# More Exercise: Methods

Problems for exercise and homework for the ["C# Fundamentals" course @ SoftUni](https://softuni.bg/trainings/3836/programming-fundamentals-with-csharp-september-2022)  
You can check your solutions in [Judge](https://judge.softuni.org/Contests/1287/Methods-More-Exercise)

## Data Types

Write a program that, depending on the first line of the input, reads an **int**, a **double** or a **string**.

* If the data type is int, multiply the number by 2.
* If the data type is real, multiply the number by 1.5 and format it to the second decimal point.
* If the data type is a string, surround the input with '$'.

Print the result on the console.

using System;

namespace \_01.\_Data\_Types

{

class Program

{

static void Main(string[] args)

{

string input = Console.ReadLine();

if (input == "int")

{

int value = int.Parse(Console.ReadLine());

dataTypes(value);

}

else if(input=="real")

{

double value = double.Parse(Console.ReadLine());

dataTypes(value);

}

else if(input=="string")

{

string value = Console.ReadLine();

dataTypes(value);

}

}

private static void dataTypes(int a)

{

Console.WriteLine(2\*a);

}

private static void dataTypes(double a)

{

Console.WriteLine($"{(1.5 \* a):F2}");

}

private static void dataTypes(string a)

{

Console.WriteLine($"${a}$");

}

}

}

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| int  5 | 10 |
| real  2 | 3.00 |
| string  hello | $hello$ |

### Hint

Try to solve the problem using only one method with different overloads.

## Center Point

You are given the coordinates of two points on a [Cartesian coordinate system](https://en.wikipedia.org/wiki/Cartesian_coordinate_system) - X1, Y1, X2, and Y2. **Create a method** that prints the point that is closest to the center of the coordinate system (0, 0) in the format (X, Y). If the points are at the same distance from the center, print only the first one.

using System;

namespace \_02.\_Center\_Point

{

class Program

{

static void Main(string[] args)

{

double X1 = double.Parse(Console.ReadLine());

double Y1 = double.Parse(Console.ReadLine());

double X2 = double.Parse(Console.ReadLine());

double Y2 = double.Parse(Console.ReadLine());

if( distanceToCentre(X1, Y1)< distanceToCentre(X2, Y2))

{ Console.WriteLine($"({X1}, {Y1})"); }

else if (distanceToCentre(X1, Y1) > distanceToCentre(X2, Y2))

{ Console.WriteLine($"({X2}, {Y2})"); }

else

{ Console.WriteLine($"({X1}, {Y1})"); }

}

private static double distanceToCentre(double a, double b)

{

double distance = Math.Sqrt(Math.Pow(Math.Abs(a), 2) + Math.Pow(Math.Abs(b), 2));

return distance;

}

}

}

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2  4  -1  2 | (-1, 2) |
| 3  -6  2  10 | (3, -6) |

## Longer Line

You are given the coordinates of four points in the 2D plane. The first and the second pair of points form two different lines. Print the longer line in the format "**(X1, Y1)(X2, Y2)**" starting with the point that is closer to the center of the coordinate system (0, 0). (You can reuse the method that you wrote for the previous problem.) If the lines are of equal length, print only the first one.

using System;

namespace Methods\_\_\_More\_Exercises

{

class Program

{

static void Main(string[] args)

{

double X1 = double.Parse(Console.ReadLine());

double Y1 = double.Parse(Console.ReadLine());

double X2 = double.Parse(Console.ReadLine());

double Y2 = double.Parse(Console.ReadLine());

double X3 = double.Parse(Console.ReadLine());

double Y3 = double.Parse(Console.ReadLine());

double X4 = double.Parse(Console.ReadLine());

double Y4 = double.Parse(Console.ReadLine());

if (length(X1, Y1, X2, Y2) < (length(X3, Y3, X4, Y4)))

{

if (distanceToCentre(X1, Y1) < distanceToCentre(X2, Y2))

{

Console.WriteLine($"({X1}, {Y1})({X2}, {Y2})");

}

else if (distanceToCentre(X1, Y1) > distanceToCentre(X2, Y2))

{

Console.WriteLine($"({X2}, {Y2})({X1}, {Y1})");

}

else

{ Console.WriteLine($"({X1}, {Y1})({X2}, {Y2})"); }

}

else if (length(X1, Y1, X2, Y2) > (length(X3, Y3, X4, Y4)))

{

if (distanceToCentre(X3, Y3) < distanceToCentre(X4, Y4))

{

Console.WriteLine($"({X3}, {Y3})({X4}, {Y4})");

}

else if (distanceToCentre(X3, Y3) > distanceToCentre(X4, Y4))

{

Console.WriteLine($"({X4}, {Y4})({X3}, {Y3})");

}

else

{ Console.WriteLine($"({X3}, {Y3})({X4}, {Y4})"); }

}

if (length(X1, Y1, X2, Y2) == (length(X3, Y3, X4, Y4)))

{

if (distanceToCentre(X1, Y1) < distanceToCentre(X2, Y2))

{

Console.WriteLine($"({X1}, {Y1})({X2}, {Y2})");

}

else if (distanceToCentre(X1, Y1) > distanceToCentre(X2, Y2))

{

Console.WriteLine($"({X2}, {Y2})({X1}, {Y1})");

}

else

{ Console.WriteLine($"({X1}, {Y1})({X2}, {Y2})"); }

}

}

private static double distanceToCentre(double a, double b)

{

double distance = Math.Sqrt(Math.Pow(Math.Abs(a), 2) + Math.Pow(Math.Abs(b), 2));

return distance;

}

private static double length(double a, double b, double c, double d)

{

double width = 0, height = 0;

if (a >= 0 && c >= 0)

{

if (a >= c)

{ width = a - c; }

else

{ width = c - a; }

}

else if (a <= 0 && c <= 0)

{

if (a >= c)

{ width = a - c; }

else

{ width = c - a; }

}

else if ((a >= 0 && c <= 0) || (a <= 0 && c >= 0))

{

width = Math.Abs(a) + Math.Abs(c);

}

if (b >= 0 && d >= 0)

{

if (b >= d)

{ height = b - d; }

else

{ height = d - b; }

}

else if (b <= 0 && d <= 0)

{

if (b >= d)

{ width = b - d; }

else

{ width = d - b; }

}

else if ((b >= 0 && d <= 0) || (b <= 0 && d >= 0))

{

width = Math.Abs(b) + Math.Abs(d);

}

double length = Math.Sqrt(Math.Pow(Math.Abs(width), 2) + Math.Pow(Math.Abs(height), 2));

return length;

}

}

}

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2  4  -1  2  -5  -5  4  -3 | (4, -3)(-5, -5) |
| 34  -3  5  9  -8  10  8  11 | (5, 9)(34, -3) |

## Tribonacci Sequence

In the **"Tribonacci" sequence**, every number is formed by the **sum of the previous 3** numbers.

You are given a number num. Write a program that printsnumnumbers from the Tribonacci sequence, on a single line, starting from 1. The input comes as a parameter named num. The value num will always be a positive integer.

using System;

namespace \_04.\_Tribonacci\_Sequence

{

class Program

{

static void Main(string[] args)

{

int number = int.Parse(Console.ReadLine());

tribonacciSequence(number);

}

static void tribonacciSequence(int number)

{

int[] arr = new int[number];

if (number == 0)

{ Console.WriteLine("0"); }

else if (number == 1)

{ Console.WriteLine("1"); }

else if (number == 2)

{ Console.WriteLine("1 1"); }

else

{

arr[0] = 1;

arr[1] = 1;

arr[2] = 2;

for (int i = 3; i < number; i++)

{

arr[i] = arr[i - 1] + arr[i - 2] + arr[i - 3];

}

for (int j = 0; j < number; j++)

{

Console.Write(arr[j] + " ");

}

}

}

}

}

### Examples

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| 4 | 1 1 2 4 |  | 8 | 1 1 2 4 7 13 24 44 |

## Multiplication Sign

You are given a number num1, num2 and num3. Write a program that finds if num1 \* num2 \* num3 (the product) is **negative**, **positive or zero**. Try to do this **WITHOUT** multiplying the 3 numbers.

using System;

namespace \_5.\_Multiplication\_Sign

{

class Program

{

static void Main(string[] args)

{

double num1 = double.Parse(Console.ReadLine());

double num2 = double.Parse(Console.ReadLine());

double num3= double.Parse(Console.ReadLine());

if(num1==0||num2==0||num3==0)

{ Console.WriteLine("zero"); }

else if((num1>0&&num2>0&&num3>0)||(num1>0&&num2<0&&num3<0)|| (num1 <0 &&num2> 0 && num3 < 0)|| (num1 <0 && num2 < 0 && num3> 0))

{ Console.WriteLine("positive"); }

else

{ Console.WriteLine("negative"); }

}

}

}

### Examples

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| 2  3  -1 | negative |  | 2  3  1 | positive |