# CMPT 360: Lab Assignment #1
# Complex Triangles

Brady Coles

last updated: January 3, 2018

## Contents

**1 Course Goals** **1**

**2 Problem Description** **1**

**3 Sample I/O** **2**

**4 Language Comparison** **2**

**5 Program Documentation** **2**
5.1 Errors and Messages . . . . . 2
5.2 Problem Solution . . . . . 3
5.3 Pseudocode . . . . . 3
5.4 C# Version Documentation . . . . . 3
5.5 Fortran Version Documentation . . . . . 4

**6 Program Listing** **4**
6.1 C# Listing . . . . . 4
6.2 Fortran Listing . . . . . 6

**7 Sample Output** **8**
7.1 C# Output . . . . . 8
7.2 Fortran Output . . . . . 8

## 1 Course Goals

This assignment fulfills the following goals:

- a group I language (C#)
- implemented on the Windows platform
- a group II language (Fortran)
- implemented on the Mac OSX platform

## 2 Problem Description

The programs detailed in this document take three points in the complex plane and calculates the side lengths and angles of the triangle those points determine.

# 3 Sample I/O

The input for these programs are three complex numbers. The output is the side lengths and angles of the triangles. For example, if the input numbers were:

```
Z1 = 3 + 2i,   Z2 = 6 + 2i,   Z3 = 3 + 6i
```

The output would be printed to the console:

```
Side Lengths
Z1Z2 = 3
Z2Z3 = 5
Z3Z1 = 4
Angles
Z1 = 90
Z2 = 53.1301024
Z3 = 36.8698976
```

# 4 Language Comparison

There are some important differences between C# and Fortran 95 that were relevant to this problem.

- In Fortran, complex numbers are an intrinsic type, with built in operators. C# requires either an additional library or for the programmer to create their own complex number type.
- C# has methods (functions), and subroutine functionality is achieved through a method with `void` return type. Fortran supports both subroutines and functions with slightly different syntax.
- C# methos return their value using a return statement. Fortran functions return the value of a variable of the same name as the function when the function ends.
- In C#, float literals are considered double precision unless otherwise specified, and functions for floating point values expect doubles. In Fortran, float literals are single precision unless otherwise specified, and functions for double precision floats have different names than regular floating point functions.
- C# fully supports OOP, while Fortran does not. Fortran modules are similar to static classes in C#.
- String formatting uses different syntax and operates in a different way. C# uses a format method, while Fortran uses a string literal.

# 5 Program Documentation

## 5.1 Errors and Messages

Both the C# and Fortran programs check for valid input, and if invalid input is found, they output a message.

`Not a valid triangle. Points must be distinct.`
One or more points are the same, therefore a triangle does not exist.

`Not a valid triangle. Points must not be collinear.`
The points all lie on a single line, therefore a triangle does not exists.

*Note*, due to floating point errors, this check sometimes validates invalid input. A suitable error margin (epsilon) was not determined, so invalid input may be accepted in extreme cases (where one point is much further away than the other points are to each other).

## 5.2 Problem Solution

The solution can be broken into the following steps.

1. Validating Input - The input is checked that no two points are equivalent, and that they are not collinear. Collinearity is determined by checking the area of the triangle, where an area of 0 implies collinearity. Area was calculated using the shoelace method (C#) or Herons formula (Fortran)
2. Side Lengths - Side lengths are given by taking the absolute value of the difference of the two (complex) endpoint of a side.
3. Angles - Angles were calculated using the law of cosines with the previously calculated side lengths.
4. Print Output

## 5.3 Pseudocode

This is a solution to the problem in psuedocode.

```
def triangle(z1, z2, z3) \\ where z1, z2, z3 are complex
    if any of z1,z2,z3 are equal or z1, z2, z3 are collinear then
        print warning and exit
    end if
    z1z2, z2z3, z3z1 = distance(z1, z2), distance(z2, z3), distance(z3, z1)
    angz1, angz2, angz3 = angle(z2z3, z1z2, z3z1), ang(z3z1, z1z2, z2z3), ang(z1z2, z2z3, z3z1)
    print sidelengths and angles
end triangle

def distance(a, b) \\ Where a, b are complex
    return |a - b|
end distance

def angle(opp, adj1, adj2) \\ sidelengths of triangle
    return acos((adj1^2 + adj2^2 - opp^2) / (2*adj1*adj2))
end angle
```

## 5.4 C# Version Documentation

The static class `ComplexTriangle`

| Public Methods | | |
|---|---|---|
| Name | Arguments | Description |
| `static Triangle` | `z1, z2, z3` of the type `ComplexNumber` representing vertices of triangle. | Prints out the side lengths and angles of the triangle determined by the input. If points are indistinct or collinear it prints a warning and ends. |

The struct `ComplexNumber`.
The struct does not define operators, it simply acts as a container for the two parts of a complex number.

| Constructors | | |
|---|---|---|
| Name | Arguments | Description |
| `ComplexNumber` | `real, imaginary` of the type `double` as the real and imaginary part of a complex number. | Creates a complex number with given values. |
| Public Fields | | |
| `double r` | Real part of the number | |
| `double i` | Imaginary part of the number | |

### 5.5 Fortran Version Documentation

Module written for Fortran 95, specifically the gfortran compiler.

Import the module `complex_triangle`.

| Public Subroutines | | |
|---|---|---|
| Name | Arguments | Description |
| `triangle` | `z1, z2, z3` double precision (kind=8) COMPLEX numbers representing vertices of triangle. | Prints out the side lengths and angles of the triangle determined by the input. If points are indistinct or collinear it prints a warning and ends. |

# 6 Program Listing

## 6.1 C# Listing

```
using System;

/// <summary>
/// Author: Brady Coles
/// This class allows the use of complex numbers (defined below) in the determining
/// of the sidelengths, area, and angles of a triangle on the Complex Plane.
/// </summary>
static class ComplexTriangle
{
    // For identification

    /// <summary>
    /// Takes three points on the complex plane, and determines the side lengths
    /// and angles of the triangle formed by them.
    /// </summary>
    /// <param name="z1">Point 1</param>
    /// <param name="z2">Point 2</param>
    /// <param name="z3">Point 3</param>
    public static void Triangle (ComplexNumber z1, ComplexNumber z2, ComplexNumber z3)
    {
        Console.WriteLine(string.Format("Triangle \n Z1: {0} \n Z2: {1} \n Z3: {2}",
            z1.ToString(), z2.ToString(), z3.ToString()));
        // Check Validity
        if (z1.Equals(z2) || z1.Equals(z3) || z2.Equals(z3))
            // Check for non-distinct numbers
        {
            Console.WriteLine("Not a valid triangle. Points must be distinct");
            return;
        }
        if (Area(z1, z2, z3) == 0)
            // Check for collinearity
        {
            Console.WriteLine("Not a valid triangle. Points must not be collinear.");
            return;
        }

        var z1z2 = Distance(z1, z2);
        var z2z3 = Distance(z2, z3);
```

```
39          var z3z1 = Distance(z3, z1);
40
41          var angZ1 = Angle(z2z3, z1z2, z3z1);
42          var angZ2 = Angle(z3z1, z1z2, z2z3);
43          var angZ3 = Angle(z1z2, z2z3, z3z1);
44
45          Console.WriteLine("Side Lengths:");
46          Console.WriteLine(string.Format(" Z1 Z2 : {0}", z1z2));
47          Console.WriteLine(string.Format(" Z2 Z3 : {0}", z2z3));
48          Console.WriteLine(string.Format(" Z3 Z1 : {0}", z3z1));
49          Console.WriteLine("Angles (degrees)");
50          Console.WriteLine(string.Format(" Z1 : {0}", angZ1));
51          Console.WriteLine(string.Format(" Z2 : {0}", angZ2));
52          Console.WriteLine(string.Format(" Z3 : {0}", angZ3));
53      }
54
55      /// <summary>
56      /// Determines the distance between two complex numbers
57      /// </summary>
58      /// <param name="z1">Number 1</param>
59      /// <param name="z2">Number 2</param>
60      /// <returns>Distance between numbers</returns>
61      private static double Distance(ComplexNumber z1, ComplexNumber z2)
62      {
63          ComplexNumber z = new ComplexNumber(z1.r - z2.r, z1.i - z2.i);
64          return Math.Pow((z.r * z.r) + (z.i * z.i), 0.5);
65      }
66
67      /// <summary>
68      /// Determines the interior angle opposite a side of a triangle.
69      /// </summary>
70      /// <param name="opposite">Side opposite the angle</param>
71      /// <param name="adjacent1">Adjacent side</param>
72      /// <param name="adjacent2">Another adjacent side</param>
73      /// <returns>The angle in degrees</returns>
74      private static double Angle(double opposite, double adjacent1, double adjacent2)
75      {
76          return Math.Acos(
77                  ((adjacent1 * adjacent1) + (adjacent2 * adjacent2) - (opposite * opposite))
78                  / (2 * adjacent1 * adjacent2)
79              )
80              * (360 / (2 * Math.PI));
81      }
82
83      /// <summary>
84      /// Determines the area of a triangle in the complex plane.
85      /// </summary>
86      /// <param name="z1"></param>
87      /// <param name="z2"></param>
88      /// <param name="z3"></param>
89      /// <returns>Area of the triangle</returns>
90      private static double Area (ComplexNumber z1, ComplexNumber z2, ComplexNumber z3)
91      {
92          // Uses shoelace formula to calculate area
```

```
        return (z1.r * (z2.i + z3.i) + z2.r * (z1.i + z3.i) + z3.r * (z1.i + z2.i)) / 2;
    }

}

/// <summary>
/// Simple complex number. Does not include operators.
/// </summary>
public struct ComplexNumber
{
    public double r;
    public double i;

    // Default constructor requires real and imaginary parts of the complex number
    public ComplexNumber(double real, double imaginary)
    {
        r = real;
        i = imaginary;
    }

    // Returns a nicely formatted string in the form a+bi
    public override string ToString()
    {
        return string.Format("{0:0.#####}{1:+0.#####;-0.#####}i", r, i);
    }
}
```

## 6.2 Fortran Listing

```
! Author: Brady Coles
! Module that can determine area, side lengths, and angles of a triangle in
! the Complex Plane.
MODULE complex_triangle
    IMPLICIT NONE
    private
    public :: triangle ! triangle is the only public subroutine
CONTAINS
    ! calculates the distance between complex points
    ! arguments are points in the complex plane
    FUNCTION distance(z1, z2)
        COMPLEX(KIND=8), INTENT(IN) :: z1, z2
        REAL(kind=8) distance
        distance = CDABS(z1 - z2)
    END FUNCTION distance

    ! calculates an angle of a triangle
    ! arguments are sidelengths.
    FUNCTION angle (opp, adj1, adj2)
        REAL(kind=8), INTENT(IN) :: opp, adj1, adj2
        REAL(kind =8) angle
        REAL, PARAMETER :: PI = 4.0 * ATAN(1.0)
        angle = DACOS((adj1**2 + adj2**2 - opp**2) / (2 * adj1 * adj2))
        angle = angle * (360.0 / (2 * PI))
    END FUNCTION angle
```

```
26
27      ! calculates the area of a triangle with Herons formula
28      ! a, b, c are sidelengths.
29      FUNCTION area(a, b, c)
30          REAL(kind=8), INTENT(IN) :: a, b, c
31          REAL(kind=8) :: area, s
32          s = (a + b + c) / 2.0d+0
33          area = DSQRT(s*(s-a)*(s-b)*(s-c))
34      END FUNCTION area
35
36      ! prints out info about triangle
37      SUBROUTINE triangle(z1, z2, z3)
38          COMPLEX(KIND=8), INTENT(IN) :: z1, z2, z3 ! points of a triangle
39          REAL(kind=8) :: z1z2, z2z3, z3z1, angz1, angz2, angz3
40          CHARACTER(20) :: fmt ='(F0.4,SP,F0.4,"i")'
41
42          ! Check if points are distinct
43          IF (z1 == z2 .or. z2 == z3 .or. z3 == z1) THEN
44              PRINT *, 'Not a valid triangle. Points must be distinct.'
45              RETURN
46          END IF
47
48          z1z2 = distance(z1, z2)
49          z2z3 = distance(z2, z3)
50          z3z1 = distance(z3, z1)
51
52          ! Check if points are collinear by getting area of triangle
53          IF (area(z1z2, z2z3, z3z1) == 0) THEN      ss ! Fails in extreme cases
54              PRINT *, 'Not a valid triangle. Points must not be collinear.'
55              RETURN
56          END IF
57
58          PRINT *, 'TRIANGLE'
59          PRINT '("  ",A,": ",F0.4,SP,F0.4,"i")', 'Z1', z1, 'Z2', z2, 'Z3', z3
60          PRINT *, 'SIDE LENGTHS'
61          PRINT *,  ' Z1 Z2 : ', z1z2
62          PRINT *,  ' Z2 Z3 : ', z2z3
63          PRINT *,  ' Z3 Z1 : ', z3z1
64          PRINT *, 'ANGLES (degrees)'
65          PRINT *, ' Z1 : ', angle(z2z3, z1z2, z3z1)
66          PRINT *, ' Z2 : ', angle(z3z1, z1z2, z2z3)
67          PRINT *, ' Z3 : ', angle(z1z2, z2z3, z3z1)
68      END SUBROUTINE triangle
69  END MODULE complex_triangle
70
71  ! A test program to try the module
72  PROGRAM test
73  USE complex_triangle
74  COMPLEX(kind=8) :: z1, z2, z3
75  z1 = COMPLEX(-2.1d+0, -0.0001d+0)
76  z2 = COMPLEX(-2098.7d+0, 15.1d+0)
77  z3 = COMPLEX(30d+0, 0.0001d+0)
78  CALL triangle(z1, z2, z3)
79  END PROGRAM
```

# 7 Sample Output

## 7.1 C# Output

```
Triangle
 Z1: 1.2+15i
 Z2: 0+0i
 Z3: 4+0i
Side Lengths:
 Z1 Z2 : 15.0479234447813
 Z2 Z3 : 4
 Z3 Z1 : 15.25909564817
Angles (degrees)
 Z1 : 15.1474446784618
 Z2 : 85.4260787400992
 Z3 : 79.4264765814391
```

## 7.2 Fortran Output

```
 TRIANGLE
  Z1: -2.1000-.0001i
  Z2: -2098.7000+15.1000i
  Z3: +30.0000+.0001i
 SIDE LENGTHS
  Z1 Z2 :    2096.6543761478688
  Z2 Z3 :    2128.7535547780089
  Z3 Z1 :    32.100000000623055
 ANGLES (degrees)
  Z1 :    179.58698551679822
  Z2 :    6.2277442378044527E-003
  Z3 :   0.40677685572767547
```