**Задание 10.1. Комплексные вычисления** (*Reflection, dynamic*)

1. Если в каталоге программы присутствует сборка из домашнего задания 2.1 (комплексные числа), то, используя Reflection и эту сборку, необходимо:
   - Не используя dynamic, вычислить значение выражения:

     $$z = \frac{(x+y)^2}{27}, \text{где } x = 2 + 3i; y = 14(\cos\left(\frac{\pi}{4}\right) + i\sin\left(\frac{\pi}{4}\right))$$

     **Т.е. одно комплексное число задано в обычном виде, а второе в тригонометрическом*
   - Распечатать результат в обычном виде и в виде модуля и аргумента (тригонометрическом виде)
   - Используя dynamic, вычислить

     $$c = \frac{(a^2 + b^2)^2}{3b}, \text{где } a = 4 + i; b = 2(\cos\left(\frac{\pi}{3}\right) + i\sin\left(\frac{\pi}{3}\right))$$

     **Т.е. одно комплексное число задано в обычном виде, а второе в тригонометрическом*
   - Распечатать результат в обычном виде и в виде модуля и аргумента (тригонометрическом виде)
2. Если в каталоге программы отсутствует сборка из домашнего задания 2.1, то:
   - Необходимо сообщить об этом пользователю
   - Используя стандартную .NET структуру System.Numerics.Complex, без Reflection, вычислить:

     $$d = 34 + e^f, \text{где } e = 1 + 2i; f = 3(\cos\left(\frac{\pi}{8}\right) + i\sin\left(\frac{\pi}{8}\right))$$

     **При этом Complex уже обладает всеми необходимыми методами для работы с комплексными числами, например, FromPolarCoordinates(), Pow() и т.д.*
   - Вывести результат пользователю

*Для созданий комплексных чисел в тригонометрическом виде используйте статический метод, создающий комплексное число по модулю и аргументу (был в домашнем задании)*

`Обратите внимание, ваше приложение не должно содержать ссылок на сборку из домашнего задания 2.1.`

**Задание 10.2.* (необязательное). Работа с делегатами через Reflection. Анализ загруженных сборок** (*Reflection, dynamic*)

1. Распечатайте список загруженных сборок при работе вашего приложения
2. Напишите и вызовите метод, который динамически через Reflection вызывает из отдельной сборки из задания 7 метод для поиска обратного значения функции sin(x)=0.5 с точностью 0,0001. Распечатайте результат.
   Средствами работы с файловой системой предусмотрите возможность отсутствия такой сборки в каталоге приложения (без необходимости использования try … catch блока). При этом программа должна продолжать работать, просто пропустив этот шаг (и шаг 3) и выдав пользователю вразумительное сообщение.
3. Напишите отдельный не статический метод, делающий такой же вызов, что и в п.2 только используя dynamic (метод поиска обратного числа должен быть не статическим). Распечатайте результат.

4. Еще раз распечатайте список загруженных сборок при работе вашей программы. Обратите внимание на состав загруженных сборок (на появление новых сборок из-за использования dynamic).

Обратите внимание, ваше приложение не должно содержать ссылок на сборку с методом поиска обратного значения.

**Примечания:**

1. Для получения загруженных в данный момент сборок используйте метод *GetAssemblies()* у текущего домена приложения:
   ```
   Assembly[] assemblies = AppDomain.CurrentDomain.GetAssemblies();
   ```
2. Для динамического создания делегата произвольного тапа используйте статический метод CreateDelegate() класса Delegate
   ```
   Delegate del = Delegate.CreateDelegate(type, method);
   ```
3. Для динамического вызова делегата используйте метод DynamicInvoke() у произвольного делегата
   ```
   object result = del.DynamicInvoke(2, 4, 3);
   ```
4. Для проверки существует ли файл на диске используйте статический метод Exists() класса System.IO.File
   ```
   bool isFileExists = File.Exists("Имя файла");
   ```