Input/Output and Mathematical Functions

01204111 Computer and Programming

Department of Computer Engineering
Kasetsart University

Cliparts are taken from http://openclipart.org

Outline

- More arithmetic operations
- Reading input
- Output formatting
- Data type conversions
- Useful mathematical functions and Math library

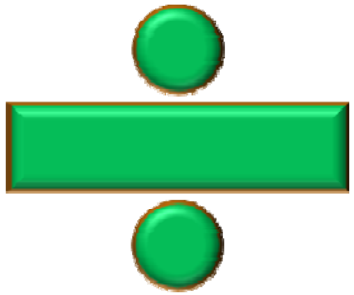
Task: *Bill Sharing with Tip*

- From the previous example, the total cost is 344 Baht. Each person must pay 344/5 = 68.8 Baht.
  - It is not easy to pay a fraction of a baht
  - Let us pay in whole amount and use the remaining amount as the tip
- Write a program to compute the amount each person has to pay and the amount of tip

Item	Price
Salad	82
Soup	64
Steak	90
Wine	75
Orange Juice	33

Bill Sharing with Tip – Ideas

- Compute the total amount as usual
- To compute the amount each has to pay
  - Divide the amount by the number of people
  - Keep only the whole amount
  - *Hint: integer division*



- Compute the remainder of the division
  - C# provides the % operator to compute the remainder from an integer division

Bill Sharing with Tip – Program

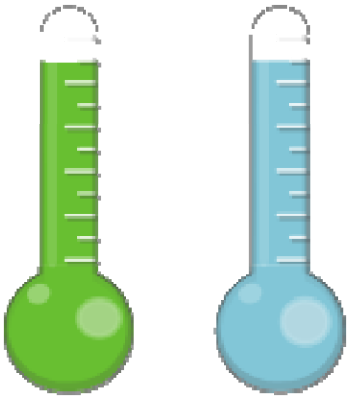
```
using System;

class Program
{
    static void Main()
    {
        int total;
        total = 82+64+90+75+33;
        Console.Write("Total amount: ");
        Console.WriteLine( total );
        Console.Write("Each has to pay: ");
        Console.WriteLine( total / 5 );
        Console.Write("Tip: ");
        Console.WriteLine( total % 5 );
    }
}
```

Task: *Celsius to Kelvin*

- The relationship between temperature in degrees Celsius (C) and Kelvin (K) is

$$K = C + 273.15$$



- Write a program that
  - Reads a temperature value in degrees Celsius from user using the keyboard
  - Then outputs the corresponding temperature in Kelvin

Celsius to Kelvin – Ideas

- We will use these steps:
  1. Read an input value from user using the keyboard; store it in a variable `celsius`
  2. Display the value of `celsius+273.15`
- Hints:
  - Recall that **Console** has the method **ReadLine()** to read an input from keyboard
  - This method returns a string value

Celsius to Kelvin – First Attempt

```
using System;

class Program
{
    static void Main()
    {
        Console.Write("Enter temperature in degrees Celcius: ");
        string celsius = Console.ReadLine();
        Console.Write("Temperature in Kelvin: ");
        Console.WriteLine(celsius+273.15);
    }
}
```

- This is the output. Is it correct?

```
Enter temperature in degrees Celcius: 37
Temperature in Kelvin: 37273.15
```

Join?

✓ Compile
✓ Run
✗ Expected output
= Logical error/ Semantics error

Strings and + Operator

- Using the + operator with a string may give an undesired result

Expression	Evaluated to	Remark
`"37" + 273.15`	`"37273.15"`	string + number
`273.15 + "37"`	`"273.1537"`	number + string
`37 + 273.15`	`310.15`	number + number

- So we must convert a string to a number before using +

Celsius to Kelvin – Revised Program

- Use the method `double.Parse`

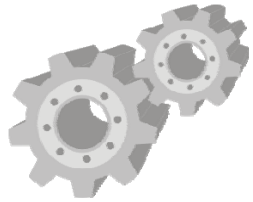
C# Practice
- Capitalise class (Eg. Int32, Console)
- Decapitalise object

```
using System;

class Program
{
    static void Main()
    {
        Console.Write("Enter temperature in degrees Celcius: ");
        double celsius = double.Parse(Console.ReadLine());
        Console.Write("Temperature in Kelvin: ");
        Console.WriteLine(celsius+273.15);
    }
}
```

object

Reading a Number – Details

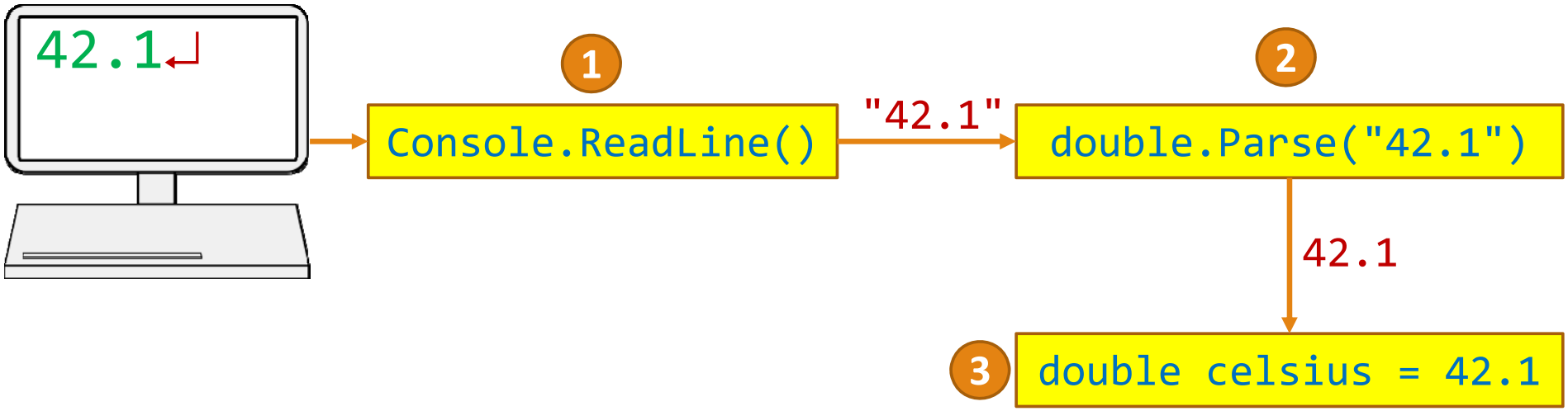


- This statement by itself contains three operations

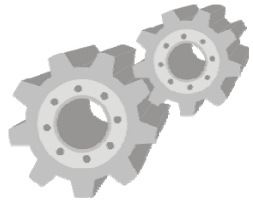
```
double celsius = double.Parse(Console.ReadLine());
```

3 2 1

- Let's break it down. Suppose the user enters `42.1` via keyboard



Parsing Other Numeric Types



- The Parse method is available for both **double** and **int** data types
  - **`double.Parse(s)`** converts the string *s* into a double

```
double gpa = double.Parse("3.41");
```

  - **`int.Parse(s)`** converts the string *s* into an int

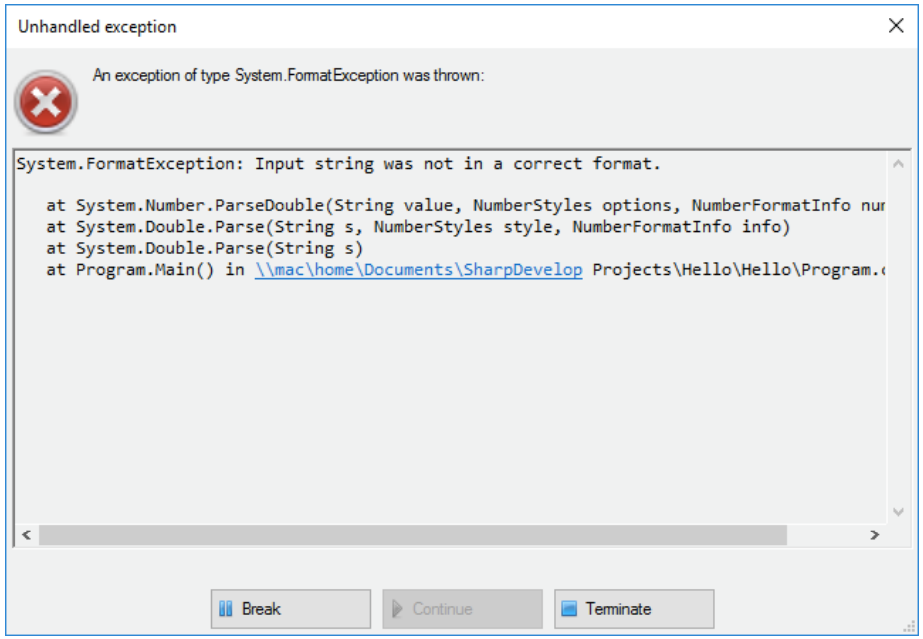
```
int age = int.Parse("21");
```

Dealing with Error

- It is possible to give an invalid input to `Console.ReadLine` that `double.Parse` is unable to convert
  - Such as `Hello`, `37+15`

```
Enter temperature in degrees Celcius: 37+15
```

- This will cause a ***run-time error***
  - It looks like this on SharpDevelop
  - When this happens, there is nothing we can do but to terminate (stop) the program



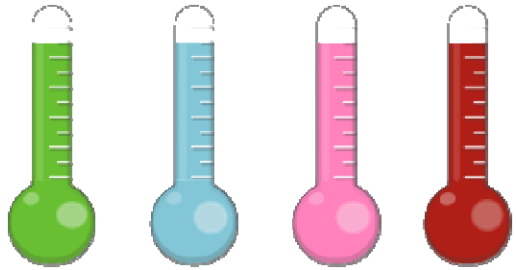
- C# provides the `TryParse` method which allows us to check for error
  - Details in later chapters

Task: Temperature Conversion

- Relationship between temperature in degrees Celsius and degrees Fahrenheit is:

$$\frac{C}{5} = \frac{R}{4} = \frac{F - 32}{9} = \frac{K - 273.15}{5}$$

where *C* , *R*, *F*, and *K* are temperature values in °C, °F, °R, and Kelvin, respectively



- Write a program that
  - Reads a temperature value in degrees Celsius
  - Then outputs the corresponding temperature values in degrees Fahrenheit,

Temperature Conversion - Ideas

- An equation like

$$\frac{C}{5} = \frac{F - 32}{9}$$

  cannot be entered into the C# program directly
  - (because C# does not know how to solve an equation)

- We need to solve the equation to find *F* ourselves

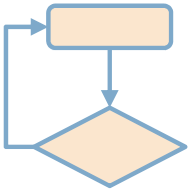
one, and only single variable

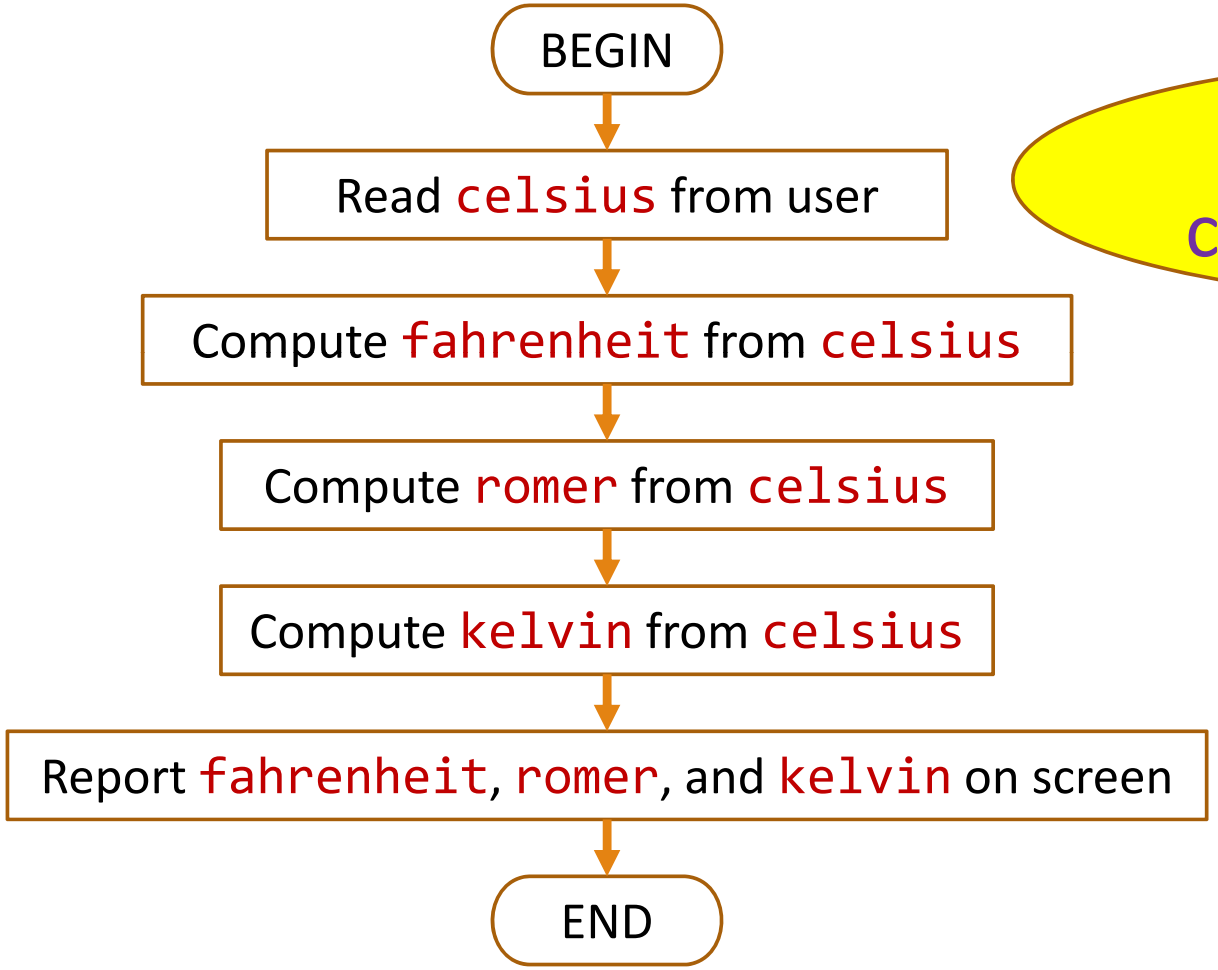
$$F = \frac{9C}{5} + 32$$

any expression

  which is directly mapped to an assignment operation

Temperature Conversion – Steps





This diagram is called a ***Flowchart***

Temperature Conversion – Program#1

```
using System;

class Program
{
    static void Main()
    {
        Console.Write("Enter temperature in degrees Celcius: ");
        double celsius = double.Parse(Console.ReadLine());
        double fahrenheit = ((celsius*9.0)/5.0)+32;
        double romer = (celsius*4)/5;
        double kelvin = celsius + 273.15;
        Console.Write(celsius);
        Console.WriteLine(" degrees Celsius is equal to:");
        Console.Write(" "); Console.Write(fahrenheit);
        Console.WriteLine(" degrees Fahrenheit");
        Console.Write(" "); Console.Write(romer);
        Console.WriteLine(" degrees Romer");
        Console.Write(" "); Console.Write(kelvin);
        Console.WriteLine(" Kelvin");
    }
}
```

Temperature Conversion – Program#2

This code uses `Console.WriteLine`'s output formatting capability

```
using System;

class Program
{
    static void Main()
    {
        Console.Write("Enter temperature in degrees Celcius: ");
        double celsius = double.Parse(Console.ReadLine());
        double fahrenheit = ((celsius*9.0)/5.0)+32;
        double romer = (celsius*4)/5;
        double kelvin = celsius + 273.15;
        Console.WriteLine("{0} degrees Celsius is equal to:", celsius);
        Console.WriteLine("  {0} degrees Fahrenheit", fahrenheit);
        Console.WriteLine("  {0} degrees Romer", romer);
        Console.WriteLine("  {0} Kelvin", kelvin);
    }
}
```

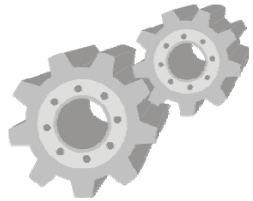
Temperature Conversion – Program#3

The :f2 inside {0} means formatting number with 2 decimal places

```
using System;

class Program
{
    static void Main()
    {
        Console.Write("Enter temperature in degrees Celcius: ");
        double celsius = double.Parse(Console.ReadLine());
        double fahrenheit = ((celsius*9.0)/5.0)+32;
        double romer = (celsius*4)/5;
        double kelvin = celsius + 273.15;
        Console.WriteLine("{0:f2} degrees Celsius is equal to:", celsius);
        Console.WriteLine("  {0:f2} degrees Fahrenheit", fahrenheit);
        Console.WriteLine("  {0:f2} degrees Romer", romer);
        Console.WriteLine("  {0:f2} Kelvin", kelvin);
    }
}
```

Output Formatting



- The methods `Console.Write` and `Console.WriteLine` provide special output formatting capabilities
- Examples:

```
int width = 30, height = 60;
Console.WriteLine("Size = {0}x{1}", width, height);
```

  - gives the output `Size = 30x60`

```
double salary = 16000;
Console.WriteLine("Salary is {0:f2}", salary);
```

  - gives the output `Salary is 16000.00`

Task: Phone Bill

- Long-distance rate for a domestic call is 2 baht/minute, while a fraction of a minute is charged as a whole minute
- For example
  - 1-minute call → 2 baht
  - 3-minute call → 6 baht
  - 5.2-minute call → 12 baht

- Write a program that
  - asks the user how many seconds is used for the call
  - then computes the total charge for the call

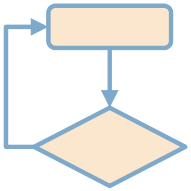
Phone Bill - Ideas

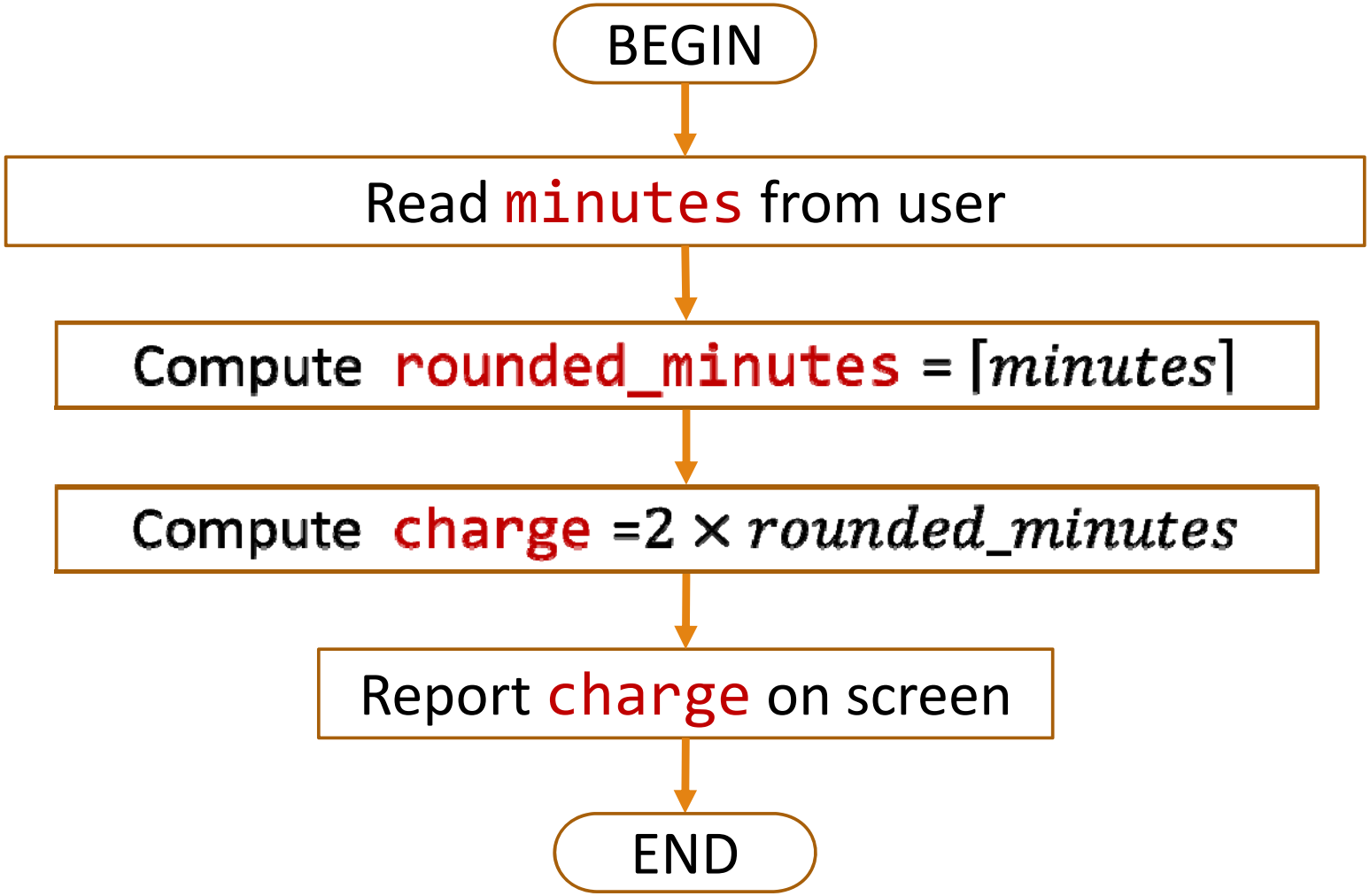
- At first, the problem looks like a typical division problem
- However, the fraction of the result must not be discarded this time, but will be rounded up to the nearest integer
  - E.g., 3 is rounded up to 3, while 3.1 is rounded up to 4
- Let x represent the call length in minutes; we want to know ***the smallest integer that is larger or equal to*** x
- Mathematically, we are computing

$$\lceil x \rceil$$

this is called
" the **ceiling** of x "

Phone Bill – Steps





Phone Bill – Program

```
 1: using System;
 2:
 3: class Program
 4: {
 5:     static void Main()
 6:     {
 7:         Console.Write("Enter call length in minutes: ");
 8:         double minutes = double.Parse(Console.ReadLine());
 9:         double rounded_minutes = Math.Ceiling(minutes);
10:         int charge = 2 * ((int)rounded_minutes);
11:         Console.WriteLine("Charge is {0} baht.", charge);
12:     }
13: }
```

explicit conversion
(Intends to convert datatype)

Phone Bill – Syntax Details

- Line 9, the expression

```
Math.Ceiling(minutes)
```

  is evaluated to $\lceil minutes \rceil$

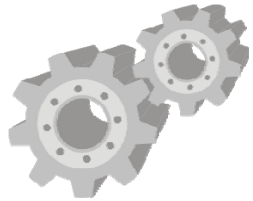
- Line 10, the expression

```
(int)rounded_minutes
```

  is called ***type casting***
  - This is required to convert a double to an int

Beginner challenge: Babylonian sqrt in C#

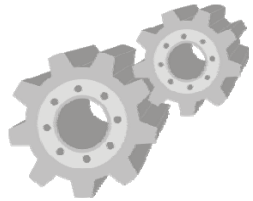
Math Library



- C# provides many mathematical functions and constants in the ***Math*** library
- Some common functions and constants are:

Expression	Evaluated to	Remark
`Math.Floor(x)`	$\lfloor x \rfloor$	Largest integer smaller or equal to x
`Math.Ceil(x)`	$\lceil x \rceil$	Smallest integer larger or equal to x
`Math.Abs(x)`	$\lvert x \rvert$	Absolute value of x
`Math.Pow(x,y)`	x^y	
`Math.Max(x,y)`	$\max(x,y)$	
`Math.Min(x,y)`	$\min(x,y)$	
`Math.Sqrt(x)`	$\sqrt{x}$	Square-root of x
`Math.PI`	π	approx.3.14159
`Math.E`	e	approx.2.71828

Data Type Conversion



- **Implicit conversion**
  - Allowed between two different numeric types, without loss of information, such as converting from an `int` to a `double`

```
int i = 5;
double d = i;
```

- **Explicit conversion**
  - ***Type casting*** converts one numeric type into another with potential loss of information

```
double d = 5.1;
int i = ((int) d);
```

  - ***String parsing*** converts a string into a numeric value

```
string s = "5.1";
double d = double.Parse(s);
```

Task: Savings Account

- When you have a savings account, the bank usually deposits interest back into your account every year
- You would like to know how much money you will have after a certain number of years
- Write a program that
  - lets user input the principle, rate (%), and years
  - outputs the amount you will have after the specified number of years

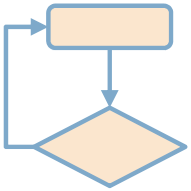
Savings Account - Ideas

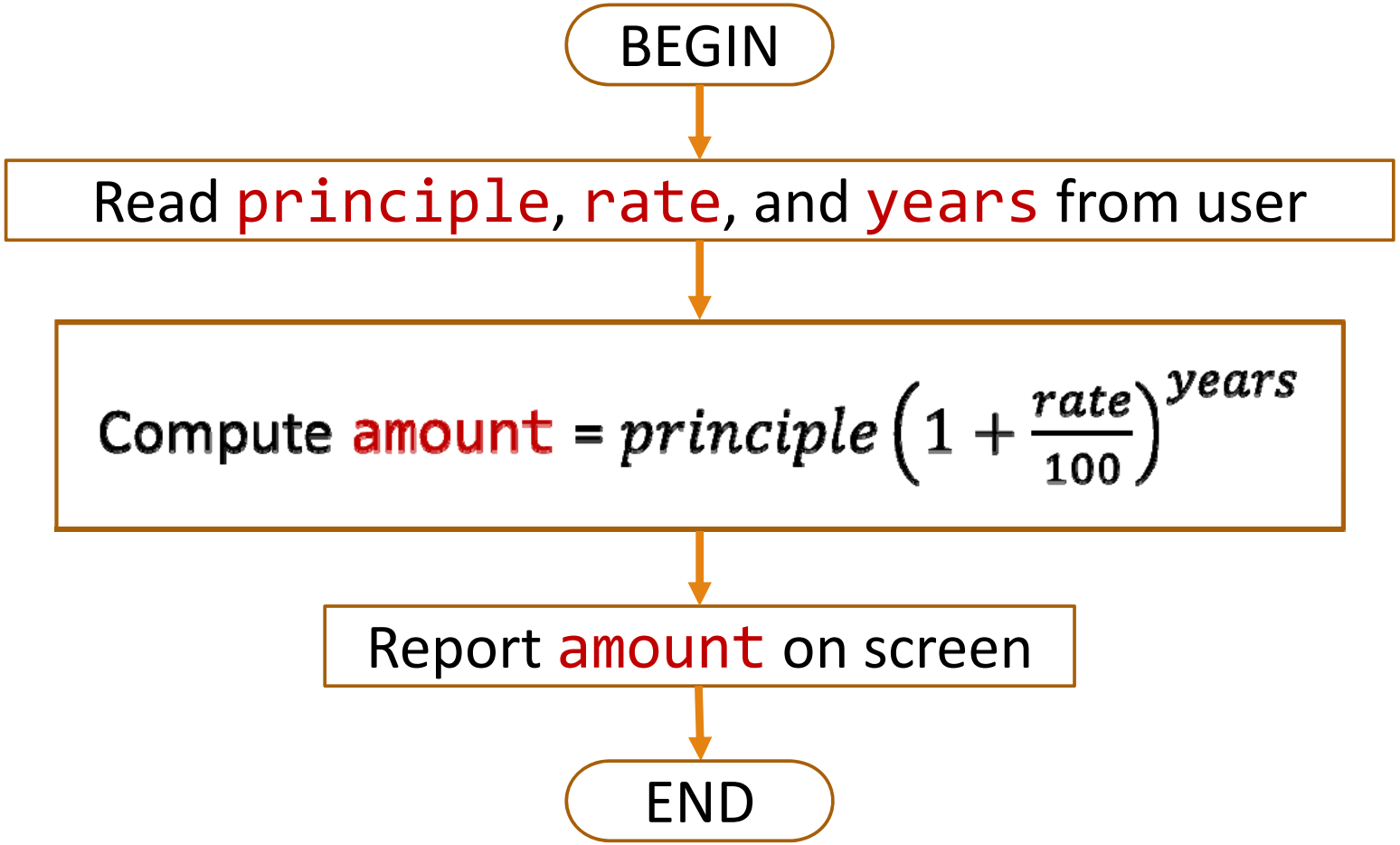
- Let us analyze the relationship among the amount in the account, principle (p), rate (r), and years (n)

Year	Amount	Rearranging
0	p	p
1	$p+\frac{pr}{100}$	$p\left(1+\frac{r}{100}\right)$
2	$p\left(1+\frac{r}{100}\right)+\frac{p\left(1+\frac{r}{100}\right)r}{100}$	$p\left(1+\frac{r}{100}\right)\left(1+\frac{r}{100}\right)=p\left(1+\frac{r}{100}\right)^2$
3	$p\left(1+\frac{r}{100}\right)^2+\frac{p\left(1+\frac{r}{100}\right)^2 r}{100}$	$p\left(1+\frac{r}{100}\right)^2\left(1+\frac{r}{100}\right)=p\left(1+\frac{r}{100}\right)^3$

- It follows that on nth year, the amount will be $p\left(1+\frac{r}{100}\right)^n$

Savings Account – Steps





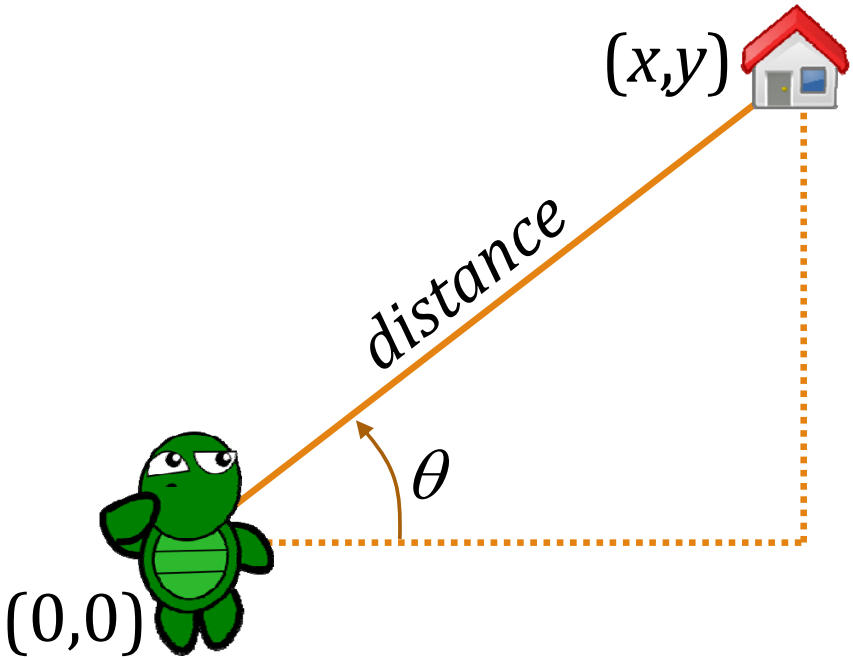
Savings Account – Program

```
using System;

class Program
{
    static void Main()
    {
        Console.Write("Principle (Baht): ");
        double principle = double.Parse(Console.ReadLine());
        Console.Write("Rate (% per year): ");
        double rate = double.Parse(Console.ReadLine());
        Console.Write("Time (years): ");
        int years = int.Parse(Console.ReadLine());
        double amount = principle * Math.Pow( 1 + (rate/100), years);
        Console.WriteLine("Amount: {0:f2}", amount);
    }
}
```

Task: Bring Turtle Home

- Our little robotic turtle is lost in the field. Please help guide him from his location at (0,0) to his home at (*x*,*y*)
- He cannot walk very fast, so we must head him to the right direction so that he can walk with the shortest distance
- Write a program to take the values *x* and *y*, then report the values of θ and *distance*



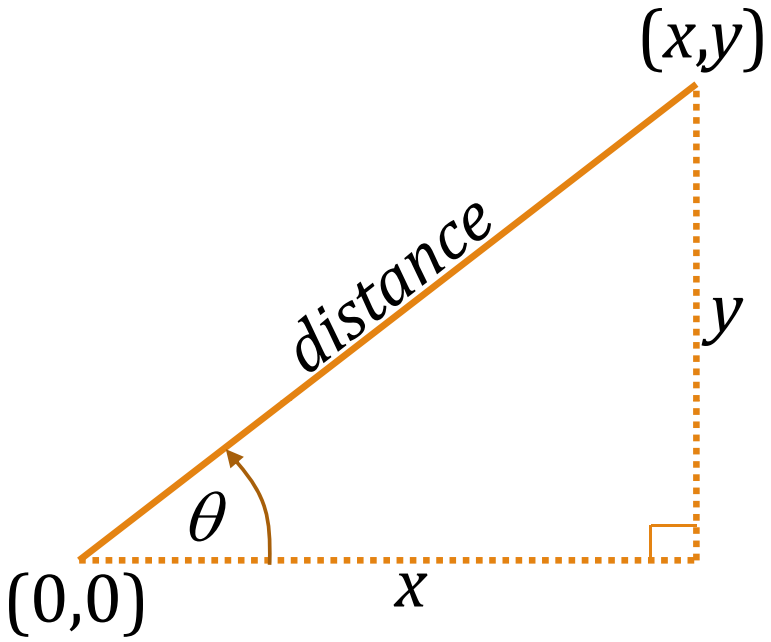
Bring Turtle Home - Ideas

- Again, we need to analyze the relationship among all the variables to solve the two unknowns
- From Pythagorean theorem

$$distance^2 = x^2 + y^2$$

- And from Trigonometry

$$\tan\theta = \frac{y}{x}$$



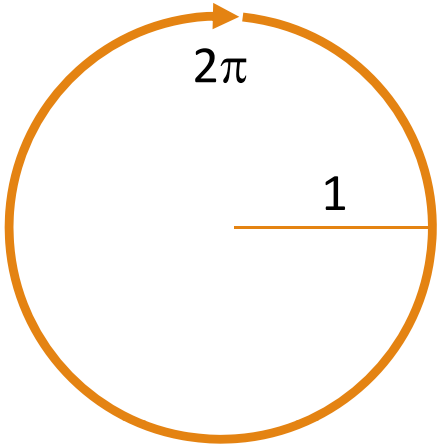
- Therefore,

$$distance = \sqrt{x^2 + y^2} \quad \text{and} \quad \theta = \arctan\frac{y}{x}$$

Caveats – Radians vs. Degrees

- In most programming languages, the unit of angles used by trigonometry functions is ***radians***, not degrees
- A full circle, 360 degrees, is 2π radians
- Therefore,

$$1 \text{ radian} = \frac{360}{2\pi} \text{ degrees}$$




```
public static double Atan(
        double d
)
```

Parameters

d

Type: System.Double
A number representing a tangent.

Return Value

Type: System.Double
An angle, θ, measured in radians, such that $-\pi/2 \leq \theta \leq \pi/2$.

Bring Turtle Home – Program

```
using System;

class Program
{
    static void Main()
    {
        Console.Write("Enter x: ");
        double x = double.Parse(Console.ReadLine());
        Console.Write("Enter y: ");
        double y = double.Parse(Console.ReadLine());

        double distance = Math.Sqrt( (x*x) + (y*y) );
        double heading = 360/(2*Math.PI) * Math.Atan(y/x);

        Console.WriteLine("Heading: {0:f2} degrees", heading);
        Console.WriteLine("Distance: {0:f2} units", distance);
    }
}
```

Conclusion

- C# provides many useful mathematical functions in its Math library
- Assigning a value to a variable of different type may require explicit type conversion syntax
  - Type casting
  - String parsing
- Most of the time spent in programming is analyzing the problem, not writing code

References

- C# string formatting syntax
  https://msdn.microsoft.com/en-us/library/txafckwd(v=vs.110).aspx
- C# Math Library
  https://msdn.microsoft.com/en-us/library/system.math(v=vs.110).aspx
- Casting and type conversions in C#
  https://msdn.microsoft.com/en-us/library/ms173105.aspx