**Евразийский национальный университет имени Л.Н.Гумилева**

**Отчет**

**по Лабораторной работе №6**

|  |  |
| --- | --- |
|  | Выполнила:  Абдакимова Лейла ИС-32  Аипова Орынбасар ИС-32  Жайсанова Дания ИС-32  Жусупекова Аида ИС-32  Рахимжанова Анеля ИС-32  Проверила: Жукабаева Т.К. |

**г. Нур-Султан**

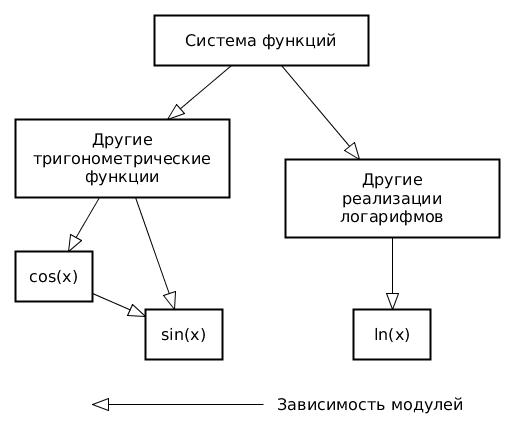
**2022**

**Лабораторная работа №****6**

Провести интеграционное тестирование программы, осуществляющей вычисление системы функций (в соответствии с вариантом).

**Правила выполнения работы:**

1. Все составляющие систему функции (как тригонометрические, так и логарифмические) должны быть выражены через базовые (тригонометрическая зависит от варианта; логарифмическая - натуральный логарифм).
2. Структура приложения, тестируемого в рамках лабораторной работы, должна выглядеть следующим образом (пример приведён для базовой тригонометрической функции sin(x)):



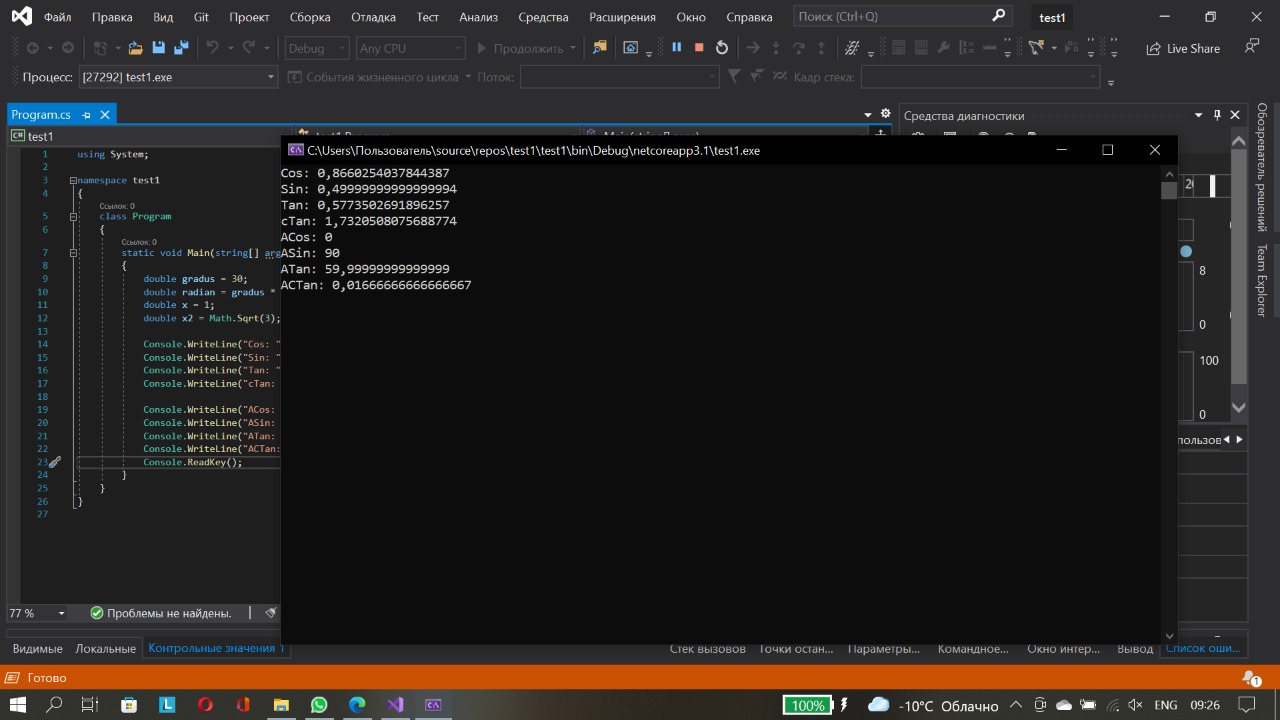
1. Обе "базовые" функции (в примере выше - sin(x) и ln(x)) должны быть реализованы при помощи разложения в ряд с задаваемой погрешностью. Использовать тригонометрические / логарифмические преобразования для упрощения функций ЗАПРЕЩЕНО.
2. Для КАЖДОГО модуля должны быть реализованы табличные заглушки. При этом, необходимо найти область допустимых значений функций, и, при необходимости, определить взаимозависимые точки в модулях.
3. Разработанное приложение должно позволять выводить значения, выдаваемое любым модулем системы, в сsv файл вида «X, Результаты модуля (X)», позволяющее произвольно менять шаг наращивания Х. Разделитель в файле csv можно использовать произвольный.

**Порядок выполнения работы:**

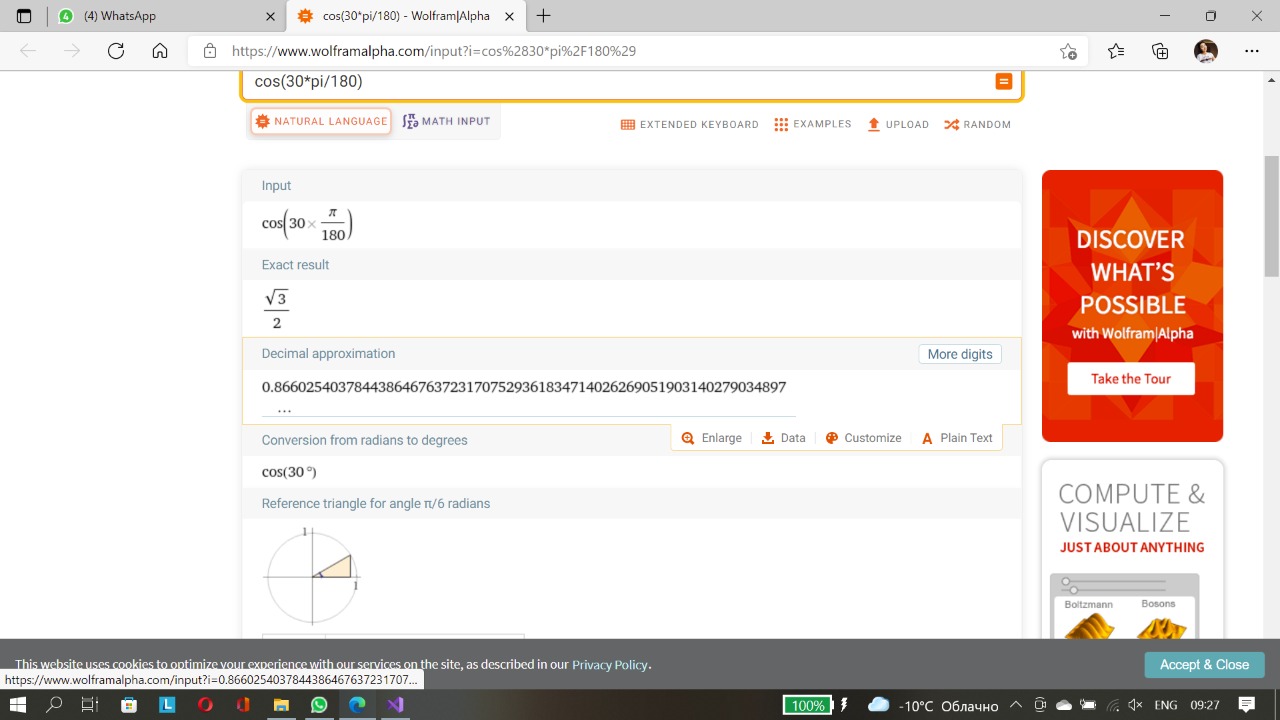
1. Разработать приложение, руководствуясь приведёнными выше правилами.
2. С помощью JUNIT4 разработать тестовое покрытие системы функций, проведя анализ эквивалентности и учитывая особенности системы функций. Для анализа особенностей системы функций и составляющих ее частей можно использовать сайт <https://www.wolframalpha.com/>.
3. Собрать приложение, состоящее из заглушек. Провести интеграцию приложения по 1 модулю, с обоснованием стратегии интеграции, проведением интеграционных тестов и контролем тестового покрытия системы функций.

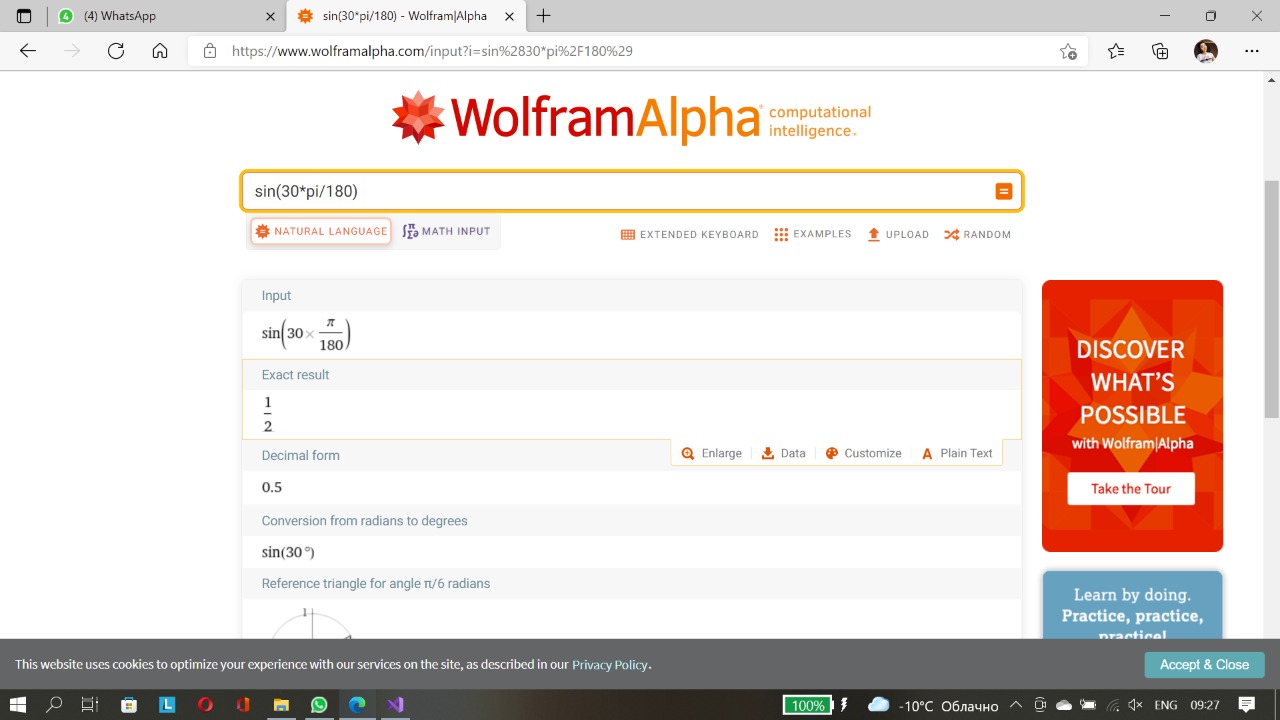
**Отчет по работе:**

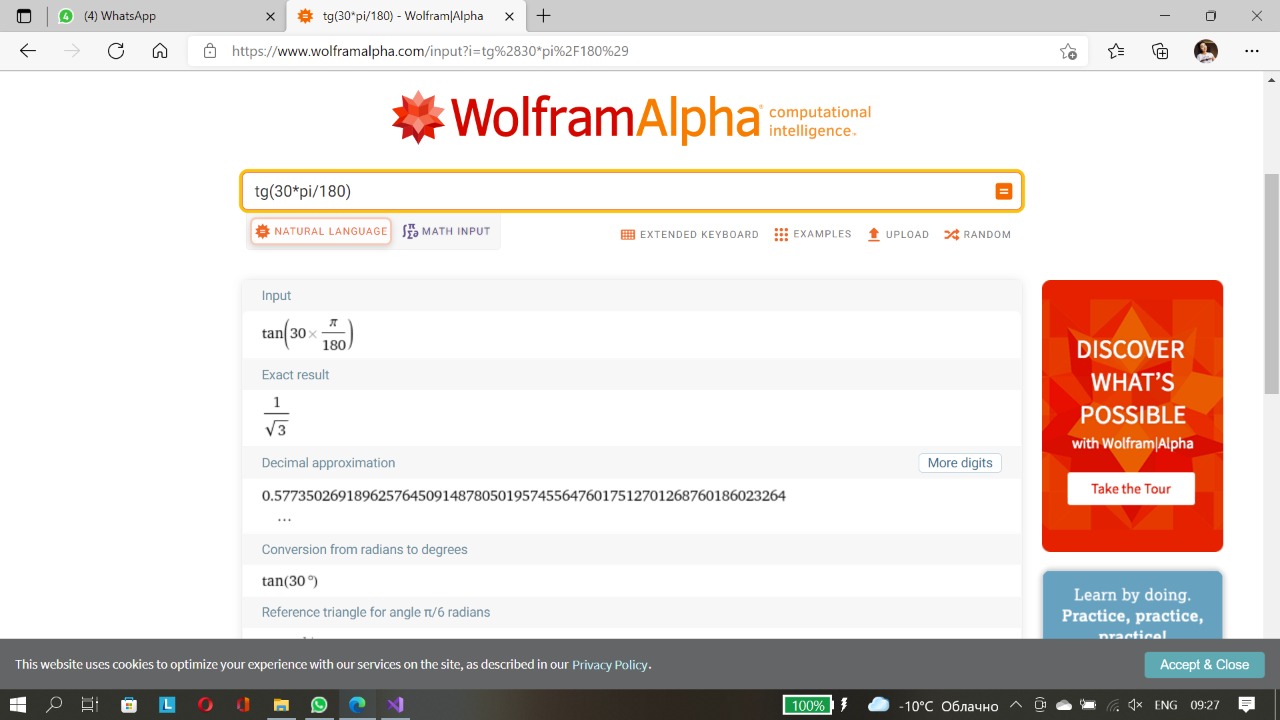
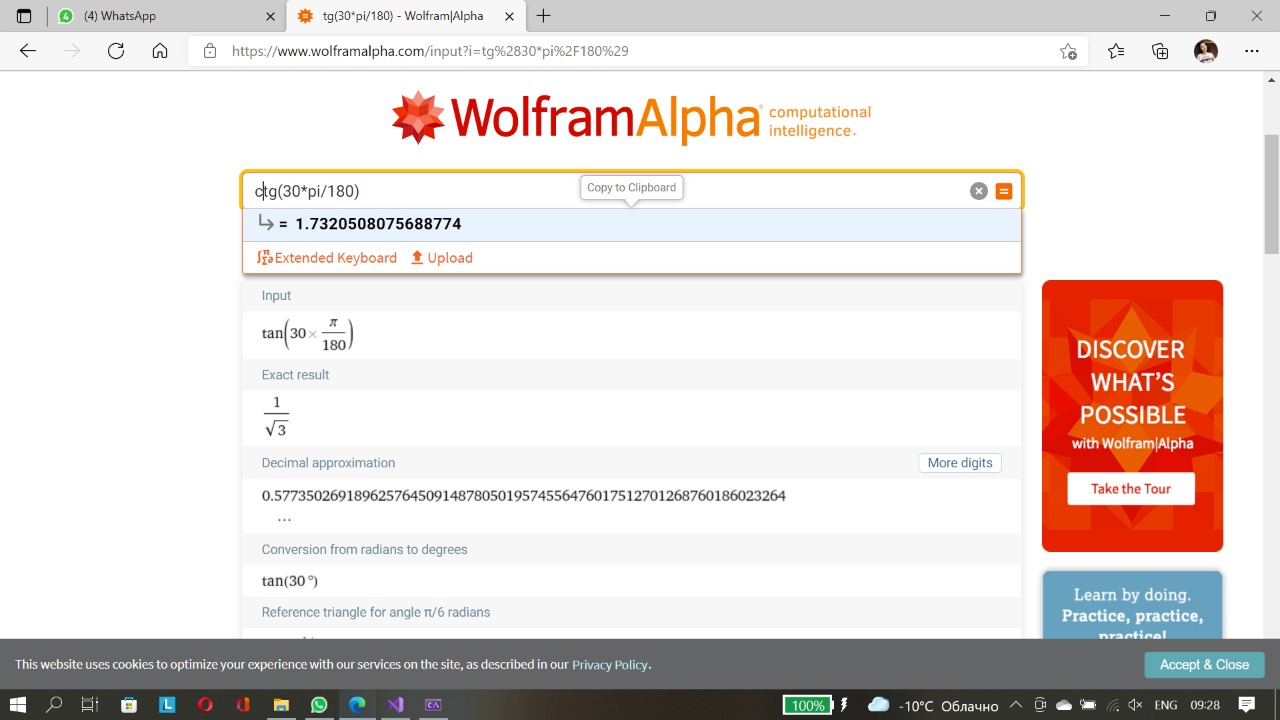
По результатам кода вышло:

****

По результатам тестирования вышло:

****

****

****

**Вопросы к защите лабораторной работы:**

**1. Цели и задачи интеграционного тестирования. Расположение фазы интеграционного тестирования в последовательности тестов; предшествующие и последующие виды тестирования ПО.**

Целью нашего тестирования является выявление багов при взаимодействии между этими программными модулями и в первую очередь направлен на проверку обмена данными между этими самими модулями.

Интеграционное тестирование / integration testing — фокусируется на взаимодействии между компонентами / модулями, системами.

Интеграционное тестирование фокусируется на взаимодействии между компонентами / модулями / подсистемами / системами.

Выделяют 2 подтипа:

Компонентное интеграционное тестирование — проверяет связи между компонентами. Может быть автоматизировано.

Системное интеграционное тестирование — проверяет связи между подсистемами / системами. Не всегда можно автоматизировать, так как часто интеграция происходит с внешним сервисом, к которому мы не имеем доступа.

**2. Алгоритм интеграционного тестирования.**

Алгоритм интеграционного тестирования:

Подготовка план интеграционных тестов

Разработка тестовых сценариев.

Выполнение тестовых сценариев и фиксирование багов.

Отслеживание и повторное тестирование дефектов.

Повторять шаги 3 и 4 до успешного завершения интеграции.

**3. Концепции и подходы, используемые при реализации интеграционного тестирования.**

Графики, построенные csv-выгрузкам, полученным в процессе интеграции приложения. Описание тестового покрытия с обоснованием его выбора. Разработанное приложение должно позволять выводить значения, выдаваемое любым модулем системы, в сsv файл вида «X, Результаты модуля», позволяющее произвольно менять шаг наращивания Х. Разделитель в файле csv можно использовать произвольный.

**4. Программные продукты, используемые для реализации интеграционного тестирования. Использование JUnit для интеграционных тестов.**

С помощью C# разработать тестовое покрытие системы функций, проведя анализ эквивалентности и учитывая особенности системы функций. Для анализа особенностей системы функций и составляющих ее частей можно использовать сайт https://www.wolframalpha.com/.

**5. Автоматизация интеграционных тестов. ПО, используемое для автоматизации интеграционного тестирования.**

Интеграционное тестирование позволяет проверить, не нарушается ли существующий код внесенными изменениями. Непрерывной интеграцией (CI) в DevOps называется процесс, который обеспечивает автоматическую компиляцию всей системы при каждом изменении базы кода, например, при отправке любого запроса на вытягивание в репозиторий Git. В следующем списке приведены распространенные примеры интеграционных тестов.

* + Средства статического анализа кода, такие как Lint и Format.
  + Выполните команду terraform validate, чтобы проверить синтаксис файлов конфигурации.
  + Выполните команду terraform plan, чтобы проверить, работает ли конфигурация должным образом.