





using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace hw1.Bowden

{

public class Complex

{

//Default constructor. Sets values to zero.

public Complex()

{

//set properties to zero because the class was created without defining the values

Real = 0;

Imaginary = 0;

}//end Default constructor

//Constructor whcih sets the initial values of the Complex class

public Complex(double real, double imag)

{

//set the values of the properties

Real = real;

Imaginary = imag;

}//end constructor

//read-only property that holds the value of the real number

public double Real { get; private set;}

//read-only property that hold the value of the imaginary number

public double Imaginary { get; private set;}

public static Complex operator +(Complex c1, Complex c2)

{

//Calculate the value of each part of the Complex Number

double real = c1.Real + c2.Real;

double imaginary = c1.Imaginary + c2.Imaginary;

//Return the operator value

return new Complex(real, imaginary);

}//end operator +

public static Complex operator - (Complex c1, Complex c2)

{

//Calculate the value of each part of the Complex Number

double real = c1.Real - c2.Real;

double imaginary = c1.Imaginary - c2.Imaginary;

//Return the operator value

return new Complex(real, imaginary);

}//end operator -

public static Complex operator \* (Complex c1, Complex c2)

{

//Calculate the value of each part of the Complex Number

double real = (c1.Real \* c2.Real) - (c1.Imaginary \* c2.Imaginary);

double imaginary = (c1.Real \* c2.Imaginary) + (c1.Imaginary \* c2.Real);

//Return the operator value

return new Complex(real, imaginary);

}//end operator \*

public static Complex operator / (Complex c1, Complex c2)

{

//Calculate the value of each part of the Complex Number

double real = (c1.Real \* c2.Real + c1.Imaginary \* c2.Imaginary) / (Math.Pow(c2.Real,c1.Imaginary) + Math.Pow(c2.Imaginary, c1.Imaginary));

double imaginary = (c1.Imaginary \* c2.Real - c1.Real \* c2.Imaginary) / (Math.Pow(c2.Real, c1.Imaginary) + Math.Pow(c2.Imaginary, c1.Imaginary));

//Return the operator value

return new Complex(real, imaginary);

}//end operator /

public override string ToString()

{

//local variables

string RetVal = "0";

//return the string value in the simplest terms so that a 0i is not displayed to the user

if(Imaginary == 0)

{

//display the real value only

RetVal = Real.ToString();

}

else

{

if(Real == 0)

{

//display the imaginary value only

RetVal = Imaginary.ToString() + "i";

}

else

{

//display both parts

RetVal = string.Format("({0} {1} {2}i)", Real, (Imaginary < 0 ? "-" : "+"), Math.Abs(Imaginary));

}

}

//return our function value

return RetVal;

}//end method ToString

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace hw1.Bowden

{

class Program

{

//Main entry point

static void Main(string[] args)

{

//change the title to the window

Console.Title = "Homework 1: Courtney Bowden";

//call the method that does most of the work

Run();

}//end Main

//this is where most of the work happens

static void Run()

{

//primary application loop

while(true)

{

//create whitespace for the user

Console.WriteLine();

//ask the user to enter a command

string UserInput = GetUserInput("Enter Command: ");

//check to make sure the command is not blank

if (string.IsNullOrWhiteSpace(UserInput)) continue;

//check for quit command

if (UserInput == "QUIT") QuitCommand();

//find out what the command was, and process it

if(UserInput == "ADD"

|| UserInput == "SUB"

|| UserInput == "MULT"

|| UserInput == "DIV")

{

//try block to catch errors

try

{

//get the terms

double real1 = GetTerms("Enter real term: ");

double imaginary1 = GetTerms("Enter imaginary term: ");

double real2 = GetTerms("Enter real term: ");

double imaginary2 = GetTerms("Enter imaginary term: ");

//calculate the result

Complex c1 = new Complex(real1, imaginary1);

Complex c2 = new Complex(real2, imaginary2);

//declare local variable Output for use inside the switch block

string Output;

//use a switch block to match the command to the output string

switch (UserInput)

{

case "ADD":

Output = string.Format("{0} + {1} = {2}", c1, c2, c1 + c2);

if (Output.Contains("Infinity"))

{

//the result is too large, tell the user to choice a smaller number

Console.WriteLine("The final result is too large. Please choose smaller input terms. ");

}

else

{

//display the output to the user

Print(Output);

}

break;

case "SUB":

Output = string.Format("{0} - {1} = {2}", c1, c2, c1 - c2);

if (Output.Contains("Infinity"))

{

//the result is too large, tell the user to choice a smaller number

Console.WriteLine("The final result is too large. Please choose smaller input terms. ");

}

else

{

//display the output to the user

Print(Output);

}

break;

case "MULT":

Output = string.Format("{0} \* {1} = {2}", c1, c2, c1 \* c2);

if (Output.Contains("Infinity"))

{

//the result is too large, tell the user to choice a smaller number

Console.WriteLine("The final result is too large. Please choose smaller input terms. ");

}

else

{

//display the output to the user

Print(Output);

}

break;

case "DIV":

if (real2 == 0 && imaginary2 == 0)

{

//notify the user that they are attempthing to divide by zero

Console.WriteLine("You cannot divide by zero. Please enter " +

"either a non-zero real number or a non-zero " +

"imaginary number for the second terms.");

continue;

}

else

{

Print(string.Format("{0} / {1} = {2}", c1, c2, c1 / c2));

}

break;

}

}

catch(Exception ex)

{

//something went wrong, write the error message to console and exit gracefully

QuitCommand(ex.Message);

}

}

else

{

//the user asked for something unrecognized, or they typed help. display the help information

HelpCommand();

continue;

}

}

}//end Run

//generic function to ask user for their input

static string GetUserInput(string InputMessage = "")

{

//write the input message to the console

Console.Write(InputMessage);

//read the input from user and returning it

return Console.ReadLine().ToUpper();

}//end GetUserInput

//this function gets the real and imaginary terms from the use

static double GetTerms(string InputMessage)

{

//this function must return a double value, so keep trying until the user gives a valid double

//local variables

double RetVal = 0.0;

//start our main loop

while(true)

{

//write the input message to the console

Console.Write(InputMessage);

//read back the result as a variant

string Result = Console.ReadLine();

//check to see if user a double value

if (double.TryParse(Result, out RetVal))

{

//return the function value

return RetVal;

}

else

{

//ask user to put in a valid number

Console.WriteLine("Please enter a valid numberic value.");

}

}

}//end GetTerms

//this function tells the user we are exiting and then it exits the program

static void QuitCommand(string ErrorMessage = "")

{

//if an error message exists then write the error message to the console

Console.WriteLine(ErrorMessage);

//tell the user goodbye

Console.WriteLine("Goodbye!!!");

//put the system to sleep for 1 second

System.Threading.Thread.Sleep(1000);

//exit

System.Environment.Exit(0);

}//end QuitCommand

//this method prints user help

static void HelpCommand()

{

//print each line of the help command

Console.WriteLine("Valid commands for this program");

Console.WriteLine("help - prints this help");

Console.WriteLine("add - Adds two complex numbers");

Console.WriteLine("sub - Subtracts two complex numbers");

Console.WriteLine("div - Divides two complex numbers");

Console.WriteLine("mult - Multiplies two complex numbers");

Console.WriteLine("quit - Exit program");

}//end HelpCommand

//prints the complex number result

static void Print(string Message)

{

//show the result and the message

Console.WriteLine();

Console.WriteLine("Result:");

Console.WriteLine(Message);

}//end Print

}

}