
{
    // these are two variables of the "standard" type "double"
    double test_sideA, test_sideB;
    test_sideA=10;
    test_sideB=15.5;

    // these are two instantiations (objects) of the class rectangle
    // i.e. two variables of the user-defined type "rectangle"
    rectangle testrec1, testrec2;

    // this is not allowed: function member computeArea() is NOT public; the user cannot access it directly
    testrec1.computeArea();

    // set values using one member function
    testrec1.inputSides(test_sideA, test_sideB);
    cout << "rectangle 1: " << endl;
    cout << "rectangle side A is: " << testrec1.getSide(1) << endl;
    cout << "rectangle side B is: " << testrec1.getSide(2) << endl;
    cout << "rectangle area is: " << testrec1.getArea() << endl;
    //cout << "rectangle perimeter is: " << testrec1.getPerimeter() << endl;
    cout << " Count of rectangle objects is: " << testrec1.getObjectCount() << endl;

    cout << endl << endl;
}
```

Similarly for member functions that are NOT public

Any question?



Example: class “Rectangle”

```
tester.cpp  RectangleExample.h X
rectangle  getSide(int sidenum)

public:
    double getArea() {return area;}
    double getSide(int sidenum)
    {
        double out;
        switch(sidenum){
            case 1:
                out=sideA;
                break;
            case 2:
                out=sideB;
                break;
            default:
                cout << "Error in getSide(): Incorrect sidenum values" << endl;
                out=-1;
        }
        return out;
    }

    void inputSides(double in_sideA, double in_sideB) { ... }

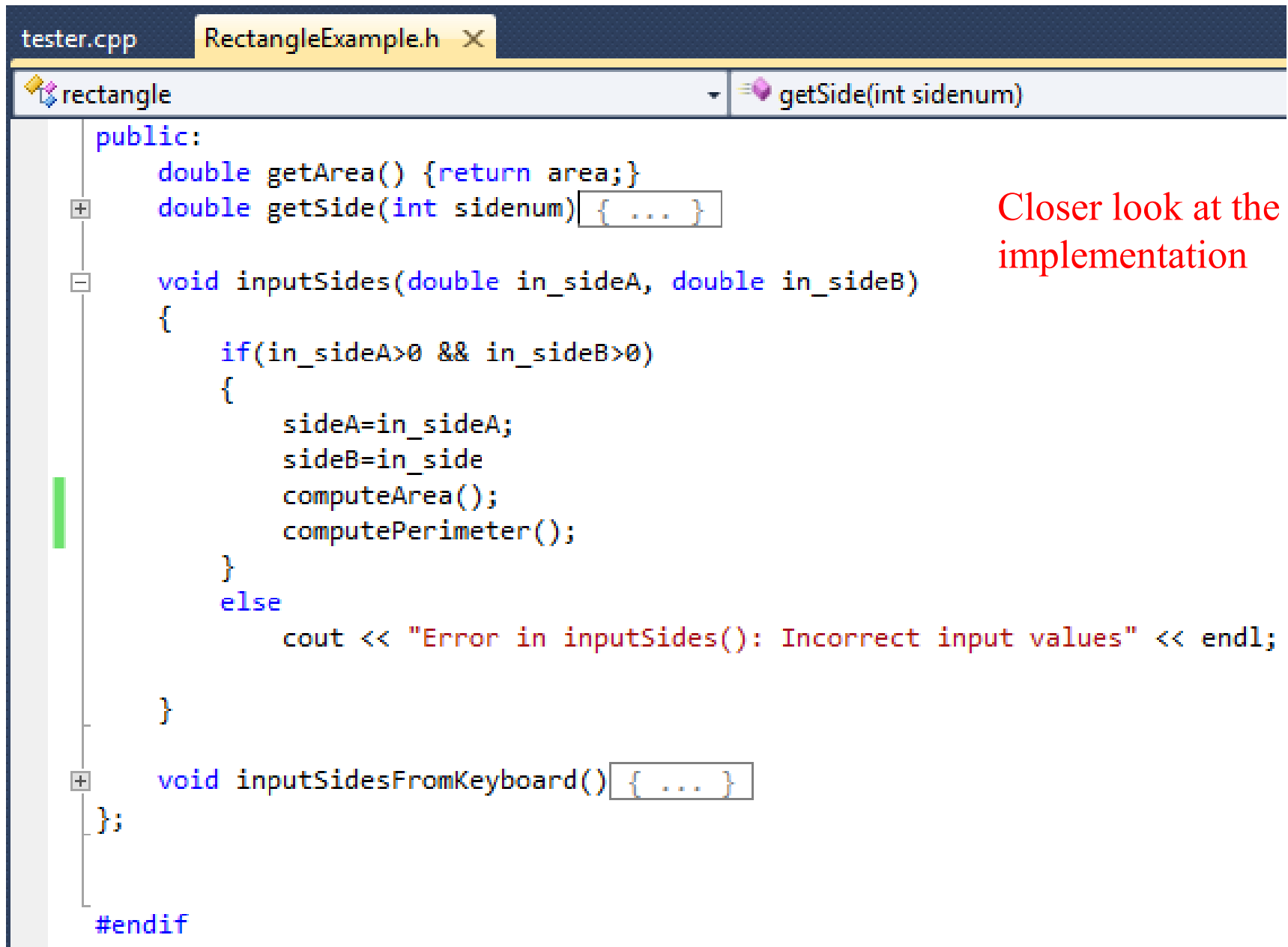
    void inputSidesFromKeyboard() { ... }

};

#endif
```

Closer look at the implementation

Example: class “Rectangle”



```
tester.cpp  RectangleExample.h X
rectangle  getSide(int sidenum)

public:
    double getArea() {return area;}
    double getSide(int sidenum) { ... }

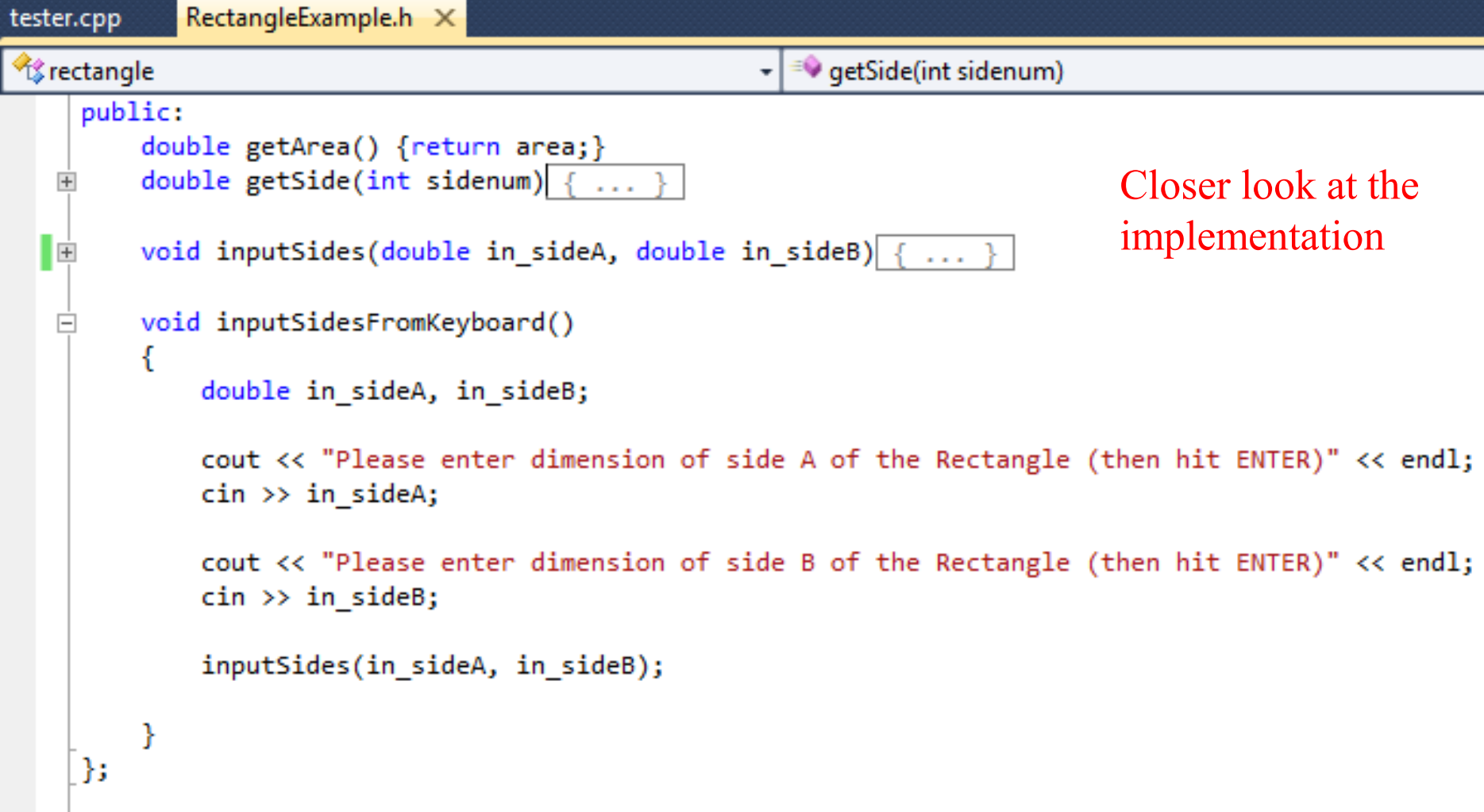
    void inputSides(double in_sideA, double in_sideB)
    {
        if(in_sideA>0 && in_sideB>0)
        {
            sideA=in_sideA;
            sideB=in_side
            computeArea();
            computePerimeter();
        }
        else
            cout << "Error in inputSides(): Incorrect input values" << endl;
    }

    void inputSidesFromKeyboard() { ... }
};

#endif
```

Closer look at the implementation

Example: class “Rectangle”



The screenshot shows a code editor with two tabs: 'tester.cpp' and 'RectangleExample.h'. The 'RectangleExample.h' tab is active, showing the implementation of a 'rectangle' class. The class has a public section with four methods: 'getArea()', 'getSide(int sidenum)', 'inputSides(double in_sideA, double in_sideB)', and 'inputSidesFromKeyboard()'. The 'getSide' method is highlighted with a mouse cursor. To the right of the code, there is a red text annotation: 'Closer look at the implementation'.

```
tester.cpp  RectangleExample.h X
rectangle  getSide(int sidenum)

public:
    double getArea() {return area;}
    double getSide(int sidenum) { ... }
    void inputSides(double in_sideA, double in_sideB) { ... }
    void inputSidesFromKeyboard()
    {
        double in_sideA, in_sideB;

        cout << "Please enter dimension of side A of the Rectangle (then hit ENTER)" << endl;
        cin >> in_sideA;

        cout << "Please enter dimension of side B of the Rectangle (then hit ENTER)" << endl;
        cin >> in_sideB;

        inputSides(in_sideA, in_sideB);
    }
};
```

Closer look at the implementation

Any question?

Relevant topics to ask questions:

- C++ Objects; member variables and functions
- C structs (difference with Object)
- C functions (difference with Object member functions)



Example: class “Rectangle”

Improve the implementation:

- Compute perimeter
- Return perimeter to the user



Notes on using the **iostream** library objects in the following appendix

Appendix: iostream

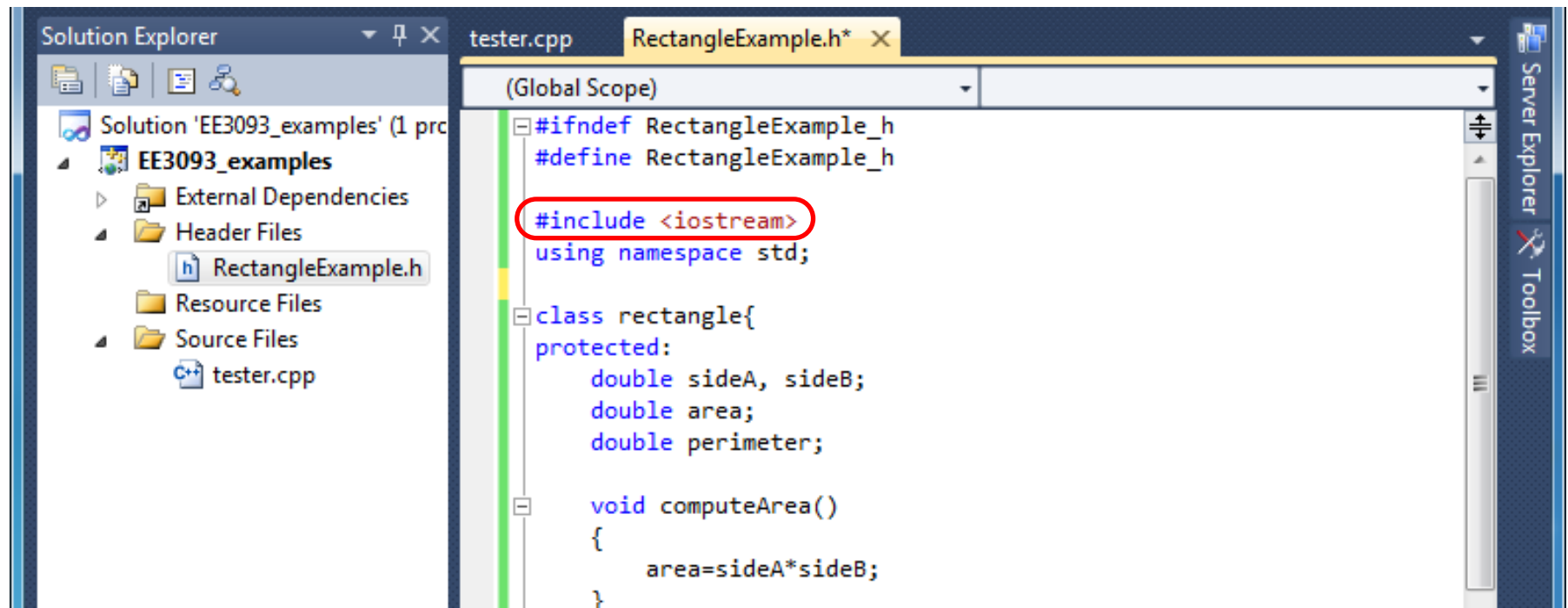
Equivalent to the input/output functionality provided in C by `stdio.h`, **iostream** provides basic input/output services for C++ programs.

The main **objects** defined in `iostream` are **cin** and **cout** that allow users to send/receive data to/from standard streams input/output.

Appendix: iostream

How to use it:

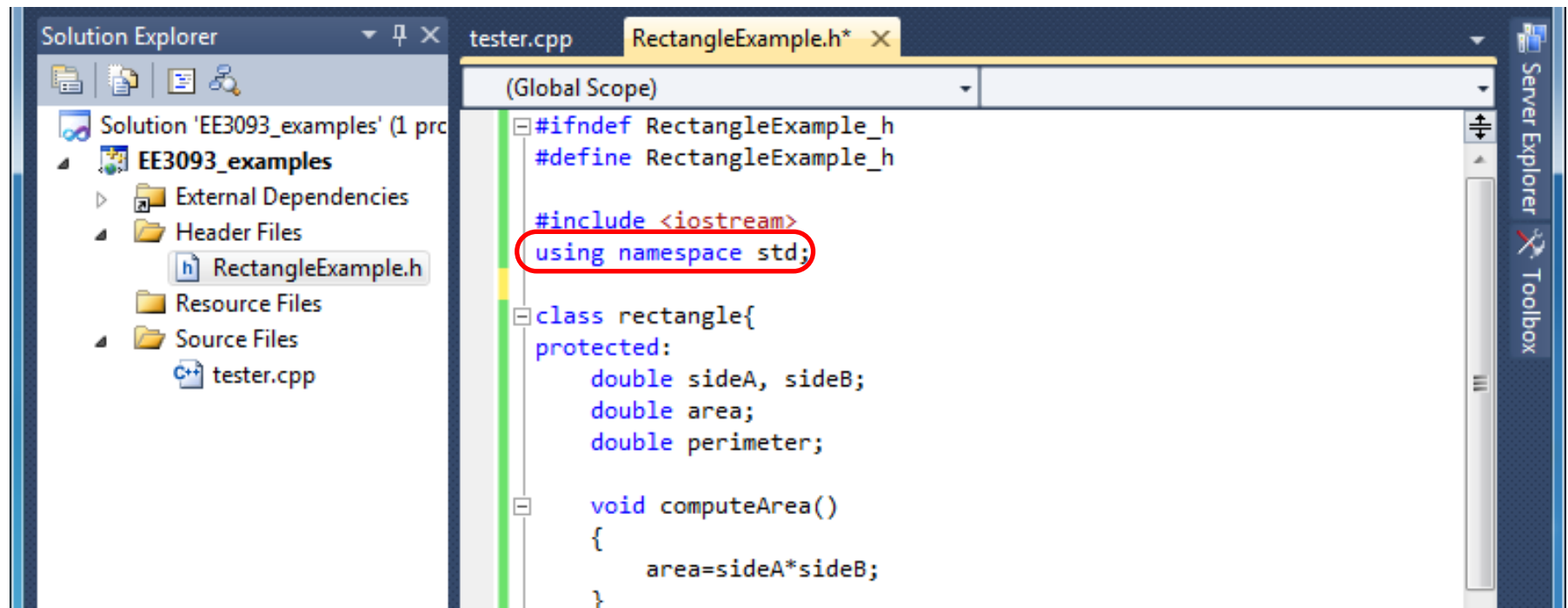
- Include the library (header file)
 - **#include** to tell the compiler to search and include the file.
 - angle brackets `< >` around the filename tell the compiler to search the file in pre-designated directories.
 - Note: C++ Standard Library Files do not have the **.h** name extension that is now only used by user-defined header files.



Appendix: iostream

How to use it:

- Include the library (header file)
- Optionally, use the namespace **std** defined in this Standard Library
 - This means you can refer to the object name **cin**, which is defined within **std**, simply as **cin**, as opposed to **std::cin**. The same applies to other class names, e.g. **cout**, **endl**, defined in **std**.



Appendix: iostream

How to use it:

- Include the library (header file)
- Optionally, use the namespace **std** defined in this Standard Library
- Output: print to screen using **cout**
 - make **cout** receive a string “like this sequence of words within inverted commas” using the operator `<<`; **cout** will then print that string to screen;
 - make **cout** receive a numerical value, be it a variable or the output of a function; **cout** will then print that value to screen (with proper formatting);
 - make **cout** receive **endl** to move to a new line when printing.

```
// set values using one member function
testrec1.inputSides(test_sideA, test_sideB);
cout << "rectangle side A is: " << testrec1.getSide(1) << endl;
cout << "rectangle side B is: " << testrec1.getSide(2) << endl;
cout << "rectangle area is: " << testrec1.getArea() << endl;
```

Appendix: iostream

How to use it:

- Include the library (header file)
- Optionally, use the namespace **std** defined in this Standard Library
- Output: print to screen using **cout**
- Input: read to keyboard **cin**
 - **cin** prompts the user to input data (into the command window, via keyboard);
 - make cin send the data (it received from keyboard) to a variable the operator `>>` ;
 - **cout** passes the value (properly formatted) to that variable.

```
] void inputSidesFromKeyboard()
{
    double in_sideA, in_sideB;

    cout << "Please enter dimension of side A of the Rectangle (then hit ENTER)" << endl;
    cin >> in_sideA;

    cout << "Please enter dimension of side B of the Rectangle (then hit ENTER)" << endl;
    cin >> in_sideB;

    inputSides(in_sideA, in_sideB);
}
```