

Изображение выглядит как текст, книга, бумага, меню

Содержимое, созданное искусственным интеллектом, может быть неверным.

Изображение выглядит как текст, бумага, книга, меню

Содержимое, созданное искусственным интеллектом, может быть неверным.



Содержание

[Введение 5](#_Toc203225274)

[1 Квантовые компьютеры 6](#_Toc203225275)

[1.1 Общие сведения о квантовых компьютерах 6](#_Toc203225276)

[1.2 Обзор применения квантовых компьютеров в различных отраслях 8](#_Toc203225277)

[1.3 Современные технологии квантовых компьютеров 10](#_Toc203225278)

[1.4 Проблемы и перспективы развития квантовых компьютеров 13](#_Toc203225279)

[2 Описание программы 15](#_Toc203225280)

[ЗАКЛЮЧЕНИЕ 21](#_Toc203225281)

[СПИСОК ЛИТЕРАТУРЫ 22](#_Toc203225282)

[Приложение 1 (обязательное) 23](#_Toc203225283)

# Введение

В последние десятилетия квантовые компьютеры перешли из области теоретических исследований в стадию активной разработки, привлекая внимание как научного сообщества, так и крупных технологических корпораций. В отличие от классических компьютеров, которые оперируют битами (0 и 1), квантовые компьютеры используют кубиты, способные находиться в суперпозиции состояний и проявлять квантовую запутанность. Это открывает принципиально новые возможности для решения задач, недоступных даже самым мощным суперкомпьютерам.

Актуальность темы обусловлена потенциальным прорывом в таких областях, как криптография, моделирование сложных молекулярных систем, оптимизация и искусственный интеллект. Уже сегодня компании, такие как IBM, Google и IonQ, демонстрируют работающие квантовые процессоры, а эксперименты вроде достижения "квантового превосходства" подтверждают практическую значимость этой технологии.

# 1 Квантовые компьютеры

## 1.1 Общие сведения о квантовых компьютерах

Квантовые компьютеры представляют собой принципиально новый класс вычислительных устройств, основанных на законах квантовой механики. В отличие от классических компьютеров, оперирующих бинарными битами, квантовые системы используют кубиты - квантовые аналоги битов, способные находиться в состоянии суперпозиции. Это означает, что кубит может одновременно принимать значения 0 и 1 с определенной вероятностью, что открывает возможности для параллельных вычислений принципиально нового типа.

Основу работы квантовых компьютеров составляют три фундаментальных квантовых явления: суперпозиция состояний, квантовая запутанность и квантовая интерференция. Суперпозиция позволяет кубиту существовать в смешанном состоянии, запутанность обеспечивает мгновенную корреляцию между частицами независимо от расстояния, а интерференция дает возможность усиливать "правильные" и подавлять "неправильные" результаты вычислений. В совокупности эти свойства обеспечивают так называемый квантовый параллелизм - способность одновременно обрабатывать экспоненциально большое количество возможных состояний. [1]

Современные квантовые компьютеры можно разделить на несколько основных типов. Универсальные квантовые компьютеры, разрабатываемые компаниями IBM, Google и другими, предназначены для выполнения произвольных квантовых алгоритмов. Альтернативный подход реализован в адиабатических квантовых компьютерах, таких как системы D-Wave, которые специализируются на решении задач оптимизации методом квантового отжига. Отдельную категорию составляют квантовые симуляторы - устройства, предназначенные для моделирования сложных квантовых систем.

Физическая реализация кубитов может быть различной. Наибольшее распространение получили сверхпроводящие кубиты, используемые в системах IBM и Google, которые работают при сверхнизких температурах. Альтернативные подходы включают ионные ловушки, отличающиеся высокой точностью операций, топологические кубиты, потенциально более устойчивые к декогеренции, и фотонные системы, перспективные для квантовых коммуникаций. [2]

## 1.2 Обзор применения квантовых компьютеров в различных отраслях

Квантовые компьютеры, благодаря своей способности обрабатывать информацию принципиально новым способом, открывают перспективы для революционных изменений в самых разных сферах человеческой деятельности. Хотя массовое внедрение этой технологии еще впереди, уже сегодня можно выделить несколько ключевых направлений, где квантовые вычисления демонстрируют наибольший потенциал.

В финансовой сфере квантовые компьютеры могут произвести настоящую революцию. Они способны значительно ускорить сложные вычисления, связанные с оценкой рисков, оптимизацией инвестиционных портфелей и моделированием финансовых рынков. Особый интерес представляет возможность быстрого анализа огромных массивов данных для выявления скрытых закономерностей и прогнозирования рыночных тенденций. Квантовые алгоритмы уже сейчас тестируются крупными банками и финансовыми институтами для решения задач монте-карловского моделирования и обнаружения мошеннических операций.

Фармацевтика и химическая промышленность — еще одна область, где ожидается прорыв благодаря квантовым технологиям. Традиционные методы компьютерного моделирования молекулярных структур сталкиваются с фундаментальными ограничениями при работе со сложными молекулами. Квантовые компьютеры, работающие по тем же законам, что и моделируемые молекулы, потенциально могут точно предсказывать свойства новых лекарственных соединений, ускорять разработку препаратов и даже помогать в создании новых материалов с заданными характеристиками. Это может сократить сроки и стоимость разработки новых лекарств в разы.

В логистике и управлении цепями поставок квантовые алгоритмы обещают радикально улучшить процессы оптимизации. Задачи маршрутизации, распределения ресурсов и управления запасами, которые требуют перебора огромного количества вариантов, идеально подходят для квантового подхода. Уже сейчас ведущие логистические компании экспериментируют с квантовыми алгоритмами для поиска оптимальных маршрутов доставки и минимизации транспортных издержек.

Искусственный интеллект и машинное обучение также могут получить значительный импульс от развития квантовых технологий. Квантовые компьютеры потенциально способны ускорять обучение нейронных сетей, особенно в задачах распознавания образов и обработки естественного языка. Особый интерес представляет возможность реализации принципиально новых алгоритмов машинного обучения, основанных на квантовых принципах.

В энергетике квантовые вычисления могут помочь в разработке новых аккумуляторов и материалов для солнечных панелей, а также в оптимизации работы энергосетей. Моделирование квантовых систем на квантовых же компьютерах открывает перспективы для создания сверхпроводников, работающих при комнатной температуре, что может привести к настоящей энергетической революции.

Кибербезопасность сталкивается с двойственным влиянием квантовых технологий. С одной стороны, квантовые компьютеры угрожают взломать современные криптографические системы, с другой — предлагают новые методы защиты информации на основе квантовой криптографии и алгоритмов, устойчивых к квантовым атакам.

Хотя практическое применение квантовых компьютеров в промышленных масштабах пока ограничено, активные исследования и тестовые внедрения в этих областях демонстрируют огромный потенциал технологии. По мере решения технических проблем и увеличения мощности квантовых систем, их влияние на различные отрасли экономики будет только расти, открывая новые горизонты для научных и технологических прорывов.

## 1.3 Современные технологии квантовых компьютеров

На сегодняшний день развитие квантовых вычислений идет по нескольким параллельным направлениям, каждое из которых предлагает уникальные подходы к реализации кубитов - базовых элементов квантовых процессоров. Эти технологии находятся на разных стадиях зрелости и обладают характерными преимуществами и ограничениями, определяющими сферы их потенциального применения.

Наиболее продвинутой и инвестиционно привлекательной платформой стали сверхпроводящие кубиты, используемые такими технологическими гигантами как IBM, Google и Intel. Эти системы работают при экстремально низких температурах, близких к абсолютному нулю, где металлические цепи демонстрируют сверхпроводящие свойства. Главное преимущество данного подхода - относительная простота масштабирования с использованием модифицированных технологий производства классических процессоров. Однако такие системы требуют сложнейших криогенных установок и остаются крайне чувствительными к внешним шумам. [3]

Альтернативное направление представляют ионные ловушки, разрабатываемые компаниями IonQ и Honeywell. В этих системах кубиты реализованы на отдельных атомах, удерживаемых в вакууме с помощью электромагнитных полей. Главное достоинство - исключительная стабильность и долгое время когерентности кубитов. Однако сложности с масштабированием и относительно низкая скорость операций пока ограничивают применение этой технологии.

Особый интерес вызывает развиваемый Microsoft подход с топологическими кубитами, которые теоретически должны быть значительно устойчивее к декогеренции благодаря использованию экзотических квазичастиц. Хотя практическая реализация этой концепции пока не достигнута, потенциальные преимущества делают данное направление крайне перспективным для создания надежных квантовых компьютеров будущего. [3]

Отдельно стоит отметить фотонные квантовые компьютеры, такие как разрабатываемые Xanadu и PsiQuantum. В этих системах кубиты кодируются в квантовых состояниях света, что позволяет работать при комнатной температуре и потенциально обеспечивает легкую интеграцию с существующей оптоволоконной инфраструктурой. Однако управление одиночными фотонами и их детектирование остаются серьезными технологическими вызовами.

Помимо универсальных квантовых компьютеров, существует отдельный класс адиабатических квантовых систем, представленных компанией D-Wave. Эти устройства специализируются на решении задач оптимизации методом квантового отжига, но не поддерживают полный набор квантовых операций. Несмотря на ограниченную область применения, они уже сегодня используются для решения практических задач в логистике и финансовом моделировании.

Современные квантовые процессоры, независимо от их физической реализации, относятся к классу так называемых "шумных промежуточных квантовых устройств" (NISQ). Они содержат от нескольких десятков до нескольких сотен кубитов, но пока не обладают достаточной надежностью для полноценного квантового превосходства. Основные усилия исследователей сейчас сосредоточены на разработке методов коррекции ошибок и повышении стабильности квантовых состояний. [3]

Развитие квантовых технологий происходит в условиях жесткой конкуренции между государственными научными центрами и частными корпорациями. Китайские исследователи добились значительных успехов в квантовой коммуникации, европейские центры сосредоточены на фундаментальных аспектах квантовых вычислений, а американские компании лидируют в создании коммерчески ориентированных квантовых процессоров.

По мере совершенствования технологий производства и управления кубитами, а также развития специализированного программного обеспечения, можно ожидать постепенного перехода от лабораторных образцов к промышленным решениям. Однако до создания универсального квантового компьютера, способного решать широкий круг практических задач, вероятно, пройдет еще не менее десятилетия интенсивных исследований и разработок. [3]

## 1.4 Проблемы и перспективы развития квантовых компьютеров

Несмотря на значительный прогресс в области квантовых вычислений, перед исследователями и инженерами по-прежнему стоят серьезные технологические вызовы, требующие фундаментальных научных прорывов и инновационных инженерных решений.

Одной из ключевых проблем остается декогеренция – потеря квантовой информации из-за взаимодействия кубитов с окружающей средой. Современные системы демонстрируют время когерентности от микросекунд до миллисекунд, что существенно ограничивает сложность выполняемых операций. Для борьбы с этим явлением разрабатываются различные подходы, включая пассивную защиту (улучшение изоляции кубитов) и активные методы квантовой коррекции ошибок, требующие значительного увеличения количества физических кубитов для создания логических кубитов.

Масштабирование квантовых систем представляет собой отдельную сложную задачу. Увеличение числа кубитов сопровождается ростом технических сложностей – от необходимости поддержания сверхнизких температур в сверхпроводящих системах до проблем управления отдельными атомами в ионных ловушках. Особую остроту приобретают вопросы обеспечения однородности характеристик кубитов и минимизации межкубитных помех в многокубитных процессорах.

Серьезным препятствием для практического применения остается отсутствие универсального программного обеспечения и алгоритмов, адаптированных для работы в условиях шумов и ошибок, характерных для современных NISQ-устройств. Разработка эффективных квантовых алгоритмов, устойчивых к ошибкам, требует тесного взаимодействия физиков, математиков и специалистов по компьютерным наукам.

Перспективы развития квантовых вычислений связаны с несколькими направлениями. В ближайшие 5-10 лет ожидается прогресс в создании гибридных систем, сочетающих классические и квантовые процессоры для решения конкретных прикладных задач. Особые надежды возлагаются на развитие топологических кубитов, которые потенциально могут обеспечить более высокую устойчивость к декогеренции.

В долгосрочной перспективе (10-20 лет) возможно создание универсальных отказоустойчивых квантовых компьютеров, способных выполнять произвольные квантовые алгоритмы. Это потребует не только технологических прорывов, но и развития новой элементной базы, систем охлаждения и методов управления квантовыми состояниями.

Особое значение приобретает развитие квантовых коммуникаций и создание квантового интернета, который позволит объединять отдельные квантовые процессоры в распределенные вычислительные системы. Параллельно ведутся работы по созданию постквантовой криптографии, устойчивой к атакам с использованием квантовых компьютеров.

Развитие квантовых технологий требует значительных инвестиций и международной кооперации. Многие страны включили квантовые вычисления в число приоритетных направлений научно-технического развития, что создает благоприятные условия для ускорения исследований в этой области. Однако до практической реализации всего потенциала квантовых вычислений предстоит преодолеть еще множество научных и технологических барьеров. [4]

# 2 Описание программы

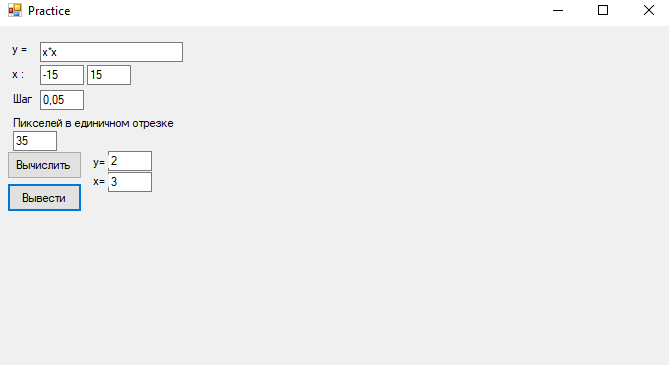
При запуске программы пользователю показывается главное окно с полями для ввода с предустановленными значениями. Пользователь может ввести саму функцию, область значений Х, шаг, с которым будут подставляться значения в функцию и количество пикселей в единичном отрезке графика, конкретное значение X для функции (Рисунок 4).  


Рисунок 4 – Пример главного окна.

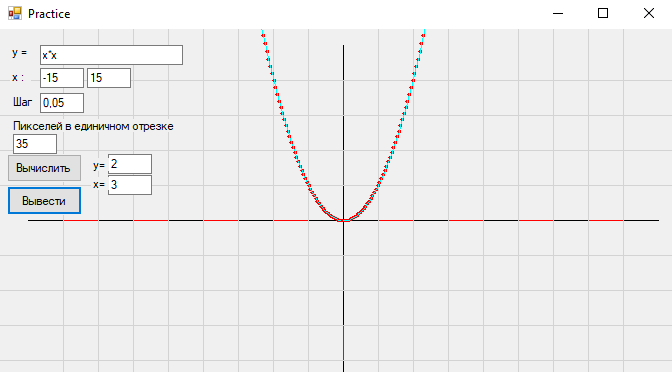
При нажатии кнопки Вывести и всех верно заполненных полях в график функции вместо переменной будут подставляться значения из диапазона значений Х с указанным шагом, каждая вычисленная точка будет выведена на график, после чего они соединятся синей линией (Рисунок 5).  


Рисунок 5 – Вывод параболы.

При нажатии кнопки Вычислить происходит отрисовка графика с выделением точки с указанным Х, выводится ее значение Y (Рисунок 6).

Программа поддерживает помимо баззовых арифметических операций возвередние в степень, функции синуса, косинуса, тангенса и котангенса (Рисунки 6-8).

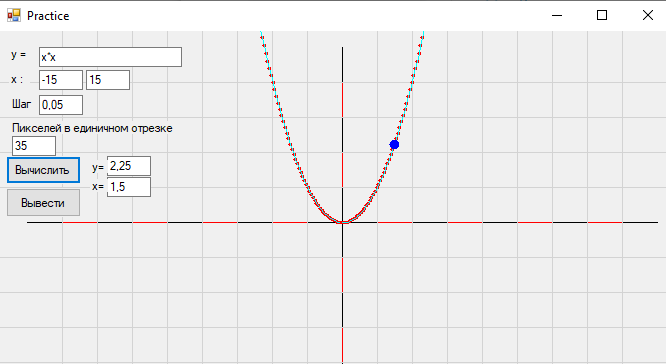


Рисунок 6 – Пример степенной функции.

Программа также поддерживает тригонометрические функции (Рисунки 7, 8).

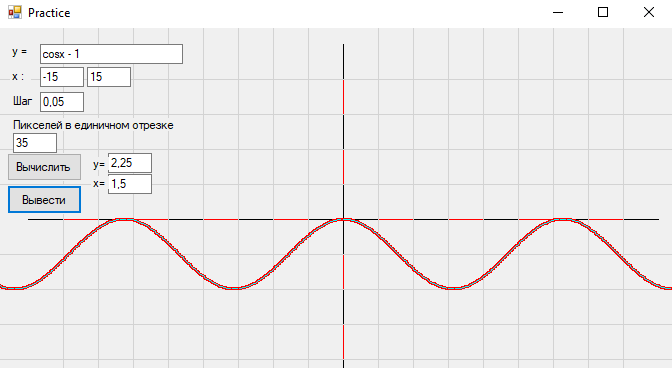


Рисунок 7 – Пример вывода функции косинуса.

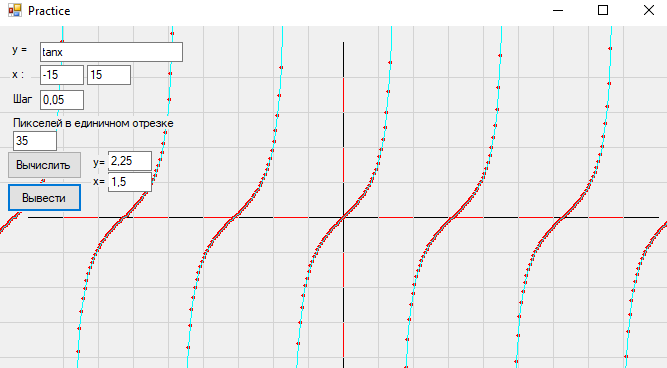


Рисунок 8 – Пример вывода функции тангенса.

Можно регулировать масштаб изображения путем изменения количества пикселей в единичном отрезке, а так же регулировать количество вычисляемых точек, указывая шаг (Рисунок 9).

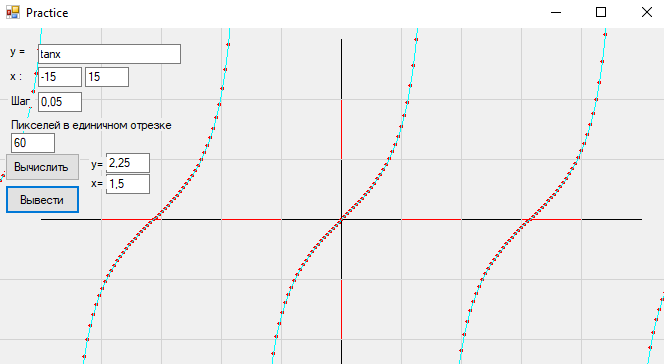


Рисунок 9 – Вывод функции тангенса с измененным масштабом и шагом

При нажатии кнопки Вывести или Вычислить предыдущий график стирается. После того, как был стёрт предыдущий график, осуществляется проверка введенных в полях значений. Если какое то из значений не прошло проверку, то новый график не отобразится.

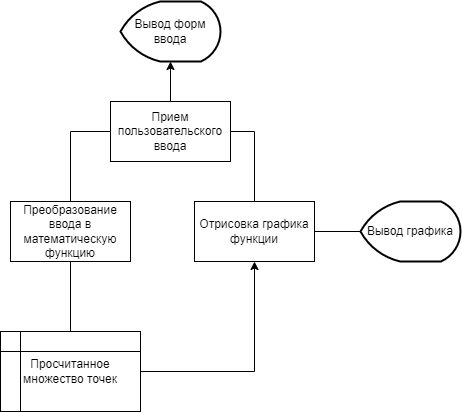


Рисунок 10 – Функицональная схема программы

Программа принимает на вход функицю f(x) в строковом формате. Затем в заданном числовом промежутке по оси Х с определенным шагом берутся отдельные числа и подставляются в функцию вместо переменной Х. Модуль рассчета переводит полученное выражение в пригодный для вычисления формат, разбивает его на отдельные операции и производит их в правильном математическом порядке. Рассчитанные значения функции в каждой точке сохраняются в опреативную память. Затем, после просчета всего числового промежутка рисуются координатные оси и на полученных осях выводятся рассчитанные точки, и между ними проводится линия. Точки выходящие за пределы экрана пропускаются. В случае нажатия на кнопку «Вычислить» помимо вышеперечисленных операций производится подсчет отдельно для искомой точки, и она выводится на экран более крупной и другим цветом (Рисунок 11).

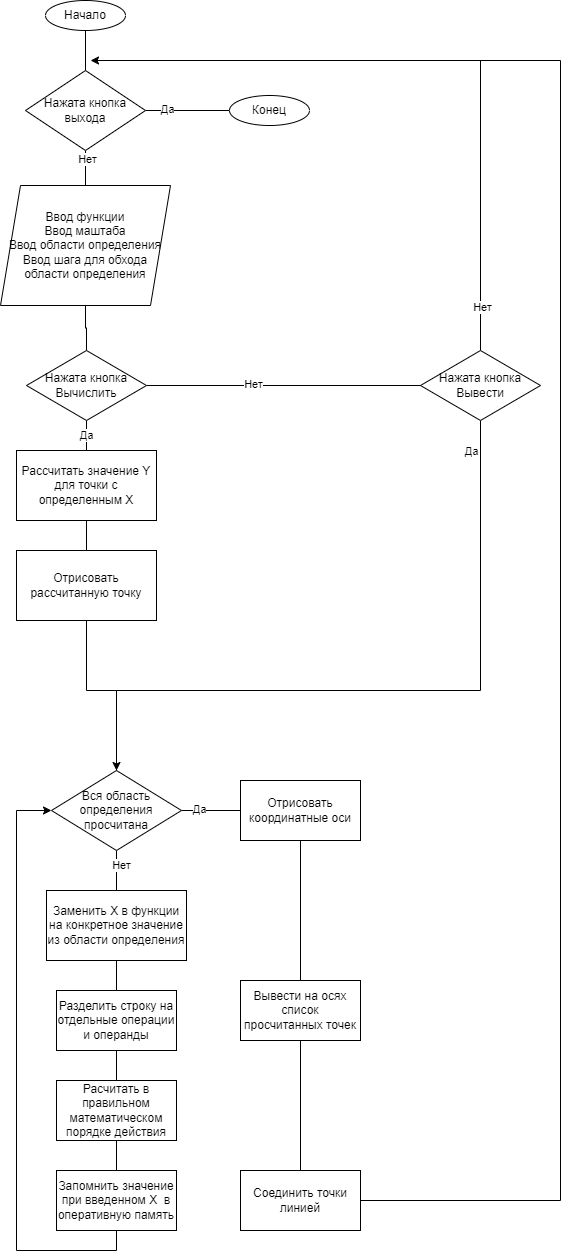


Рисунок 11 – Блок-схема программы

# ЗАКЛЮЧЕНИЕ

Несмотря на значительный прогресс в последние годы, современные квантовые компьютеры все еще сталкиваются с серьезными техническими вызовами, включая проблему декогеренции, ошибок квантовых операций и сложности масштабирования систем. Тем не менее, уже сейчас можно наблюдать первые практические применения этой технологии в области криптографии, оптимизации и моделирования молекулярных систем.

Перспективы развития квантовых вычислений выглядят чрезвычайно многообещающими. По мере решения существующих технических проблем и увеличения количества устойчивых кубитов, можно ожидать революционных изменений во многих областях науки и промышленности. Однако для полного раскрытия потенциала этой технологии потребуются значительные усилия как в области фундаментальных исследований, так и в разработке практических приложений.

В ходе выполнения работы были получены необходимые навыки, изучены все поставленные вопросы и достигнуты обозначенные цели. В следствие чего, считаю прохождение практики успешной.

# СПИСОК ЛИТЕРАТУРЫ

1. Прескилл Дж. Квантовая информация и квантовые вычисления: учебное пособие / Дж. Прескилл; пер. с англ. под ред. А.С. Холево. - М.: РХД, 2011. - 464 с.

2. Коваленко И. И., Прилипко В. К., Физические основы квантовых вычислений. Динамика кубита. — СПб.: Лань, 2019. — 214 с.

3. Лэдд Т.Д. Квантовые компьютеры / Т.Д. Лэдд [и др.] // Nature. - 2010. - Т. 464, № 7285. - С. 45-53. - DOI: 10.1038/nature08812.

4. Садовничий В.А. Квантовые технологии: проблемы и перспективы. - М.: МГУ, 2021. - 512 с.

# Приложение 1 (обязательное)

**Листинг файла Form1.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Linq.Expressions;

using System.Diagnostics;

using System.Xml.Schema;

namespace SummerPractice

{

public partial class Form1 : Form

{

Graphics g;

Point center;

int scale = 20;

double gap = 0.5;

double xAbs = 0;

public Form1()

{

InitializeComponent();

}

public void drawXLine()

{

Pen pen;

int count = (center.X) / scale;

for (int i = 0; i < count; i++)

{

if (i % 2 == 0)

{

pen = Pens.Black;

}

else

{

pen = Pens.Red;

}

g.DrawLine(pen, center.X + scale \* i, center.Y, center.X + scale \* (i+1), center.Y);

g.DrawLine(pen, center.X + scale \* -i, center.Y, center.X + scale \* (-i-1), center.Y);

}

}

public void drawYLine()

{

Pen pen;

int count = (center.Y) / scale;

for (int i = 0; i < count; i++)

{

if (i % 2 == 0)

{

pen = Pens.Black;

}

else

{

pen = Pens.Red;

}

g.DrawLine(pen, center.X, center.Y + scale \* i, center.X, center.Y + scale \* (i+1));

g.DrawLine(pen, center.X, center.Y + scale \* -i, center.X, center.Y + scale \* (-i-1));

}

}

private List<Point> calculateFunction()

{

List<Point> points = new List<Point>();

string func = input.Text;

for (double x = -xAbs; x <= xAbs; x += gap)

{

double result = calculatePoint(func, x);

points.Add(new Point(center.X + (int)(x \* scale), center.Y - (int)(result \* scale)));

}

return points;

}

private double calculatePoint(string func, double x)

{

string xValue = String.Format("{0:F10}", x);

string expr = func.Replace("x", xValue);

expr = expr.Replace(",", ".");

double res = StringToFormula.Eval(expr);

return res;

}

private void showResult()

{

List<Point> points = calculateFunction();

for (int i = 0; i < points.Count; i++)

{

if(checkPointInBounds(points[i]) == false)

{

continue;

}

drawPoint(points[i]);

if (i > 0)

{

if (checkPointInBounds(points[i-1]) == false)

{

continue;

}

g.DrawLine(Pens.Aqua, points[i - 1], points[i]);

}

}

}

private bool checkPointInBounds(Point point)

{

if (Math.Abs(this.Width - point.X) > this.Width\*2 || Math.Abs(this.Height - point.Y) > this.Height\*2)

{

return false;

}

return true;

}

private void drawPoint(Point point)

{

g.FillEllipse(Brushes.Red, point.X-2, point.Y-2, 4, 4);

}

private bool parseInput()

{

if (!Int32.TryParse(scaleTextBox.Text, out scale))

{

return false;

}

if (!double.TryParse(gapTextBox.Text, out gap))

{

return false;

}

if (!double.TryParse(xAbsTextBox.Text, out xAbs))

{

return false;

}

return true;

}

private void paintButton\_Click(object sender, EventArgs e)

{

g = CreateGraphics();

g.Clear(DefaultBackColor);

center = new Point(this.Width / 2, this.Height / 2);

if (parseInput())

{

drawXLine();

drawYLine();

showResult();

}

}

}

}

# Приложение 1 (обязательное)

**Листинг файла StringToFormula.cs**

using System;

using System.Collections.Generic;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace SummerPractice

{

public static class StringToFormula

{

private static readonly string[] operators = { "+", "-", "/", "%", "\*", "^"};

private static readonly string[] functions = {"sin", "cos", "tan", "cot"};

private static readonly Func<double, double, double>[] operations = {

(a1, a2) => a1 + a2,

(a1, a2) => a1 - a2,

(a1, a2) => a1 / a2,

(a1, a2) => a1 % a2,

(a1, a2) => a1 \* a2,

(a1, a2) => Math.Pow(a1, a2),

};

private static readonly Func<double, double>[] trigonometryOperations = {

(a1) => Math.Sin(a1),

(a1) => Math.Cos(a1),

(a1) => Math.Tan(a1),

(a1) => Math.Cos(a1) / Math.Sin(a1),

};

public static bool TryEval(string expression, out double value)

{

try

{

value = Eval(expression);

return true;

}

catch

{

value = 0.0;

return false;

}

}

public static double Eval(string expression)

{

if (string.IsNullOrEmpty(expression))

return 0.0;

if (double.TryParse(expression, out double value))

return value;

List<string> tokens = GetTokens(expression);

tokens.Add("$"); // Append end of expression token

Stack<double> operandStack = new Stack<double>();

Stack<string> operatorStack = new Stack<string>();

int tokenIndex = 0;

while (tokenIndex < tokens.Count - 1)

{

string token = tokens[tokenIndex];

string nextToken = tokens[tokenIndex + 1];

switch (token)

{

case "(":

{

string subExpr = GetSubExpression(tokens, ref tokenIndex);

operandStack.Push(Eval(subExpr));

continue;

}

case ")":

throw new ArgumentException("Mis-matched parentheses in expression");

case "sin":

case "cos":

case "tan":

case "cot":

if (nextToken == "(")

{

int newIndex = tokenIndex + 1;

string subExpr = GetSubExpression(tokens, ref newIndex);

double computedValue = trigonometryOperations[Array.IndexOf(functions, token)](Eval(subExpr));

operandStack.Push(computedValue);

tokenIndex = newIndex;

}

else

{

string computableNumber = nextToken;

if (nextToken == "-")

{

computableNumber += tokens[tokenIndex + 2];

tokenIndex++;

}

double computedValue = trigonometryOperations[Array.IndexOf(functions, token)](double.Parse($"{computableNumber}", CultureInfo.InvariantCulture));

operandStack.Push(computedValue);

tokenIndex += 2;

}

continue;

// Handle unary ops

case "-":

case "+":

{

if (!IsOperator(nextToken) && operatorStack.Count == operandStack.Count)

{

operandStack.Push(double.Parse($"{token}{nextToken}", CultureInfo.InvariantCulture));

tokenIndex += 2;

continue;

}

}

break;

}

if (IsOperator(token))

{

while (operatorStack.Count > 0 && OperatorPrecedence(token) <= OperatorPrecedence(operatorStack.Peek()))

{

if (!ResolveOperation())

{

throw new ArgumentException(BuildOpError());

}

}

operatorStack.Push(token);

}

else

{

operandStack.Push(double.Parse(token, CultureInfo.InvariantCulture));

}

tokenIndex += 1;

}

while (operatorStack.Count > 0)

{

if (!ResolveOperation())

throw new ArgumentException(BuildOpError());

}

return operandStack.Pop();

bool IsOperator(string token)

{

return Array.IndexOf(operators, token) >= 0;

}

int OperatorPrecedence(string op)

{

switch (op)

{

case "^":

return 3;

case "\*":

case "/":

case "%":

return 2;

case "+":

case "-":

return 1;

default:

return 0;

}

}

string BuildOpError()

{

string op = operatorStack.Pop();

string rhs = operandStack.Any() ? operandStack.Pop().ToString() : "null";

string lhs = operandStack.Any() ? operandStack.Pop().ToString() : "null";

return $"Operation not supported: {lhs} {op} {rhs}";

}

bool ResolveOperation()

{

string op = operatorStack.Pop();

double rhs = operandStack.Pop();

double lhs = operandStack.Pop();

operandStack.Push(operations[Array.IndexOf(operators, op)](lhs, rhs));

Console.WriteLine($"Resolve {lhs} {op} {rhs} = {operandStack.Peek()}");

return true;

}

}

private static string GetSubExpression(List<string> tokens, ref int index)

{

StringBuilder subExpr = new StringBuilder();

int parenlevels = 1;

index += 1;

while (index < tokens.Count && parenlevels > 0)

{

string token = tokens[index];

switch (token)

{

case "(": parenlevels += 1; break;

case ")": parenlevels -= 1; break;

}

if (parenlevels > 0)

subExpr.Append(token);

index += 1;

}

if (parenlevels > 0)

throw new ArgumentException("Mis-matched parentheses in expression");

return subExpr.ToString();

}

private static List<string> GetTokens(string expression)

{

string operators = "()^\*/%+-";

string digits = "0123456789.,";

string[] funcs = {

"sin",

"cos",

"tan",

"cot"

};

List<string> tokens = new List<string>();

StringBuilder sb = new StringBuilder();

expression = expression.Replace(" ", string.Empty);

for(int i = 0; i < expression.Length; i++)

{

char c = expression[i];

if (operators.IndexOf(c) >= 0)

{

if ((sb.Length > 0))

{

tokens.Add(sb.ToString());

sb.Length = 0;

}

tokens.Add(c.ToString());

}

else if(digits.IndexOf(c) == -1 && i+2 < expression.Length)

{

string func = expression.Substring(i, 3);

if(Array.IndexOf(funcs, func) >= 0) //got func

{

if ((sb.Length > 0))

{

tokens.Add(sb.ToString());

sb.Length = 0;

}

tokens.Add(func);

}

}

else if (digits.IndexOf(c) >= 0)

{

sb.Append(c);

}

}

if ((sb.Length > 0))

{

tokens.Add(sb.ToString());

}

return tokens;

}

}

}

Изображение выглядит как текст, меню, бумага, книга

Содержимое, созданное искусственным интеллектом, может быть неверным.

Изображение выглядит как текст, бумага, книга, Бумажное изделие

Содержимое, созданное искусственным интеллектом, может быть неверным.

