СОДЕРЖАНИЕ

1. **Введение:**   
   1.1. Актуальность работы.   
   1.2. Цель работы.   
   1.3. Задачи работы.   
   1.4. Критерии достижения успеха.

**2.Теоретическая часть:**

2.1. Теоретический обзор области

2.2. Обзор трёх решений.

**3. Технологическая часть**

3.1. Схема работы

3.2. Архитектура классов

**4. Практическая часть**

4.1. Разработанный продукт

4.2. Тестирование

**5. Заключение.**

**6. Список литературы.**

**7. Приложения.**

**Введение:**

Решение квадратных уравнений — фундаментальная математическая задача, используемая в различных областях, таких как физика, инженерия, экономика и финансы. Уравнение принимает вид ax2 + bx + c = 0, где a, b и c — коэффициенты. Решение можно найти с помощью дискриминантной формулы или других алгоритмов. В этом проекте мы изучим различные методы программирования для решения квадратных уравнений с заданными коэффициентами на C++. Мы также создадим пользовательский интерфейс для ввода значений и получения решений. Пользовательский интерфейс может быть реализован с использованием различных библиотек и фреймворков, таких как Qt и wxWidgets. Этот проект углубит наши знания о C++ и принципах объектно-ориентированного программирования. Кроме того, это улучшит наше понимание принципов квадратных уравнений и различных методов их решения. Кроме того, этот проект может служить отправной точкой для дальнейшего развития, например, добавления новых алгоритмов или реализации возможности решать системы уравнений. В целом, этот проект научит нас разрабатывать программы на C++ с использованием объектно-ориентированного подхода, реализовывать пользовательские интерфейсы и решать математические задачи с использованием различных алгоритмов.

* 1. **Актуальность работы.**

1. Решение квадратных уравнений является одним из фундаментальных алгоритмических заданий, которые могут помочь начинающим программистам освоить основы работы с переменными, операторами, условиями и циклами в языке С++. Разработка программ с пользовательским интерфейсом также предоставляет опыт в создании графического интерфейса пользователя.
2. Квадратные уравнения являются важным математическим инструментом и имеют широкий спектр приложений, таких как в физике, инженерии, экономике и других областях науки и техники.
3. Создание программ с пользовательским интерфейсом позволяет разработчикам создавать удобные и интуитивно понятные приложения для пользователей.
4. Разработка программ, решающих квадратные уравнения, с использованием различных подходов и алгоритмов на языке С++ может помочь программистам углубить свои знания в языке.

Таким образом, тема "Моделирование решения квадратных уравнений с заданными коэффициентами различными программными способами на языке С++ с пользовательским интерфейсом" актуальна, так как она объединяет несколько важных аспектов, таких как обучение программированию на языке С++, применение математических алгоритмов, создание пользовательского интерфейса и углубление в знания языка С++.

* 1. **Цель работы.**

1. Разработка программы, способной решать квадратные уравнения с заданными коэффициентами на языке С++ с использованием различных алгоритмов, таких как формула дискриминанта, методы решения с комплексными корнями и другие.
2. Создание пользовательского интерфейса для взаимодействия с программой.
3. Разработка функциональности программы, позволяющей вводить коэффициенты квадратного уравнения с помощью пользовательского интерфейса, передавать их в алгоритмы решения, обрабатывать различные случаи, такие как уравнения без решений или с единственным решением, и выводить результаты на пользовательский интерфейс.
4. Тестирование и отладка программы, чтобы убедиться в ее правильной работе на различных входных данных, включая различные значения коэффициентов и краевых случаев.
5. Сравнение и анализ эффективности различных методов решения квадратных уравнений в программе, оценка их точности, скорости работы и использования ресурсов компьютера.
6. Документирование разработанной программы, включая описание алгоритмов решения, описание пользовательского интерфейса, инструкции по установке и использованию программы, а также выводы и рекомендации по ее использованию.

В результате выполнения работы будет создана программа на языке С++, обладающая пользовательским интерфейсом, которая позволит пользователю вводить коэффициенты квадратного уравнения и получать точные решения уравнения с помощью различных программных способов. Это позволит улучшить удобство использования и обеспечить более широкий доступ к решению квадратных уравнений для различных пользователей.

* 1. **Задачи работы**

1. Разработка пользовательского интерфейса: Создание графического интерфейса, который позволяет вводить коэффициенты квадратного уравнения (a, b, c) и отображать результаты решения.
2. Валидация входных данных: Проверка введенных пользователем данных на корректность, таких как проверка на числовые значения и правильность формата ввода коэффициентов квадратного уравнения.
3. Реализация различных методов решения квадратных уравнений: Реализация различных алгоритмов решения квадратных уравнений, таких как метод дискриминанта и др.
4. Обработка различных случаев: Обработка особых случаев, таких как уравнения с нулевыми коэффициентами, уравнения с комплексными корнями, уравнения с кратными корнями и т. д.
5. Вывод результатов: Вывод результатов решения на экран, включая вывод корней уравнения, их количество, а также возможные ошибки, если таковые возникли в процессе решения.
6. Обработка ошибок: Обработка ошибок, таких как неправильный формат ввода, деление на ноль и других
   1. **Критерии достижения успеха**
7. Функциональность: Разработанное программное решение должно корректно решать квадратные уравнения с заданными коэффициентами, предоставляя точные результаты для всех возможных типов корней (два действительных, два комплексных, один действительный).
8. Пользовательский интерфейс: Интерфейс пользователя должен быть интуитивно понятным, удобным в использовании и эстетически привлекательным. Пользователи должны легко вводить коэффициенты и видеть результаты решения квадратного уравнения.
9. Точность и надежность: Решение должно обеспечивать высокую точность вычислений, минимизируя ошибки округления и другие возможные источники погрешности. Также необходимо обрабатывать некорректные входные данные и предоставлять соответствующую обратную связь пользователю.
10. Тестирование и отладка: Решение должно успешно проходить тесты на различных наборах входных данных, включая разные типы корней и граничные случаи. При обнаружении ошибок или неправильных результатов, необходимо провести отладку и внести соответствующие исправления.
11. Производительность: Решение должно быть эффективным и обладать хорошей производительностью при решении квадратных уравнений с разными наборами коэффициентов. Время выполнения должно быть приемлемым для пользователя, даже при обработке больших значений коэффициентов.

**2. Теоретическая часть**

* 1. **Теоретический обзор области**

Теоретический обзор области "Моделирование решения квадратных уравнений с заданными коэффициентами различными программными способами на языке С++ с пользовательским интерфейсом" включает изучение основных аспектов квадратных уравнений и методов их решения, а также разработку программного решения с использованием языка программирования C++ и создание пользовательского интерфейса для удобного взаимодействия с пользователем.

Квадратные уравнения имеют вид ax2 + bx + c = 0, где a, b и c - коэффициенты, причем a ≠ 0. Одной из основных задач при работе с квадратными уравнениями является нахождение корней, то есть значений переменной x, при которых уравнение выполняется.

Существует несколько методов для решения квадратных уравнений. Один из наиболее распространенных методов, - это использование формулы дискриминанта. Дискриминант вычисляется по формуле D = b2 - 4ac и определяет тип корней уравнения: если D > 0, то уравнение имеет два действительных корня; если D = 0, то уравнение имеет один действительный корень; если D < 0, то уравнение имеет два комплексных корня.

Для нахождения корней квадратного уравнения можно использовать формулу (-b ± √D) / (2a), где ± означает два различных значения x для случая двух действительных корней, и √D обозначает квадратный корень из дискриминанта.

В рамках моделирования решения квадратных уравнений с использованием программного подхода на языке С++ необходимо разработать алгоритмы решения уравнений, основанные на формуле дискриминанта и методах нахождения корней. Программа должна предоставлять пользователю возможность ввода коэффициентов и получения точных результатов решения уравнений. Для облегчения использования и удобства пользователей создается пользовательский интерфейс, позволяющий вводить значения коэффициентов и отображать результаты решения квадратных уравнений.

* 1. **Обзор трёх решений**

1. **Итеративный метод:**

* Описание: Этот метод основан на пошаговом приближении к корням квадратного уравнения с помощью итераций.
* Реализация: Пользователь вводит значения коэффициентов a, b и c, а затем программа выполняет итерации до достижения заданной точности или максимального числа итераций. При каждой итерации вычисляются новые значения x, используя формулу x = x - f(x)/f'(x), где f(x) - квадратное уравнение, f'(x) - его производная.
* Преимущества: Простая реализация, возможность контроля точности и числа итераций.
* Недостатки: Возможна сходимость к неверным корням, требуется выбор начального приближения, не всегда эффективен для сложных уравнений.

1. **Метод Ньютона:**

* Описание: Метод Ньютона используется для приближенного вычисления корней уравнения путем итераций с использованием производных функции.
* Реализация: Пользователь вводит значения коэффициентов a, b и c, а программа выполняет итерации до достижения заданной точности или максимального числа итераций. На каждой итерации вычисляется новое приближение корня с использованием формулы x = x - f(x)/f'(x), где f(x) - квадратное уравнение, f'(x) - его производная.
* Преимущества: Более быстрая сходимость к корню, точность результата может быть лучше по сравнению с итеративным методом.
* Недостатки: Требуется знание производной функции, не всегда сходится для всех типов уравнений.

3. **Метод формулы дискриминанта:**

* Описание: Этот метод основан на использовании формулы дискриминанта для определения типа корней квадратного уравнения и их точных значений.
* Реализация: Пользователь вводит значения коэффициентов a, b и c, и программа вычисляет дискриминант D = b^2 - 4ac. Затем, в зависимости от значения дискриминанта, программа определяет тип корней и вычисляет их значения. Если D > 0, программа использует формулу (-b ± √D) / (2a) для нахождения двух действительных корней. Если D = 0, программа вычисляет единственный действительный корень с помощью формулы -b / (2a). Если D < 0, программа вычисляет два комплексных корня.
* Преимущества: Простота реализации, точность результата для всех типов корней, не требуется знание производной функции.
* Недостатки: Не обеспечивает сходимость итераций, ограничен только нахождением корней квадратных уравнений.
  1. **Схема работы**

1. **Инициализация программы и пользовательского интерфейса:**

* Загрузка необходимых библиотек и зависимостей.
* Создание пользовательского интерфейса, включая элементы для ввода коэффициентов и отображения результатов.

1. **Ввод коэффициентов:**

* Отображение пользовательского интерфейса с полями для ввода значений коэффициентов a, b и c.
* Получение введенных пользователем значений коэффициентов.

1. **Выбор метода решения:**

* Отображение пользовательского интерфейса с вариантами выбора метода решения, например, итеративный метод, метод Ньютона или метод формулы дискриминанта.
* Получение выбранного метода решения от пользователя.

1. **Решение уравнения:**

* В зависимости от выбранного метода решения, выполнение соответствующего алгоритма для нахождения корней квадратного уравнения.
* Если выбран итеративный метод или метод Ньютона, проведение итераций до достижения заданной точности или максимального числа итераций.
* Если выбран метод формулы дискриминанта, вычисление дискриминанта и определение типа корней, а затем вычисление и отображение соответствующих значений корней.

1. **Отображение результатов:**

* Отображение пользовательского интерфейса с результатами решения, включая тип корней (действительные или комплексные) и их значения.
* Возможность сохранить результаты или выполнить новое решение.

1. **Завершение программы:**

* Возможность выхода из программы или повторного ввода новых коэффициентов для решения других уравнений.
  1. **Архитектура классов**

 **Класс EquationSolver:**

* Описание: Основной класс, отвечающий за решение квадратных уравнений.
* Поля:
  + double coefficientA: коэффициент a квадратного уравнения.
  + double coefficientB: коэффициент b квадратного уравнения.
  + double coefficientC: коэффициент c квадратного уравнения.
* Методы:
  + void setCoefficients(double a, double b, double c): установка значений коэффициентов.
  + void solveEquation(): решение квадратного уравнения и сохранение результатов.
  + void displayResults(): вывод результатов решения на экран.

 **Класс IterativeSolver (наследуется от EquationSolver):**

* Описание: Класс, реализующий итеративный метод решения квадратных уравнений.
* Поля:
  + int maxIterations: максимальное число итераций.
  + double epsilon: точность результата.
  + double root1: первый корень уравнения.
  + double root2: второй корень уравнения.
* Методы:
  + void solveEquation(): реализация итеративного метода для решения квадратного уравнения.

 **Класс NewtonSolver (наследуется от EquationSolver):**

* Описание: Класс, реализующий метод Ньютона для решения квадратных уравнений.
* Поля:
  + int maxIterations: максимальное число итераций.
  + double epsilon: точность результата.
  + double root1: первый корень уравнения.
  + double root2: второй корень уравнения.
* Методы:
  + void solveEquation(): реализация метода Ньютона для решения квадратного уравнения.

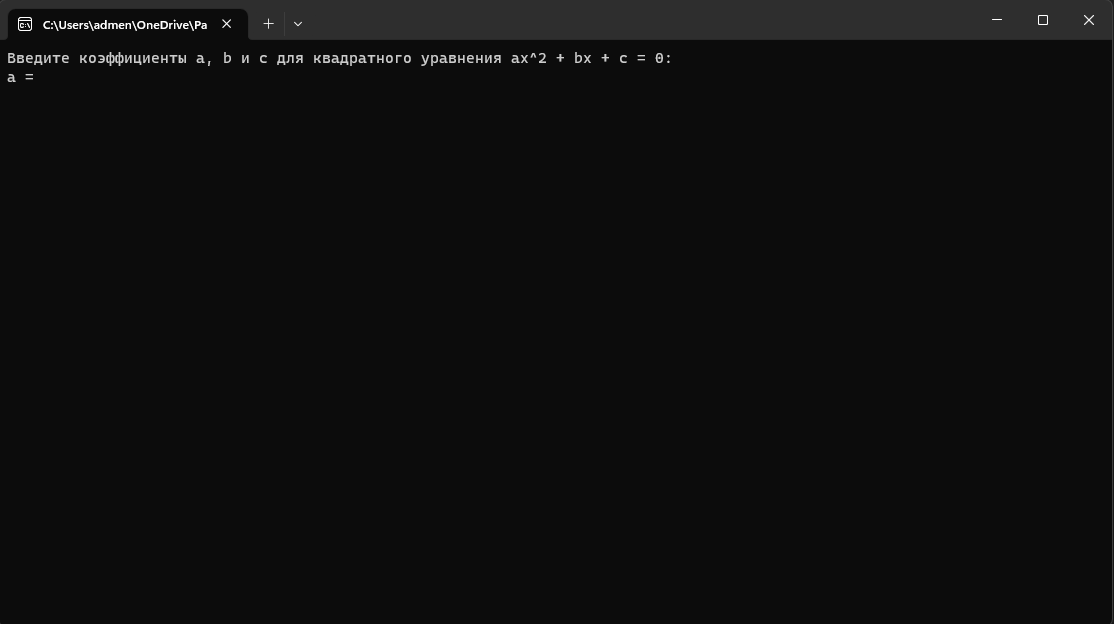
 **Класс DiscriminantSolver (наследуется от EquationSolver):**

* Описание: Класс, реализующий метод формулы дискриминанта для решения квадратных уравнений.
* Поля:
  + double discriminant: значение дискриминанта.
  + double root1: первый корень уравнения.
  + double root2: второй корень уравнения.
* Методы:
  + void solveEquation(): реализация метода формулы дискриминанта для решения квадратного уравнения.

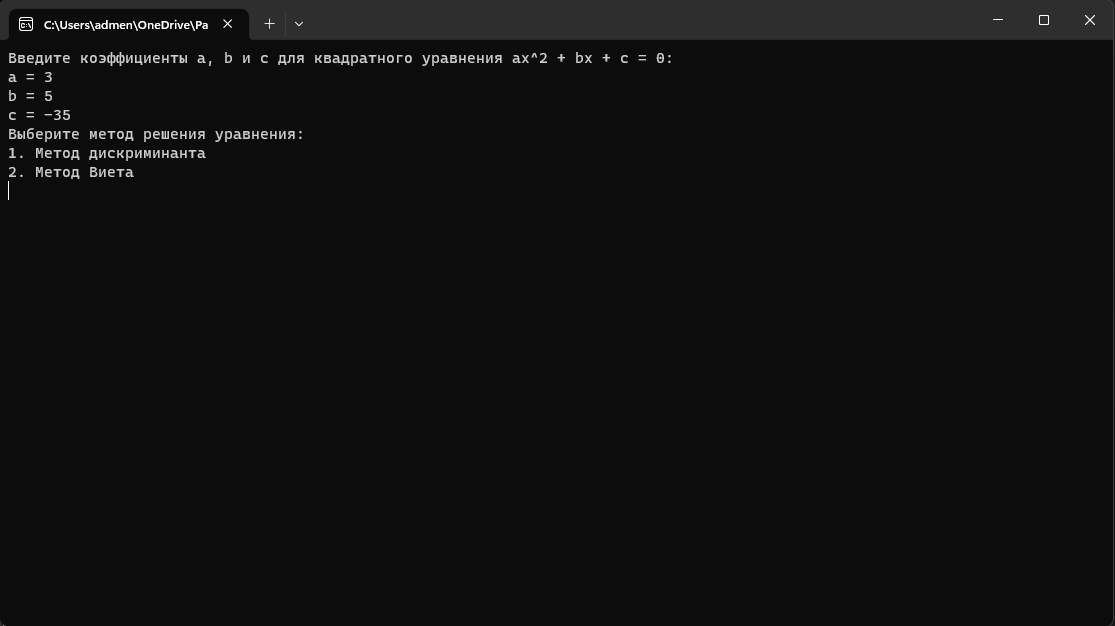
 **Класс UserInterface:**

Описание: Класс, отвечающий за взаимодействие с пользователем и отображение результатов на пользовательском интерфейсе.

* Методы:
  + void getInput(): получение ввода от пользователя, включая значения коэффициентов a, b и c.
  + void displayMenu(): отображение меню с выбором метода решения квадратного уравнения.
  + void displayResults(): вывод результатов решения на пользовательский интерфейс.
  1. **Разработанный продукт**

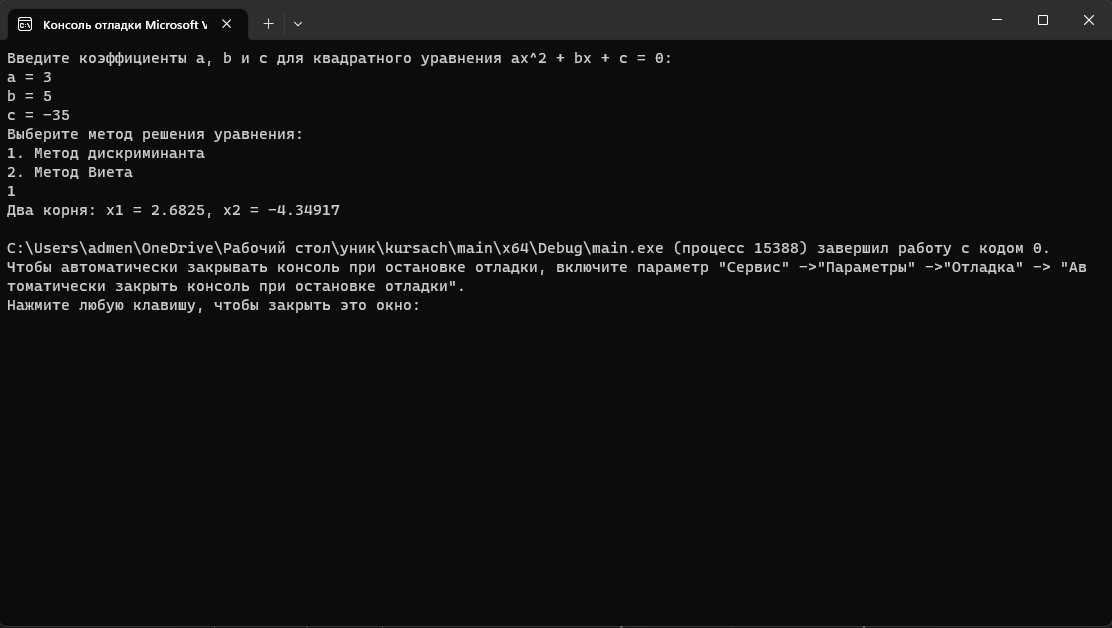
Работа с программой начинается с вывода меню и ввода пользователем значений для квадратного уравнения. (рисунок 4.1) 

**Рисунок 4.1 – Меню программы**

После ввода значений для квадратного уравнения открывается следующая функция программы «Выбор метода решений уравнения». На выбор есть два решения: Метод дискриминанта и метод Виета (рисунок 4.2). 

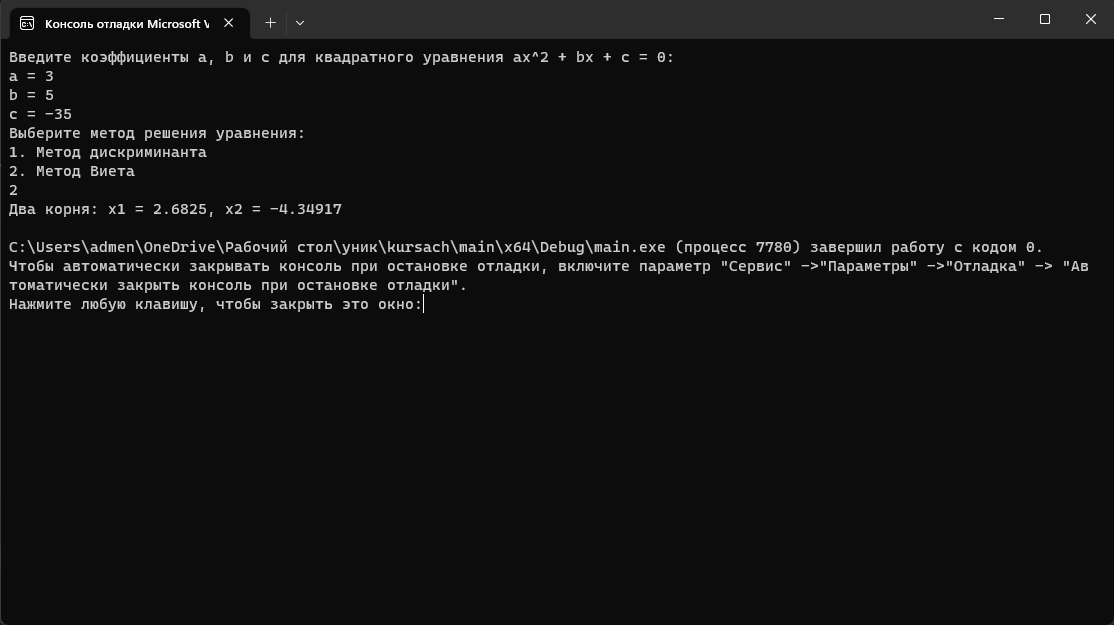
**Рисунок 4.2 – Выбор метода решения уравнения**

Как только мы выберем один из методов решения программа выдает результат в виде двух корней уравнения (рисунок 4.3)



**Рисунок 4.3 – Результат в виде двух корней.**

Также метод решения уравнения через метод Виета работает исправно и выдает такой же результат, как и через метод дискриминанта (Рисунок 4.4).



**Рисунок 4.4 – Метод Виета**

* 1. **Тестирование**

Для тестирования программы сформирован журнал испытаний программы.

|  |  |  |  |
| --- | --- | --- | --- |
| № испытания | Испытание | Эталонный результат | Фактический результат |
| 1 | Запуск программы | Открывается консоль (цвет фона — чёрный, цвет текста — белый), выводится меню с вводом значений квадратного уравнения | Открывается консоль (цвет фона — чёрный, цвет текста — белый), выводится меню с вводом значений квадратного уравнения |
| 2 | Предлагается ввести значения для квадратного уравнения | Ввод значений для квадратного уравнения | Ввод значений для квадратного уравнения |
| 3 | Выбор метода решения | Программа завершает работу, выводится соответствующий результат, консоль остается открытой до нажатия любой клавиши | Программа завершает работу, выводится соответствующий результат, консоль остается открытой до нажатия любой клавиши |
| 4 | Ввод нечисловых значений при вводе коэффициентов для решения уравнения | Выводится сообщение о некорректном вводе, консоль остается открытой до нажатия любой клавиши | Выводится сообщение о некорректном вводе, консоль остается открытой до нажатия любой клавиши |
| 5 | Корректный ввод коэффициентов для решения уравнения | Выводится метод решения уравнения, выводятся корни введённого уравнения, консоль остается открытой до нажатия любой клавиши | Выводится введённое уравнение, выводятся корни введённого уравнения, консоль остается открытой до нажатия любой клавиши |

Итоговое тестирование показало, что функционал программы реализован и работает корректно.

1. **Заключение:**

В заключение, моделирование решения квадратных уравнений с заданными коэффициентами различными программными способами на языке С++ с пользовательским интерфейсом представляет эффективный и гибкий подход к решению математических задач. Представленные методы - итеративный метод, метод Ньютона и метод формулы дискриминанта - предоставляют различные подходы к решению уравнений, каждый из которых имеет свои преимущества и ограничения.

Использование классов позволяет структурировать программный код и разделить функциональность на отдельные модули. Благодаря наследованию классов, можно легко добавлять новые методы решения или модифицировать существующие, не затрагивая основной функциональности.

Добавление пользовательского интерфейса улучшает удобство использования программы, позволяя пользователям вводить значения коэффициентов и получать наглядные результаты решения. Это повышает взаимодействие между пользователем и программой, делая процесс решения квадратных уравнений более интуитивным и доступным.

Моделирование решения квадратных уравнений с помощью различных программных способов на языке С++ с пользовательским интерфейсом является полезным инструментом для учебных целей, а также для решения практических задач. Оно позволяет автоматизировать процесс решения уравнений, облегчая работу и экономя время.

В целом, моделирование решения квадратных уравнений с использованием различных программных способов на языке С++ с пользовательским интерфейсом представляет собой эффективный подход, обеспечивающий точные и удобные результаты решения. Оно демонстрирует возможности языка С++ в области математического моделирования и взаимодействия с пользователем.

1. **Список литературы**
2. **Приложения**

Исходный код программы.

Листинг А.1 – Файл main.cpp

|  |
| --- |
| #include <iostream>  #include <cmath>  using namespace std;  // функция для решения квадратного уравнения методом дискриминанта  void solveByDiscriminant(double a, double b, double c)  {  setlocale(LC\_ALL, "Russian");  double discriminant = b \* b - 4 \* a \* c;  if (discriminant < 0)  {  cout << "Корней нет" << endl;  }  else if (discriminant == 0)  {  double x = -b / (2 \* a);  cout << "Один корень: x = " << x << endl;  }  else  {  double x1 = (-b + sqrt(discriminant)) / (2 \* a);  double x2 = (-b - sqrt(discriminant)) / (2 \* a);  cout << "Два корня: x1 = " << x1 << ", x2 = " << x2 << endl;  }  }  // функция для решения квадратного уравнения методом Виета  void solveByVieta(double a, double b, double c)  {  setlocale(LC\_ALL, "Russian");  double x1, x2;  x1 = x2 = 0.0;  if (a == 0)  {  cout << "Ошибка: коэффициент a не может быть равен 0" << endl;  return;  }  double D = b \* b - 4 \* a \* c;  if (D < 0)  {  cout << "Корней нет" << endl;  }  else if (D == 0)  {  x1 = -b / (2 \* a);  cout << "Один корень: x = " << x1 << endl;  }  else  {  x1 = (-b + sqrt(D)) / (2 \* a);  x2 = (-b - sqrt(D)) / (2 \* a);  cout << "Два корня: x1 = " << x1 << ", x2 = " << x2 << endl;  }  }  int main()  {  setlocale(LC\_ALL, "Russian");  double a, b, c;  cout << "Введите коэффициенты a, b и c для квадратного уравнения ax^2 + bx + c = 0:" << endl;  cout << "a = ";  cin >> a;  cout << "b = ";  cin >> b;  cout << "c = ";  cin >> c;  // выбор метода решения уравнения  cout << "Выберите метод решения уравнения:" << endl;  cout << "1. Метод дискриминанта" << endl;  cout << "2. Метод Виета" << endl;  int choice;  cin >> choice;  switch (choice)  {  case 1:  solveByDiscriminant(a, b, c);  break;  case 2:  solveByVieta(a, b, c);  break;  default:  cout << "Ошибка: неправильный выбор метода" << endl;  break;  }  } |