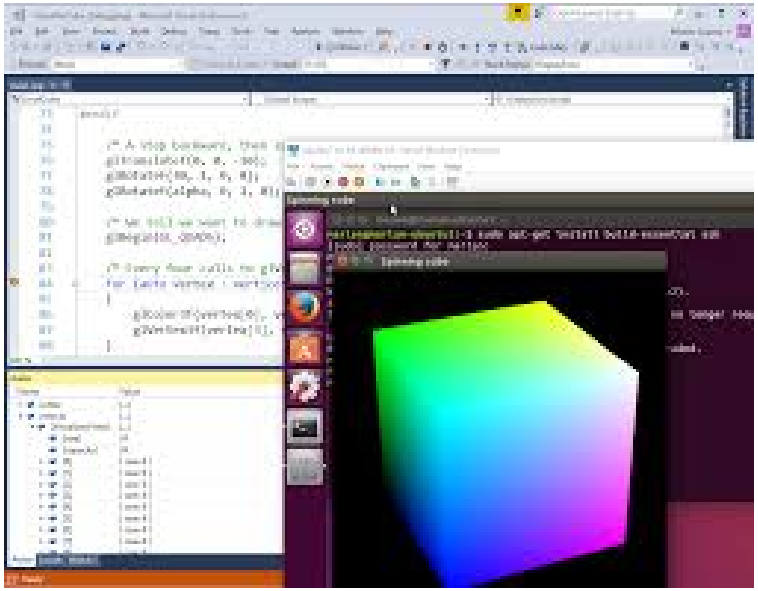
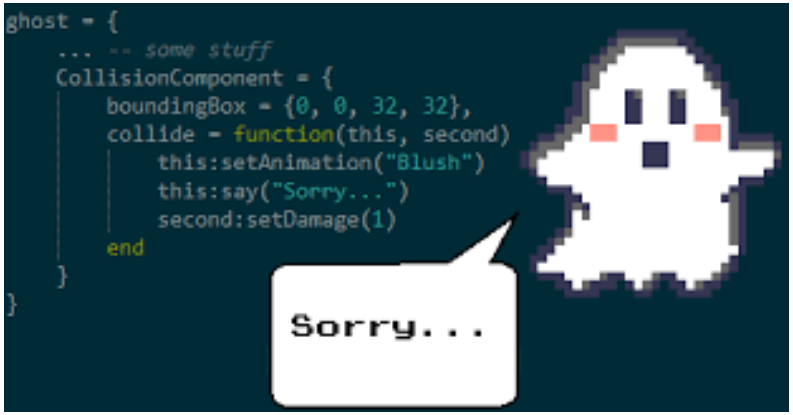
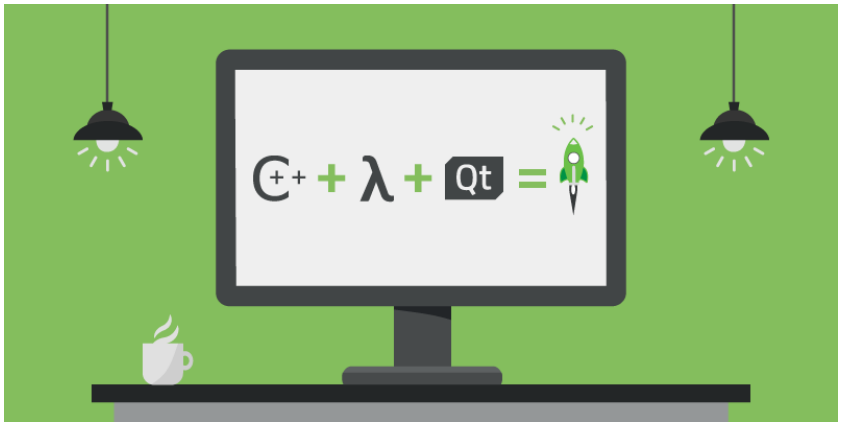
C/C++ Programming: Intro to C++ (2/3)







Course **EE3093** – Lecturer: Dr F. Verdicchio

Any question?

Relevant topics to ask questions:

- C++ Objects; syntax, member variables and functions
- C structs (difference with Object)
- C functions (difference with Object member functions)
- Latest example(s) discussed in class; use of cin, cout.

Example: class “Rectangle”

Improve the initial implementation:

- An object should be properly initialized before it can be used meaningfully.

Example: class “Rectangle”

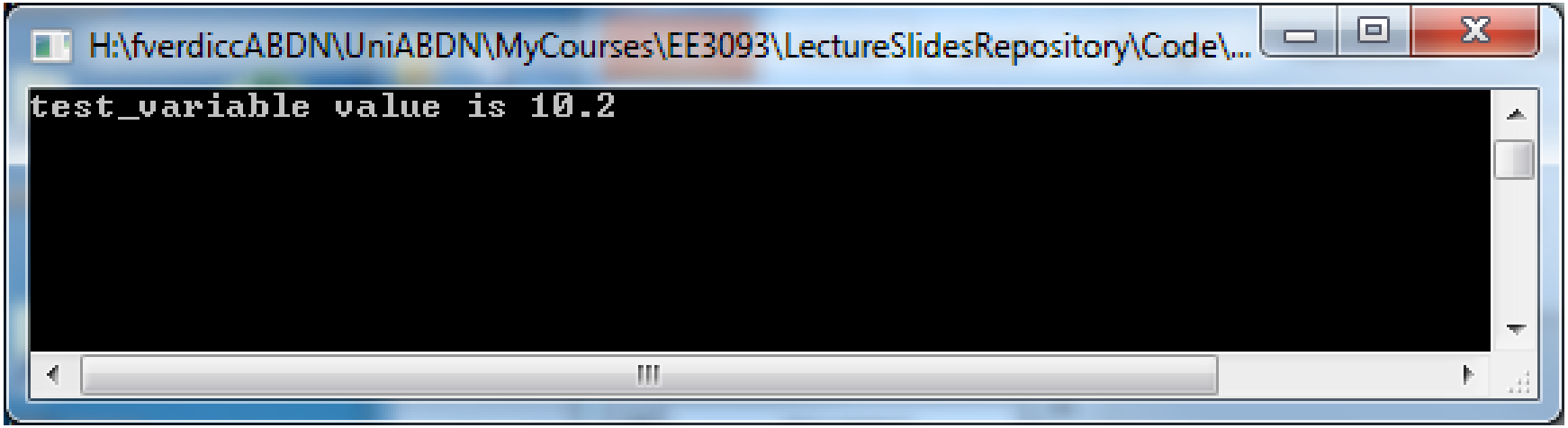
Improve the initial implementation:

- An object should be properly initialized before it can be used meaningfully.
  - Consider a C equivalent example (a C variable in place of a C++ object)

```
void C_example()
{
    double test_variable;
    test_variable = 10.2;
    cout << "test_variable value is " << test_variable << endl;
}
```

test_variable instantiated (“created”) here.

Value assigned by user here



Example: class “Rectangle”

Improve the initial implementation:

- An object should be properly initialized before it can be used meaningfully.
  - Consider a C equivalent example (a C variable in place of a C++ object)

```
void C_example()
{
    double test_variable;
    //test_variable = 10.2;
    cout << "test_variable value is " << test_variable << endl;
}
```

What happens now (value assignment commented out)??

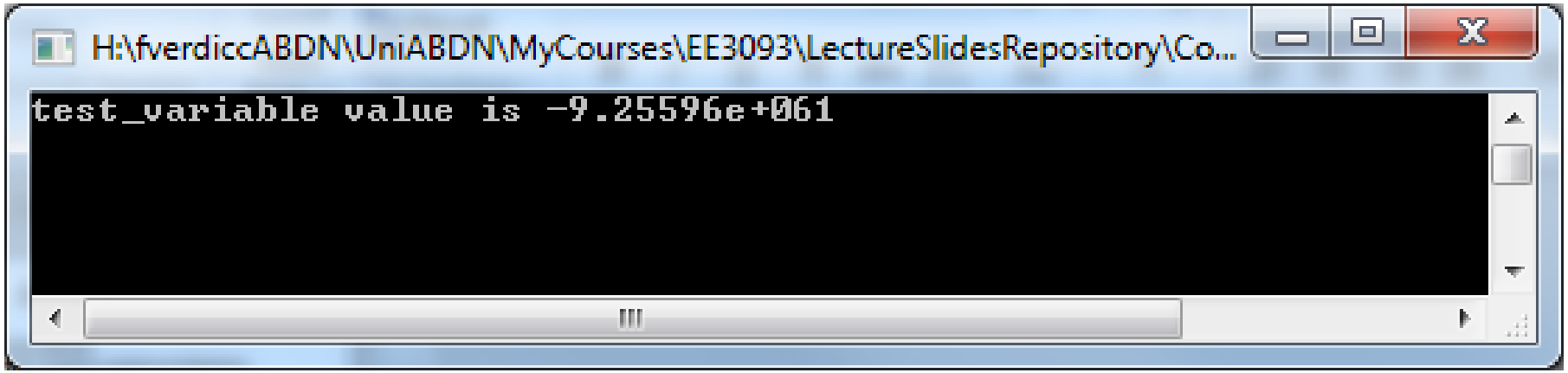
Example: class “Rectangle”

Improve the initial implementation:

- An object should be properly initialized before it can be used meaningfully.
  - Consider a C equivalent example (a C variable in place of a C++ object)

```
void C_example()
{
    double test_variable;
    //test_variable = 10.2;
    cout << "test_variable value is " << test_variable << endl;
}
```

The variable was not assigned a value by the user; variable value is then “random”



Example: class “Rectangle”

Improve the initial implementation:

- An object should be properly initialized before it can be used meaningfully.
  - Back to Cpp objects

Example: class "Rectangle"

Improve the initial implementation:

- An object should be properly initialized before it can be used meaningfully; how do we ensure that happens?

testrec1 and testrec2 are instantiated ("created") here.

```
rectangle testrec1, testrec2;

// 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 << endl << endl;

// set values using the other member function
testrec2.inputSidesFromKeyboard();
cout << "rectangle 2: " << endl;
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"

Improve the initial implementation:

- An object should be properly initialized before it can be used meaningfully; how do we ensure that happens?

testrec1 and testrec2 are initialized by the user (assigning a value to each sides) here.

```
rectangle testrec1, testrec2;

// 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 << endl << endl;

// set values using the other member function
testrec2.inputSidesFromKeyboard();
cout << "rectangle 2: " << endl;
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"

Improve the initial implementation:

- An object should be properly initialized before it can be used meaningfully; how do we ensure that happens?

testrec1 and testrec2 are used here (output side values, area).

```
rectangle testrec1, testrec2;

// 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 << endl << endl;

// set values using the other member function
testrec2.inputSidesFromKeyboard();
cout << "rectangle 2: " << endl;
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”

Improve the initial implementation:

- An object should be properly initialized before it can be used meaningfully; how do we ensure that happens?

What would happen now?

```
rectangle testrec1, testrec2;

// set values using one member function

;
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 << endl << endl;

// set values using the other member function

cout << "rectangle 2: " << endl;
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"

Improve the initial implementation:

- An object should be properly initialized before it can be used meaningfully; how do we ensure that happens?

What would happen now? testrec1, testrec2 are instantiated and used as before, but the sides have not been assigned a value. Can we at least notify the user?

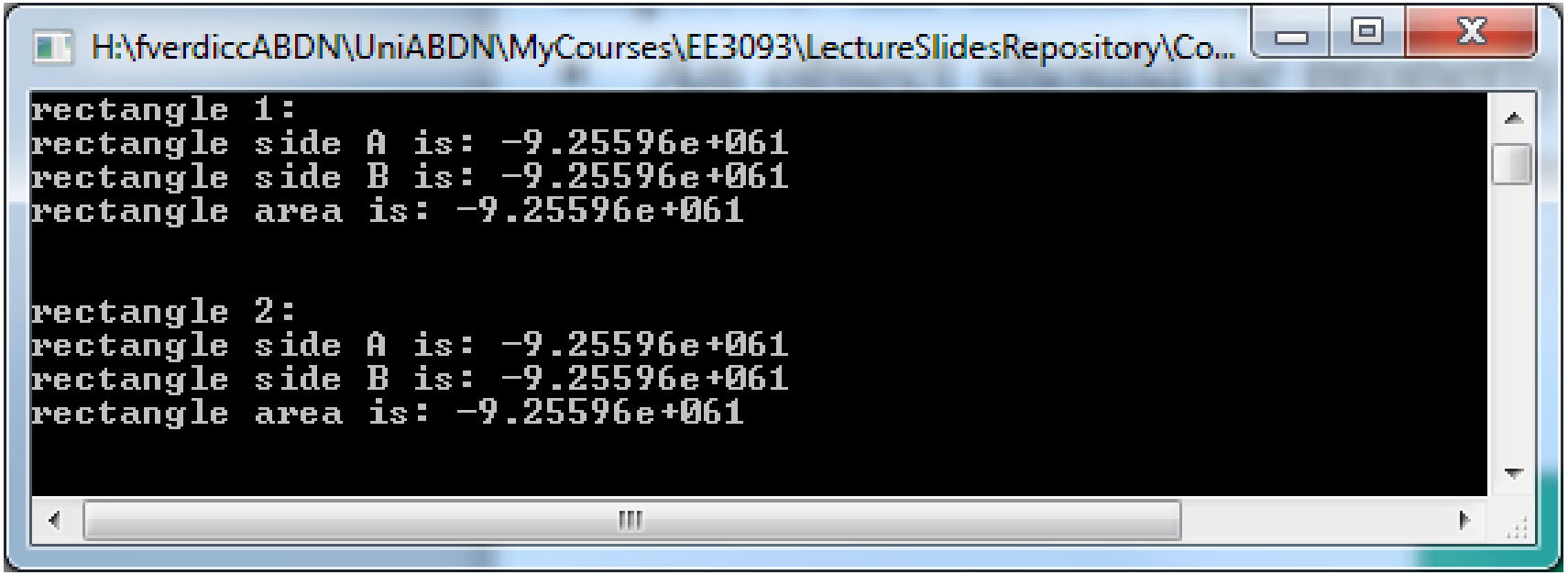
```
rectangle testrec1, testrec2;

// set values using one member function

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 << endl << endl;

// set values using the other member function

cout << "rectangle 2: " << endl;
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”

Improve the initial implementation:

- An object should be properly initialized before it can be used meaningfully; how do we ensure that happens?

Let’s see the result



Example: class “Rectangle”

Improve the initial implementation:

- Rectangle initialization

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

    // initialization flag
    bool init_flag;

    // functions
    void computeArea() { ... }
    void computePerimeter() { ... }
```

Example: class “Rectangle”

Improve the initial implementation:

- Rectangle initialization; how it can be used to determine when an object is used before initialization

```
bool isInitialized(){return init_flag;}
double getSide(int sidenum)
{
    double out;
    if(isInitialized())
    {

        switch(sidenum){
        case 1:
            out=sideA;
            break;
        case 2:
            out=sideB;
            break;
        default:
            cout << "Error in getSide(): Incorrect sidenum values" << endl;
            out=-1;
        }
    }
    else
    {
        cout << "Error in getSide(): Rectangle is not initialized" << endl;
        out=-1;
    }

    return out;
}
```

Example: class “Rectangle”

Improve the initial implementation:

- Rectangle initialization; set the object as initialized when appropriate values for each side are supplied.

```
bool isInitialized(){return init_flag;}
double getSide(int sidenum) { ... }

void inputSides(double in_sideA, double in_sideB)
{
    if(!isInitialized())
    {
        if(in_sideA>0 && in_sideB>0)
        {
            sideA=in_sideA;
            sideB=in_sideB;
            init_flag=true;
            computeArea();
            computePerimeter();
        }
        else
            cout << "Error in inputSides(): Incorrect input values" << endl;
    }
    else
        cout << "Error in inputSides(): Rectangle is already initialized " << endl;

}
```

Example: class “Rectangle”

Improve the initial implementation:

- Rectangle initialization: how can we ensure a member variable (e.g. init_flag) is set, or a function is called, as soon as an object is instantiated?

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

    // initialization flag
    bool init_flag;

    // functions
    void computeArea() { ... }
    void computePerimeter() { ... }
```

Example: class “Rectangle”

Improve the initial implementation:

- Rectangle Initialization: use the object **constructor** member function

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

    // initialization flag
    bool init_flag;

    // functions
    void computeArea() { ... }
    void computePerimeter() { ... }

public:
    // constructor
    rectangle()
    {
        init_flag=false;
    }
```

C++ object Constructor

Constructor :

- Member function with the same name as the class;
- Typically without input parameters (you can also add versions that require input parameters)
- Automatically called by an object as soon as it is created (only once)

```
rectangle testrec1, testrec2;

// 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 << endl << endl;

// set values using the other member function
testrec2.inputSidesFromKeyboard();
cout << "rectangle 2: " << endl;
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;
```

testrec1 and testrec2 are instantiated ("created") here; the constructor is called for each of them.

Example: class “Rectangle”

Improve the initial implementation:

- An object should be properly initialized before it can be used meaningfully; how do we ensure that happens?

Let’s see the result NOW

```
rectangle testrec1, testrec2;

// set values using one member function
;
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 << endl << endl;

// set values using the other member function

cout << "rectangle 2: " << endl;
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”

Improve the initial implementation:

- An object should be properly initialized before it can be used meaningfully; how do we ensure that happens?

Let’s see the result NOW

H:\fverdiccABDN\UniABDN\MyCourses\EE3093\LectureSlidesRepository\Code\EE3093_exampl...

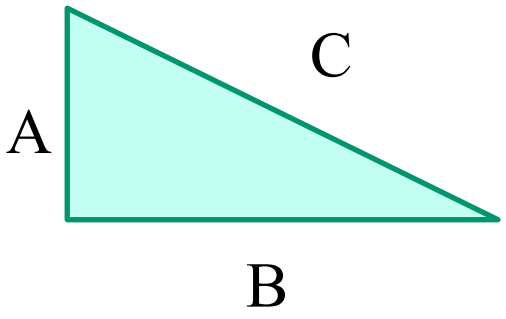
```
rectangle 1:
Error in getSide(): Rectangle is not initialized
rectangle side A is: -1
Error in getSide(): Rectangle is not initialized
rectangle side B is: -1
Error in getArea(): Rectangle is not initialized
rectangle area is: 0


rectangle 2:
Error in getSide(): Rectangle is not initialized
rectangle side A is: -1
Error in getSide(): Rectangle is not initialized
rectangle side B is: -1
Error in getArea(): Rectangle is not initialized
rectangle area is: 0
```

Any question?

Second example: class “Right Triangle”

Write a Class Right Triangle (similar to rectangle):

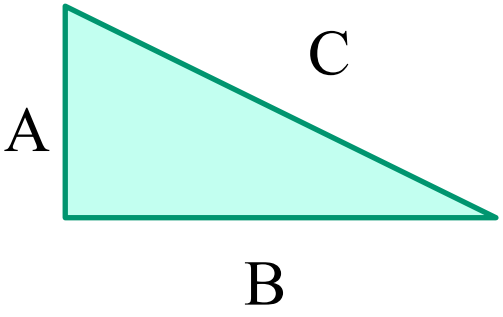
- Determine the relevant “internal variables”
- Write member functions for the user to:
  - input A & B (C is set according to the Pythagorean theorem);
  - output sides, area, perimeter;



Second example: class “Right Triangle”

Write a Class Right Triangle (similar to rectangle):

- Determine the relevant “internal variables”
- Write member functions for the user to:
  - input A & B (C is set according to the Pythagorean theorem);
  - output sides, area, perimeter;

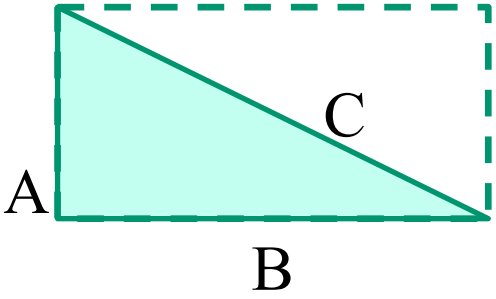


Second example: class “Right Triangle”

Write a Class Right Triangle (similar to rectangle):

- Determine the relevant “internal variables”
- Write member functions for the user to:
  - input A & B (C is set according to the Pythagorean theorem);
  - output sides, area, perimeter;

Insight for simple implementation:
A & B are specified by the user; C is given by the Pythagorean theorem

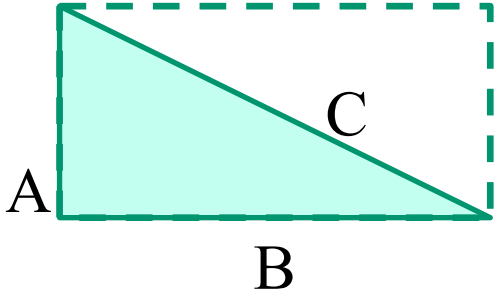


How to implement it:

- Design a class containing:
  - an object Rectangle to hold Side A & B
  - Side C
- Design functions (exploit those in Rectangle) :
  - Public: set / get data;
  - Private: compute area and perimeter.

Second example: class “Right Triangle”

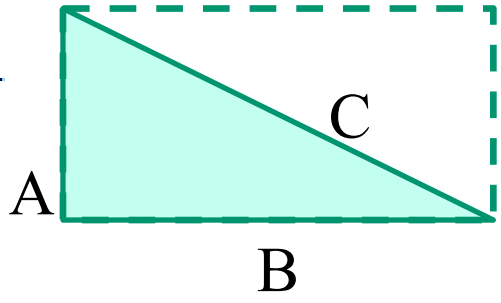
```
#include "RectangleExample.h"

class right_triangle{
protected:
    // variables

    // use an object rectangle to hold information about A & B
    rectangle SideAandSideB;

    // Hypotenuse
    double sideC;

    double area;
    double perimeter;

    // functions
    void computeArea() { ... }
    void computePerimeter() { ... }
    void computeHypotenuse() { ... }
public:
    double getPerimeter() { ... }
    double getArea() { ... }

    bool isInitialized(){return SideAandSideB.isInitialized();}
    double getSide(int sidenum) { ... }
    void inputSides(double in_sideA, double in_sideB) { ... }
    void inputSidesFromKeyboard() { ... }
};
```



Second example: class "Right Triangle"

Let's start looking at the implementation of Public functions

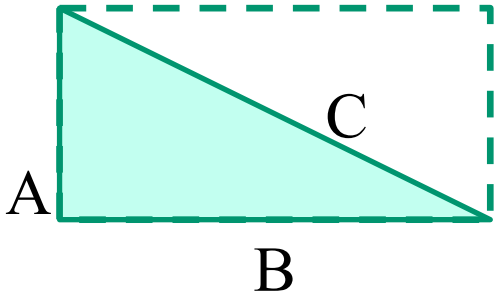
```
public:
    double getPerimeter()
    {
        if(SideAandSideB.isInitialized())
            return perimeter;
        else
            cout << "Error in getPerimeter(): Triangle is not initialized " << endl;
        return 0;
    }
    double getArea()
    {
        if(SideAandSideB.isInitialized())
            return area;
        else
            cout << "Error in getArea(): Triangle is not initialized " << endl;
        return 0;
    }

    bool isInitialized(){return SideAandSideB.isInitialized();}

    double getSide(int sidenum) { ... }
    void inputSides(double in_sideA, double in_sideB) { ... }
    void inputSidesFromKeyboard() { ... }
};
```



Second example: class "Right Triangle"

Implementation of Public functions: functions that input data

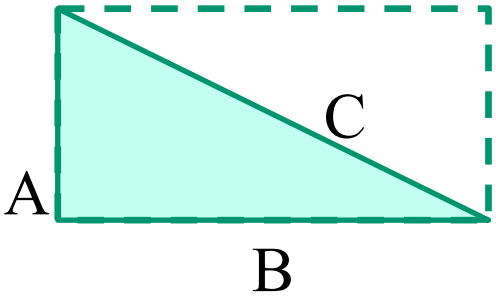
```
public:
    double getPerimeter() { ... }
    double getArea() { ... }

    bool isInitialized(){return SideAandSideB.isInitialized();}
    double getSide(int sidenum) { ... }
    void inputSides(double in_sideA, double in_sideB)
    {
        if(!isInitialized())
        {
            if(in_sideA>0 && in_sideB>0)
            {
                // set A and B (this initializes the rectangle)
                SideAandSideB.inputSides(in_sideA,in_sideB);

                // set C
                computeHypotenuse();

                // now the triangle is initializd; compute area and perimeter
                computeArea();
                computePerimeter();
            }
            else
                cout << "Error in inputSides(): Incorrect input values" << endl;
        }
        else
            cout << "Error in inputSides(): Triangle is already initialized " << endl;

    }
    void inputSidesFromKeyboard() { ... }
};
```



Second example: class "Right Triangle"

Implementation of Public functions: functions that input data

```
public:
    double getPerimeter() { ... }
    double getArea() { ... }

    bool isInitialized(){return SideAandSideB.isInitialized();}
    double getSide(int sidenum) { ... }
    void inputSides(double in_sideA, double in_sideB) { ... }
    void inputSidesFromKeyboard()
    {
        if(!isInitialized())
        {
            double in_sideA, in_sideB;

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

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

            // use these vlaues to set A, B and C;
            // this initializes the rectangle, hence the triangle
            inputSides(in_sideA, in_sideB);

            if(isInitialized())
                cout << " Dimension of side C of the Triangle is: " << sideC << endl;
        }
        else
            cout << "Error in inputSidesFromKeyboard(): Triangle is already initialized " << endl;
    }
};
```

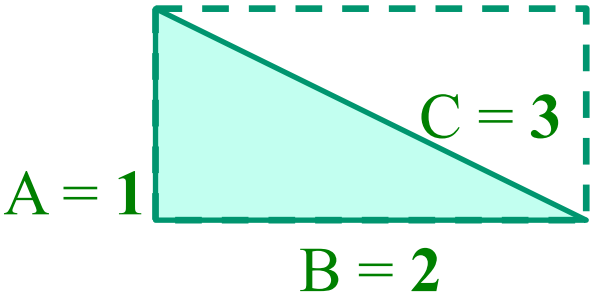


Second example: class "Right Triangle"

```
public:
    double getPerimeter() { ... }
    double getArea() { ... }
    bool isInitialized(){return SideAandSideB.isInitialized();}
    double getSide(int sidenum)
    {
        double out;
        if(isInitialized())
        {

            switch(sidenum){
            case 1:
            case 2:
                out=SideAandSideB.getSide(sidenum);
                break;
            case 3:
                out=sideC;
                break;
            default:
                cout << "Error in getSide(): Incorrect sidenum value" << endl;
                out=-1;
            }
        }
        else
        {
            cout << "Error in getSide(): Triangle is not initialized" << endl;
            out=-1;
        }
        return out;
    }
    void inputSides(double in_sideA, double in_sideB) { ... }
    void inputSidesFromKeyboard() { ... }
};
```

Implementation of Public functions that output data

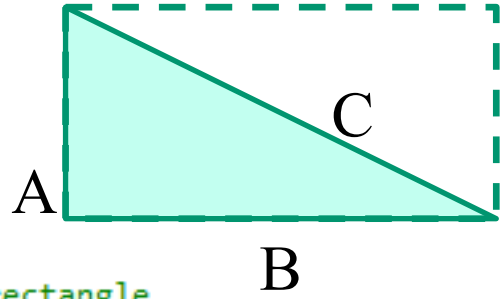
Convention for *sidenum*:



Second example: class “Right Triangle”

Implementation of Private functions

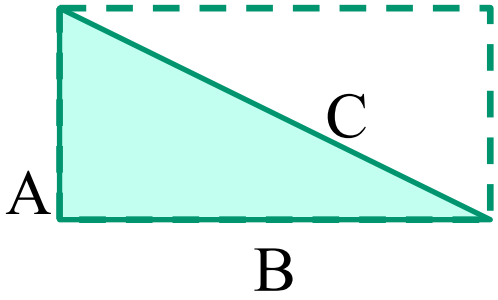
```
#include "RectangleExample.h"

class right_triangle{
protected:
    // variables
    // use an object rectangle to hold information about A & B
    rectangle SideAandSideB;

    // Hypotenuse
    double sideC;

    double area;
    double perimeter;

    // functions
    void computeArea()
    {
        // the area of the right triangle is half that of the rectangle
        if(SideAandSideB.isInitialized())
            area=(SideAandSideB.getArea())/2;
    }
    void computePerimeter()
    {
        // the area of the right triangle is: A + B + C;
        //  A + B is half the perimeter of the rectangle;
        if(SideAandSideB.isInitialized())
            perimeter=( (SideAandSideB.getPerimeter())/2 ) + sideC;
    }
    void computeHypotenuse() { ... }
public:
```



Second example: class “Right Triangle”

Implementation of Private functions

```
#include "RectangleExample.h"

class right_triangle{
protected:
    // variables
    // use an object rectangle to hold information about A & B
    rectangle SideAandSideB;

    // Hypotenuse
    double sideC;

    double area;
    double perimeter;

    // functions
    void computeArea() { ... }
    void computePerimeter() { ... }
    void computeHypotenuse()
    {
        if(isInitialized())
        {
            // given A and B the value of C is imposed by the Pythagorean theorem
            double in_sideA=SideAandSideB.getSide(1);
            double in_sideB=SideAandSideB.getSide(2);
            sideC=sqrt( (in_sideA*in_sideA) + (in_sideB*in_sideB) );
        }
    }
public:
```



Any question?