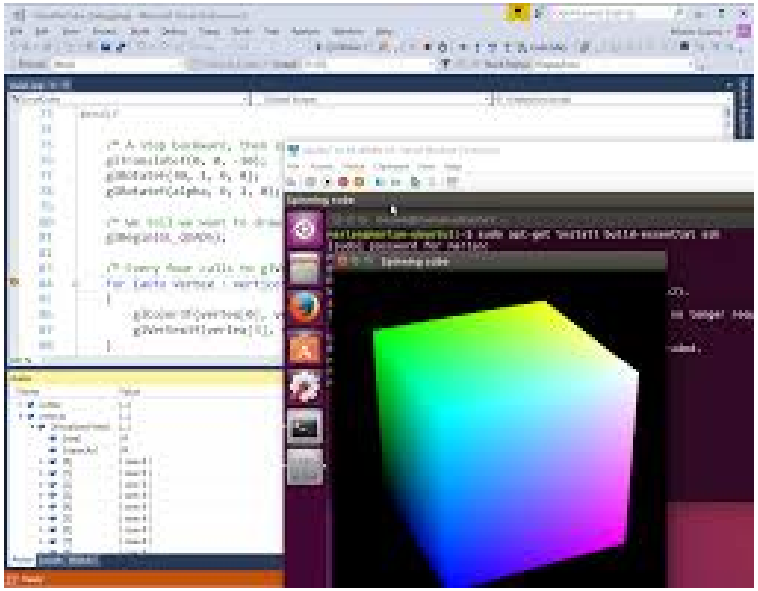
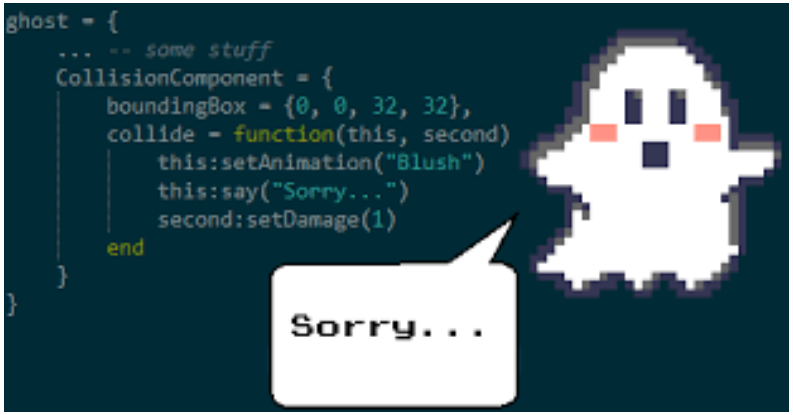
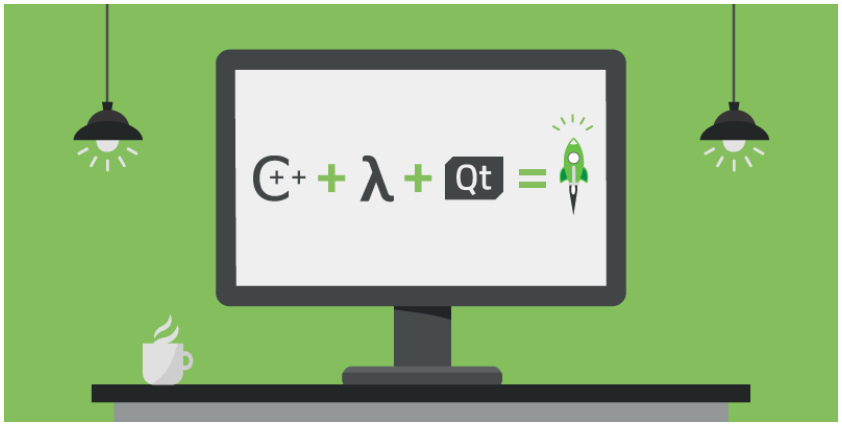
C/C++ Programming: Intro to C++ (3/3)







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

Any question?

Relevant topics to ask questions:

- C++ Objects; syntax, member variables and functions
- C++ Objects: Constructors; composite objects.

Array of objects

Objects can be used to form arrays:

```
void array_of_object()
{
    double test_sideA, test_sideB;

    static const int arraysize=10;

    // these are N instantiations (objects) of the class rectangle
    // i.e. variables of the user-defined type "rectangle"
    rectangle rct_array[arraysize];

    for(int i=0; i<arraysize; i++)
    {
        test_sideA=10 + i;
        test_sideB=20 + i;
        rct_array[i].inputSides(test_sideA, test_sideB);
        cout << "rectangle position "<< i << ":" << endl;
        cout << "rectangle side A is: " << rct_array[i].getSide(1) << endl;
        cout << "rectangle side B is: " << rct_array[i].getSide(2) << endl;
        cout << "rectangle area is: " << rct_array[i].getArea() << endl;
        cout << "rectangle perimeter is: " << rct_array[i].getPerimeter() << endl << endl << endl;
    }

}
```

Array of objects

Objects can be used to form arrays:

```
void array_of_object()
{
    double test_sideA, test_sideB;

    static const int arraysize=10;

    // these are N instantiations (objects) of the class rectangle
    // i.e. variables of the user-defined type "rectangle"
    rectangle rct_array[arraysize];

    for(int i=0; i<arraysize; i++)
    {
        test_sideA=10 + i;
        test_sideB=20 + i;
        rct_array[i].inputSides(test_sideA, test_sideB);
        cout << "rectangle position "<< i << ":" << endl;
        cout << "rectangle side A is: " << rct_array[i].getSide(1) << endl;
        cout << "rectangle side B is: " << rct_array[i].getSide(2) << endl;
        cout << "rectangle area is: " << rct_array[i].getArea() << endl;
        cout << "rectangle perimeter is: " << rct_array[i].getPerimeter() << endl << endl << endl;
    }

}
```

This is static allocation of 10 elements: the size of the array is known to the compiler;

Array of objects

Objects can be used to form arrays:

```
void array_of_object()
{
    double test_sideA, test_sideB;

    static const int arraysize=10;

    // these are N instantiations (objects) of the class rectangle
    // i.e. variables of the user-defined type "rectangle"
    rectangle rct_array[arraysize];

    for(int i=0; i<arraysize; i++)
    {
        test_sideA=10 + i;
        test_sideB=20 + i;
        rct_array[i].inputSides(test_sideA, test_sideB);
        cout << "rectangle position "<< i << ":" << endl;
        cout << "rectangle side A is: " << rct_array[i].getSide(1) << endl;
        cout << "rectangle side B is: " << rct_array[i].getSide(2) << endl;
        cout << "rectangle area is: " << rct_array[i].getArea() << endl;
        cout << "rectangle perimeter is: " << rct_array[i].getPerimeter() << endl << endl << endl;
    }

}
```

The constructor is automatically invoked for all 10 elements.

Array of objects

Objects can be used to form arrays:

- Example: scan the array and identify those that satisfy certain criteria
  - Area above a certain threshold AND sideA length is an odd number

Array of objects

Objects can be used to form arrays:

- Example: scan the array and identify those that satisfy certain criteria
  - Area above a certain threshold AND sideA length is an odd number

```
void array_of_object()
{
    double test_sideA, test_sideB;
    double area, area_threshold=300;
    const int arraysize=10;

    // these are N instantiations (objects) of the class rectangle
    // i.e. variables of the user-defined type "rectangle"
    rectangle rct_array[arraysize];

    for(int i=0; i<arraysize; i++)
    {
        test_sideA=10 + i;
        test_sideB=20 + i;
        rct_array[i].inputSides(test_sideA, test_sideB);

        area=rct_array[i].getArea();
        int int_test_sideA=(int) rct_array[i].getSide(1);

        if( (area>area_threshold) && (!(int_test_sideA%2))  )
        {
            cout << "rectangle position "<< i << ":" << endl;
            cout << "rectangle side A is: " << rct_array[i].getSide(1) << endl;
            cout << "rectangle area is: " << rct_array[i].getArea() << endl;
        }
    }

}
```

Example: class “Rectangle” with colors

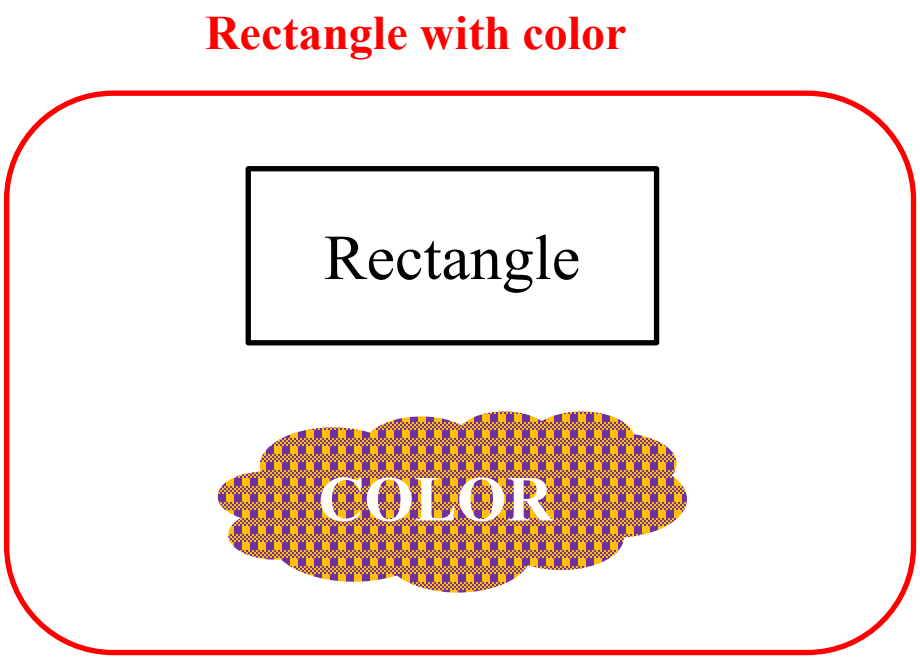
A possible implementation: composite object

- Rectangle object
- Color (enumerator)

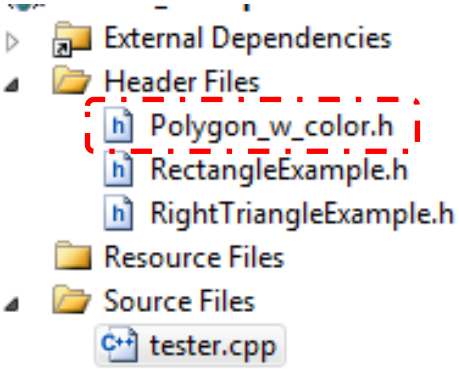
Example: class “Rectangle” with colors

A possible implementation: **composite object**

- Rectangle object
- Color object (based on an enumerator)



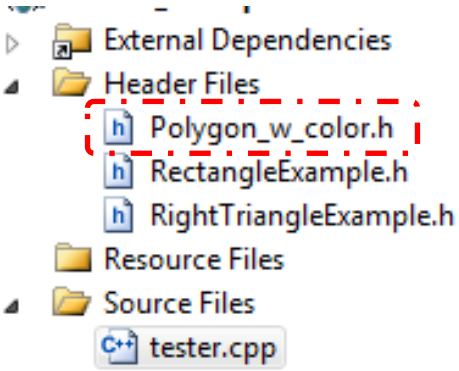
Example: class “Rectangle” with colors




```
#include "RectangleExample.h"

// enumerator: finite set of choices for the colors
// blank and color_stop "bookend" the allowed colors
enum polygonColorOptions{blank=0, white, red, orange, yellow, green,
    light_blue, dark_blue, purple, color_stop};

class polygonColor{
private:
    polygonColorOptions color;
public:
    polygonColor(){color=blank;}
    void setColor(polygonColorOptions inp_color){ ... }
    polygonColorOptions getColor(){return color;}

    void inputColorFromKeyboard(){ ... }
    void printColorInfo(){ ... }

    void inputRandomColor(){ ... }
};

class rectangleWcolor{
public:
    rectangle the_rectangle;
    polygonColor the_color;
public:
    void inputFromKeyboard(){ ... }
    void printInfo(){ ... }
    void inputRandomValues(double max_val=100){ ... }
};
```

Rectangle is a composite object:

Example: class “Rectangle” with colors




```
#include "RectangleExample.h"

// enumerator: finite set of choices for the colors
// blank and color_stop "bookend" the allowed colors
enum polygonColorOptions{blank=0, white, red, orange, yellow, green,
    light_blue, dark_blue, purple, color_stop};

class polygonColor{
private:
    polygonColorOptions color;
public:
    polygonColor(){color=blank;}
    void setColor(polygonColorOptions inp_color){ ... }
    polygonColorOptions getColor(){return color;}

    void inputColorFromKeyboard(){ ... }
    void printColorInfo(){ ... }

    void inputRandomColor(){ ... }
};

class rectangleWcolor{
public:
    rectangle the_rectangle;
    polygonColor the_color;
public:
    void inputFromKeyboard(){ ... }
    void printInfo(){ ... }
    void inputRandomValues(double max_val=100){ ... }
};
```

- Rectangle object

Example: class “Rectangle” with colors

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

```
#include "RectangleExample.h"

// enumerator: finite set of choices for the colors
// blank and color_stop "bookend" the allowed colors
enum polygonColorOptions{blank=0, white, red, orange, yellow, green,
    light_blue, dark_blue, purple, color_stop};

class polygonColor{
private:
    polygonColorOptions color;
public:
    polygonColor(){color=blank;}
    void setColor(polygonColorOptions inp_color){ ... }
    polygonColorOptions getColor(){return color;}

    void inputColorFromKeyboard(){ ... }
    void printColorInfo(){ ... }

    void inputRandomColor(){ ... }
};

class rectangleWcolor{
public:
    rectangle the_rectangle;
    polygonColor the_color;
public:
    void inputFromKeyboard(){ ... }
    void printInfo(){ ... }
    void inputRandomValues(double max_val=100){ ... }
};
```

- PolygonColor object

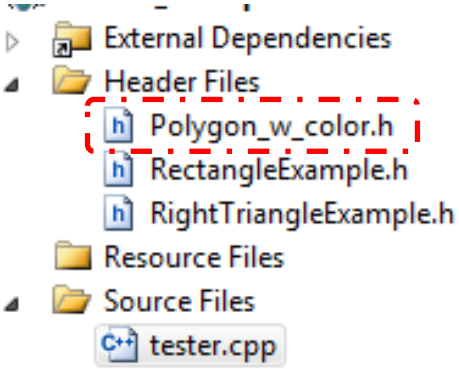
Example: class “Rectangle” with colors

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

```
#include "RectangleExample.h"

// enumerator: finite set of choices for the colors
// blank and color_stop "bookend" the allowed colors
enum polygonColorOptions{blank=0, white, red, orange, yellow, green,
    light_blue, dark_blue, purple, color_stop};

class polygonColor{
private:
    polygonColorOptions color;
public:
    polygonColor(){color=blank;}
    void setColor(polygonColorOptions inp_color){ ... }
    polygonColorOptions getColor(){return color;}

    void inputColorFromKeyboard(){ ... }
    void printColorInfo(){ ... }

    void inputRandomColor(){ ... }
};

class rectangleWcolor{
public:
    rectangle the_rectangle;
    polygonColor the_color;
public:
    void inputFromKeyboard(){ ... }
    void printInfo(){ ... }
    void inputRandomValues(double max_val=100){ ... }
};
```

C enumerator: Represented by the machine as an integer

Example: class “Rectangle” with colors




```
#include "RectangleExample.h"

// enumerator: finite set of choices for the colors
// blank and color_stop "bookend" the allowed colors
enum polygonColorOptions{blank=0, white, red, orange, yellow, green,
    light_blue, dark_blue, purple, color_stop};

class polygonColor{
private:
    polygonColorOptions color;
public:
    polygonColor(){color=blank;}
    void setColor(polygonColorOptions inp_color){ ... }
    polygonColorOptions getColor(){return color;}

    void inputColorFromKeyboard(){ ... }
    void printColorInfo(){ ... }

    void inputRandomColor(){ ... }
};

class rectangleWcolor{
public:
    rectangle the_rectangle;
    polygonColor the_color;
public:
    void inputFromKeyboard(){ ... }
    void printInfo(){ ... }
    void inputRandomValues(double max_val=100){ ... }
};
```

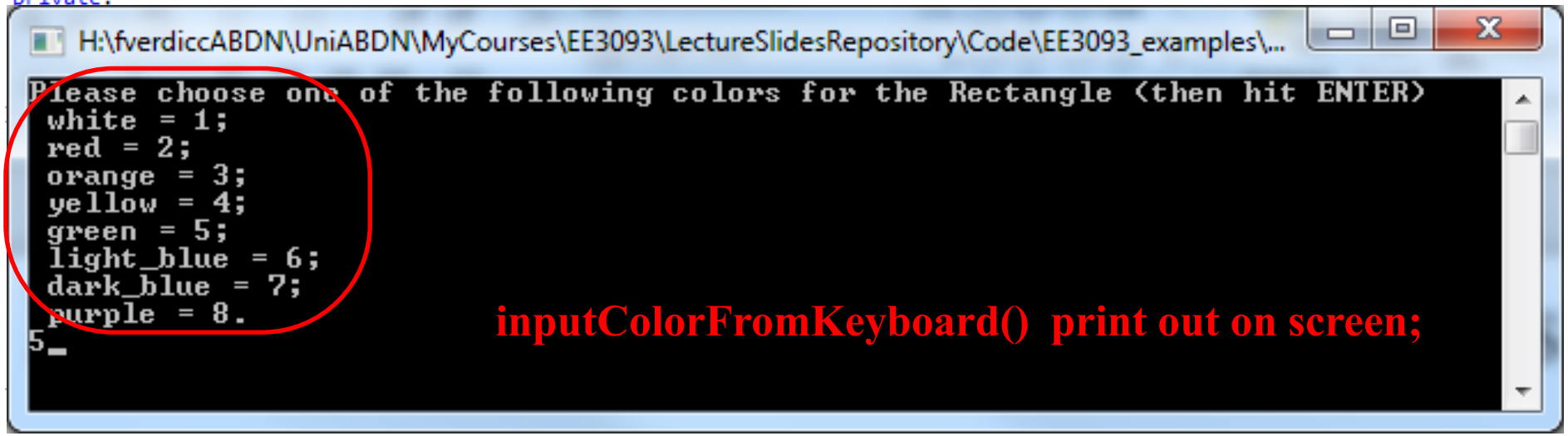
Public functions:

Example: class “Rectangle” with colors

```
class polygonColor{
private:
    polygonColorOptions color;
public:
    polygonColor(){color=blank;}
    void setColor(polygonColorOptions inp_color)
    {
        // only set once
        if(color==blank)
        {
            // make sure input color is within the limits
            if( (inp_color>blank) &&  (inp_color<color_stop) )
                color=inp_color;
        }
    }
    polygonColorOptions getColor(){return color;}

    void inputColorFromKeyboard()
    {
        if(color==blank)
        {
            int inp_color;
            cout << "Please choose one of the following colors for the Rectangle (then hit ENTER)" << endl;
            cout << " white = " << white << ";" << endl;
            cout << " red = " << red << ";" << endl;
            cout << " orange = " << orange << ";" << endl;
            cout << " yellow = " << yellow << ";" << endl;
            cout << " green = " << green << ";" << endl;
            cout << " light_blue = " << light_blue << ";" << endl;
            cout << " dark_blue = " << dark_blue << ";" << endl;
            cout << " purple = " << purple << "." << endl;
            cin >> inp_color;
            setColor((polygonColorOptions)inp_color);
        }
        else
            cout << "Error in inputColorFromKeyboard(): Color is already initialized " << endl;
    }
```

Example: class “Rectangle” with colors

```
class polygonColor{
private:
```



inputColorFromKeyboard() print out on screen;

```
    void inputColorFromKeyboard()
    {
        if(color==blank)
        {
            int inp_color;
            cout << "Please choose one of the following colors for the Rectangle (then hit ENTER)" << endl;
            cout << " white = " << white << ";" << endl;
            cout << " red = " << red << ";" << endl;
            cout << " orange = " << orange << ";" << endl;
            cout << " yellow = " << yellow << ";" << endl;
            cout << " green = " << green << ";" << endl;
            cout << " light_blue = " << light_blue << ";" << endl;
            cout << " dark_blue = " << dark_blue << ";" << endl;
            cout << " purple = " << purple << "." << endl;
            cin >> inp_color;
            setColor((polygonColorOptions)inp_color);
        }
        else
            cout << "Error in inputColorFromKeyboard(): Color is already initialized " << endl;
    }
```

Example: class “Rectangle” with colors

```
class polygonColor{
private:
```

H:\fverdiccABDN\UniABDN\MyCourses\EE3093\LectureSlidesRepository\Code\EE3093_examples\...

```
Please choose one of the following colors for the Rectangle (then hit ENTER)
 white = 1;
 red = 2;
 orange = 3;
 yellow = 4;
 green = 5;
 light_blue = 6;
 dark_blue = 7;
 purple = 8.
5_
```

User input;

```
    void inputColorFromKeyboard()
    {
        if(color==blank)
        {
            int inp_color;
            cout << "Please choose one of the following colors for the Rectangle (then hit ENTER)" << endl;
            cout << " white = " << white << ";" << endl;
            cout << " red = " << red << ";" << endl;
            cout << " orange = " << orange << ";" << endl;
            cout << " yellow = " << yellow << ";" << endl;
            cout << " green = " << green << ";" << endl;
            cout << " light_blue = " << light_blue << ";" << endl;
            cout << " dark_blue = " << dark_blue << ";" << endl;
            cout << " purple = " << purple << "." << endl;
            cin >> inp_color;
            setColor((polygonColorOptions)inp_color);
        }
        else
            cout << "Error in inputColorFromKeyboard(): Color is already initialized " << endl;
    }
```

Example: class “Rectangle” with colors

```
void printColorInfo()
{
    if(color!=blank)
    {
        cout << "Rectangle color is: ";
        switch(color){
        case white:
            cout << " white." << endl;
            break;
        case red:
            cout << " red." << endl;
            break;
        case orange:
            cout << " orange." << endl;
            break;
        case yellow:
            cout << " yellow." << endl;
            break;
        case green:
            cout << " green." << endl;
            break;
        case light_blue:
            cout << " light_blue." << endl;
            break;
        case dark_blue:
            cout << " dark_blue." << endl;
            break;
        case purple:
            cout << " purple." << endl;
            break;
        default:
            cout << "Error in printInfo(): Color num not recognized" << endl;
        }
    }
    else
        cout << "printInfo(): Color is not initialized " << endl;
}
```

Example: class “Rectangle” with colors

```
// enumerator: finite set of choices for the colors
// blank and color_stop "bookend" the allowed colors
enum polygonColorOptions{blank=0, white, red, orange, yellow, green,
    light_blue, dark_blue, purple, color_stop};

class polygonColor{
private:
    polygonColorOptions color;
public:
    polygonColor(){color=blank;}
    void setColor(polygonColorOptions inp_color){ ... }
    polygonColorOptions getColor(){return color;}

    void inputColorFromKeyboard(){ ... }
    void printColorInfo(){ ... }

    void inputRandomColor()
    {
        // this produces a random integer within [0 , RAND_MAX]
        int rand_value=rand();
        // this produces a random int within [1 , color_stop-1]
        int inp_color = (int)(   ( ((double)rand_value)/RAND_MAX ) * (color_stop-2) + 1 );

        setColor( (polygonColorOptions) inp_color);
    }
};
```

Example: class “Rectangle” with colors

```
// enumerator: finite set of choices for the colors
// blank and color_stop "bookend" the allowed colors
enum polygonColorOptions{blank=0, white, red, orange, yellow, green,
    light_blue, dark_blue, purple, color_stop};

class polygonColor{
private:
    polygonColorOptions color;
public:
    polygonColor(){color=blank;}
    void setColor(polygonColorOptions inp_color){ ... }
    polygonColorOptions getColor(){return color;}

    void inputColorFromKeyboard(){ ... }
    void printColorInfo(){ ... }

    void inputRandomColor()
    {
        // this produces a random integer within [0 , RAND_MAX]
        int rand_value=rand();
        // this produces a random int within [1 , color_stop-1]
        int inp_color = (int)(   ( ((double)rand_value)/RAND_MAX ) * (color_stop-2) + 1 );

        setColor( (polygonColorOptions) inp_color);
    }
};
```

Use library function rand:

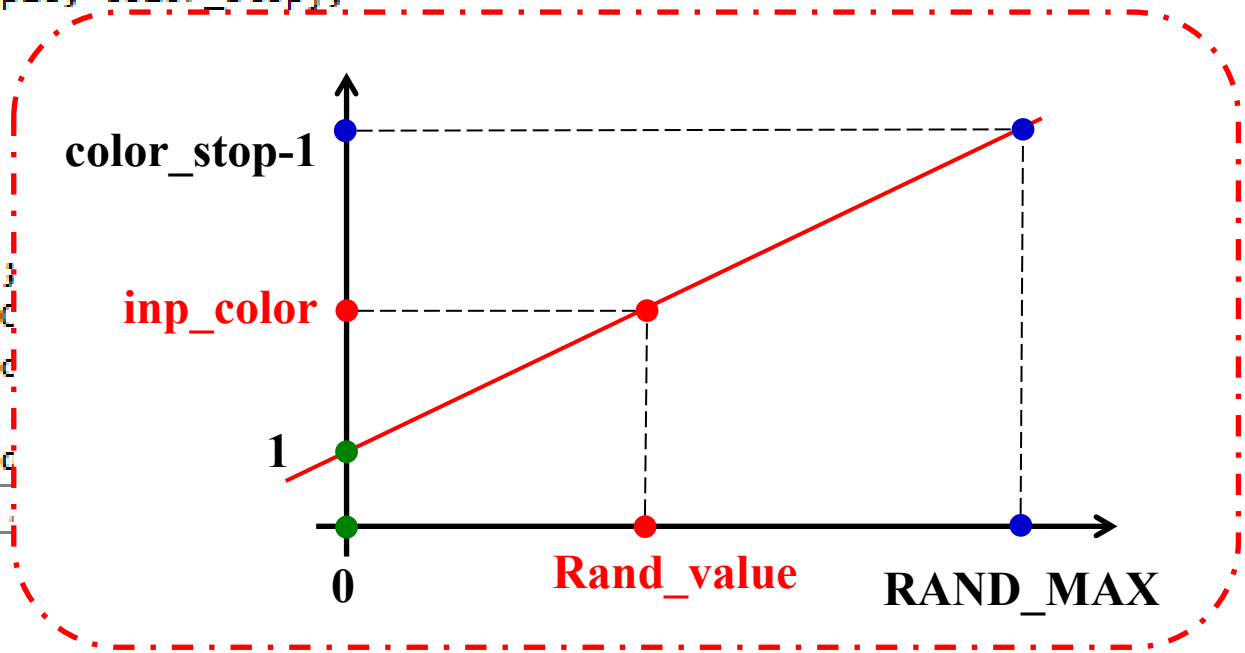
#include <stdlib.h>

Example: class “Rectangle” with colors

```
// enumerator: finite set of choices for the colors
// blank and color_stop "bookend" the allowed colors
enum polygonColorOptions{blank=0, white, red, orange, yellow, green,
    light_blue, dark_blue, purple, color_stop};

class polygonColor{
private:
    polygonColorOptions color;
public:
    polygonColor(){color=blank;}
    void setColor(polygonColorOptions inp_color) { ... }
    polygonColorOptions getColor(){return color;}

    void inputColorFromKeyboard() { ... }
    void printColorInfo() { ... }

    void inputRandomColor()
    {
        // this produces a random integer within [0 , RAND_MAX]
        int rand_value=rand();
        // this produces a random int within [1 , color_stop-1]
        int inp_color = (int)(   ( ((double)rand_value)/RAND_MAX ) * (color_stop-2) + 1 );

        setColor( (polygonColorOptions) inp_color);
    }
};
```

Example: class “Rectangle” with colors

```
// enumerator: finite set of choices for the colors
// blank and color_stop "bookend" the allowed colors
enum polygonColorOptions{blank=0, white, red, orange, yellow, green,
    light_blue, dark_blue, purple, color_stop};

class polygonColor{
private:
    polygonColorOptions color;
public:
    polygonColor(){color=blank;
    void setColor(polygonColor(
    polygonColorOptions getCold

    void inputColorFromKeyboard
    void printColorInfo() { ...

    void inputRandomColor()
    {
        // this produces a random integer within [0 , RAND_MAX]
        int rand_value=rand();
        // this produces a random int within [1 , color_stop-1]
        int inp_color = (int)(   ( ((double)rand_value)/RAND_MAX ) * (color_stop-2) + 1 );

        setColor( (polygonColorOptions) inp_color);
    }
};
```



Example: class “Rectangle” with colors

```
// enumerator: finite set of choices for the colors
// blank and color_stop "bookend" the allowed colors
enum polygonColorOptions{blank=0, white, red, orange, yellow, green,
    light_blue, dark_blue, purple, color_stop};

class polygonColor{
private:
    polygonColorOptions color;
public:
    polygonColor(){color=blank;}
    void setColor(polygonColorOptions inp_color){ ... }
    polygonColorOptions getColor(){return color;}

    void inputColorFromKeyboard(){ ... }
    void printColorInfo(){ ... }

    void inputRandomColor()
    {
        // this produces a random integer within [0 , RAND_MAX]
        int rand_value=rand();
        // this produces a random int within [1 , color_stop-1]
        int inp_color = (int)(   ( ((double)rand_value)/RAND_MAX ) * (color_stop-2) + 1 );

        setColor( (polygonColorOptions) inp_color);
    }
};
```

Variables are of integer type

Example: class “Rectangle” with colors

```
// enumerator: finite set of choices for the colors
// blank and color_stop "bookend" the allowed colors
enum polygonColorOptions{blank=0, white, red, orange, yellow, green,
    light_blue, dark_blue, purple, color_stop};

class polygonColor{
private:
    polygonColorOptions color;
public:
    polygonColor(){color=blank;}
    void setColor(polygonColorOptions inp_color){ ... }
    polygonColorOptions getColor(){return color;}

    void inputColorFromKeyboard(){ ... }
    void printColorInfo(){ ... }

    void inputRandomColor()
    {
        // this produces a random integer within [0 , RAND_MAX]
        int rand_value=rand();
        // this produces a random int within [1 , color_stop-1]
        int inp_color = (int)(   ( ((double)rand_value)/RAND_MAX ) * (color_stop-2) + 1 );

        setColor( (polygonColorOptions) inp_color);
    }
};
```

Typecast one of them to double to force the double-precision computation; otherwise integer arithmetic (division) is used

Example: class “Rectangle” with colors

```
// enumerator: finite set of choices for the colors
// blank and color_stop "bookend" the allowed colors
enum polygonColorOptions{blank=0, white, red, orange, yellow, green,
    light_blue, dark_blue, purple, color_stop};

class polygonColor{
private:
    polygonColorOptions color;
public:
    polygonColor(){color=blank;}
    void setColor(polygonColorOptions inp_color) { ... }
    polygonColorOptions getColor(){return color;}

    void inputColorFromKeyboard() { ... }
    void printColorInfo() { ... }

    void inputRandomColor()
    {
        // this produces a random integer within [0 , RAND_MAX]
        int rand_value=rand();
        // this produces a random int within [1 , color_stop-1]
        int inp_color = (int)(   ( ((double)rand_value)/RAND_MAX ) * (color_stop-2) + 1 );

        setColor( (polygonColorOptions) inp_color);
    }
};
```

Example: class “Rectangle” with colors

```
class rectangleWcolor{
public:
    rectangle the_rectangle;
    polygonColor the_color;
public:
    void inputFromKeyboard()
    {
        the_rectangle.inputSidesFromKeyboard();
        the_color.inputColorFromKeyboard();
    }
    void printInfo()
    {
        if( (the_rectangle.isInitialized()) || (the_color.getColor()!=blank) )
        {
            cout << endl << " ----------- " << endl;
            the_rectangle.printRectangleInfo();
            the_color.printColorInfo();
            cout << " ----------- " << endl;
        }
        else
            cout << " printInfo(): Rectangle w Color not intialized" << endl;
    }
    void inputRandomValues(double max_val=100)
    {
        the_rectangle.inputRandomSides(max_val);
        the_color.inputRandomColor();
    }
};
```

Example: class “Rectangle” with colors

```
class rectangleWcolor{
public:
    rectangle the_rectangle;
    polygonColor the_color;
public:
    void inputFromKeyboard()
    {
        the_rectangle.inputSidesFromKeyboard();
        the_color.inputColorFromKeyboard();
    }
    void printInfo()
    {
        if( (the_rectangle.isInitialized()) || (the_color.getColor()!=blank) )
        {
            cout << endl << " ----------- " << endl;
            the_rectangle.printRectangleInfo();
            the_color.printColorInfo();
            cout << " ----------- " << endl;
        }
        else
            cout << " printInfo(): Rectangle w Color not intialized" << endl;
    }
    void inputRandomValues(double max_val=100)
    {
        the_rectangle.inputRandomSides(max_val);
        the_color.inputRandomColor();
    }
};
```

Random values within (0, max_val] assigned to each side

Example: class “Rectangle” with colors

Public Member function of class rectangle:

```
void inputRandomSides(double max_val=100)
{
    if(!isInitialized())
    {
        double in_sideA, in_sideB;
        int rand_valueA=0;
        int rand_valueB=0;

        // check that max_val is a meaningful value
        if(max_val<=0)
            max_val=100;

        // this produces a random integer within (0 , RAND_MAX]
        while(rand_valueA==0)
            rand_valueA=rand();
        while(rand_valueB==0)
            rand_valueB=rand();
        // this produces a random double within (0 , max_val]
        in_sideA=( ((double)rand_valueA)/RAND_MAX )*max_val;
        in_sideB=( ((double)rand_valueB)/RAND_MAX )*max_val;
        inputSides(in_sideA, in_sideB);
    }
    else
        cout << "Error in inputRandomSides(): Rectangle is already initialized " << endl;
}
```

Random values within (0, max_val] assigned to each side;
If the function is called without an argument inputRandomSides() the default value 100 is used for max_val

Example: class “Rectangle” with colors

```
void array_of_random_object_example()
{
    double test_sideA, test_sideB, random_max, area, area_threshold=300;
    polygonColorOptions pcolr, pcolr_target=white;
    const int arraysize=100;
    int count=0;
    // these are N instantiations (objects) of the class rectangle
    // i.e. variables of the user-defined type "rectangle"
    rectangleWcolor rct_array[arraysize];
    random_max=2*(sqrt(area_threshold));

    // initialize random seed via srand()
    // using the current time to set the seed
    srand(time(NULL));
    /////

    for(int i=0; i<arraysize; i++)
    {
        rct_array[i].inputRandomValues(random_max);
        rct_array[i].printInfo();
    }
    cout << endl << endl << "      ////////////  " << endl << endl;
    for(int i=0; i<arraysize; i++)
    {
        area=rct_array[i].the_rectangle.getArea();
        pcolr=rct_array[i].the_color.getColor();
        if( (area>area_threshold) && (pcolr==pcolr_target)  )
        {
            cout << "rectangle position "<< i << ":" << endl;
            cout << "rectangle area is: " << area << endl;
            cout << "rectangle color is: " << pcolr << endl;
            cout << endl;
            count++;
        }
    }
    cout << endl << " " << count << " rectangles out of " << arraysize << " matched the search criteria." << endl;
}
```

Any question?