Subroutines & Libraries

01204111 Computer and Programming

Department of Computer Engineering
Faculty of Engineering
Kasetsart University.

Cliparts are taken from http://openclipart.org

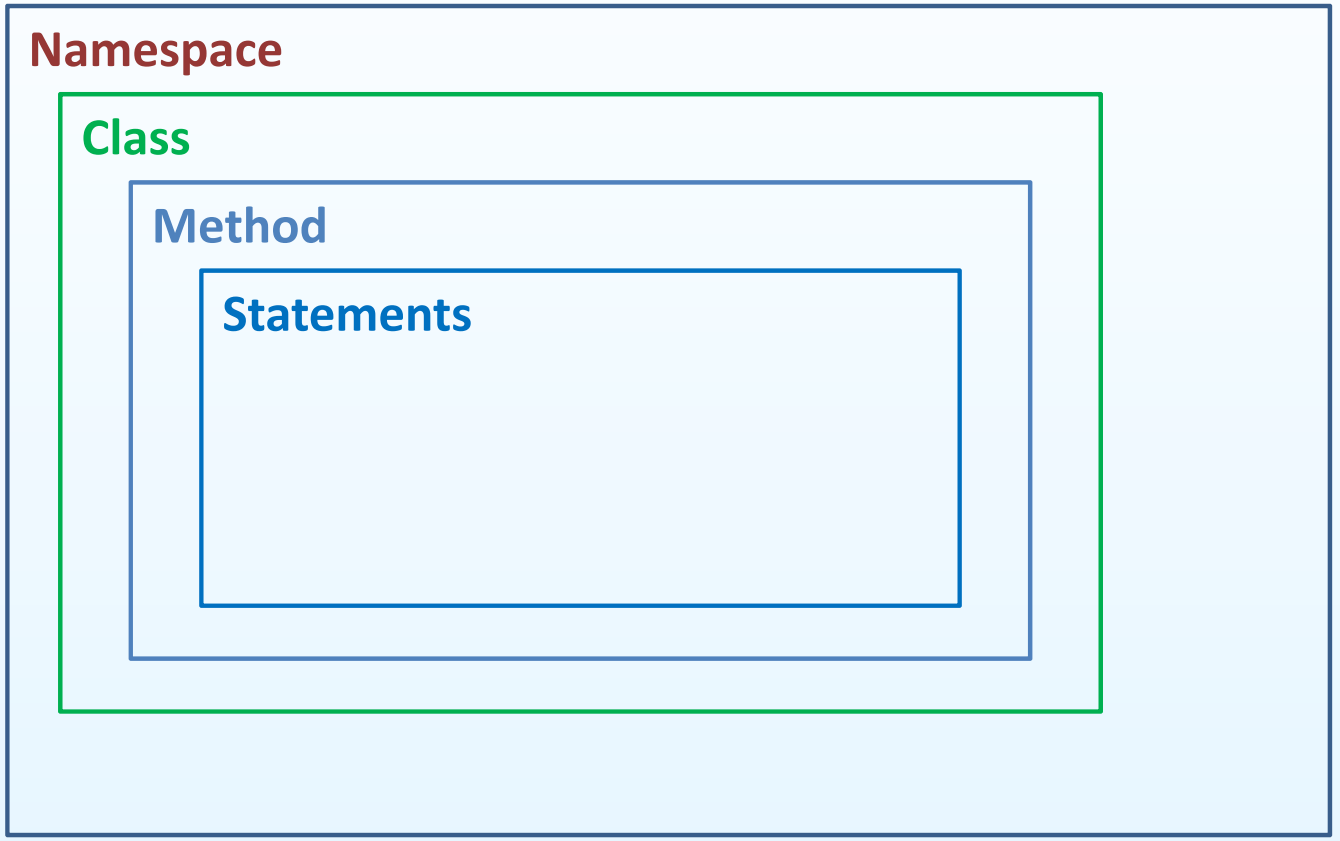
Department of Computer Engineering Kasetsart University

MIKE LABORATORY

Outline

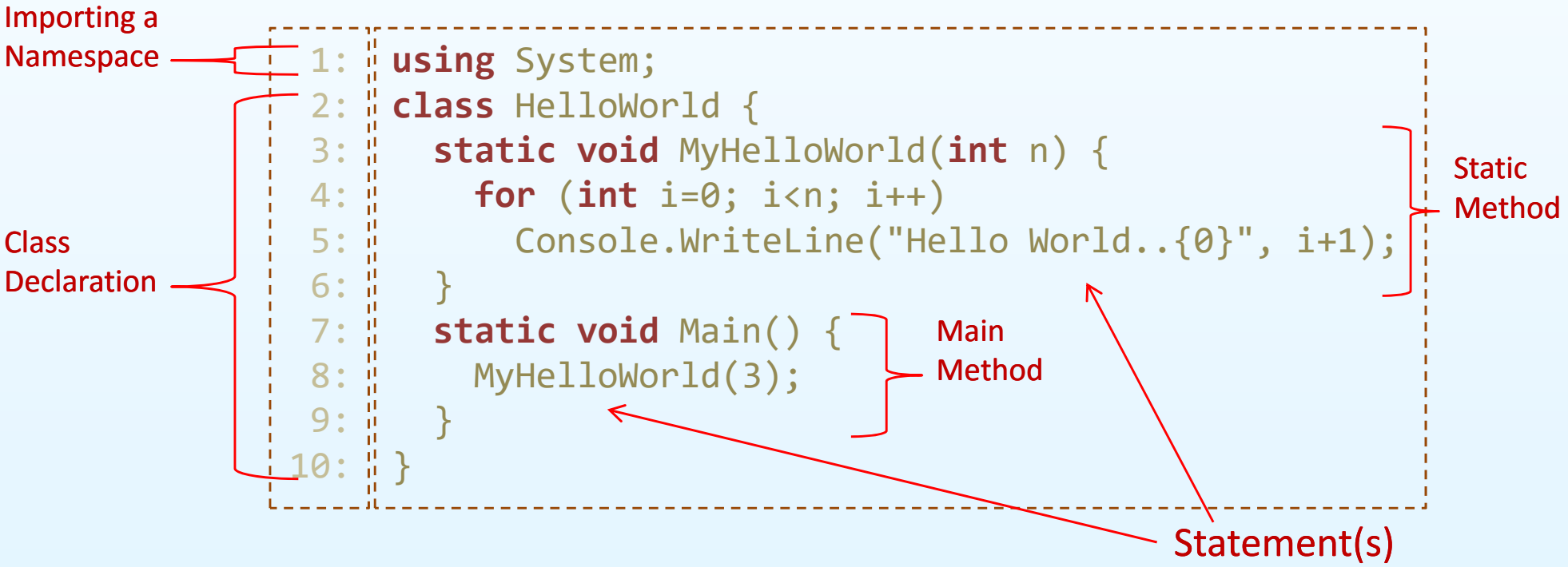
- Components of method
- Parameter passing

C# Elements



A Simple C# Program

- A C# program consists of at least one class.
- A class consists of at least one method.
- C# always starts execution at the **`Main()`** method.



Methods

- A method is a code block that contains a series of statements.
- The `Main()` method is the entry point for every C# application and it is called when the program is started.

```
 1: using System;
 2: class HelloWorld {
 3:   static void MyHelloWorld(int n) {
 4:     for (int i=0; i<n; i++)
 5:       Console.WriteLine("Hello World..{0}", i+1);
 6:   }
 7:   static void Main() {
 8:     MyHelloWorld(3);
 9:   }
10: }
```

Primitive Methods

- `Parse()` and `ToString()` methods

```
string str = "30";
int x = int.Parse(str);
int number = 0;
bool success = int.TryParse(str, out number);
string y = number.ToString();
```

- `Convert.To`*`XXX`*`()` methods

```
int a = Convert.ToInt32("30");
double b = Convert.ToDouble("30.25");
```

- `MaxValue` and `MinValue` properties

```
int a = int.MaxValue;
string s = a.ToString();
```

- Equal methods

```
int x = 10, y = 11, z = 10;
Console.WriteLine("{0} {1}", x==y, x.Equals(z));
```

Math Methods

```
Console.WriteLine(Math.Abs(-12.34));                  // 12.34
Console.WriteLine(Math.Ceiling(3.29));      // 4
Console.WriteLine(Math.Floor(3.29));                  // 3
Console.WriteLine(Math.Cos(Math.PI/4));     // 0.707106781186548
Console.WriteLine(Math.Exp(1));             // 2.71828182845905
Console.WriteLine(Math.Log(4));             // 1.38629436111989(base e)
Console.WriteLine(Math.Log10(100));         // 2
Console.WriteLine(Math.Max(2,3));           // 3
Console.WriteLine(Math.Pow(5,3));           // 125
Console.WriteLine(Math.Round(12.345678,4));// 12.3457
Console.WriteLine(Math.Sqrt(2));            // 1.4142135623731
```

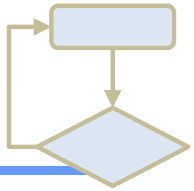
Task: Circle Area

- Program will ask the user to input the radius value of a circle, calculate the circle's area, and then print the resulting circle's area to screen.

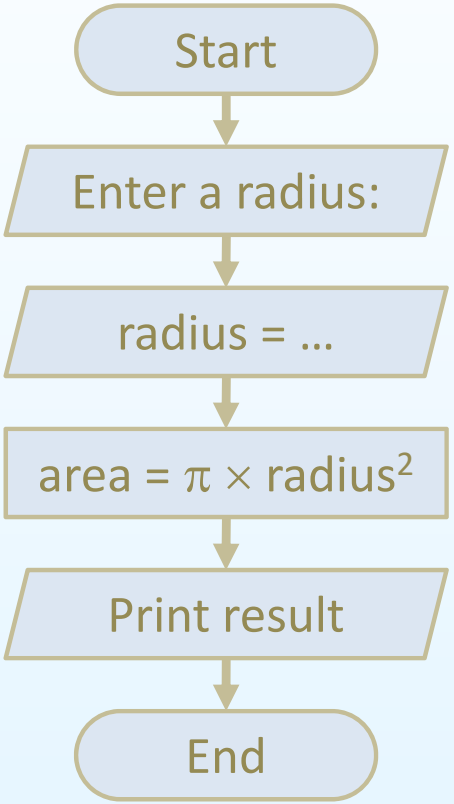
Circle Area - Ideas

- Need to know what is the radius of the underlying circle
- Compute the circle's area
  - area = $\pi \times$ radius $\times$ radius
- Show the result to screen

Circle Area - Steps

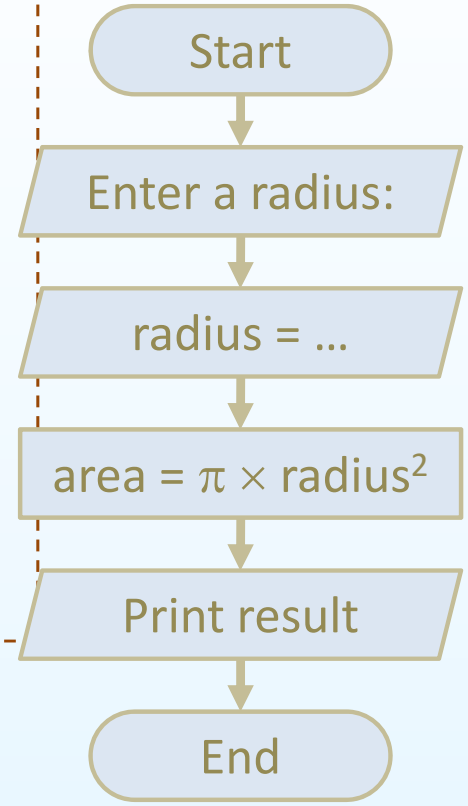


- Tell user to input the radius to the program
- Get input radius from the user
- Calculate the Area
  - area = $\pi \times$ radius $\times$ radius
- Print the resulting Area
- Pause the screen

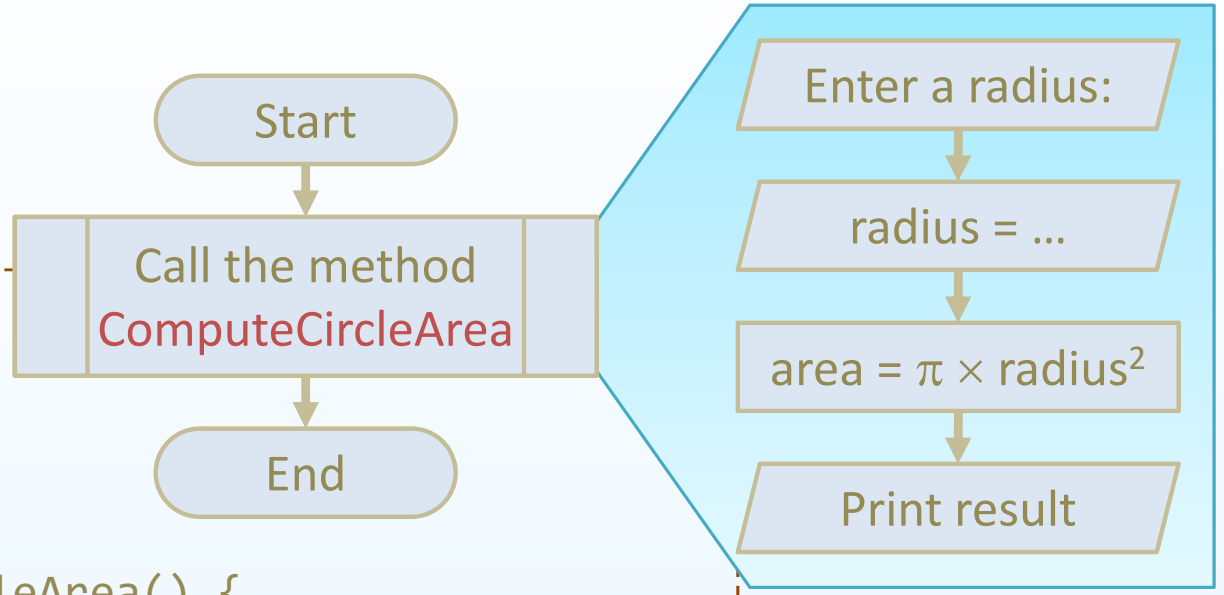


Circle Area (v1) - Program

```
 1: using System;
 2: class CircleArea {
 3:   static void Main() {
 4:     Console.Write("Enter a radius: ");
 5:     double radius = double.Parse(Console.ReadLine());
 6:     double area = Math.PI*radius*radius;
 7:     Console.WriteLine(
           "Area of a circle with radius {0} is {1}",
           radius, area);
 8:     Console.ReadKey(true);
 9:   }
10: }
```

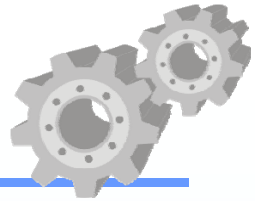


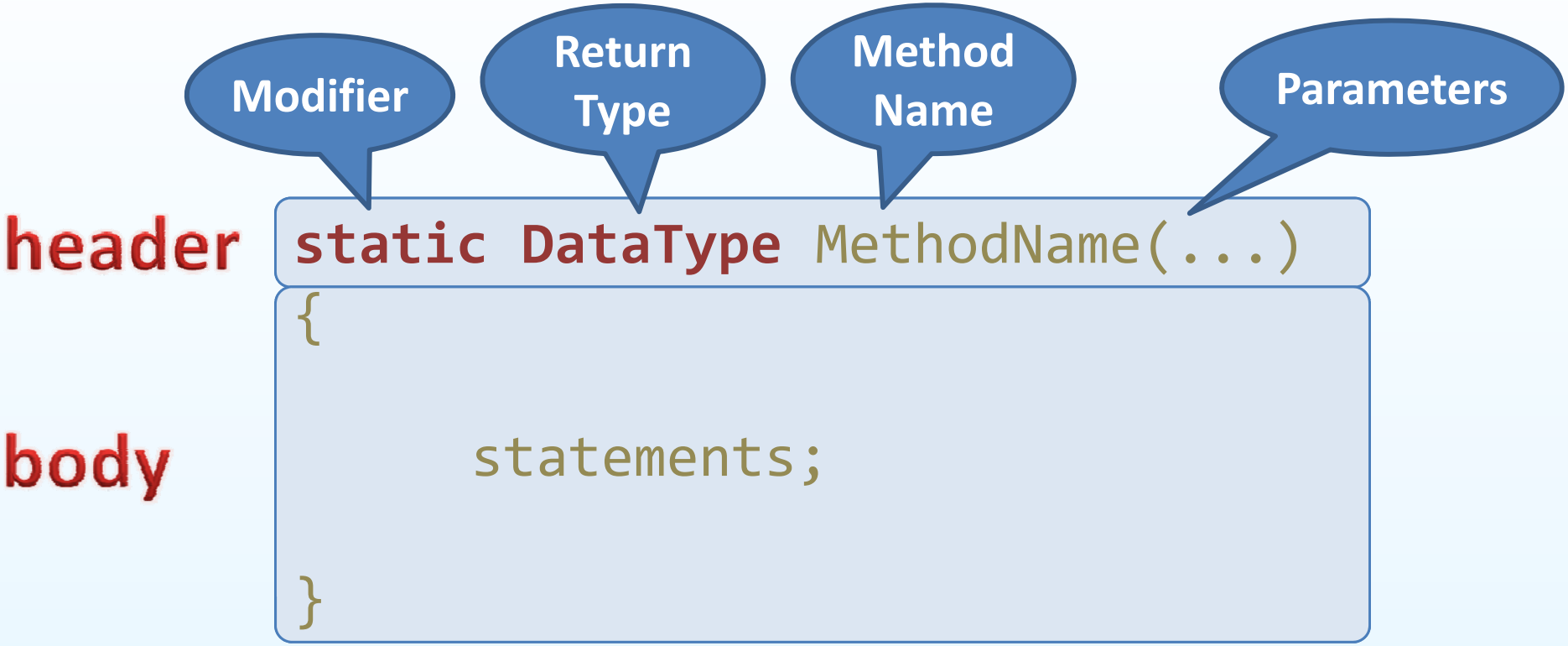
Circle Area (v2) - Program



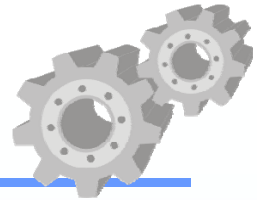

```
 1: class CircleArea {
 2:   static void Main() {
 3:     ComputeCircleArea();
 4:     Console.ReadKey(true);
 5:   }
 6:   static void ComputeCircleArea() {
 7:     Console.Write("Enter a radius: ");
 8:     double radius = double.Parse(Console.ReadLine());
 9:     double area = Math.PI*radius*radius;
10:     Console.WriteLine(
            "Area of a circle with radius {0} is {1}",
            radius, area);
11:   }
12: }
```

Method Declaration





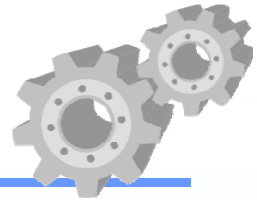
Return Type



- Indicate what type of value is returned when the method is completed
- For a non-returned value method, we use `void` keyword:

```
static void ComputeCircleArea() {
    ...
}
```

Method Name



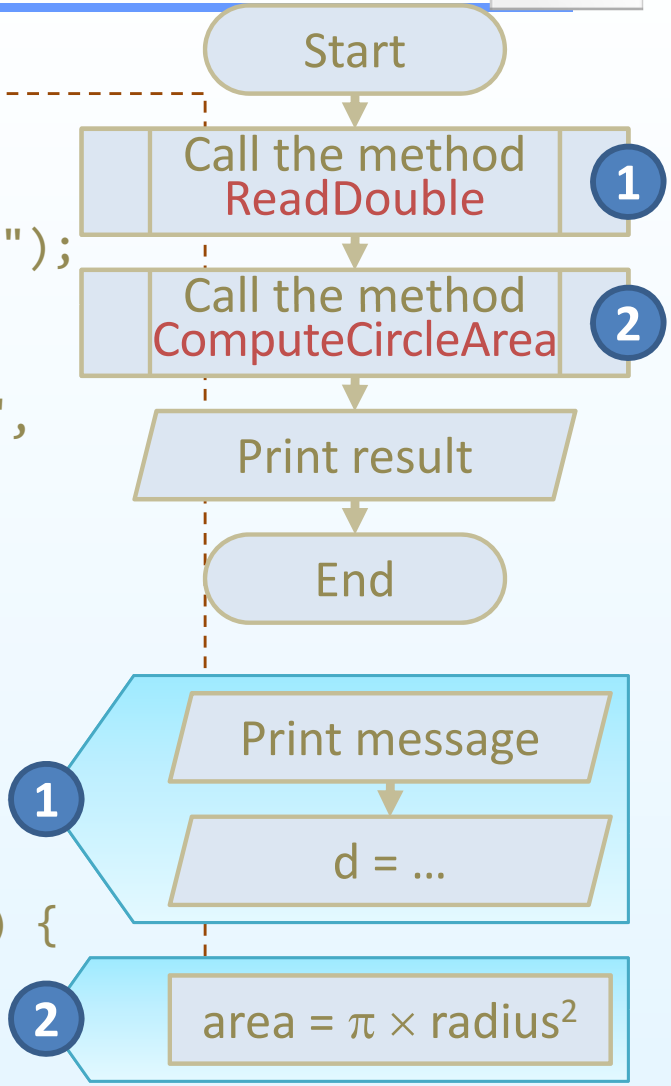
- Follow the rules for creating an identifier
- Examples:

```
static void ComputeCircleArea() {
    ...
}
```

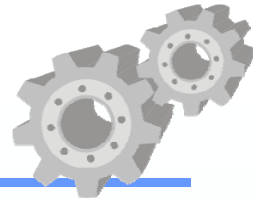
  - `CalculateSalesTax()`
  - `AssignSectionNumber()`
  - `DisplayResults()`
  - `ConvertInputValue()`
  - …

Circle Area (v3) - Program

```
 1: class CircleArea {
 2:   static void Main() {
 3:     double radius = ReadDouble("Enter a radius: ");
 4:     double area = ComputeCircleArea(radius);
 5:     Console.WriteLine(
            "Area of a circle with radius {0} is {1}",
            radius, area);
 6:     Console.ReadKey(true);
 7:   }
 8:   static double ReadDouble(string prompt) {
 9:     Console.Write(prompt);
10:     double d = double.Parse(Console.ReadLine());
11:     return d;
12:   }
13:   static double ComputeCircleArea(double radius) {
14:     return Math.PI * radius * radius;
15:   }
16: }
```



Return Type



- Indicate what type of value is returned when the method is completed
- For a returned value method, we use the primitive type:

```
static double ReadDouble(string prompt) {
    ...
}
```

```
static double ComputeCircleArea(double radius) {
    ...
}
```

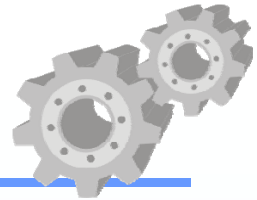
Caveats – `return` Statement

- Require the `return` statement for all returned value methods
- Return a compatible value

```
static double ReadDouble(string prompt) {
    Console.Write(prompt);
    double d = double.Parse(Console.ReadLine());
    return d;
}
```

```
static double ComputeCircleArea(double radius) {
    return Math.PI * radius * radius;
}
```

Parameters



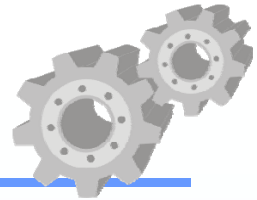
- Appear inside parentheses
- Include pairs of data type and identifier

```
static double ReadDouble(string prompt) {
    ...
}
```

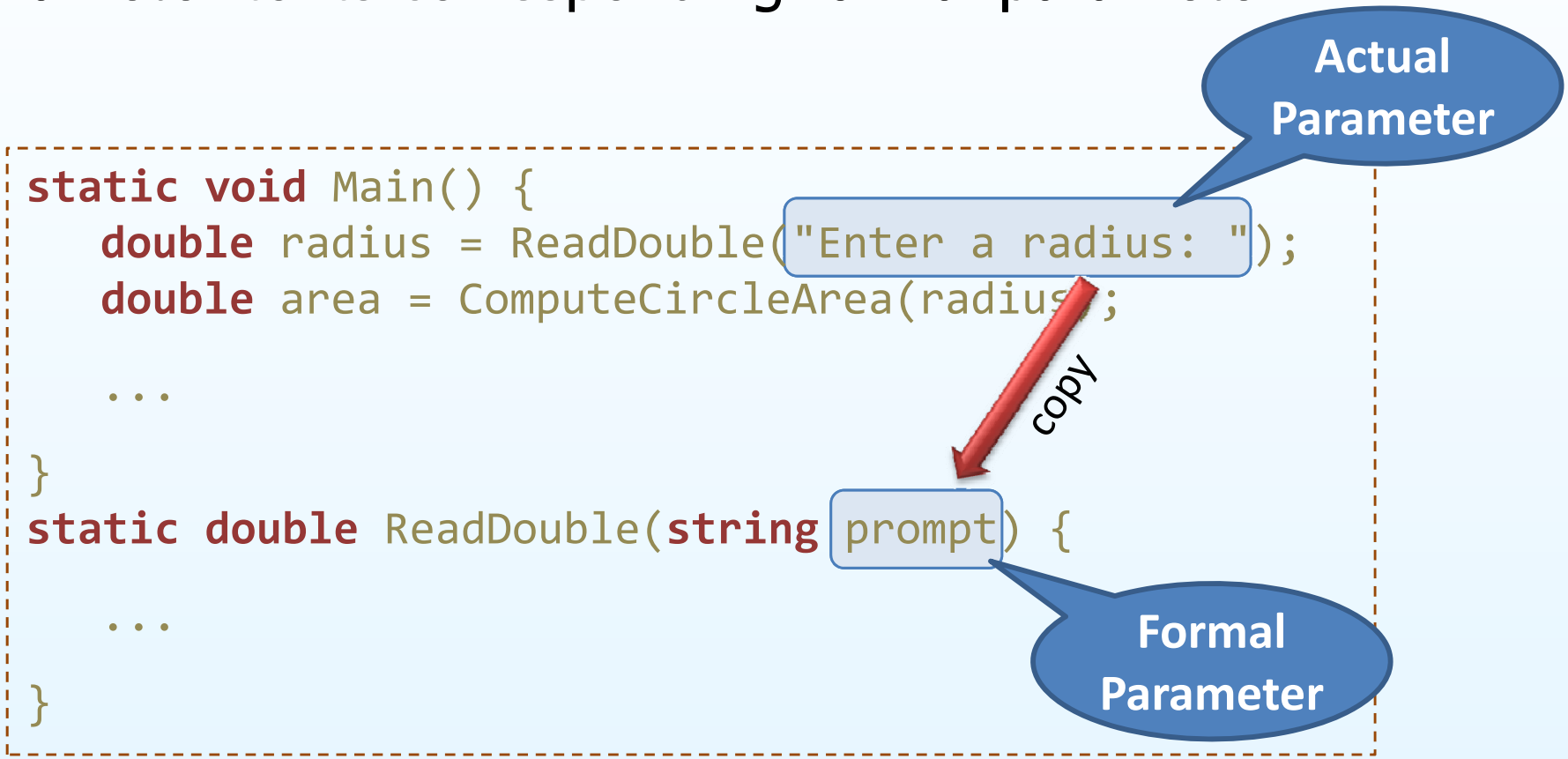
```
static double ComputeCircleArea(double radius) {
    ...
}
```

```
static void GetXY(double x, double y) {
    ...
}
```

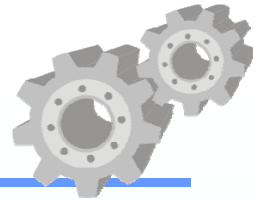
Pass by Value Parameters



- Mechanism of copying the value from an actual parameter to its corresponding formal parameter



Pass by Value Parameters



- Mechanism of **copying** the value from an **actual parameter** to its corresponding **formal parameter**

```
static void Main() {
    double radius = ReadDouble("Enter a radius: ");
    double area = ComputeCircleArea(radius);

    ...

}
static double ComputeCircleArea(double radius) {

    ...

}
```

Actual Parameter

copy

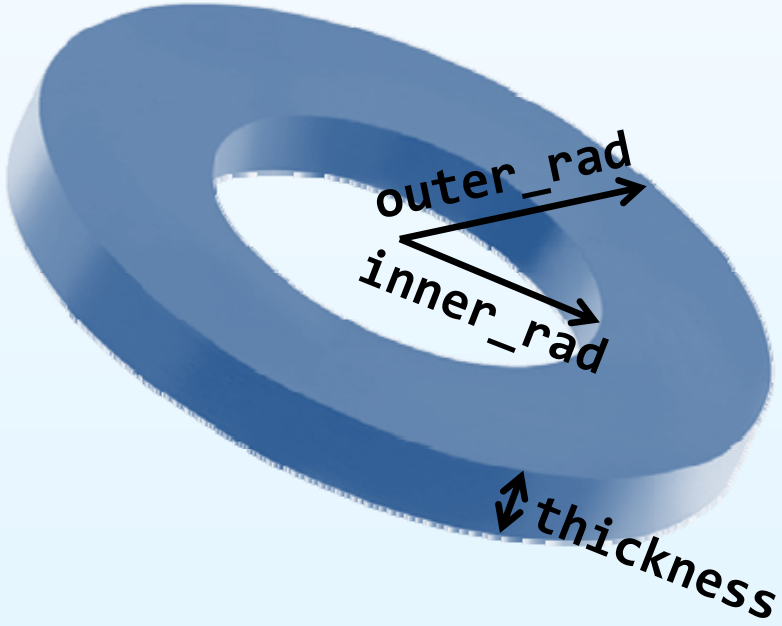
Formal Parameter

Task: Flat Washers

- You work for a hardware company that manufactures flat washers. To estimate shipping costs, your company needs a program that computes the weight of a specified quality of flat washers.

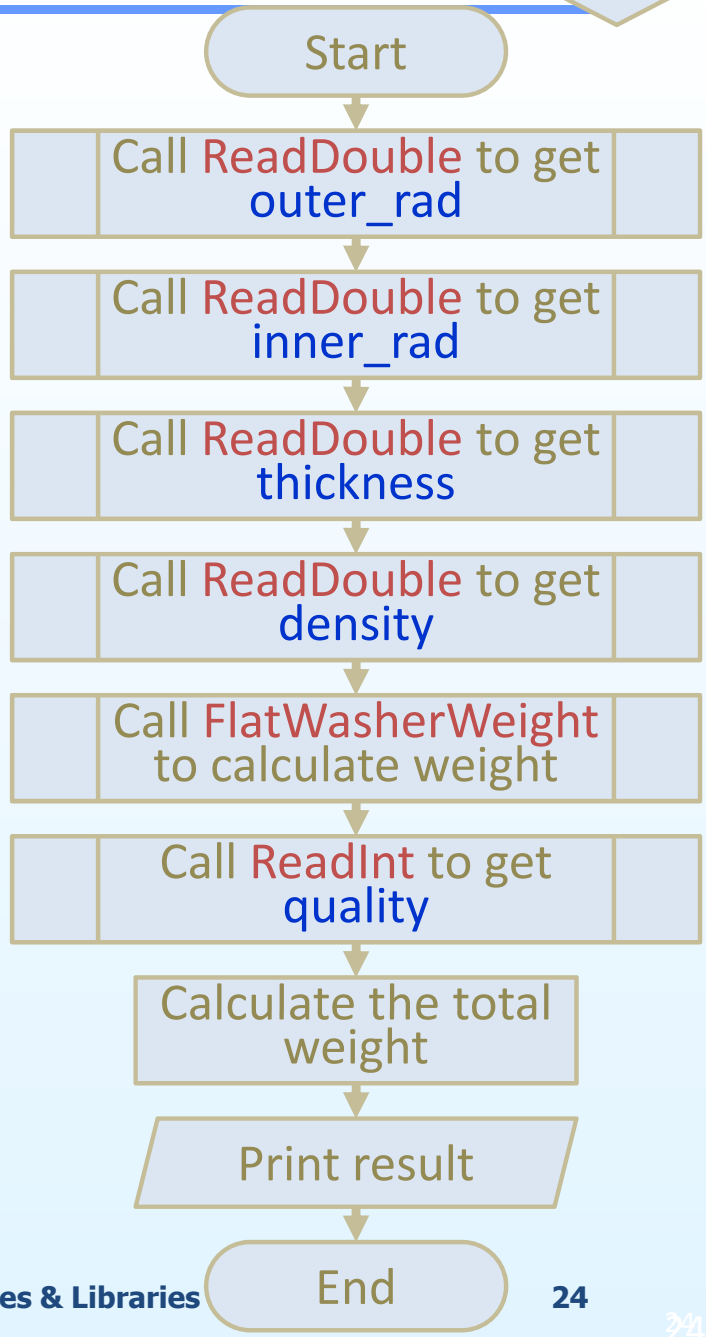
Flat Washers - Ideas

- A flat washer resembles a small donut (see the figure).
- To compute the weight of a single flat washer, you need to know its *rim area*, *thickness*, and *density* of the material
  - Here, we can reuse the **`ReadDouble()`** and **`ComputeCircleArea()`** methods of the previous CircleArea_v3 program
- Requirements:
  - Radius of flat washer
  - Radius of hole
  - Thickness
  - Density
  - Quantity



Flat Washers - Steps

- Get the washer's outer radius, inner radius, thickness, and the material density
- Compute the weight of one flat washer
  - unit_weight = rim_area × thickness × density
- Get quantity of washers
- Compute the weight of batch of washers
  - total_weight = unit_weight × quantity
- Print the resulting weight of batch
- Pause the screen



Flat Washers - Program

```
 1: using System;
 2: class FlatWasher {
 3:   static void Main() {
 4:     double outer_rad = ReadDouble("Enter the outer radius (cm.): ");
 5:     double inner_rad = ReadDouble("Enter inner radius (cm.): ");
 6:     double thickness = ReadDouble("Enter thickness (cm.): ");
 7:     double density   = ReadDouble("Enter density (g/cubic cm.): ");
 8:     double unit_weight = FlatWasherWeight(outer_rad, inner_rad,
                                              thickness, density);
 9:     int    quantity  = ReadInt("Enter the quantity (pieces): ");
10:     double total_weight = unit_weight * quantity;
11:     Console.WriteLine("Weight of the batch is {0:f2} grams",
                          total_weight);
12:     Console.ReadKey(true);
13:   }
      ... // codes lines 14-30 will be continued in subsequent pages
31: }
```

Flat Washers - Program

```
14:   static double ReadDouble(string prompt) {
15:     Console.Write(prompt);
16:     double d = double.Parse(Console.ReadLine());
17:     return d;
18:   }
19:   static int ReadInt(string prompt) {
20:     Console.Write(prompt);
21:     int i = int.Parse(Console.ReadLine());
22:     return i;
23:   }
24:   static double ComputeCircleArea(double radius) {
25:     return Math.PI * radius * radius;
26:   }
27:   static double FlatWasherWeight(double outer_rad, double inner_rad,
                                     double thickness, double density) {
28:     double rim_area = ComputeCircleArea(outer_rad)
                        - ComputeCircleArea(inner_rad);
29:     return rim_area * thickness * density;
30:   }
```

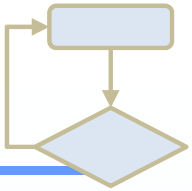
Task: Average of Three

- Program will ask three integer input values from the user, calculate the average of those three values, and then print the result to screen.

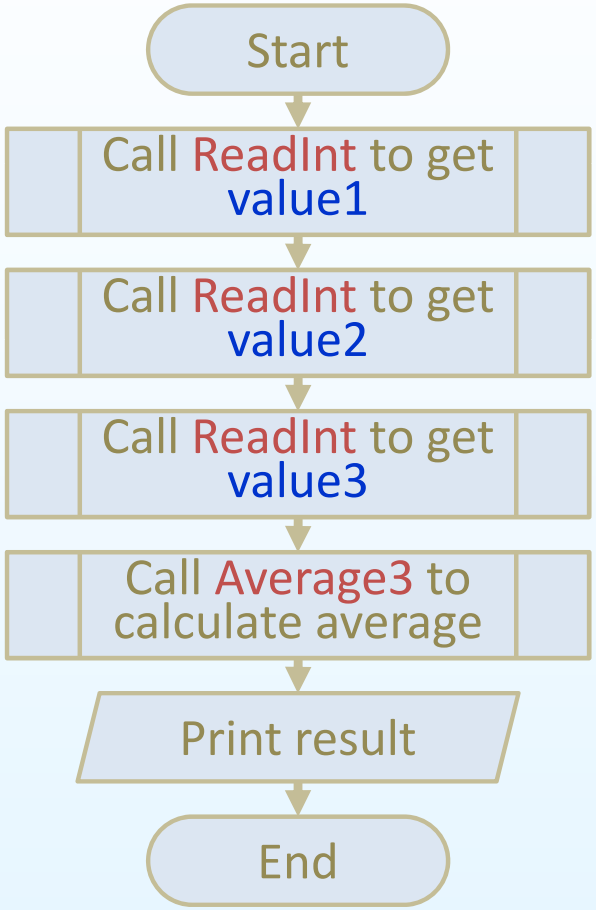
Average of Three - Ideas

- Need to know the three integer values, i.e., value1, value2, value3
- Compute the average
  - average = (value1 + value2 + value3)/3
- Show the result to screen

Average of Three - Steps



- Get input three input integer values from the user
- Calculate the average
  - average = (value1 + value2 + value3)/3
- Print the resulting average
- Pause the screen



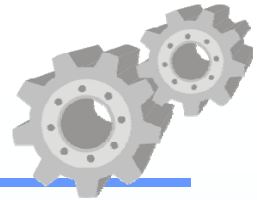
Average of Three (v1) - Program

```
 1: class AverageOfThree {
 2:   static void Main() {
 3:     /* read three integers */
 4:     int value1 = ReadInt("1st value: ");
 5:     int value2 = ReadInt("2nd value: ");
 6:     int value3 = ReadInt("3rd value: ");
 7:     /* compute and output their average */
 8:     Console.WriteLine("average is {0:f4}",
                          Average3(value1, value2, value3));
 9:     Console.ReadKey(true);
10:   }
11:   static double Average3(int x, int y, int z) {
12:     return (x+y+z)/3.0;
13:   }
14:   static int ReadInt(string prompt) {
15:     Console.Write(prompt);
16:     int i = int.Parse(Console.ReadLine());
17:     return i;
18:   }
19: }
```

Average of Three (v2) - Program

```
 1: class AverageOfThree {
 2:   static void Main() {
 3:     /* read three integers */
 4:     int value1, value2, value3;
 5:     Read3Integers(out value1, out value2, out value3);
 6:     Console.WriteLine("average is {0:f4}",
                          Average3(value1, value2, value3));
 7:     Console.ReadKey(true);
 8:   }
 9:   static double Average3(int x, int y, int z) { return (x+y+z)/3.0; }
10:   static void Read3Integers(out int x, out int y, out int z) {
11:     x = ReadInt("1st value: ");
12:     y = ReadInt("2nd value: ");
13:     z = ReadInt("3rd value: ");
14:   }
15:   static int ReadInt(string prompt) {
16:     Console.Write(prompt);
17:     int i = int.Parse(Console.ReadLine());
18:     return i;
19:   }
20: }
```

Pass by Reference Parameters



- Mechanism of a reference type, i.e., the formal parameter refers to the corresponding actual parameter
  - We use (for now) the `out` keyword.

```
static void Main() {
    int value1, value2, value3;
    Read3Integers(out value1, out value2, out value3);

    ...
}
static void Read3Integers(out int x, out int y, out int z) {

    ...
}
```

Actual Parameter

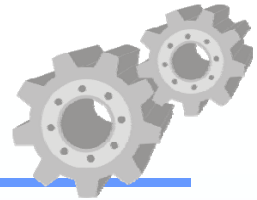
refer to

refer to

refer to

Formal Parameter

Pass by Reference Parameters



```
static void Main() {
   int value1, value2, value3;
   Read3Integers(out value1, out value2, out value3);
   ...
}
static void Read3Integers(out int x, out int y, out int z) {
   x = ReadInt("1st value: ");
   y = ReadInt("2nd value: ");
   z = ReadInt("3rd value: ");
}
```

Now, value1 is 10 too.

Now, value2 is 20 too.

Now, value3 is 25 too.

refer to

refer to

refer to

Suppose the user input 10

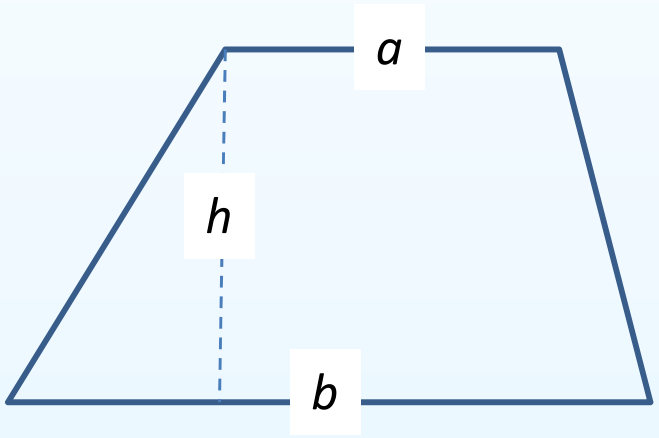
Suppose the user input 20

Suppose the user input 25

Task: Trapezoid

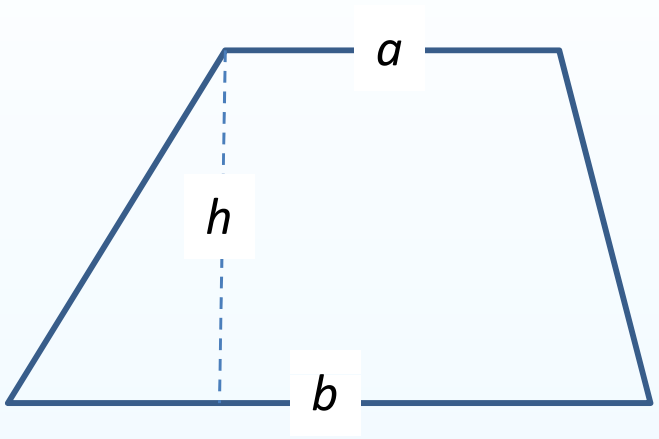
- In Euclidean geometry, a convex quadrilateral with at least one pair of parallel sides is referred to as a **trapezoid.**

(ref: https://en.wikipedia.org/wiki/Trapezoid)



$$area = \frac{a+b}{2}h$$

Trapezoid - Ideas

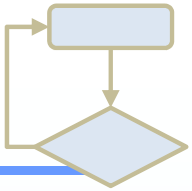


$$area = \frac{a+b}{2}h$$

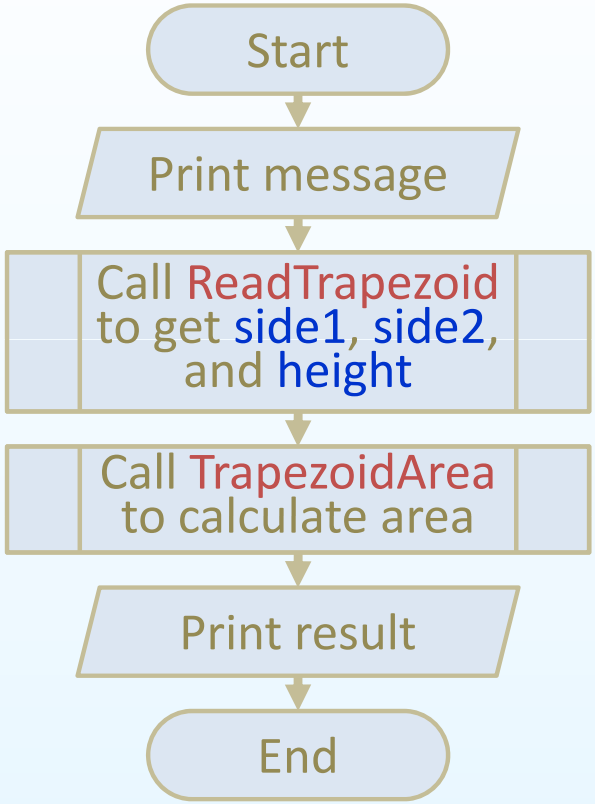
- From the above trapezoid formula, if we let *a* be close to zero (e.g., 1E-14), the area of the triangle (a trapezoid with a near-zero collateral) can also be approximately by

$$area = \frac{b}{2}h$$

Trapezoid - Steps



- Get three double values from the user:
  - (parallel) side1
  - (parallel) side2
  - height
- Calculate the trapezoid area
  - area = ((side1 + side2)/2) × height
- Print the resulting area
- Pause the screen

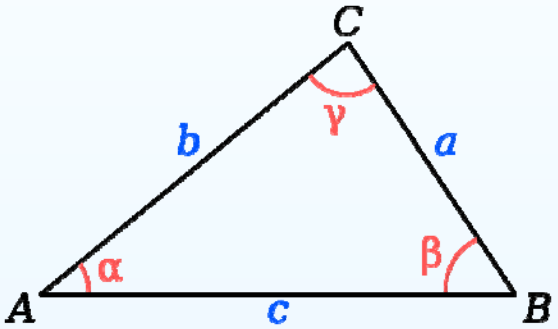


Trapezoid - Program

```
 1: class Trapezoid {
 2:   static void Main() {
 3:     double side1, side2, height;
 4:     Console.WriteLine("Give me the size of your trapezoid.");
 5:     ReadTrapezoid(out side1, out side2, out height);
 6:     Console.WriteLine("Trapezoid's area is {0}",
                            TrapezoidArea(side1, side2, height));
 8:     Console.ReadKey(true);
 9:   }
10:   static double TrapezoidArea(double a, double b, double h)
11:   { return 0.5*(a+b)*h; }
12:   static void ReadTrapezoid(out double a, out double b, out double h) {
13:     // read two parallel side lengths (a and b), and height of a trapezoid (h)
14:     a = ReadDouble("Parallel side 1's length: ");
15:     b = ReadDouble("Parallel side 2's length: ");
16:     h = ReadDouble("Height: ");
17:   }
18:   static double ReadDouble(string prompt) {
19:     Console.Write(prompt);
20:     double d = double.Parse(Console.ReadLine());
21:     return d;
22:   }
23: }
```

Task: Triangle Area (Heron)

- In geometry, **Heron's formula** (sometimes called Hero's formula), named after Hero of Alexandria, gives the area of a triangle by requiring no arbitrary choice of side as base or vertex as origin, contrary to other formulas for the area of a triangle, such as half the base times the height or half the norm of a cross product of two sides.



(ref: https://en.wikipedia.org/wiki/Heron's_formula)

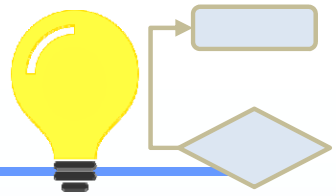
- Heron's formula states that the area of a triangle whose sides have lengths *a*, *b*, and *c* is

$$area = \sqrt{a(s-a)(s-b)(s-c)}\ ,$$

where *s* is the semiperimeter of the triangle; that is,

$$s = \frac{a+b+c}{2}$$

Triangle Area (Heron) - Ideas + Step



- Get the x-y coordinate of the triangle's 3 vertices
- Calculate the length of the lines *a*, *b*, and *c* which are connected to those 3 vertices
- Calculate the semiperimeter
- Calculate the triangle's area using the Heron's formula
- Print the resulting area
- Pause the screen

Triangle Area (Heron) - Program

```
 1: class Program {
 2:   static void Main() {
 3:   // X-Y coordinates of the triangle's 3 vertices
 4:   double x1, y1, x2, y2, x3, y3;
 5:
 6:   ReadTriangle(out x1, out y1, out x2, out y2, out x3, out y3);
 7:   Console.WriteLine("area of the triangle is {0:f4}",
                        TriangleArea(x1, y1, x2, y2, x3, y3));
 8:
 9:   Console.ReadKey(true);
10:
11:   }
      ... // codes lines 12-46 will be continued in subsequent pages
47: } // test with (15,15) (23,30) (50,25), area = 222.5
```

Triangle Area (Heron) - Program

```
12: // read X-Y co-ordinates of 3 vertices of a triangle
13: static void ReadTriangle(out double x1, out double y1,
                             out double x2, out double y2,
                             out double x3, out double y3) {
14:   Console.WriteLine(
        "Enter X-Y coordinates of the three vertices of your triangle:");
15:   Console.WriteLine("1st vertex:");
16:   x1 = ReadDouble("x? ");
17:   y1 = ReadDouble("y? ");
18:   Console.WriteLine("2nd vertex:");
19:   x2 = ReadDouble("x? ");
20:   y2 = ReadDouble("y? ");
21:   Console.WriteLine("3rd vertex:");
22:   x3 = ReadDouble("x? ");
23:   y3 = ReadDouble("y? ");
24: }
25: static double ReadDouble(string prompt) {
26:   Console.Write(prompt);
27:   return double.Parse(Console.ReadLine());
28: }
```

Triangle Area (Heron) - Program

```
29: // Given the 3 vertices, compute triangle area using Heron's Formula
30: static double TriangleArea(double x1, double y1,
                               double x2, double y2,
                               double x3, double y3) {
31:   // the famous variables of Heron's Formula
32:   double s, a, b, c;
33:   a = LineLength(x1, y1, x2, y2);
34:   b = LineLength(x2, y2, x3, y3);
35:   c = LineLength(x3, y3, x1, y1);
36:   s = 0.5*(a+b+c);
37:   // the Heron's formula itself
38:   return Math.Sqrt(s*(s-a)*(s-b)*(s-c));
39: }
40: // Given X-Y coordiates of 2 points, compute the line length that joins them
41: static double LineLength(double x1, double y1, double x2, double y2){
42:   return Math.Sqrt(Sqr(x1-x2)+Sqr(y1-y2)); // 2D Line length formula
43: }
44: static double Sqr(double x) {
45:   return x*x;
46: }
```

Conclusion

- Types of method
  - Non-returned value method
  - Returned value method
- Actual parameters VS. Formal parameters
- Types of parameter passing
  - Pass by value
  - Pass by reference (only, `out` keyword)