***Международный образовательный комплекс «Гармония «Школа №97», город Ижевск***

*Автор: Васильев Георгий Михайлович, школа №97, 10В класс*

*Научные руководители:*

*Кузнецова Екатерина Андреевна,   
учитель математики  
Ившин Александр Николаевич,   
учитель информатики*

***Тема:***

***Визуализированная алгоритмизация тригонометрических уравнений с отбором корней на С#***

***Ижевск***

***2021 год***

Оглавление

[Введение 2](#_Toc66352958)

[Зарождение идеи проекта 4](#_Toc66352959)

[Тригонометрия 4](#_Toc66352960)

[Особенности тригонометрических уравнений 5](#_Toc66352961)

[Классификация уравнений 6](#_Toc66352962)

[Выбранный язык программирования 7](#_Toc66352963)

[Что из себя представляет С# 8](#_Toc66352964)

[Этапы реализации проекта 9](#_Toc66352965)

[Оболочка приложения 9](#_Toc66352966)

[Алгоритм решения 11](#_Toc66352967)

[Реализация отбора корней 13](#_Toc66352968)

[Итог работы 14](#_Toc66352969)

[Рабочая программа 14](#_Toc66352970)

[Справка к использованию 14](#_Toc66352971)

[Необработанные случаи 14](#_Toc66352972)

[Использованная литература 15](#_Toc66352973)

[Приложение 16](#_Toc66352974)

*«Самый важный аспект разработки программного обеспечения — ясно понять, что вы пытаетесь создать»*

*Бьерн Страуструп, создатель С++*

# **Введение**

С каждым годом информационные технологии всё сильнее проникают в нашу жизнь. Если в 2006 году количество принятых вакансий IT сферы составило всего 41 тысячу, то 2021 это число увеличилось до невообразимых 240 тысяч. То есть за какие-то 15 лет информационные технологии развились настолько, что необходимость квалифицированных специалистов растёт в геометрической прогрессии.

Конечно же, информационный прогресс не может не повлиять на жизнь населения многих стран, в том числе и на детей. Дети нового поколения рождаются в веке информационных технологий, со всех сторон они окружены различными устройствами, которые так или иначе упрощают нашу жизнь. Но как это сказывается на самом поколении? Кто-то скажет, что оно потеряно, и вырастут лишь одни лоботрясы и лентяи, однако на практике подрастающие дети сильно заинтересованы окружающими их технологиями.

В качестве доказательства можно привести рейтинг результатов Единого Государственного Экзамена по предмету «Информатика и ИКТ», напрямую связанного с информационными технологиями. В 2010 его выбрали 62652 человека (≈7% от общего количества сдающих), а в 2020 уже 83610 (≈14% от общего количества сдающих). Несмотря на невысокое численное повышение сдающих, процентное соотношения возросло в целых 2 раза. Объясняется это понижением общего количества подрастающего поколения в 2020 году, из-за чего экзамен в этом году сдавало на 33% меньше выпускников.

Ещё 20-30 лет назад каждый школьник мечтал о собственной ЭВМ, способной в миг решить за них любую задачу из алгебры или геометрии. С массовым внедрением калькуляторов, эти мечты частично осуществились, ведь сейчас каждый школьник может не только ждать и надеяться, что такое устройство появится, но и может запрограммировать его сам. Программирование преподаётся во многих школах России, существует множество курсов и возможностей самому осуществлять собственные мечты в реальность, это и заманивает столько новых людей в IT сферу.

С другой стороны, такие приложения, как например, PhotoMath, способное вычислять примеры, решать уравнения и неравенства практически любой сложности, уже созданы и многие считают, что незачем изобретать уже изобретённый велосипед. Но не стоит забывать, что приложения не всесильны, и даже команда лучших разработчиков и математиков могут упустить некоторые типы заданий, которые не могут быть решены иным приложением.

При использовании PhotoMath, я столкнулся с проблемой отсутствия возможности задавать числовой промежуток при решении тригонометрических уравнений, что не позволяло быстро и точно проверить собственные ответы.

Поэтому мне пришла идея написать собственный «калькулятор», который помог бы студентам и школьникам с решением и проверкой этих уравнений.

**Проблема**: можно ли запрограммировать решение тригонометрических уравнений с возможностью отбора корней, и на каком языке программирования это будет лучше всего реализовать

**Объект**: программирование и один из разделов математики – тригонометрия

**Предмет исследования**: тригонометрические уравнения из ЕГЭ

**Цель**: написать программу для решения максимально возможного количества тригонометрических уравнений с возможностью отбора корней

**Задачи**:   
- Проанализировать существующие виды тригонометрических уравнений из ЕГЭ за последние 10 лет  
- Провести сравнительный анализ подходящих языков программирования  
- Создать оконную оболочку для записи уравнения  
- Написать универсальный код для решения тригонометрических уравнений

**Методы**: теоретический анализ и синтез, классификация и обобщение материала, сравнение наиболее подходящих языков программирования

# **Зарождение идеи проекта**

При подготовке к Единому Государственному Заданию по предмету профильной алгебры попадается множество различных заданий, однако конкретно тригонометрические уравнения, которые присутствуют как в первой части (9 задание КИМ 2020 года), так и во второй (13 задание), является одним из основных заданий профильной математики. По статистике, целых 35% сдающих профильную математику справляются с 13 заданием. Учитывая тот факт, что это задание из части С, то есть задание с развёрнутым ответом. Это наивысшее значение среди иных подобных заданий. Задания такого типа есть абсолютно в каждом КИМе, что делает подготовку к ним универсальной. Учитывая это, меня поразило, как задания по одной, казалось бы, несложной теме можно получить целых 3 первичных балла, поэтому я принялся целенаправленно изучать эту тему. Изучив материалы за последние 10 лет, нашёл закономерность: 9 задание составляют числовые преобразования, направленные в основном на умение использовать базовые свойства корней, степеней, переменных, логарифмов и, собственно, тригонометрических функций. Задание под номером 13 на 70% составлено исключительно из тригонометрических уравнений повышенной сложности, для решения которых необходимо лишь уметь применять тригонометрические формулы.

## **Тригонометрия**

Термин произошёл от древнегреческого τρίγωνον «треугольник» и μετρέω «измеряю», то есть *измерение треугольников* и появился в 1595 году в эпоху Позднего Возрождения. Тригонометрия является важнейшим разделом математики, который изучает свойства и функции треугольников. К примеру техника триангуляции из раздела тригонометрии, позволяет рассчитать расстояние между недалёкими звёздами в космосе, контролировать системы навигации спутников и конструировать здания с оптимальным отношением размеров на основе системы треугольников.

Основы тригонометрии даются как отношение сторон в прямоугольном треугольнике в 9 классе, как раздел геометрии, дальнейшее изучение продолжается только в 10 или 11 классе, в зависимости от учебника алгебры, где начинается прямой курс изучения тригонометрических уравнений. Курс рассчитан на полгода, за время которого даются все основные понятия и тригонометрические преобразования, вроде формул двойного угла и суммы/разности тригонометрических функций. И только в последний момент вводится понятие отбора корней на заданном промежутке, которое необходимо для решения тригонометрического уравнения из второй части под буквой «Б».

## **Особенности тригонометрических уравнений**

Главной особенностью тригонометрических уравнений, относительно линейных и квадратичных, являлось то, что вводилась дополнительная переменная k (или n), показывающая периодичность пересечения функции и функции y=0. То есть теперь каждое уравнение имело бесконечное количество корней с определенным промежутком, который и надо было искать.

Отбор корней в свою очередь был введён для того, чтобы ограничить промежуток корней для точного их вычисления, а не в общем виде, как требуется написать ответ под буквой «А» задания №13

К тому же не стоит забывать, что график синусоиды и косинусоиды имеет значения только от -1 до 1 включительно, а значит если уравнение вида , где , то оно не будет иметь решений. Такой же «подвох» есть и в тангенсоидных уравнениях, в которых хоть область допустимых значений и не имеет границ (то есть в аналогичном уравнении, корни имеются при любых t), однако при разложении , мы получаем , следовательно быть нулём не может, а значит вводится условие: , что очень важно при решении уравнений. То же самое касается и , где также не может равняться нулю.

В школьный курс изучения тригонометрии также входят уравнения вида   
, где это любая обратная от тригонометрической функция, то есть arcsin/arctg и др. Эти функции возвращают угол по значению соответственной им функции. Обратные тригонометрические функции необходимы для записи решения уравнения в общем виде.

Но не стоит забывать, что обратные тригонометрические функции кардинально отличаются от самих тригонометрических функций. Они не имеют периодичности и существуют только на отрезке [-π; π] (arcsin/arccos) или на (-π; π) (arctg/arccctg). К тому же, уравнения такого вида в заданиях ЕГЭ не встречаются, именно поэтому, в рамках данной работы, уравнения вида не рассматривались при решении поставленной цели.

## **Классификация уравнений**

Изучив тригонометрические уравнения из заданий экзамена, пришёл к выводу, что они все в процессе своего решения используют тригонометрические преобразования и после этого принимают один из следующих видов (иногда представляют собой произведение нескольких таких уравнений).

В результате проведённого исследования у меня получилась следующая классификации:

**Рис. 1**

# **Выбранный язык программирования**

Собираясь написать проектно-исследовательскую работу по программированию, я твёрдо решил, что в этот раз программа будет именно визуализированная, а не консольная, коей была работа «Алгоритмическое решение финансовых задач»[[1]](#footnote-1) за 2019 год. Консольные программы хоть и просты в исполнении, внешний вид и удобство использования таковых оставляет желать лучшего. Поэтому мне предстояло с нуля изучать хотя бы базовую визуализацию в программирования.

К моменту написания работы я знал 2 языка, это интерпретируемый Python и компилируемый C++, компилируемые языки, в отличие интерпретируемых работают быстрее и не требуют переработки кода в программу каждый раз при запуске. В обоих из них есть возможность визуализации приложения (OpenGL в С++ и TKinter в Python), однако была одна очень крупная проблема: вручную прописывать все параметры объекта (размеры, цвет, положение в окне) было бы колоссально долго, ведь для полноценного калькулятора со всеми тригонометрическими функциями требуется немалое количество различных объектов. К примеру, в моей программе 46 объектов, и весь фрагмент кода, отвечающий за параметры кнопок занимает 713 строк.

Для решения моей проблемы требовался так называемый «визуализированный язык программирования», предназначенный напрямую для работы с объектами. Из популярнейших это или Java, или С#.

C# как, по мнению многих, попытка Microsoft создать собственный Java.  
По синтаксису эти языки программирования очень схожи. Но в C# есть преимущества на мой взгляд, делегаты, события, свойства. Но, к сожалению, стек технологий у Java шире и больше, по сравнению с C#. Простому разработчику, наверное, все равно, но разработчику, который в будущем будет искать работу... Просмотрев вакансии я понял, что от C# джуниора требуют меньше, чем от Java джуниора.  
На википедии есть статья сравнение C# и Java, и там есть очень хорошая цитата.  
Кик Рэдек из Microsoft считает С# более сложным языком, чем Java. По его мнению, «язык Java был построен таким образом, чтобы уберечь разработчика от стрельбы себе в ногу» (англ. «Java was built to keep a developer from shooting himself in the foot»), а «С# был построен так, чтобы дать разработчику пистолет, но оставить его на предохранителе» (англ. «C# was built to give the developer a gun but leave the safety turned on»).  
На Кворо 1 из разработчиков писал, что Java более многословен, а C# более точен и менее многословен.  
В целом, C# мне нравится больше, чем Java. Он является ответвлением от С++, поэтому даже большинство команд, операций и типов переменных абсолютно идентичны. С C# я начал, C# изучал, много практиковался. Java более многословен, и все таки это другой язык программирования.

## **Что из себя представляет С#**

Как уже было сказано выше, С# (C Sharp) - визуализированный язык, к тому же в отличие от Python, является компилируемым. То есть программный код переводится в машинный единожды и преобразуется в приложение расширением формата .exe, который в последствии можно использовать на любой другой ЭВМ без программного кода. В этом его преимущество. Тот же Java, который также подходил по возможности визуализации, является не полностью компилируемым, так как переводит программный код в машинный не сразу, а через так называемый промежуточный байт-код для JVM (Java Virtual Machine), что замедляет время выполнения программы.

Сам по себе язык C# был разработан на рубеже второго и третьего тысячелетий под личным руководством инженера Microsoft, Андерса Хейлсберга, которого и называют «отцом» этого языка. Создавался язык ради разработки Microsoft .NET Framework. Это особая платформа, на которой полностью построена Windows, все окна и оболочка операционной системы.

Установив это расширение в компилятор (в моём случае Visual Studio Community 2019), можно создать полноценное окно Windows (Windows Form) с возможностью растягивания, скрытия, развёртывания на весь экран и закрытия.

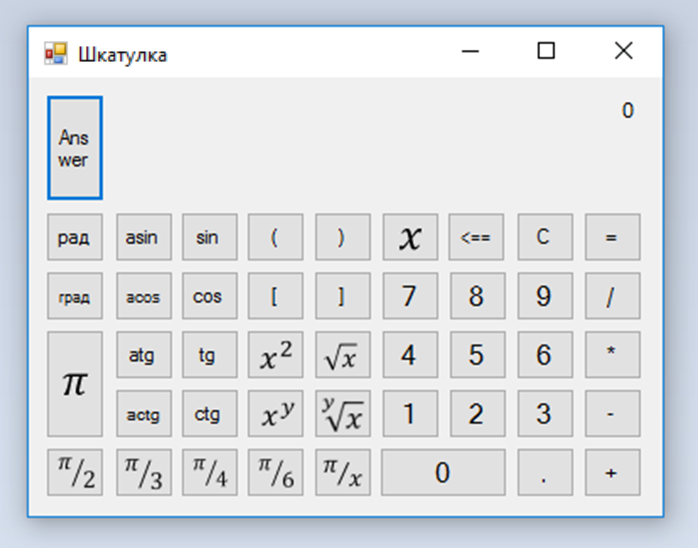
Также несравненным преимуществом визуализированного языка С# перед С++ является упрощённая визуализация. Как уже было сказано выше, в OpenGL С++ для создания единственной кнопки необходимо множество параметров, в С# же все они создаются в отдельном скрипте проекта «.Designer». Программисту лишь остаётся как в игре собрать «пазл» из кнопок, лейблов и текстбоксов, которым можно изменять цвет в удобном меню, менять размеры и положение перетягиванием, в общем всё сделано для упрощения работы программисту.

# **Этапы реализации проекта**

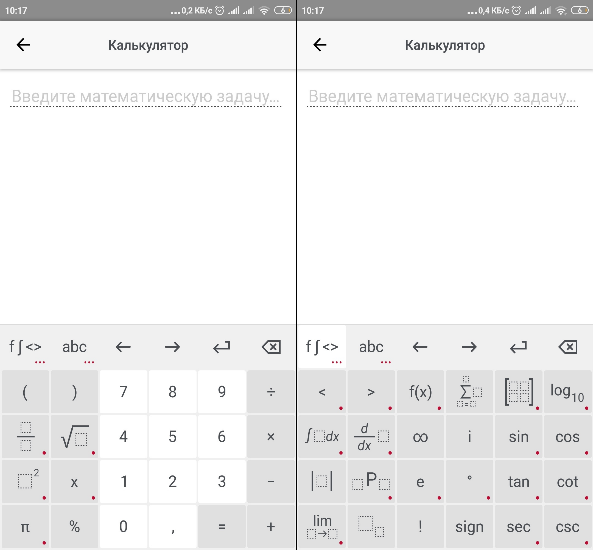
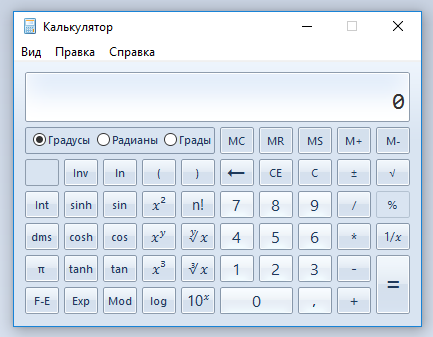
## **Оболочка приложения**

**I**. Первоначальной задачей было решить, какой сделать оболочку программы. По воле случая мне уже предоставлялась возможность писать игру на Unity с помощью скриптинга на С#, однако эта идея мне не понравилась тем, что делать из обычного калькулятора «игру» было неэффективно с точки зрения выделяемой памяти и времени. Тогда-то и пригодилось расширение .NET, позволяющее быстро и экономно создать Windows Form.

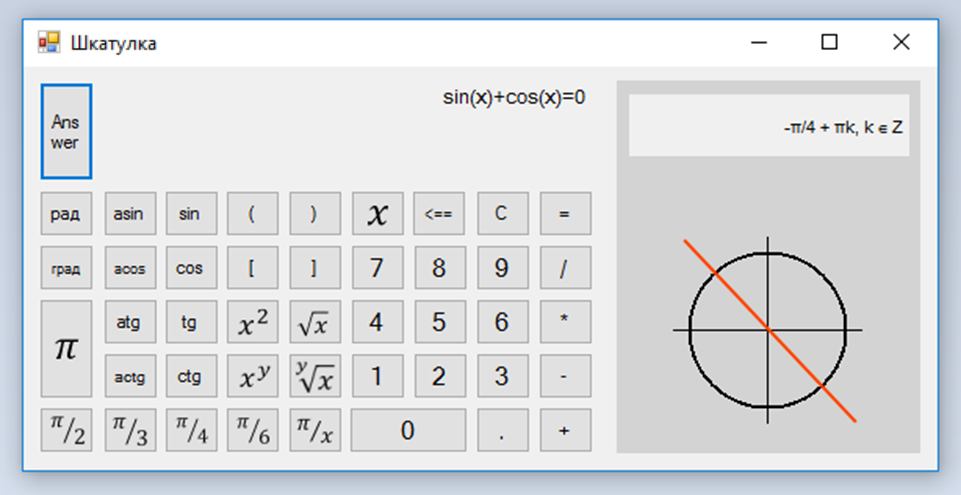
**II**. Вторым шагом решить, как же всё-таки будет выглядеть оболочка. Выбор решался между интерфейсами стандартного инженерного калькулятора Windows и PhotoMath. В первом было слишком много лишнего для программы (логарифмы, экспонента и модуль), во втором же возникала сложность реализации с «плавающей клавиатурой», где за одной кнопкой может содержаться дополнительное поле с вводимыми символами. Поэтому решено было создать оболочку на основе калькулятора Windows с небольшими видоизменениями. На рисунке 2.1 представлен получившийся прототип, обладающий всем необходимым для записи тригонометрических уравнений и выполненный на основе двух других оболочек, представленных ниже, рисунок 2.2 и 2.3.



**Рис. 2.1** Созданный мной прототип

 **Рис. 2.2** Интерфейс PhotoMath **Рис. 2.3** Инженерный калькулятор Windows

В стандартный калькулятор была добавлена возможность добавления переменной для решения именно уравнений, ведь первоначально калькулятор используется для прямых вычислений. Соответственно понадобилась и отдельная кнопка для вывода ответа на уравнение (кнопка Answer, обведена синим). Ответ же выводится в специально выдвигающейся части приложения, которая появляется только при выводе ответа. Рисунок 3.



**Рис. 3**

**III**. С созданием окна со всеми кнопками необходимо было реализовать возможность записи уравнения. Первоначально это казалось простейшей задачей, однако с каждым тестом программы выяснились всё новые и новые случаи ошибочной записи уравнения. К примеру, при записи каждой части уравнения возникает необходимость проверки правильности расставления скобок, количества запятых в одном числе не может превышать одно и так далее. Таким образом, возникла идея автоматизировать ввод уравнения, для того чтобы исключить возникновения случайных ошибок, можно сказать описок при вводе. Полный список всех «автоматических исправлений» для записи уравнение см. в приложение 2.

С этой системой «автоматических исправлений» при записи уравнения практически не может возникнуть ошибки, которая могла бы повлиять на получение ответа.

**IV**. Получив необходимую строку с уравнением, приступаем к его решению. Алгоритм решения не использует возможности нейронной сети, поэтому на ввод принимаются только полностью приведённые уравнения, где одна и та же тригонометрическая функция не встречается дважды, то есть все коэффициенты уже вычислены. Но благодаря такому решению на вычисление ответа необходимо не более одной десятой секунды.

## **Алгоритм решения**

Рассмотрим, как работает моя программа на примере уравнения из ЕГЭ[[2]](#footnote-2), задание под номером 13

a) Решите уравнение

б) Найдите все его корни, принадлежащие отрезку

Разберём решение этого уравнения.

Далее решение разделяется на письменный вид (тот, каким решал бы ученик на экзамене) и вывод самой программы, способной решить задание на данном этапе.

|  |  |
| --- | --- |
| A)  Б) Выполним отбор корней с помощью тригонометрической окружности.  -5  Серым обозначены границы отбора корней Чёрным – корни уравнения на окружности | Из-за невозможности решения произведения скобок, необходимо записывать уравнения по очереди для получения ответа. |

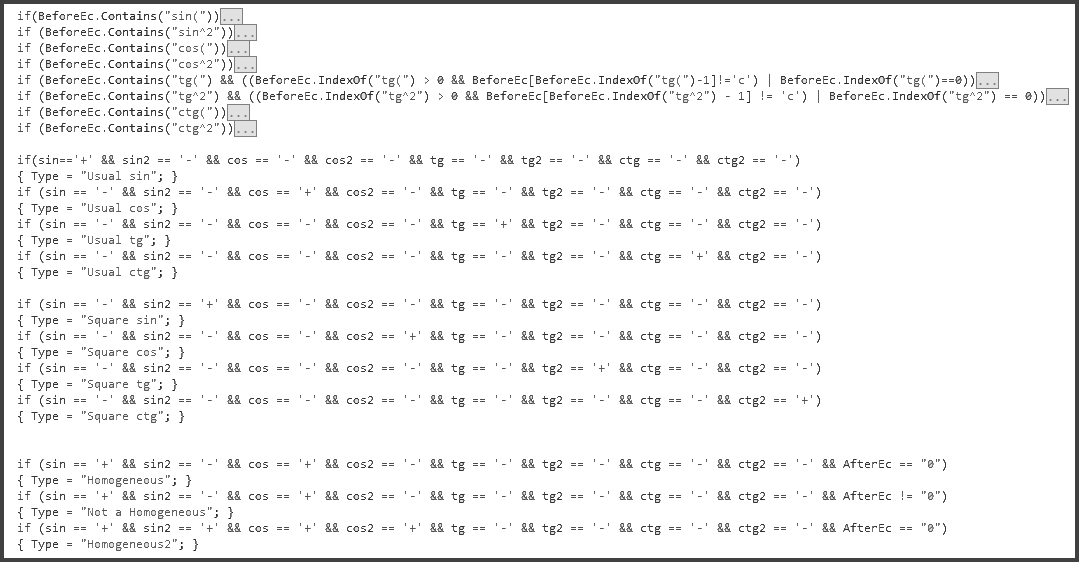
Ответ: А)

Б)

**I**. Первоначально требовалось определить тип записанного уравнения.

Реализовано это через проверку наличия определённых тригонометрических функций в выражении.

Для каждого вида тригонометрического уравнения составлен шаблон уравнения, описывающий вид этого уравнения. Для определения вида уравнения «примеряем» шаблон на введенное выражения с помощью условного оператора. Такая условная конструкция не содержит циклов, что делает проверку классификации очень быстрой и эффективной. На рисунке 4 представлен фрагмент кода, отвечающий за программную классификацию уравнений.



**Рис. 4** Фрагмент кода

**II**. Определив тип уравнения, программе осталось лишь выбрать уникальное решение для этого вида. На этом этапе пришлось полностью заносить таблицу в вывод ответов, иначе, если решение существует, будет выведена формула в общем виде. После решения сразу же удлиняется окно, и справа высвечивается ответ.

**III**. В случае, если существуют корни на заданном промежутке, то они выведутся после решения в общем виде в формате «решение + nπ», чтобы было понятнее, через какой период корни чередуются и сколько их всего.

## **Реализация отбора корней**

**I**. Весь ввод и вывод информации до сих пор выполнялся через объект С# label, текстовое поле, в котором напрямую писать невозможно, это и мешало в реализации финальной функции. К счастью в C# есть объект TextBox, который пользователь может напрямую изменять. Он и был использован в программе

**II**. Вторая сложность заключалась в том, что через клавиатуру невозможно ввести значение π, как неотъемлемой части тригонометрии и промежутков уравнений, эта проблема решена через возможность написания греческого символа двумя латинскими «pi», которые впоследствии программа преобразует в числовое значение.   
 **III**. Получив множество решений заданного уравнения, программа сравнивает его через коэффициент k и последовательно выводит все решения заданного промежутка через запятую.

# **Итог работы**

## **Рабочая программа**

В результате проектно-исследовательской работы была написана полноценная программа на языке С#, способная решать приведённые уравнения из ЕГЭ, представленных в классификации. Более того программа не только показывает множество решений уравнения на числовой окружности, но выполняет отбор корней на заданном промежутке. Цель проектной работы выполнена. Примеры работы программы см. Приложение 1.

Как уже было сказано на странице 5 «Особенности тригонометрических уравнений», программа не работает с обратными тригонометрическими функциями, однако для полноты оболочки и возможности дальнейшего развития проекта, возможности записи и тех уравнений были добавлены в калькулятор. Были вложены все силы, дабы исключить любые неполадки при работе с программой, поэтому составлена «справка к использованию», дабы ответ был точным и приложение не выдало исключение.

В дальнейшем планируется сделать возможным алгоритмизированное решение любых уравнений из самих заданий ЕГЭ с использованием нейронной самообучающейся сети, что позволит программе проводить все преобразования по вычислению коэффициентов и использование тригонометрических тождеств для упрощения уравнения. Это сделает программу абсолютно универсальной и возможной к комфортному использованию. В дополнение к этому планируется перенести приложение на мобильные платформы (Android и IOS), как помощь старшеклассникам с проверкой решения тригонометрических уравнений в школе. Если идея будет воплощена в жизнь, то с помощью рекламы появится и возможность монетизации продукта.

## **Справка к использованию**

-Написано должно быть обязательно уравнение, то есть и до, и после знака «равно».

-Все тригонометрические функции должны быть написаны только до знака «равно».

-Чтобы в поле для промежутка записать π, необходимо английскими символами написать «pi».

## **Необработанные случаи**

-Программа решает только полностью приведённые уравнения

-Уравнения, содержащие произведение слагаемых

-Уравнения, имеющие иной, кроме (х) аргумент

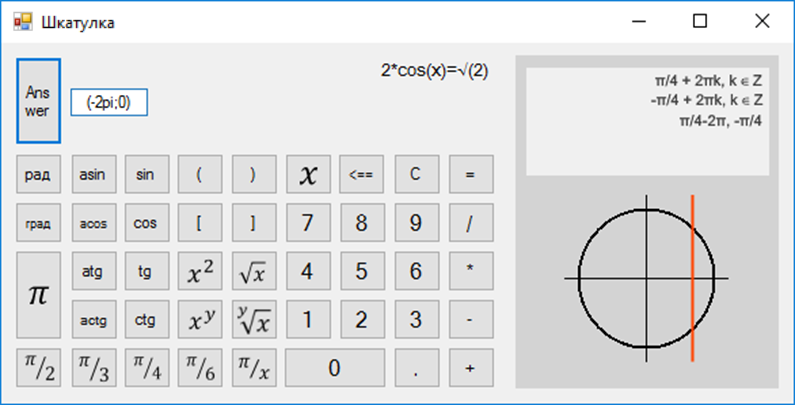
# **Использованная литература**

1. Бонд Д.Г. Unity и C# Геймдев от идеи до реализации – СПб, 2019.
2. Бхаргава А. Грокаем алгоритмы. Иллюстрированное пособие для программистов и любопытствующих. – СПб, 2017.
3. Васильева Н.Л. «Математика. В помощь выпускнику школы и абитуриенту» – СПб, 2019.
4. Круглински Д. и др. Программирование на Microsoft Visual C++ 6.0 для профессионалов – СПб, 2001.
5. Мордкович А.Г. Алгебра и начала математического анализа. 10 класс. В 2 ч. Ч. 1. Учебник для учащихся общеобразовательных учреждений (профильный уровень) – М, 2009
6. Письменный Д.Т Готовимся к экзамену по математике: математика для старшеклассников – М, 2008.
7. Сканави М.И. Сборник задач по математике для поступающих во втузы – М, 2013.
8. Федеральный институт педагогических измерений: [Электронный ресурс]. URL: <https://fipi.ru/>. (Дата обращения: 06.03.2021).
9. ЕГЭ-2021: задания, ответы, решения. Обучающая система Дмитрия Гущина: [Электронный ресурс]. URL: <https://math-ege.sdamgia.ru/>. (Дата обращения: 04.03.2021).
10. Ларин Александр Александрович. Математика. Репетитор: [Электронный ресурс]. URL: <http://alexlarin.net/>. (Дата обращения: 04.03.2021).
11. Wikipedia. Свободная энциклопедия: [Электронный ресурс]. URL: <https://ru.wikipedia.org>. (Дата обращения: 01.03.2021).

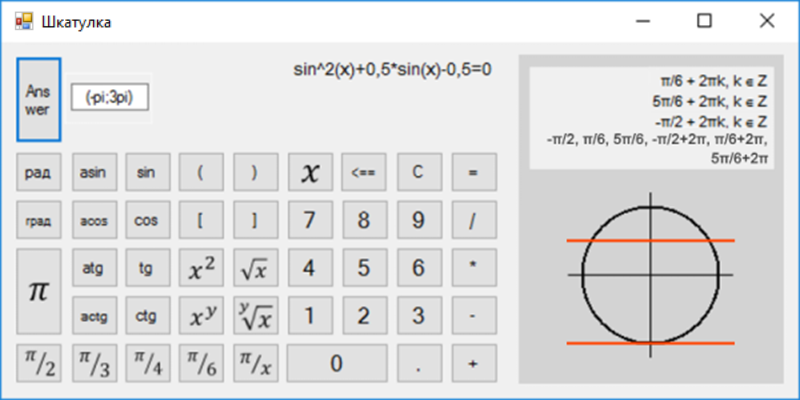
# **Приложение**

Приложение 1

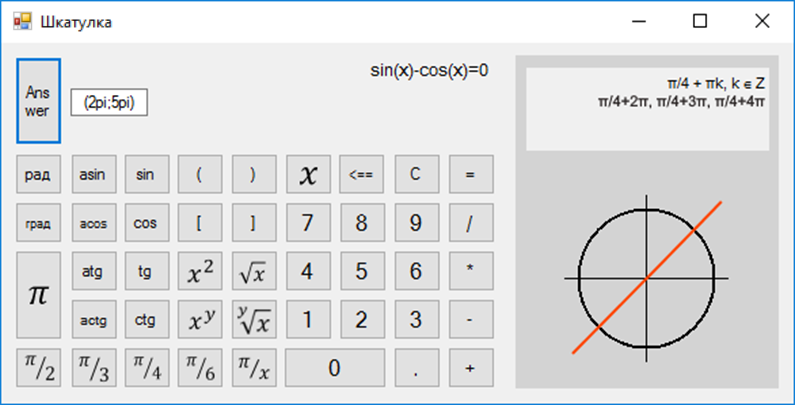
* 1. Пример решения простейшего косинусоидного уравнения по классификации



* 1. Пример решения квадратичного уравнения синуса.



* 1. Пример решения однородного уравнения первой степени.



Приложение 2

|  |
| --- |
| Запрограммирована невозможность поставить 2 десятичные запятые в одном числе |
| При написании запятой без целой части, за целую часть берётся 0 |
| Все коэффициенты перед х/тригонометрическими функциями и скобками автоматически записываются через знак умножения |
| Запрограммирована невозможность поставить 2 арифметических знака друг за другом |
| Если тригонометрическая функция первой степени, скобка, ограничивающая аргумент функции подставится автоматически |
| При использовании квадратного корня, скобка откроется автоматически |
| Разработан функционал кнопок «<=» и «С» |
| При нажатии знака «=» или «Answer» все незакрытые скобки будут закрыты |

1. Опубликовано в сборнике работ конференции Юность науке и технике 2020 по адресу: <http://www.izhdvorec.ru/meropriyatiya/npk-yunost-nauke-i-tekhnike/1738-uvazhaemye-uchastniki-respublikanskoj-npk-yunost-nauke-i-tekhnike> [↑](#footnote-ref-1)
2. Задание взято из демонстрационного варианта ЕГЭ 2021 с сайта   
   https://math-ege.sdamgia.ru [↑](#footnote-ref-2)